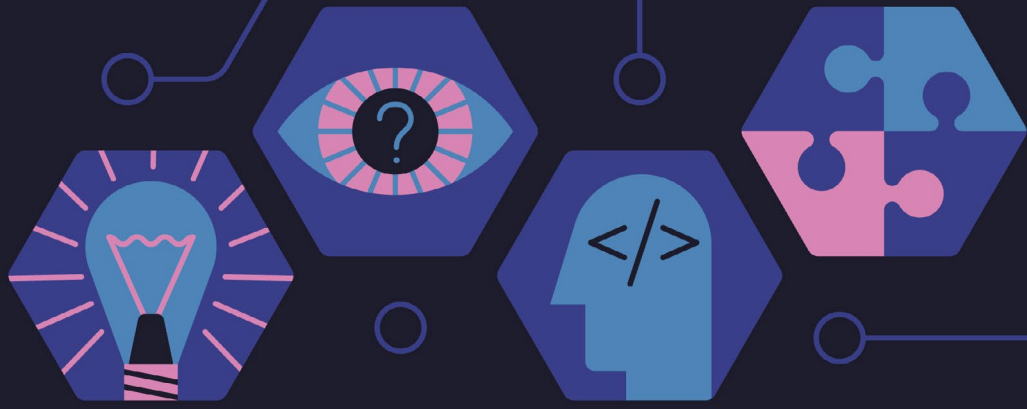


(Hello
World)



THE
BIG BOOK

OF

COMPUTING
PEDAGOGY





Raspberry Pi

Learn with the Raspberry Pi Foundation

Free for everyone anywhere in the world



Teaching resources

Discover training, resources, and guidance to help you teach computing with confidence.

teachcomputing.org



Project library

Browse our 200+ online project guides that include step-by-step instructions for learners.

projects.raspberrypi.org



Digital making at home

Check out our code-along videos and take part in Astro Pi Mission Zero from home.

raspberrypi.org/learn

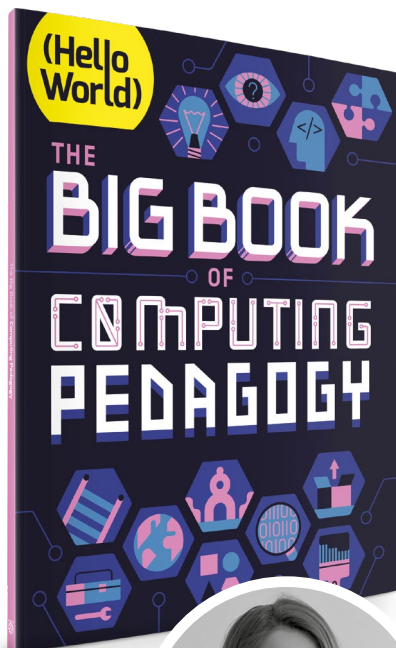


Support for parents

Watch our support tutorials and access engaging resources for your child.

raspberrypi.org/learn

HELLO, WORLD!



After nearly five years of Hello World, we're delighted to welcome you to our first-ever special edition, *The Big Book of Computing Pedagogy*! This special edition focuses on approaches to teaching computing in the classroom, and includes some of our favourite pedagogically themed articles from previous issues of Hello World, as well as a couple of never-seen-before pieces. It is structured around twelve pedagogical principles, originally developed by the Raspberry Pi Foundation for the National Centre for Computing Education in England. These principles, which are introduced on page five, are based on up-to-date research about the best ways of approaching the teaching and learning of computing.

Computing education is still relatively new, and is constantly changing and adapting. Despite leaving school less than ten years ago, my days in the computer lab were limited to learning about how to add animations on PowerPoints and trying out basic Excel formulas (and yes, there was still the odd mouse with a ball knocking about!). Computing education research is even younger, and the Raspberry Pi Foundation is proud to be an important part of this growing space, having launched the Raspberry Pi Computing Education Research Centre at the University of Cambridge in the summer.

The Big Book of Computing Pedagogy aims to be your companion to learning about tried and tested approaches to teaching computing. As with all Hello World issues, we've worked to bridge the gap between research and practice, giving you accessible chunks of research, followed by stories of trusty educators trying out an approach in their classroom or educational space.

We'd love to hear what you think of this very exciting collection. Get in touch at contact@helloworld.cc or on Twitter [@HelloWorld_Edu](https://twitter.com/HelloWorld_Edu).

Gemma Coleman
Editor

EDITORIAL

Editor

Gemma Coleman

Subeditors

Louise Richmond and Amy Rutter

Subscriptions

Joshua Crossman

Social Media

Neena Patel and Claire MacDonald

Pedagogical consultant

James Robinson

DESIGN

criticalmedia.co.uk

Head of Design

Lee Allen

Designers

Ty Logan, Sam Ribbits

Photography

Raspberry Pi Foundation, Adobe Stock

Graphics

Rob Jervis

Cover and illustrations

©Muti, Folio Art

CONTRIBUTORS

Efthimia Aivaloglou, Samantha Baloro, George Boukeas, Claire Buckler, Mark Calleja, Katharine Childs, Josh Crossman, Paul Curzon, Jonathan Dickens, Catherine Elliott, Lucia Floriánová, Rebecca Franks, Ben Garside, Sway Grantham, Will Grey, Shuchi Grover, Ben Hall, Amanda Haughs, Katie Henry, Felienne Hermans, Nic Hughes, Simon Humphreys, Thom Kunkeler, Hayley Leonard, Linda Liukas, Ursula Martin, Karl Maton, Tilman Michaeli, Gemma Moine, Carrie Anne Philbin, Mareen Przybylla, Oliver Quinlan, Neil Rickus, Calvin Robinson, James Robinson, Laura Sach, Stefan Seegerer, Cynthia Selby, Sue Sentance, Lucinda Tuttiert, Jane Waite, Matthew Wimpenny-Smith



This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

Hello World is a joint collaboration:



Hello World is published by the Raspberry Pi Foundation, 37 Hills Road, Cambridge, CB2 1NT. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to skills, products, or services referred to in the magazine. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).

FOREWORD

As a former computer science (CS) professor and industry professional, I have recently taken on a new role that permits me to take deep dives into studying CS education at primary and secondary levels. In the USA alone, this means investigating what it means to teach CS to over 50,000,000 students, all with their own backgrounds, cultures, interests, and academic goals.

Given that schools in the USA are under state and local district control, efforts to convince over 13,000 school districts and 50 states (plus US territories) that CS is not a fad and deserves a place in the classroom have been slow — but steadily growing. This raises a critical question for each school and district as they embrace this change: how do we train teachers to teach a brand new subject area in a way that is most impactful for all students?

There is no doubt that the United Kingdom is ahead of the USA in its path of training teachers to teach CS to all students. Computing at School (CAS), the Raspberry Pi Foundation, and many other organisations have provided training and free resources to teachers, often relying on education researchers to provide important guidance and direction in finding promising teaching practices, such as in this special edition of Hello World.

When I started to peruse the draft for *The Big Book of Computing Pedagogy*, I was simply stunned. I found the ready-to-consume content to be solidly based on research evidence and

tried-and-true best practices from teachers themselves. This resource provides valuable insights into introducing computing to students via unplugged activities, integrating the Predict–Run–Investigate–Modify–Make (PRIMM) pedagogical model, and introducing physical devices for computing — all written in a way that teachers can adopt and use in their own classrooms.

We share in your successes of bringing CS to all students in the UK, while at the same time work to gain an understanding of the challenges encountered in training teachers and how those challenges are being mitigated. This first special edition of Hello World is another concentrated effort to mitigate those challenges by providing timely, evidence-based resources in bite-sized pieces that are ready for consumption — not just for teachers in the UK, but all around the world. I have no doubt that this will be a valuable resource to both novice and experienced teachers, now and in the future.



Monica McGill, EdD
Founder and CEO
CSEdResearch.org

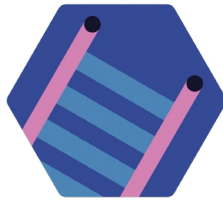
Computer Science Teachers Association (CSTA)
Board Member
Association of Computing Machinery — Women
(ACM-Women) North America Committee,
Immediate Past Chair

HOW WE TEACH COMPUTING

Much of the work of the Raspberry Pi Foundation is underpinned by our twelve principles of computing pedagogy



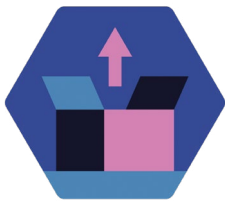
LEAD WITH CONCEPTS



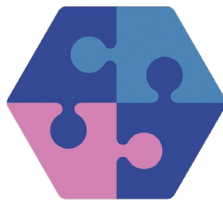
STRUCTURE LESSONS



MAKE CONCRETE



**UNPLUG, UNPACK,
REPACK**



WORK TOGETHER



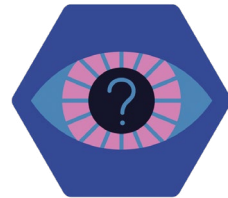
**READ AND EXPLORE
CODE FIRST**



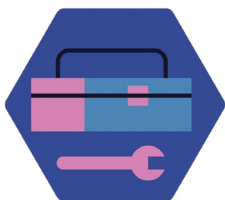
**FOSTER PROGRAM
COMPREHENSION**



MODEL EVERYTHING



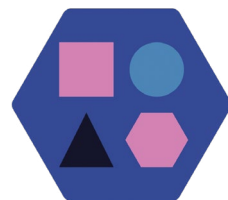
**CHALLENGE
MISCONCEPTIONS**



CREATE PROJECTS



GETS HANDS-ON



ADD VARIETY

CONTENTS



LEAD WITH CONCEPTS

- 10 CONCEPT MAPS**
A tool for planning, teaching, learning, and assessment
- 13 VELA CONCEPTS**
Non-programming activities to teach programming concepts
- 14 THE 'RIGHT' WAY?**
Learning when to simplify concepts
- 16 LEARNING GRAPHS**
A tool to plan for progression

STRUCTURE LESSONS

- 20 COGNITIVE LOAD THEORY**
How thoughtful instructional design can reduce cognitive load
- 22 THE PRIMM APPROACH**
A framework to structure programming lessons
- 25 UDL**
Universal Design for Learning in computing
- 28 CODING & 21ST-CENTURY SKILLS**
A framework developing coding alongside critical thinking, collaboration, creativity, and communication skills
- 30 CURRICULUM DESIGN**
The ABC approach to curriculum design, introducing online and blended formats to the classroom

MAKE CONCRETE

- 34 CULTURALLY RELEVANT PEDAGOGY**
Drawing upon student experience and cultural knowledge to make learning more relevant
- 36 LEARNING THROUGH MAKING**
Constructionism as a way of making concepts concrete
- 38 SCRATCHMATHS**
Integrating maths and computing
- 40 SCRATCH ENCORE**
A culturally relevant Scratch curriculum
- 41 ENGINEERING SKILLS**
Nurturing engineering skills in all students
- 42 PLAYING WITH PLUGS**
Breaking down abstract concepts

UNPLUG, UNPACK, REPACK

- 46 SEMANTIC WAVES**
The ideal conceptual journey for novice learners to follow
- 49 GO UNPLUGGED**
The value of unplugged activities
- 50 CRAZY CHARACTERS**
Reviewing a lesson activity using semantic waves

WORK TOGETHER

- 56 PEER INSTRUCTION**
Combining carefully chosen MCQs with peer discussion
- 58 PAIR PROGRAMMING**
Pairing learners to work through programming problems
- 60 COLLABORATIVE PROBLEM-SOLVING**
Could the future of learning be working together?
- 62 ENCOURAGING TALK**
Collaboration and communication in the computing curriculum
- 64 VERSION CONTROL**
A collaborative programming tool

READ AND EXPLORE CODE FIRST

- 68 CODE TRACING**
Reading and analysing code first to develop program comprehension
- 70 READ BEFORE YOU WRITE**
Evidence-based approaches for helping pupils to read code
- 72 ASSEMBLY LANGUAGE**
Using Raspberry Pi to teach assembly language to A level students



FOSTER PROGRAM COMPREHENSION

- 78 THE BLOCK MODEL**
A useful tool for understanding aspects of program comprehension
- 80 PARSON'S PROBLEMS**
Exercises to develop learners' program comprehension
- 82 WRITING CODE**
Evidence-based approaches for helping pupils to write code
- 84 THE I IN PRIMM**
Creating opportunities for students to investigate code fully

MODEL EVERYTHING

- 90 WORKED EXAMPLES**
Creating blueprints for solving new but related problems
- 92 LIVE CODING**
Developing programming solutions in real time
- 94 VIDEOS AND SELF-EXPLANATION**
Another approach to teaching programming
- 96 MODELLING FOR LEARNERS**
Top tips for implementing the modelling approach
- 98 WATCH AND LEARN**
How online video is changing the way we teach computer science
- 100 SMELLY CODE**
Passing on best practice when teaching programming to primary students

CHALLENGE MISCONCEPTIONS

- 104 ALTERNATIVE CONCEPTIONS**
Research into the importance of challenging misconceptions
- 106 ASSESSMENT FOR LEARNING**
Diagnosing students' learning needs by asking the right questions
- 108 METAPHORS AND MISCONCEPTIONS**
Metaphors and methods to avoid misconceptions
- 111 MULTIPLE CHOICE**
How to write effective MCQs
- 112 INTRODUCTORY PROGRAMMING**
Addressing misconceptions in introductory programming

CREATE PROJECTS

- 116 PROJECT-BASED LEARNING**
Applying programming knowledge to real-world scenarios
- 118 DIGITAL PROJECTS**
Research about how children make digital projects
- 122 A PATH TO AGENCY**
Projects as a tool for students to take charge of their own learning
- 124 DESIGN JOURNALS**
Introducing design journals to primary school students

GET HANDS-ON


- 130 PHYSICAL COMPUTING**
The methods and benefits of bringing physical computing to the classroom
- 132 PHYSICAL COMPUTING IN THE CLASSROOM**
Guidelines for planning and applying physical computing lessons
- 134 REFLECTIONS**
Considering and overcoming the barriers to physical computing
- 137 PRIMARILY PI**
Teaching our youngest learners physical computing

ADD VARIETY

- 143 VARIETY IN TEACHING**
Introducing variety in teaching and assessment of programming activities
- 146 STORYTELLING**
Using children's literature to teach computing to primary school pupils
- 148 RETRIEVAL PRACTICE**
Bringing information to mind to boost learning
- 152 THE INCLUSIVE CLASSROOM**
Approaches to making computing lessons more accessible and inclusive
- 154 ART AND ALGORITHMS**
Creating algorithmic art in primary computing lessons
- 156 PROGRAMMING AND PLAY**
How embracing play can make you a better educator



LEAD WITH CONCEPTS

- 10** CONCEPT MAPS
 - 13** USING NON-PROGRAMMING
ACTIVITIES TO TEACH
PROGRAMMING CONCEPTS
 - 14** THE CODE'S NOT ALL RIGHT
 - 16** LEARNING GRAPHS: TOOLS TO
PLAN FOR PROGRESSION
- 

As a discipline, computing is broad, with connections to many other subject areas. It is also rich in concepts (for example variables, binary numbers, or the ‘fetch, decode, execute’ cycle). While some are quite tangible, many are very abstract in nature. As an educator, it can be easy to get distracted by the technology, tool, programming language, or context that your learners are using. However, it is the concepts that should be the main focus of any teaching. With a rich knowledge of these concepts, pupils can build a solid understanding of all areas of computing, which can then be applied to any new tools, contexts, programming languages, and projects they encounter.

You can support learners’ conceptual understanding by using key terms and vocabulary consistently to build a shared understanding. Concepts can be explored and connected using tools such as concept maps. You can also use glossaries to help students define key terminology and help them revise and revisit those terms in the future. Regular recall and revision should form part of your everyday practice, to ensure that these concepts and the terms that describe them become embedded in your students’ understanding and vocabulary.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:** concept maps
- **What the research says:** using non-programming activities to teach programming concepts
- Simplifying concepts
- Learning graphs: tools to plan progression



CONCEPT MAPS

These versatile educational tools can help learners and educators capture, communicate, and assess knowledge

Concept maps are “graphical tools for organising and representing knowledge”.¹ In education, they can capture the knowledge of subject experts, educators, and learners, so they can be used for planning, teaching, learning, and assessment. A related set of concepts is shown in boxes (nodes) with lines connecting them to indicate relationships. These links are labelled (unlike in mind maps) to describe the relationships between them. Therefore, following a link between concepts forms some meaning or a short proposition. For example, in a concept map about data structures, you are likely to find the proposition: list → is an example of a → data structure.

Concepts may be structured in a hierarchy. Links that span across different branches of the hierarchy are called cross-links, and uncover deeper connections.

A concept map captures knowledge relevant to a focus question that provides context for the map and helps direct its construction and comprehension. Some nodes in a map may not correspond to concepts, but may be concrete examples of concepts instead. Concept maps are a means for “externalising cognition, making mental models visible so that they can be compared and combined”.² As such, they can be useful for representing the knowledge of both educators and learners, making them a versatile educational tool.

How to make them

Whether you want to use concept maps for planning, to share with your learners, or as a task for them to complete, you can find examples online (helloworld.cc/cmapex), and a guide to constructing them follows below.

Determine the focus question: Specify what the knowledge represented in a concept map will be about. The focus question can be broad, such as, “How are images represented using binary digits?” or specific, such as, “How does this piece of code work?”

Populate a short list of concepts:

Determine the list of concepts, ideas, or keywords that are relevant to the focus question. If it makes sense, order them according to how general they are, or how relevant they are to the focus question. This is the first step towards structuring the concepts into a hierarchy.

Explore the relationships between concepts:

Link related concepts with an arrow and label the links to specify their relationships. You should be able to form a meaningful statement when following a link from one concept to another.

Improve and refine: Building concept maps is always an iterative process, and concept maps should never be considered finished.

When using concept maps with learners, the process of constructing them, or even simply reading and interpreting them, must be modelled by the educator. Building a concept map can also be a collaborative activity between either the learners and the educator or small groups of learners.

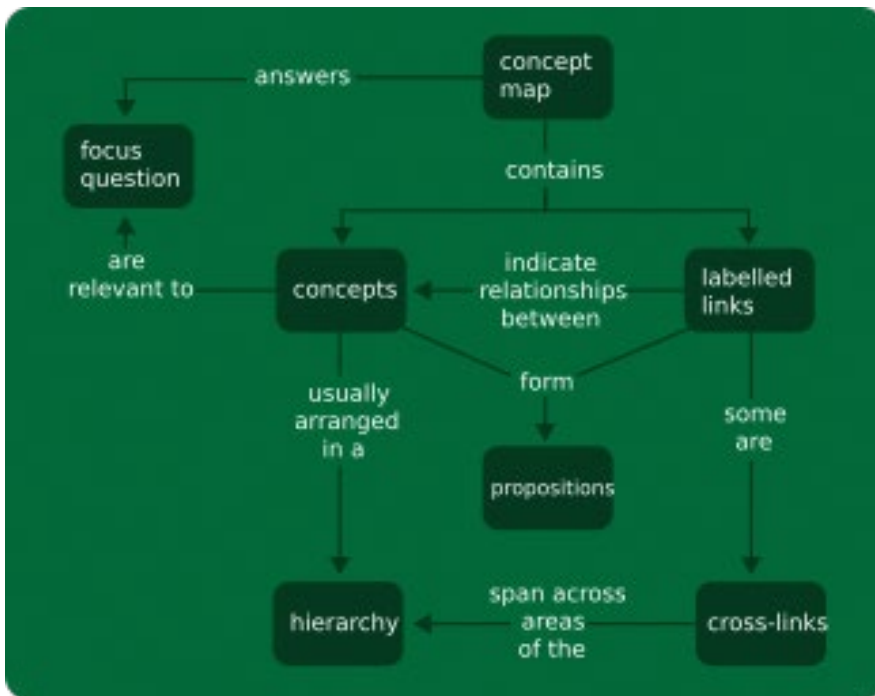
How they can be used in education

Educators can use concept maps to capture the knowledge that they aim to convey to their learners. Concept maps have a specific structure, which places restrictions on their expressive power. Therefore, in order to represent knowledge with a concept map,



■ Concept maps can be used to represent the knowledge of both teachers and learners

Credit: Vasily Merkushev/stock.adobe.com



educators have to break that knowledge down into short propositions. This is a challenging but illuminating process of introspection and iterative refinement that will help them create a representation of their teaching content in a simple, distilled form.

Concept maps that capture subject knowledge can be used by educators in a variety of ways. They can inform the

knowledge is introduced, which will support learners to organise this new information and connect it to existing schemas (clusters of connected ideas). Concept maps can also be used as study guides and revision tools.

Finally, they can be used for assessment in a number of ways. For example, educators could ask learners to fill in concepts or links, or to extend an existing concept map. Learners could repeat this process over

“ CONCEPT MAPS ARE USEFUL FOR REPRESENTING THE KNOWLEDGE OF BOTH EDUCATORS AND LEARNERS

planning of a lesson or sequence of lessons; they can serve as a means of communication with other educators, especially if they are drawn collaboratively; and they can also be used to support teaching and assessment.

Concept maps can be presented to learners as a supplementary way to provide or summarise information. Educators should start with a minimal skeleton map that will be iteratively extended when new

several lessons: they could start with a short list of concepts, which they could then incorporate into their concept maps as the lessons progress. This would help identify misconceptions or gaps in understanding at every step. Educators could use these activities to aid recall or assess prior knowledge, or to prompt learners to summarise new concepts while connecting them to form propositions.

SUMMARY

Components

- Concepts, which are often arranged in a hierarchy, form the basis of concept maps.
- Links connect the concepts to indicate relationships between them. Cross-links are links across different branches of the hierarchy.
- Labels on the links (usually verbs) specify the relationships between the concepts.
- A focus question provides context for the concept map and guides its construction.
- Propositions, or units of meaning, are formed by following a link from one concept to another, e.g. “A concept map answers a focus question.”
- In a concept map, examples are instances of concepts that may be included to make concepts clearer and more concrete.

Construction

- Determine the focus question
- Shortlist (and possibly order) the relevant concepts
- Connect concepts from the shortlist to form propositions
- Proceed iteratively: connecting, rearranging, and refining

Uses

- Informing the planning of learning experiences
- Facilitating communication and collaboration between educators
- Presenting or summarising information for learners
- Supporting learners to connect new information to existing knowledge
- Assessing learners’ prior knowledge, misconceptions, and understanding of new information

For concept maps to have a positive impact, they need to be a fully integrated feature of the teaching and learning process.

Credit: Prostock-studio/stock.adobe.com



■ Concept maps can be used as study or revision tools

- ▶ Concept maps should be tightly integrated into the teaching and learning process, and they should only be used for assessment if they have been consistently used in teaching.

“ CONCEPT MAPS REPRESENT KNOWLEDGE VISUALLY

Why they should be used

There is sufficient evidence to suggest that an integrated use of concept maps in teaching can be beneficial to learners, especially less confident learners, or those who struggle with reading.^{3, 4, 5}

Concept maps represent knowledge visually, highlighting the structure and

the connections between concepts. Their potential to enhance learning, retention, and transfer is often linked to cognitive load theory: as learners struggle to identify concepts, and as the connections between concepts place a burden on working memory, concept maps aid schema construction within long-term memory by “serving as a kind of template or scaffold to help organise knowledge and to structure it, even though the structure must be built up piece by piece”.¹ This links back to the suggestion that concept maps should be developed iteratively, through “an orderly sequence of iterations between working memory and long-term memory, as new knowledge is being received and processed”.¹

There are many ways in which you can use this beneficial educational tool. You can find examples at helloworld.cc/cmapex, and use them to support trialling this planning, learning, and assessment technique in your classroom. (HAW)

REFERENCES

- ¹ Novak, J. D., & Cañas, A. (2008). *The Theory Underlying Concept Maps and How to Construct and Use Them*. Florida Institute for Human and Machine Cognition. helloworld.cc/concept1
- ² Wilson, G. (2019). *Teaching Tech Together*. CRC Press. helloworld.cc/concept2
- ³ Nesbit, J. C., & Adesope, O. O. (2006). Learning With Concept and Knowledge Maps: A Meta-Analysis. *Review of Educational Research*, 76(3), 413-448. helloworld.cc/concept3
- ⁴ Horton, P. B., McConney, A. A., Gallo, M., Woods, A. L., Senn, G. J., & Hamelin, D. (1993). An investigation of the effectiveness of concept mapping as an instructional tool. *Science Education*, 77(1), 95-111. helloworld.cc/concept4
- ⁵ Cañas, A. J., Coffey, J. W., Carnot, M. J., Feltovich, P., Hoffman, R. R., Feltovich, J., & Novak, J. D. (2003). *A Summary of Literature Pertaining to the Use of Concept Mapping Techniques and Technologies for Education and Performance Support*. Florida Institute for Human and Machine Cognition. helloworld.cc/concept5

USING NON-PROGRAMMING ACTIVITIES TO TEACH PROGRAMMING CONCEPTS

STORY BY Thom Kunkeler

Programming takes place on a computer, but research has shown the potential of using non-programming activities to teach concepts to novice learners. The study, conducted by Shuchi Grover and her colleagues at three urban schools in the US, included 16 non-programming activities over a 20-day programme. The research team found that the learning gains from students who followed the intervention were significantly higher than those of students who followed the regular computer science curriculum.

“PROGRAMMING LANGUAGES CAN BE HARD TO GRASP

From the moment learners engage with any programming language or tool, they will encounter variables, expressions, loops, and abstraction — the VELA concepts — in one way or another.

Some learners struggle with these concepts, as they assume that variables can have multiple values at the same

time, whereas others might not be able to distinguish between what goes inside a loop and what precedes a loop. Although visual block-based programming environments such as Scratch, Alice, and Snap! have helped with navigating through different concepts, students still find it difficult to make sense of programming languages.

For this study, the research team designed non-programming activities to help students engage with and understand the foundation of programming concepts. Activities such as Story Variables and Cats and Ladders were designed to encourage students to come up with a definition of a variable collaboratively,

before applying this definition in practice. In the first activity, students discussed multiple examples in which variables were present, such as: “Last week, I bought a pen for \$1.50; now it costs \$3”, to identify what the variable is and how it changes over time.

In the second activity, students put their definition into practice by determining the length of a ladder required to reach distraught cats in Cats and Ladders. Students produced a range of possible values that were needed to rescue the cats, and engaged in abstraction by synthesising new variables based on existing ones. Try the activities yourself! (HW)

TRY IT FOR YOURSELF!

During the 20-day research intervention, students participated in 16 activities. The research team gathered data based on a pre- and post-assessment of students’ introductory programming skills, analysis of their Scratch projects, and interviews with teachers and students. There were higher learning gains for students following the intervention, but

the findings were not linked to gender or prior academic preparation, and all grades participating in the study showed similar learning gains. For teachers, these findings should be taken as an incentive to experiment with using non-programming activities to help with teaching difficult concepts. Give some of these non-programming activities a go!

FURTHER READING

- ✓ Grover, S., Jackiw, N., & Lundh, P. (2019). Concepts before coding: non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education*, 29(3) 1-30. helloworld.cc/nonprog

Examples of activities	Description
Story Variables	Introducing the idea of changing quantities in the real world and giving them meaningful names
Cats and Ladders	Naming variables and creating expressions in a non-programming context
Three Switches	Turning light switches on and off using Booleans, Boolean operators, and abstractions
Alarm Clock	Modelling real-world alarm clock situations using variables, arithmetic, logical expressions, and abstractions

THE CODE'S NOT ALL RIGHT

How do you decide when to teach a concept the right way, and when to simplify it?

It's a pretty common classroom task — you have some data of various types, and you want to teach students how to print the data out as part of a sentence, using Python. Here's my example data:

```
number = 1
person = "Scott"
```

The exam board wants students to know how to concatenate, so you might choose to teach the + operator, although it's a bit of a pain that the students will also have to put the spaces in the right place and remember to cast the integer to a string:

```
print("Thunderbird " +
      str(number) + " is piloted by "
      + person)
```

Alternatively, you might teach your students to separate the different parts with commas to avoid the casting issue. You either don't realise, or choose to disregard, that this isn't actually concatenation: it's taking advantage of the way Python's `print()` function works:

```
print("Thunderbird", number,
      "is piloted by", person)
```

You could also choose to avoid the casting issue in a different way by teaching `format()`, but will this confuse the students?

```
print("Thunderbird {} is
      piloted by {}".format(number,
                             person))
```

There are probably multiple other ways to achieve the same goal, each with their own benefits and drawbacks. Some people will argue vehemently for one or another (there's a long-running debate about `format()` within the Raspberry Pi Education Team!), but which one is right?

Building a mental model

When teaching beginners, there is a conflict to resolve: should you teach the way a more experienced programmer might approach a task, or teach the learner to use a suboptimal method, which either produces quick results or allows them to better access and build a mental model of what the program is doing?

When I was a teacher, I frequently shared resources online, to help others and save my fellow teachers valuable time. Occasionally I would receive feedback on my resources from IT professionals — some was extremely helpful, and some was less so. The helpful feedback felt collaborative and helped improve the outcome for children. These professionals helped me to improve my subject knowledge by providing a friendly suggestion of an alternative way

to approach a problem, where I hadn't realised that a different approach existed. This was extremely welcome, and I am indebted to people such as David Whale, Martin O'Hanlon, and Andy Stanford-Clark for their generous mentoring.

I would also sometimes receive feedback on my resources from IT professionals which highlighted occasions when I had deliberately made a decision to teach a concept in a suboptimal way, for a pedagogical reason. These developers pointed out so-called flaws and suggested what they thought were better ways of doing things, but they often forgot that, just because they are good at programming, that doesn't necessarily mean that they are good at teaching programming. I've encountered developers who expect eight-year-olds to have mastery of concepts that are not introduced until A



■ What thought processes do learners go through to create a simple game?

level, and who forget that much of what they know is not basic, obvious, or picked up by telling a child once.

No shortcuts to failure

It is sometimes difficult for a teacher to take off their experience goggles and notice when they are cutting short a learner's thinking process. Here's a recent example from when I was writing the Raspberry Pi resource SLUG! (helloworld.cc/slug). The project is a clone of the classic Snake game, in which the player moves the slug around an 8 × 8

- Whether the slug was moving into a coordinate containing a vegetable (to eat it)

Both of these would require a record of the vegetables. I was neatly sidestepping a problem before it had happened, and providing a more efficient approach. However, the learner's goal, at that point, was simply to see the vegetables appear. Sometimes, you purposely want to allow your learners to fall into holes and write inefficient code, so that they understand why they have to improve it!

SOMETIMES, YOU WANT TO ALLOW YOUR LEARNERS TO FALL INTO HOLES AND WRITE INEFFICIENT CODE

LED matrix, and red vegetables appear at random locations for the slug to eat.

The algorithm I gave the learners to implement, to randomly create vegetables and display them on the LED matrix, was as follows:

- Pick an x, y random coordinate on the LED matrix
- Check whether this coordinate is currently inhabited by the slug
- If it is, repeat steps one and two until you pick a location that is outside the slug
- Draw the vegetable on the LED matrix
- Store the vegetable's coordinate in a list of vegetables

When my colleague tested the resource, they asked, "Why are you storing the vegetable's coordinates in a list?" I realised I had imposed my own shortcut, drawn from my experience, on the learners without intending to, and without realising that I had done so.

I knew from experience that later on in the program, you would need to know:

1. How many vegetables existed on the screen (to avoid having too many)

Which approach is right?

Perhaps disappointingly, I don't have the silver bullet answer. It depends entirely on your learners — and that's why you're a teacher, and your job is safe from being taken over by robots! In my opinion, you should decide how to teach concepts in the way that provides the greatest benefit to the learners you have. Are you using the turtle module with some young kids who struggle with typing? Go ahead and use `from turtle import *` so there's less for them to type. Perhaps your GCSE class is particularly good this year, so why not throw in some dictionaries and make a GUI? Is there an Oxbridge hopeful in your A-level class? Show them several different approaches, and have a conversation about the benefits and drawbacks of each.

It's usually at this point that someone says, "I can't believe you're condoning teaching students things that are **wrong** — you're a terrible teacher!" To be totally clear, I am not advising you to deliberately teach bad practice (this is not a free pass to use `break!`). Instead, I am saying that it's sometimes acceptable to simplify concepts which introduce inefficiency. It's true that if you teach a programming concept in a simplified way so that a learner can understand it, you may later find it hard to persuade the learner to

HONEY, I SHRUNK THE CODE

You have a list of numbers represented as strings. You want to convert them all to integers.

```
numbers = ["23", "534",
           "52", "98", "87897"]
```

One method a student might use to achieve this would be to iterate through the list of numbers, converting each to an integer in turn and adding the result to a new list.

```
method1 = []
for n in numbers:
    method1.append(int(n))
```

However, an experienced developer might use a more efficient list comprehension:

```
method2 = [int(n) for n
            in numbers]
```

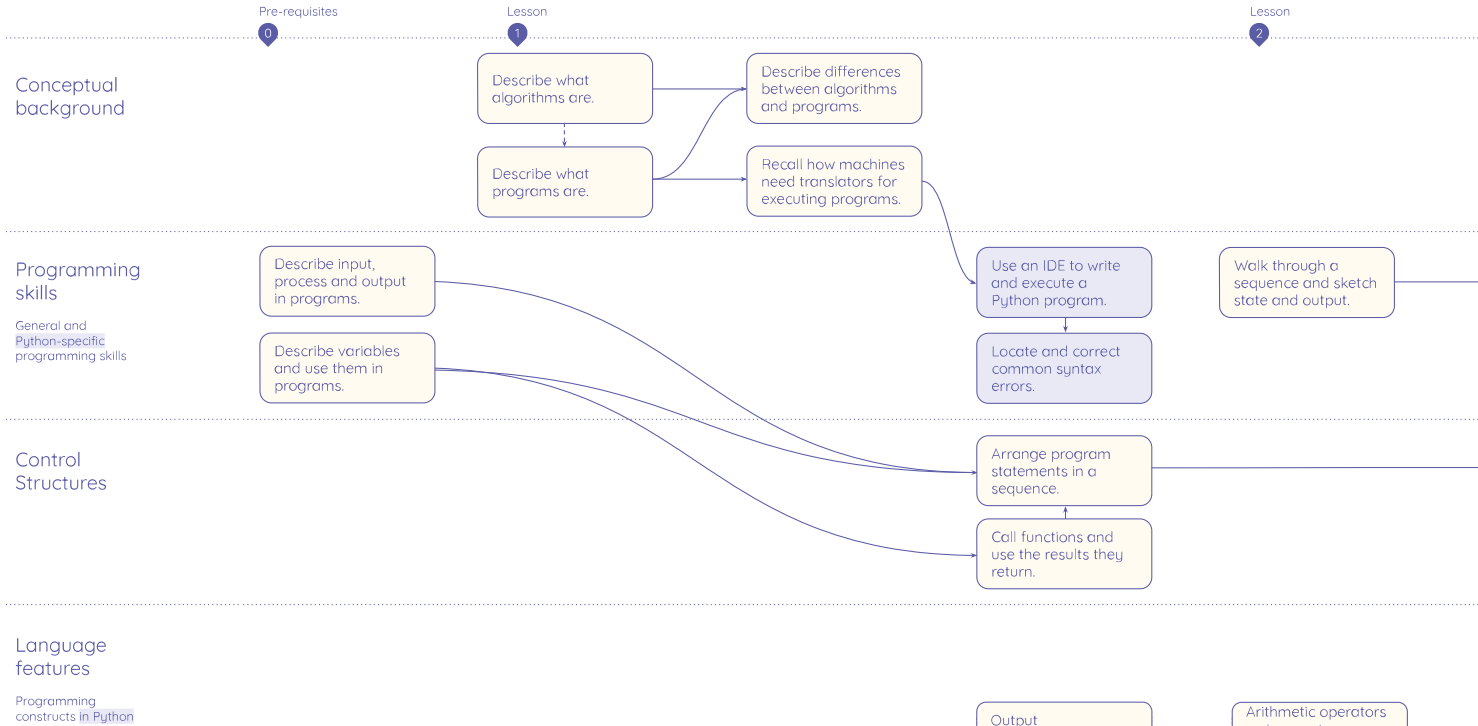
Which method is the best one to teach your learners?

abandon their safety blanket and adopt a better programming practice. However, some students will never make it to later if you made the starting hurdle too high for them to begin with. Which is best? Only you can decide. [\(HW\)](#)



LAURA SACH

Laura is a former head of department and has many years' experience teaching computer science in the classroom. She now leads the A level content team for Isaac Computer Science at the Raspberry Pi Foundation ([@codeboom](https://twitter.com/codeboom)).



LEARNING GRAPHS: TOOLS TO PLAN FOR PROGRESSION

Carrie Anne Philbin outlines an approach to mapping the computing curriculum



CARRIE ANNE PHILBIN

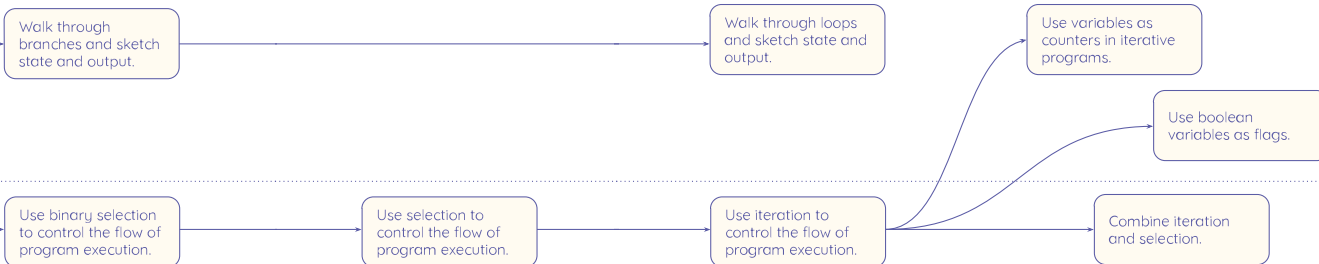
Carrie Anne is director of educator support at the Raspberry Pi Foundation, leading the development of resources to teach computing.

A progression framework is the backbone of any subject curriculum. It illustrates the sequence in which pupils learn, noting where they establish a core understanding of a topic in order to progress. As part of the National Centre for Computing Education, my team developed a bank of teaching resources that deliver the computing curriculum in England. In order to do this successfully, we studied a lot of progression frameworks, examination specifications, and even some research papers. We found that there are two quite different ways of presenting progression that show what should be taught and when it should be taught, as well as information on how or why concepts should be taught.

The first approach is to create a categorisation of skills and concepts in a list or table. Sequencing is shown by having objectives listed by Key Stage (learning stages in England), year group, or the age of the learners. Examples of this approach include the Computing at School computing progression pathways (helloworld.cc/path), and the Massachusetts Digital Literacy and Computer Science Curriculum Framework (accessible from helloworld.cc/standards). They are essentially lists of required knowledge, bundled by theme.

Another approach is to use a map of possible trajectories through learning waypoints — key building blocks of learning — and how they connect to each other. This approach highlights

Lesson 3 Lesson 4 Lesson 5 Lesson 6



This resource is available online at nccpe.io/prod-1a. Resources are updated regularly — please check that you are using the latest version. This resource is licensed under the Open Government Licence, version 3. For more information on this licence, see nccpe.io/oag

■ In this learning graph for Python programming with Year 8 pupils, solid arrows show prerequisites while dotted arrows show a suggested order of learning

where prerequisite knowledge needs to be mastered before students can move on, as well as the dependent knowledge contained in other nodes (represented by the boxes in the graphs; see image above), each containing one part of the computing curriculum that needs to be mastered in order to progress. Cambridge Mathematics ([cambridgемaths.org](https://www.cambridgемaths.org)) is leading the way in “developing a flexible and interconnected digital framework to help reimagine mathematics education 3–19”. We’ve been lucky enough to learn from their work, which has helped us to create learning graphs.

A tool for teachers

The learning graphs organise computing content — concepts, knowledge, skills, and objectives — into interconnected networks. We found that nodes often form clusters corresponding to specific themes, and we could connect them if they represent two adjacent waypoints in the learning process. Depending on the context, the

LEARNING GRAPHS SHOW OPPORTUNITIES TO ASSESS LEARNERS’ UNDERSTANDING AT LANDMARK POINTS IN A LESSON OR UNIT

nodes in a learning graph could contain anything, from the contents of a curriculum strand across an entire Key Stage, to the learning objectives of a six-lesson unit. When our team started working on a unit, the learning graphs were in a fluid state: the team would first uncover the structure of the content and the possible journeys through it, without being bound to a specific teaching pathway. The graphs would eventually reach a fixed state, where the nodes were further structured and arranged to reflect our suggestions on the order in which the content could actually be delivered.


We believe that learning graphs can be useful to teachers on a whole new level.

They directly inform lesson planning, but they also add value by showing opportunities to assess understanding at landmark points in a lesson or unit. By checking that students are grasping the concepts, teachers are able to think more about how they are teaching. They can revisit knowledge that perhaps didn’t land with learners the first time.

All progression frameworks are subjective, and with little research into computing education, we rely on teachers’ experience of combining the **what** we teach and **how** to teach it, to help inform this work. If you’ve not taken a look at our learning graphs, you can access them via helloworld.cc/tcc. [\(HW\)](#)



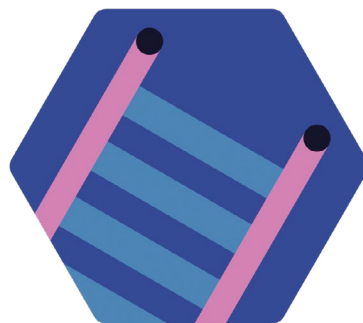
STRUCTURE LESSONS

- 20** COGNITIVE LOAD THEORY
 - 22** THE PRIMM APPROACH
 - 25** UNIVERSAL DESIGN FOR
LEARNING IN COMPUTING
 - 28** CODING AND 21ST-CENTURY SKILLS
 - 30** COULD CURRICULUM DESIGN
BE AS SIMPLE AS ABC?
- 

As in any other subject, computing lessons benefit from structure, planning, and a well-thought-out learning journey. There are several frameworks that educators can use to help them structure their computing lessons. For instance, before asking students to create something new, you might ask them first to use an existing example and modify it before moving on to creating their own. This is the Use–Modify–Create framework, and it can be really useful in helping learners to progress from working with examples created by experts to building something that they understand and own themselves.

If we think specifically about programming, a well-evidenced and popular framework that has been developed over the last few years is PRIMM. This acronym stands for Predict–Run–Investigate–Modify–Make, representing the different stages of a lesson or series of lessons.

Structuring lessons using such frameworks ensures that differentiation can be built in at various stages of the lesson. It also helps to reduce cognitive load and ensures that students are supported, engaged, and challenged at the right moments.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:** cognitive load theory
- **What the research says:** using PRIMM
- **What the research says:** the Universal Design for Learning framework in computing
- **Coding and 21st-century skills:** a framework
- **The ABC curriculum design process**

COGNITIVE LOAD THEORY

Thoughtful instructional design and structure can help avoid overloading working memory and maximise learning experiences in computing

Cognitive load theory (CLT) is a learning theory concerned with the limits of our working memory. There are many instructional methods that educators can use to avoid overloading working memory to maximise learning experiences. Computing educators should consider how they can structure lessons and present materials in light of this theory.

A model of cognition

CLT builds on the idea that human memory has two distinct areas: short-term working memory and long-term memory. While our long-term memory can be seen as essentially infinite, our working memory is extremely limited, with different studies suggesting a processing capacity of between three and nine pieces of novel information.¹ This capacity can easily become overloaded, impacting on our

ability to process the information we're presented with.

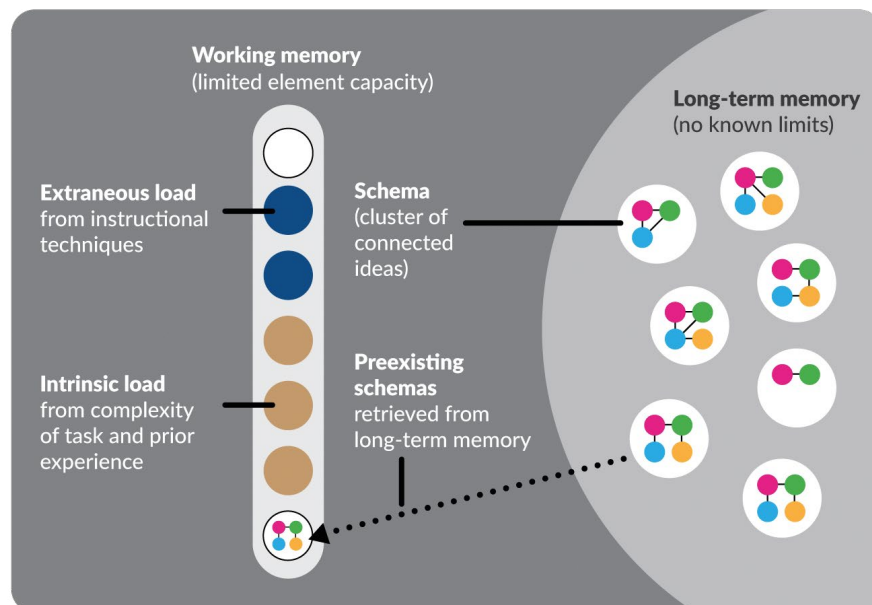
As we learn from our experiences, new information is stored in our long-term memory for future recall. Over time, these disparate elements of information are connected with existing understanding into collections of related knowledge, or schemas. The goal of effective learning design should be to facilitate the movement of new ideas and information from working memory into conceptually sound schemas.

Balancing the load

Sweller's^{1,2} research suggests that there are two key stresses, or cognitive loads, acting on the learner during a learning episode: intrinsic load and extraneous load. The intrinsic load placed on the learner relates to the complexity (number of and interactivity

of elements) of the concept(s) or skill(s) being taught, the gap between the new learning and their existing understanding, and any misconceptions they already hold. Educators should take steps to optimise intrinsic load wherever possible.

The manner in which new concepts are presented, explored, and applied can lead to an unnecessary extraneous load being placed on the learner. Having to juggle too much new information, or unnecessary information, from multiple sources can place an increased load on the learner. Through considered instructional design, educators can minimise the extraneous load in their activities.



■ Educators should consider the cognitive, intrinsic, and extraneous load put on learners when they introduce and present new concepts

“ OUR WORKING MEMORY IS EXTREMELY LIMITED

Managing intrinsic load

As outlined, intrinsic load stems from the gap between the learner's existing understanding and the new knowledge, and the complexity of the new concept or skill being taught. In seeking to reduce the load placed on learners, we need to reduce this gap. Educators should ensure that they are aware of prerequisite knowledge and learners' existing understanding. When planning the progression between concepts, it can be helpful to use learning graphs, which map connections and dependencies between concepts (see page 16).

Break the learning up into smaller tasks, or even individual elements. After being taught

EFFECT	IMPLICATIONS FOR COMPUTING EDUCATORS
<p>Split attention effect Learners must combine information from multiple sources, which increases cognitive load</p>	<ul style="list-style-type: none"> Combine diagrams with labels and related explanations Annotate programs using comments, in particular identifying common sections or patterns, known as sub-goals
<p>Redundancy effect Learners must process and disregard repeated or unnecessary information, which increases cognitive load</p>	<ul style="list-style-type: none"> Avoid redundant information in diagrams and explanations Use accessible language Minimise 'boilerplate code' (having to write a lot of code to accomplish only minor functionality)
<p>Transient information effect Information that doesn't persist must be stored in working memory, which increases cognitive load</p>	<ul style="list-style-type: none"> Provide learners with programming cheat sheets or reference guides Share or create concept maps⁴ with learners, and highlight concepts and their relationships, to provide scaffolding and reference material
<p>Multimodal effect Visual and oral information are processed separately, which reduces cognitive load</p>	<ul style="list-style-type: none"> Combine static images, animations, and oral presentation to spread the load When modelling a process, narrate or prompt self-explanation of the thought process to make it visible to learners
<p>Worked example effect Worked examples provide learners with scaffolding and support to develop generalised solutions, which reduces cognitive load</p>	<ul style="list-style-type: none"> Use partially or fully worked examples to provide possible solutions to problems, e.g. programming tasks, binary/denary conversions, compression algorithms, etc. Use worked examples to model problem-solving processes, including specifying, decomposing, prototyping, and testing
<p>Collective working memory effect Task elements are shared between a group, which reduces cognitive load</p>	<ul style="list-style-type: none"> Use techniques such as pair programming to share work between learners and thereby spread the load Poor communication between learners can add a cognitive load cost, which could eliminate the benefits of this effect

in isolation, these elements can be revisited, and connections made between them. This is useful for highly complex concepts, but a balance is needed in the classroom, where teaching time is constrained and breaking down too far could result in a lengthy and disjointed learning sequence.³

Optimising instructional design

To manage the extraneous load placed on learners, consider how you present your materials. There are a number of observable effects in CLT that positively or negatively affect the cognitive load that learners experience. See the table above for some effects that are relevant to the teaching of computing, and how you can both employ and avoid them. ^(HW)

REFERENCES

- Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive Architecture and Instructional Design: 20 Years Later. *Educational Psychology Review*, 31(2), 261-292. helloworld.cc/cognitive1
- Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, 12, 257-285. helloworld.cc/cognitive2
- Reif, F. (2008). *Applying Cognitive Science To Education: Thinking And Learning In Scientific And Other Complex Domains*. MIT Press. helloworld.cc/cognitive3
- Mühling, A. (2016). Aggregating concept map data to investigate the knowledge of beginning CS students. *Computer Science Education*, 26(3), 176-191. helloworld.cc/cognitive4

SUMMARY

Human memory

- Working memory is extremely limited, with capacity for as few as three to nine new ideas at once
- Long-term memory has no known limits
- Learning occurs when new knowledge is transferred from working memory to long-term memory
- Schemas are structured collections of prior learning that can be recalled from long-term memory
- Schemas only occupy a single space in working memory

Cognitive load theory

- Cognitive load** is a stress on a learner's working memory, reducing their ability to acquire new learning
- Intrinsic load** relates to the complexity of the learning task and the learner's existing understanding
- Extraneous load** is any additional stress placed on the learner due to the way in which the material is presented

Managing intrinsic load

- Awareness of learners' prior experience and understanding helps to predict where cognitive overload may occur
- Breaking down the learning into suitable learning episodes can help manage cognitive load

Implications for instruction

- Combine text and graphics when presenting, and remove irrelevant detail
- Present information both visually and orally, as appropriate, without adding additional load
- Use worked examples to provide scaffolding for novices
- Use collaborative techniques such as pair programming, which distribute the cognitive load among learners

THE PRIMM APPROACH

A scaffolded approach to structuring programming lessons, PRIMM encourages students to read code before they write it

P PRIMM is an approach that can help teachers structure lessons in programming. PRIMM stands for Predict–Run–Investigate–Modify–Make, representing the different stages of a lesson or series of lessons. PRIMM promotes discussion between learners about how programs work, and the use of starter programs to encourage the reading of code before writing it.

The five stages of PRIMM

Educators work their way through the following stages when using the PRIMM approach:

Predict: Students discuss a starter program and predict what it might do. They can draw or write out what they think will be

the output. At this level, the focus is on the function of the code.

Run: Students run the provided starter program so that they can test their prediction and discuss it with their peers.

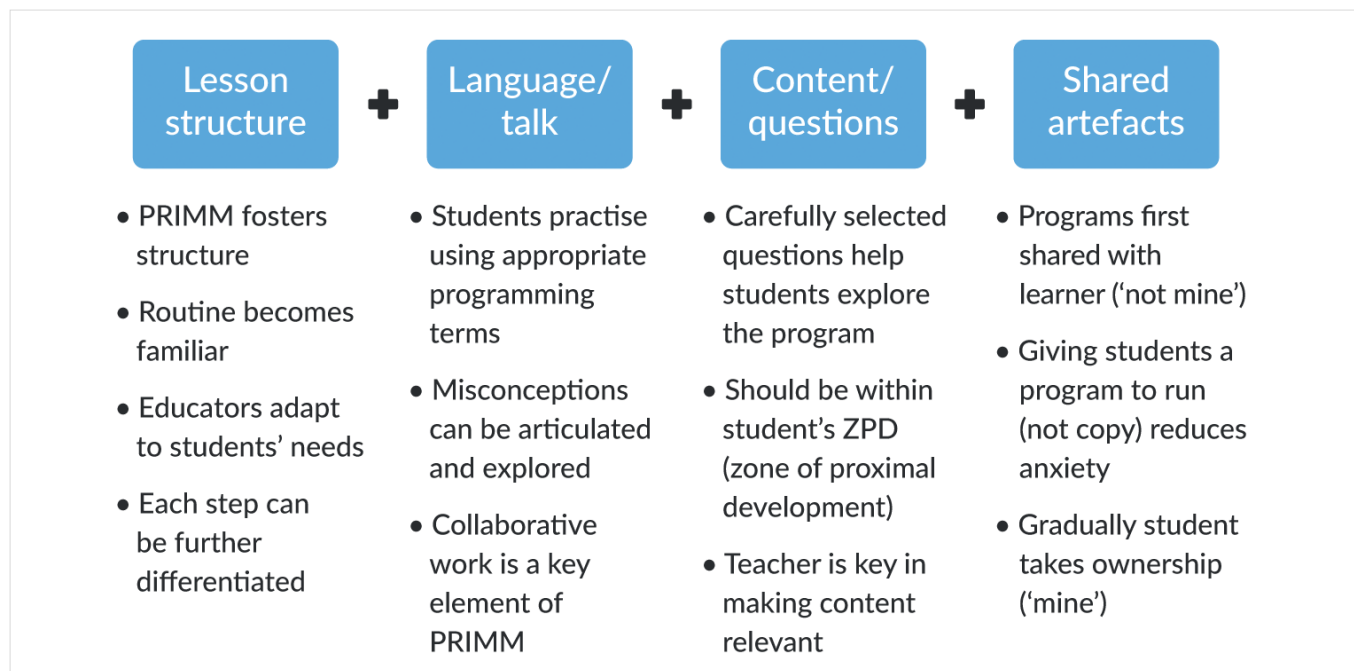
Investigate: The teacher provides a range of activities to explore the structure of the code. This could include tracing, explaining, annotating, and debugging.

Modify: Students edit the program to change its functionality via a sequence of increasingly challenging exercises. The transfer of ownership moves from the code being 'not mine' to 'partly mine' as students gain confidence by extending the function of the code.

Make: Students design a new program that uses the same structures, but solves a new problem (that is, has a new function).

You may not be able to go through all the stages in one lesson, and you may focus on one stage more than another. Using the PRIMM framework gives you a way of labelling what you are doing when you are teaching programming.

The PRIMM approach builds and draws on other research in computing education, including Use-Modify-Create¹, tracing and reading code before writing², the Abstraction Transition Taxonomy³, and the Block Model⁴. The focus on language and talk, and the use of starter programs, draw on sociocultural perspectives on how children learn programming.



■ Planning a lesson using PRIMM



■ PRIMM encourages students to work together and discuss their problem-solving

Credit: stock.adobe.com/kegfire

Encouraging talk in the classroom

Classroom discussion is an important aspect of the teaching of many subjects, but it isn't often referred to when it comes to the teaching of programming. Many PRIMM activities are carried out in pairs, and we already know that pair programming is an effective form of learning, and involves learners practising articulating what to do when writing a program. PRIMM goes a step further and encourages Predict and Investigate activities to be carried out in pairs or small groups, away from the computer. This has the following benefits:

- Talking about a program and how it works helps learners to find the right terminology to use to articulate their understanding.

 **Predict**

 **Run**

 **Investigate**

 **Modify**

 **Make**

■ The five stages of the PRIMM approach to structuring programming lessons

“ PRIMM BUILDS AND DRAWS ON DECADES OF RESEARCH

Having a common language to talk about programming constructs is important

- Actually verbalising out loud the steps of a program that are difficult to understand can help learners to focus on smaller elements at a time
- Through dialogue with others, we can ask and answer questions, and learn from others

Read before you write

The first activity in a PRIMM-like lesson involves predicting what a small segment of code will do when it is run. It doesn't require stating how it will do that, just the outcome. This shouldn't be an assessed exercise, so that all children are encouraged to have a go, and it's important that it is low stakes. Sometimes the output can be drawn, and sometimes the teacher will provide some sample inputs, depending on what kind of code it is.

SUMMARY

PRIMM is a way of structuring programming lessons that focuses on:

- Reading code before you write it
- Working collaboratively to talk about programs
- Reducing cognitive load by unpacking and understanding what code is doing
- Gradually taking ownership of programs when ready

The five stages of PRIMM are:

Predict

- Focus on the function of the code
- Encourage discussion
- Work in pairs or threes

Run

- Provide students with working code to run
- Learners check against predictions

Investigate

- Use a variety of activities, such as tracing, annotating, and questioning
- Encourage students to discuss and work with the code in pairs or small groups

Modify

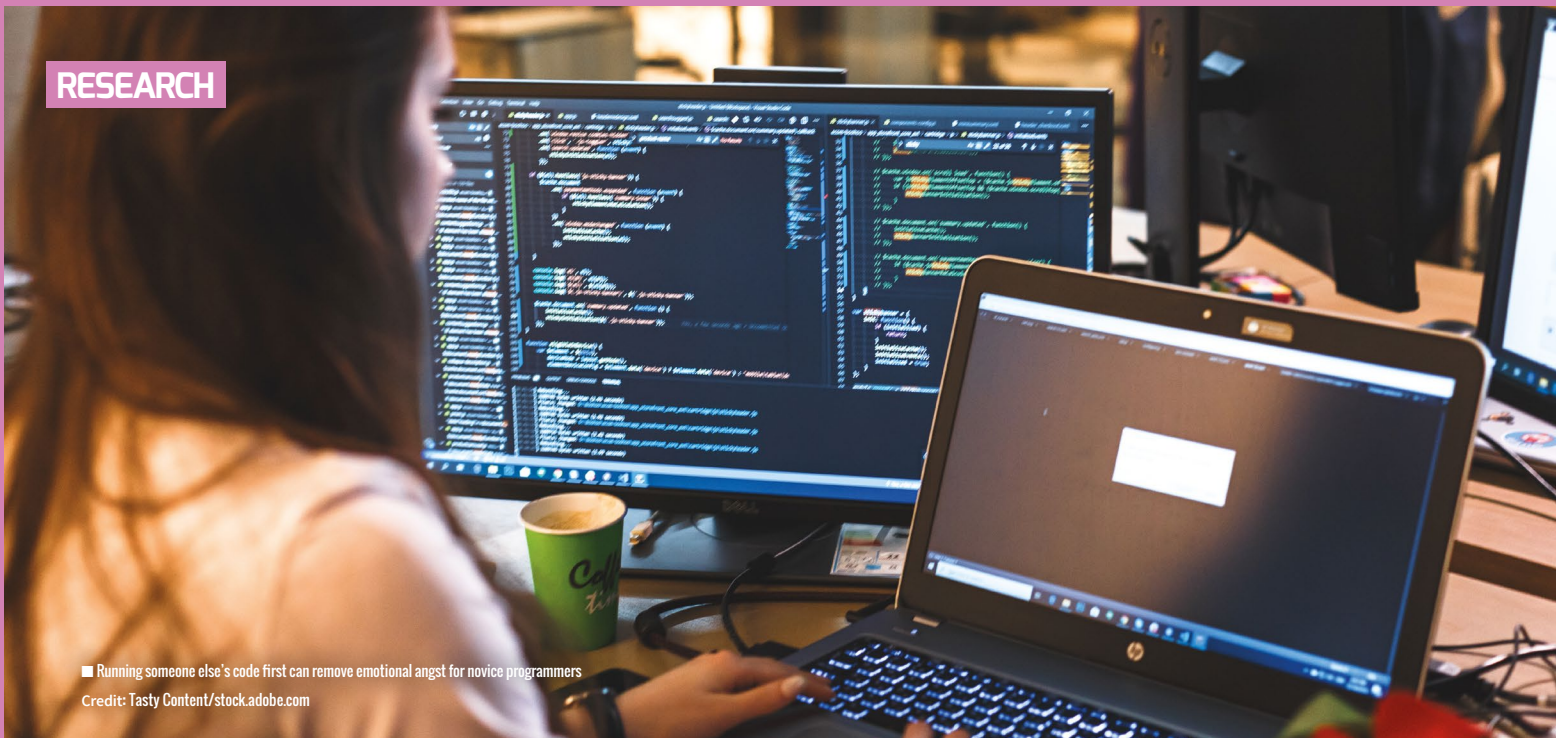
- Modify code in small steps to add new functionality
- Apply what has been learnt about the structure of the code
- Gradual increase in difficulty

Make

- Create a new program
- Practise the programming skills that have been learnt
- Can be a design task or an open task

Does it work?

- A study in 2018 with 500 learners aged 11-14 showed improved learning outcomes after 8-12 weeks of programming lessons using PRIMM⁶
- PRIMM has been put into practice by many teachers in primary and secondary schools around the world



■ Running someone else's code first can remove emotional angst for novice programmers
Credit: Tasty Content/stock.adobe.com

REFERENCES

- ¹ Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*. 29(2-3), 136-176 helloworld.cc/primm1
- ² Lee, I. et al. (2011). Computational thinking for youth in practice. *ACM Inroads*. 2(1), 32-37. helloworld.cc/primmref2
- ³ Lister, R. et al. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*. 36(4), 119-150. helloworld.cc/primm3
- ⁴ Cutts, Q. et al. (2012). The abstraction transition taxonomy: developing desired learning outcomes through the lens of situated cognition. *Proceedings of the Ninth Annual International Conference on International Computing Education Research*. New York, ACM. 63-70. helloworld.cc/primm4
- ⁵ Schulte, C. (2008). Block Model: An Educational Model of Program Comprehension as a Tool for a Scholarly Approach to Teaching. *Proceedings of the Fourth International Workshop on Computing Education Research*. New York, ACM. 149-160. helloworld.cc/primm5
- ⁶ Venables, A., Tan, G. & Lister, R. (2009). A closer look at tracing, explaining and code writing skills in the novice programmer. *Proceedings of the Fifth International Workshop on Computing Education Research*. New York, ACM. 117-128. helloworld.cc/primm6
- ⁷ Walqui, A. (2006). Scaffolding instruction for English language learners: A conceptual framework. *International Journal of Bilingual Education and Bilingualism*. 9(2), 159-180. helloworld.cc/primm7

▶ This aspect of PRIMM builds on decades of research that has shown that reading code before writing it is an effective way to learn programming. For example, work by Lister and colleagues over many years highlighted the importance of reading code and being able to trace what it does before writing new code. Comparing tracing skills to code writing, they demonstrated that novices require a 50 percent tracing code accuracy before they can independently write code with confidence.⁶

Not starting from scratch

It can be very stressful for novice programmers to write code into a blank editor window. The syntax needs to be right, or quite intimidating error messages can appear. It's easy to be put off having a go, or for teachers to resort to getting students to copy code that they don't yet understand. By running a program that the teacher has written, the learner doesn't have ownership of that starter program and

```
from turtle import *
def square ():
    for counter in range (4):
        forward(100)
        right(90)

square()
left(45)
square()
```

■ Reading code before writing it is a key element of PRIMM

does not experience the emotional angst if it doesn't work. That's why in PRIMM, the Run stage involves running a program provided on a shared drive to check the prediction. Gradually, once the student has some understanding of how the code works, they can modify the code and take ownership of the new functionality.

Drawing on sociocultural theory

Social constructivism, in particular the work of the psychologist Vygotsky, can frame our understanding of novice programmers and their learning. This interpretation of the learning process can help us to develop effective pedagogical strategies. Vygotsky proposed that mediated activity promotes higher mental processes, and identified three major forms of mediation: material tools, psychological tools (including language), and interaction with other human beings. Mediation allows learners to act as apprentices before internalising new ideas, and sociocultural theory suggests that movement from the social plane to the cognitive plane supports the learning of skills and knowledge⁷. With the PRIMM approach, the starter programs that are shared and discussed can be seen as being on the social plane, with a mediated progression to the cognitive plane once they are understood and internalised.⁵

If you want to delve deeper into PRIMM, see page 84 for more about the Investigate stage. (HW)

UNIVERSAL DESIGN FOR LEARNING IN COMPUTING

STORY BY Hayley Leonard

Universal Design for Learning (UDL) is a framework for considering how tools and resources can be used to reduce barriers and support all learners. Based on findings from neuroscience, it has been developed over the last 30 years by CAST, a nonprofit education research and development organisation based in the USA. UDL is currently used across the globe, with research showing that it can be an efficient approach for designing flexible learning environments and accessible content.

Engaging a wider range of learners is an important issue in computing, which is often not chosen as an optional subject by girls or those from some minority ethnic groups. Researchers at

the Creative Technology Research Lab (CTRL) in the USA have been investigating how UDL principles can be applied to computer science, to improve learning and engagement for all students. They have adapted the UDL guidelines to a computing education context and begun to explore how teachers use the framework in their own practice. The hope is that understanding and adapting how the subject is taught could help to increase the representation of all groups in computing.

A scientific approach

The UDL framework is based on neuroscientific evidence that highlights how different areas or networks in the brain work together to process information during

learning. Importantly, there is variation across individuals in how each of these networks functions and interacts with each other. This means that a traditional approach to teaching, in which a main task is differentiated for students with special educational needs, may miss out on the variation in learning between all students across different tasks.

The guidelines highlight opportunities to consider learner differences when planning lessons. The framework is structured according to three main principles, which are directly related to three networks in the brain that play a central role in learning. It encourages educators to plan multiple, flexible methods of *engagement* in learning (affective networks), *representation* of the teaching materials (recognition networks), and opportunities for *action and expression* of what has been learnt (strategic networks).

The three principles of UDL are each expanded into guidelines and checkpoints that allow educators to identify the different methods of engagement, representation, and expression to be used in a particular lesson. Each principle is also broken down into activities that allow learners to *access* the learning goals, remain engaged and *build* on their learning, and begin to *internalise* the approaches to learning, so that they are empowered for the future. ▶



Credit: Andrew Ebrahim/unsplash



Credit: CDC/unsplash

EXAMPLES OF UDL GUIDELINES FOR COMPUTER SCIENCE EDUCATION FROM THE CREATIVE TECHNOLOGY RESEARCH LAB

Multiple means of engagement	Multiple means of representation	Multiple means of action and expression
<p>Provide options for recruiting interests</p> <p>Give students choice (software, project, topic)</p> <p>Allow students to make projects relevant to their culture and age</p>	<p>Provide options for perception</p> <p>Model computing through physical representations as well as interactive whiteboard/videos, etc.</p> <p>Select coding apps and websites that allow adjustment of visual settings (e.g. font size/contrast) and that are compatible with screen readers</p>	<p>Provide options for physical action</p> <p>Include CS unplugged activities that show physical relationships of abstract computing concepts</p> <p>Use assistive technology, including a larger or smaller mouse or touchscreen devices</p>
<p>Provide options for sustaining effort and persistence</p> <p>Utilise pair programming and group work with clearly defined roles</p> <p>Discuss the integral role of perseverance and problem-solving in computer science</p>	<p>Provide options for language, mathematical expressions, and symbols</p> <p>Teach and review computing vocabulary (e.g. code, animations, algorithms)</p> <p>Provide reference sheets with images of blocks, or with common syntax when using text</p>	<p>Provide options for expression and communication</p> <p>Provide sentence starters or checklists for communicating in order to collaborate, give feedback, and explain work</p> <p>Provide options that include starter code</p>
<p>Provide options for self-regulation</p> <p>Break up coding activities with opportunities for reflection, such as turn and talk, or written questions</p> <p>Model different strategies for dealing with frustration appropriately</p>	<p>Provide options for comprehension</p> <p>Encourage students to ask questions as comprehension checkpoints</p> <p>Use relevant analogies and make cross-curricular connections explicit</p>	<p>Provide options for executive function</p> <p>Embed prompts to stop and plan, test, or debug throughout a lesson or project</p> <p>Demonstrate debugging with think-alouds</p>

Each principle of the UDL framework is associated with three areas of activity that may be considered when planning lessons or units of work. It will not be the case that each area of activity should be covered in every lesson, and some may prove more important in particular contexts than others. The full table and explanation can be found at *Israel, M., Lash, T., & Jeong, G. (2017). Utilizing the Universal Design for Learning Framework in K-12 Computer Science Education. Project TACTIC: Teaching All Computational Thinking Through Inclusion and Collaboration. Retrieved from University of Illinois, Creative Technology Research Lab website: helloworld.cc/ctrl.*



Credit: Design Cells/stock.adobe.com

▶ Applying UDL to computer science education

While an advantage of UDL is that the principles can be applied across different subjects, it is important to think carefully about what activities to address these principles could look like in the case of computer science.

Researchers at CTRL, led by Maya Israel, have identified key activities, some of which are presented in the table on the opposite page. These guidelines will help educators anticipate potential barriers to learning and plan activities that can overcome them, or adapt activities from those in existing schemes of work, to help engage the widest possible range of students in the lesson.

UDL in the classroom

As well as suggesting approaches to applying UDL to computer science education, the research team at CTRL has investigated how teachers are using UDL in practice. Israel and colleagues worked with four novice computer science teachers in US elementary schools to train them in the use of UDL and look at how they applied the framework in their teaching.

The research found that the teachers were most likely to include in their teaching multiple means of engagement, followed by multiple methods of representation. For example, they all offered choice in their students' activities and provided materials

“ THE GUIDELINES HELP EDUCATORS ANTICIPATE BARRIERS TO LEARNING AND PLAN ACTIVITIES TO OVERCOME THEM

in different formats (such as oral and visual presentations and demonstrations). They were less likely to provide multiple means of action and expression, and mainly addressed this principle through supporting students in planning work and checking their progress against their goals.

Although the study included only four teachers, it highlighted the flexibility of the UDL approach in catering for different needs within variable teaching contexts. More research will be needed in future, with larger samples, to understand how successful the approach is in helping a wide range of students to achieve good learning outcomes.

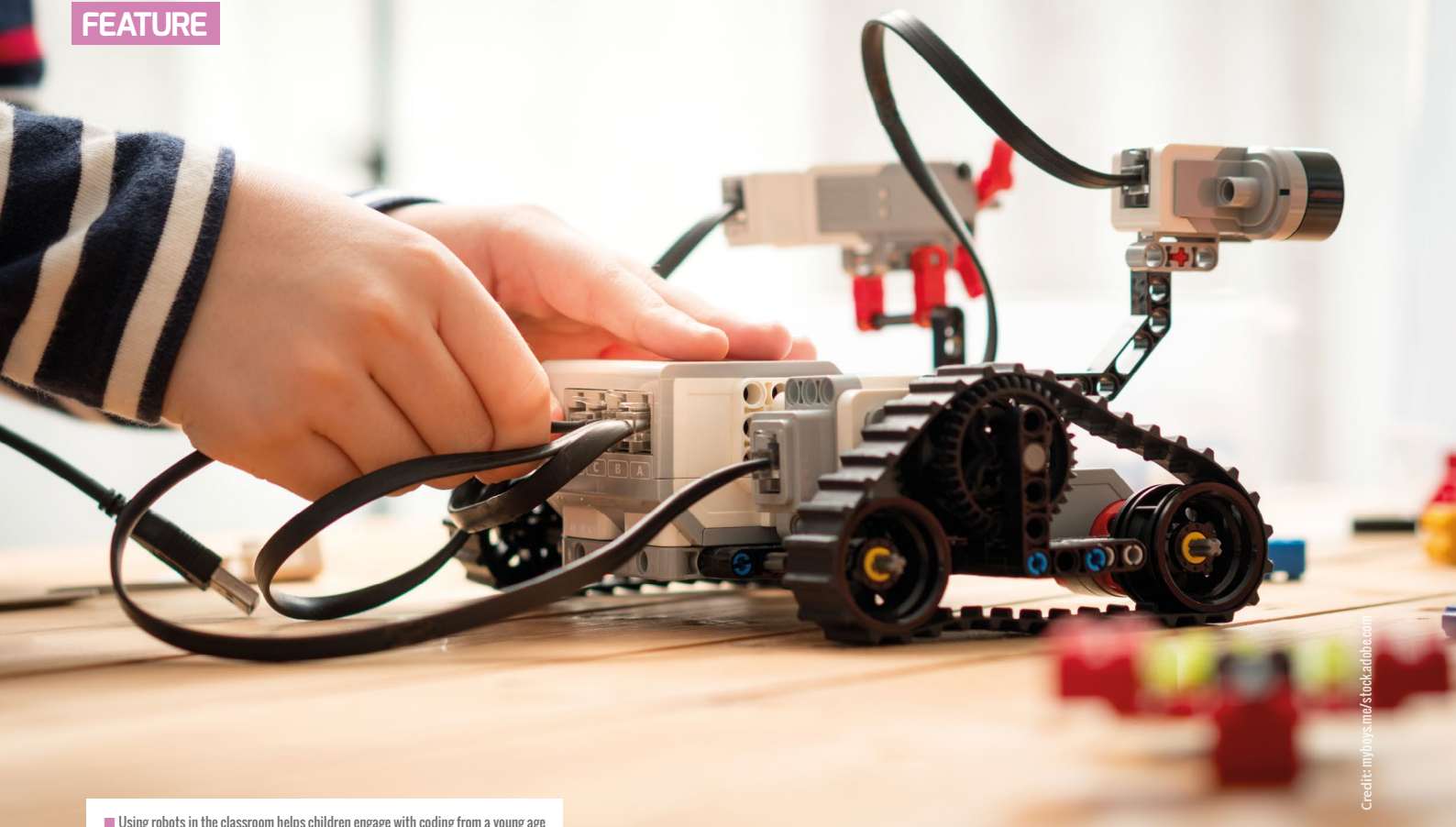
Find out more about using UDL

There are numerous resources designed to help teachers learn more about the UDL framework and how to apply it to teaching computing. The CAST website (helloworld.cc/cast) includes an explainer video and the detailed UDL guidelines. The Creative Technology Research Lab website has computing-specific ideas and lesson plans using UDL (helloworld.cc/ctrl).

Maya Israel has previously presented her research at one of the online Raspberry Pi Foundation computing education research seminars (helloworld.cc/mayaseminar). There are more seminars like this, which are free to attend and open to anyone from anywhere around the world. For more information about upcoming and previous series and to sign up to attend, please visit helloworld.cc/RPFseminars. (HW)

FURTHER READING

- ✓ Israel, M., Jeong, G., Ray, M., & Lash, T. (2020). Teaching Elementary Computer Science Through Universal Design for Learning. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 1220-1226. helloworld.cc/udl1
- ✓ Rose, D. H., & Strangman, N. (2007). Universal design for learning: Meeting the challenge of individual learning differences through a neurocognitive perspective. *Universal Access in the Information Society*, 5(4), 381-391. helloworld.cc/udl2



■ Using robots in the classroom helps children engage with coding from a young age

Credit: mjbays.me/stockadobe.com

CODING AND 21ST-CENTURY SKILLS

Ursula Martin shows how to include coding in a learning framework that equips students for an uncharted future

How should schools prepare students for lives in a digital society with a dizzying pace of change? Since the 1980s, educators, governments, charities, NGOs, and companies have been trying to answer this question. The 21st-century learning framework, created in the USA and applied in numerous countries, attempts to equip students with the skills they will need. It moves away from rote learning towards an engagement with higher-order skills.

The four Cs of the learning framework are often used by educators in the USA to guide their teaching. They are collaboration, communication, creativity, and critical thinking. As coding becomes integral to how students use technology in the classroom, integrating coding into the four Cs of the 21st-century learning framework takes education to the next

level and helps to prepare students for digital careers. This preparation cannot begin too early; it should begin in elementary school and continue to develop as students move on into high school.

Coding and critical thinking

Critical thinking is a skill that students need to develop in the classroom, to help them to dispel misconceptions about what they are learning. This also helps students to ask questions that will lead them to a better understanding of what they are learning. Using coding to teach critical-thinking skills helps students to solve the problems they identify. For example, let's say there is a problem with lights being left on in the classroom when the room is empty. Students could code an Arduino to turn the lights off if no motion is detected

in the room. This would require students to use their critical-thinking skills to determine how to solve this problem.

Coding and collaboration

Collaboration is simply working with others to accomplish a specific goal. This may not seem too hard, but it is a skill that many students lack, and one that they truly need, as they will eventually become a part of a society that functions best through collaborating with each other. Any student can learn to code, but having students code together opens a door to developing better collaboration skills, as well as enhancing their skills in other subject areas. For example, students could develop their maths skills by learning to code Dash and Dot robots to travel on a specific path. The path could be designed



Students can enhance their maths skills when learning code

by groups of students working together, using basic geometry that the students would have learnt in their maths classes. Encouraging students to collaborate to accomplish this task helps them to understand that everyone's thought process is not the same, and that different opinions can help to create a great project and provide a richer outcome than one person working alone.

previously learnt. For example, if students are tasked with coding a drone to fly, they must be able to work with others to communicate exactly what they would like their drone to do (fly, flip, and so on) and how long they want that action to take

“ INTEGRATING CODING INTO THE LEARNING FRAMEWORK TAKES EDUCATION TO THE NEXT LEVEL AND HELPS TO PREPARE STUDENTS FOR DIGITAL CAREERS

Coding and communication

Communication is key in education and in the workforce. Students must learn good communication skills to be effective in every aspect of their daily routine, whether in the classroom or in their working lives. Coding introduces a different type of communication ability to the learning process. As students learn to code, they learn to communicate by speaking and writing about what they want to create, or what they have

place for. This means using apps such as the educational programming platform Tynker, to figure out how to accomplish this task. In this way, students not only learn a new language, but they also learn how to communicate what they have learnt by having the drone perform the tasks they coded it to do. Communication is such an important skill for students to develop and effectively use, and acquiring coding knowledge can play a major part in learning this skill.

Coding and creativity

Creativity, as part of the learning process, allows students to learn by thinking outside the box. This helps students to take ownership of their learning in a non-traditional way. When coding is added to this part of the learning process, it introduces an opportunity for students to use different types of technology to create what they are learning about. For example, as students study biogeochemical cycles, they learn about the carbon cycle and how it interchanges with oxygen. A project on this could include building a small greenhouse, growing a plant in it, and coding a Raspberry Pi to open and close the windows of the greenhouse as the oxygen and carbon dioxide levels reach certain levels. This level of creativity would help students to understand just how these biogeochemical cycles work.

A different way to learn

Framing educational activities using the four Cs of 21st-century learning challenges students to learn in a non-traditional way. Adding coding to this process changes the dynamics of learning altogether. Giving students the opportunity to take ownership of their learning will help teachers to become facilitators and ensure that students better understand how learning works. (HW)



URSULA MARTIN

Ursula has been an educator for 18 years. During that time, she has taught biology, anatomy and physiology, and environmental science. She is a Raspberry Pi Certified Educator and a District Level Technology Resource Teacher in Mobile, Alabama, USA.

COULD CURRICULUM DESIGN BE AS SIMPLE AS ABC?

A curriculum design process known as ABC is helping classroom teachers review and plan more balanced online learning activities

Arena Blended Connected (ABC) curriculum design is a popular and well-respected process used in universities worldwide. Based on Professor Diana Laurillard’s research and designed and developed by University College London (UCL), the process guides educators to create or adapt sequences of learning modules to include online and blended elements. The process centres decisions around the type of learning that will take place rather than the technology that will be used.

In early 2020, the Computing at School (CAS) Research working group ran a series of training sessions to introduce teachers at primary and secondary schools to the process, as it had potential benefits to support the planning of learning during school closures. Working with the team at UCL, CAS Research developed training that presented the basics of ABC. A group of teachers then set to work trialling and adapting ABC to their schools’ needs.

ABC explained

ABC is based on the idea that there are six learning types (not to be confused with the now-debunked visual, auditory, and kinaesthetic learning styles): acquisition, investigation, discussion, collaboration, practice, and production. Educators review the type of learning that they would like to occur for a particular learning objective and design an activity to meet the need.

To help educators, the ABC method makes use of cards with suggestions of potential activities for each type. The potential activities are either a face-to-face or an online option. As the computing teachers who were taking part in the ABC training started using the standard ABC cards, they soon realised that they needed to create their own sets for their own situations. Rather than having just a face-to-face and an online option, the group decided to include three online options. These were low-tech, mid-tech, and high-tech: some teachers, and their students in

some schools, had the skills and experience to use quite complex software, whereas in other situations, a more low-tech option was needed. Two primary teachers and two secondary teachers created sets of context-specific cards. Since then, a set of cards has also been created for delivering professional development.

Uses and challenges

The computing teachers who trialled ABC found the cards were a great way to help them and their colleagues plan for remote teaching. The cards have been used for quick reviews of current planning and to help develop new activities.

ABC has been used both by individual teachers to review their lessons, and by groups of teachers. In one school, the cards have been very popular across the whole school, and in another context, they are being used to support professional development for teachers across many schools. Some of the teachers have also used the ABC process for reviewing other subjects, including English and maths. In several primary settings, both specialist and generalist teachers have started to use cards. Teachers have used them not only to work out remote activities, but also to help them flip between remote and in-school delivery.

When developing the cards, teachers audited the content to be delivered and the resources that were usually used. The process focused them on thinking about alternative pedagogies. Approaches such as flipped learning were encouraged, moving away from an emphasis on passive learning (acquisition) through watching videos. Some teachers also reported that the process

<p>Acquisition</p> <p>Learning through acquisition is about what learners are listening to: hearing a lecture or podcast, reading from books or websites, or watching demos or videos.</p>	<p>Collaboration</p> <p>Learning through collaboration embraces mainly discussion, practice, and production. Building on investigations and acquisition, it is about taking part in the process of knowledge building itself.</p>	<p>Discussion</p> <p>Discussion requires learners to articulate their ideas and questions, and to challenge and respond to the ideas and questions from the teacher and/or their peers.</p>
<p>Investigation</p> <p>Investigation encourages learners to take an active and exploratory approach to learning, to search for and evaluate a range of new information and ideas.</p>	<p>Practice</p> <p>Practice enables knowledge to be applied in context. The learner modifies actions according to the task, and uses feedback to improve. Feedback may come from self-reflection, peers or the teacher, or from activity outcomes.</p>	<p>Production</p> <p>Production is how the teacher motivates learners to consolidate what they have learnt by articulating their current conceptual understanding and reflecting on how they used it in practice.</p>

■ Unlike now-debunked learning styles, the learning types in ABC are not distinct for a specific activity. Source: UCL

AN EXAMPLE OF AN ABC CARD

Creator of card: Matthew Wimpenny-Smith		Date created: 19 May 2020		Context (e.g. phase, subject, school): Primary Key Stage 2	
Card name: Practice					
Current activity	Alternative activities				
	Low-tech	Mid-tech		High-tech	
Worksheet-based task is photocopied and handed out in class	Worksheet is converted to digital format and emailed to learner. Learner prints and completes worksheets, then photographs or scans and emails back.	Worksheet converted to editable file and then posted on Google Classroom as one copy per child, or on Seesaw app. Learner completes digitally via keyboard or digital inking.		Worksheet becomes Google Form or other web-based activity for the learner to complete, for example interactive Seesaw activity.	
	Prep time: quick Feedback time: slow	Prep time: some initially Feedback time: quick and all digital		Prep time: high initially Feedback time: could be very fast through self-marking	

linked nicely with other frameworks for integrating technology in the classroom, such as the SAMR (Substitution, Augmentation, Modification, and Redefinition) model and the TPACK (Technological Pedagogical Content Knowledge) model.

Teachers who trialled ABC noted that those schools who had already implemented a standardised approach to remote learning (such as using Google Classroom, Microsoft Teams, etc.) were at a great advantage. However, ABC can also help schools that are new to remote teaching. Sets of ABC cards have been created for different technology options, and these can be shared, giving less experienced teachers a menu of ideas to select from.

One of the main benefits of the ABC approach is that it allows for the development of remote teaching that isn't just centred around acquisition, production, and the recall of information and facts, but can expand into collaboration, investigation, and discussion. ABC particularly highlights where learning is passive, and also supports teachers to include assessment.

Embedding formative and summative assessment is a significant challenge in designing remote education. Often, quite high-tech options are needed to provide timely feedback to pupils. Learning platforms such as Google Classroom can be used for online marking. Rubrics can be set up and

shared with pupils, for self-assessment, questions asked in live lessons, and quizzes used for asynchronous learning. Teachers have found that ABC has helped them tackle the assessment challenge by pinpointing potential strategies and opportunities for introducing assessment.

Other challenges to introducing ABC have been not about the process itself, but more about the contexts for learners and schools. One such challenge has been the range of pupils' access to technology, while another involves safeguarding restrictions. Some schools found their pupils had to share devices with siblings or parents who were working from home, or that they only had access to a phone or tablet device in the evenings. The low-tech, mid-tech, and high-tech options on the cards helped teachers to begin to address these technical and timing issues.

However, to know what low-tech, mid-tech, and high-tech would look like in context, teachers needed to ask families what devices would be available to pupils, and when. It would therefore be useful, as part of the ABC process, to include a first step of auditing the devices available to your pupils. The audit needs to be done before you start your cards. Similarly, you also need to take into account the devices and platforms available to teachers, and their level of expertise.

Further challenges for the introduction of ABC are those of CPD and time. Teachers need training and time to enable them to develop these cards in preparation for the next period of remote education. In one school, senior management was concerned about how long the process would take, and the additional teacher workload. However, the teacher trialling the process is looking at how ABC can be integrated into PPA time once training has been delivered.


Taking it further and training in ABC

You can read about the original ABC process at helloworld.cc/abc; to find out how it has been adapted to our current computing school context, visit helloworld.cc/abc_cas. There, you can find a resource describing ABC, and sets of example cards, along with templates to create your own sets. A set of resources to share the ABC process with teachers is also available for CAS members at helloworld.cc/casinabox. Please contact Jane Waite (@janewaite) if you would like more information about using ABC in schools. ^(HW)

JANE WAITE, MATTHEW WIMPENNY-SMITH, CLAIRE BUCKLER, CALVIN ROBINSON & NIC HUGHES



MAKE CONCRETE

- 34** CULTURALLY RELEVANT PEDAGOGY
 - 36** LEARNING THROUGH MAKING
 - 38** SCRATCHMATHS: INTEGRATING COMPUTING AND MATHEMATICS
 - 40** SCRATCH ENCORE: A CULTURALLY RELEVANT SCRATCH CURRICULUM
 - 41** “ENGINEERS MAKE THINGS THAT HELP PEOPLE”
 - 42** DON’T TOUCH THE PLUGS OR BREAK ANYTHING!
- 

M

any of the ideas that pupils will encounter in computing are abstract and complex, and need to be illustrated using concrete examples and activities. Bringing these abstract concepts to life with real-world contextualised examples, and connecting them to other subjects, will help pupils assimilate new ideas into their existing understanding.

There are many ways in which you might make an abstract concept more concrete. One example that computing teachers may be familiar with is the use of unplugged activities, in which you take a computing concept and explore it in a non-computing context. You can also make use of analogies and storytelling to help connect a new concept with familiar experiences and comparable examples.

When considering concrete examples or providing context, you should reflect on the diversity of experience and culture of your learners. Use this knowledge of your learners to help you to deliver culturally responsive learning experiences.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:** culturally relevant pedagogy
- **What the research says:** learning through making
- **What the research says:** integrating computing and maths
- **What the research says:** a culturally relevant Scratch curriculum
- Cultivating engineering skills
- Hands-on learning to break down abstract concepts

CULTURALLY RELEVANT PEDAGOGY

Introducing culturally relevant pedagogical practices to your classroom can make all learners feel welcome and represented, and that computing is for them

In order for computing to be relevant, concrete, engaging and accessible to all learners, educators should reflect on their curriculum, materials, and teaching practices. Educators can draw on the full breadth of student experiences and cultural knowledge, facilitate projects that have personal meaning for learners, and discuss issues of bias and social justice.

What is culturally relevant pedagogy?

Culturally relevant pedagogy¹ is a framework for teaching that emphasises the importance of incorporating and valuing

all learners' knowledge, ways of learning, and heritage. It promotes the development of learners' critical consciousness of the world, and encourages them to ask questions about ethics, power, privilege, and social justice. Culturally relevant pedagogy emphasises opportunities to address issues that are important to learners and their communities.

Culturally responsive teaching² builds on the framework above to identify a range of teaching practices that can be implemented in the classroom. These include:

- Drawing on learners' cultural knowledge and experiences to inform the curriculum
- Providing opportunities for learners to choose personally meaningful projects and to express their own cultural identities
- Exploring issues of social justice and bias

It is important when using the term 'culture' that we do not focus on only one characteristic, such as ethnicity or race. While a person's ethnic background can contribute to their culture, a person's cultural identity can be based on a number of influences, including their age, gender, where they live, their family income, and their religious beliefs. Making computing a subject that is responsive to these different elements of learners' cultural identities will engage a wider range of learners.

Implementing culturally relevant pedagogy

The Raspberry Pi Foundation (RPF) has created guidelines for implementing

culturally relevant and responsive computing curriculum design and teaching (helloworld.cc/crpguidelines). These guidelines identify three main focus areas, with some key elements outlined below:

- Curriculum
- Teaching approaches
- Learning materials

Contextualise computing

Can you, for example, bring in the social, historical, or political context of a particular development in technology, ensuring that not only the dominant culture is represented? Can you make cross-curricular links to other subjects, or link to specific times in the school calendar (for example, Black History Month)? Understanding the relevance of theoretical concepts to real life is important in keeping a wide range of learners engaged in computing.

Allow student choice and promote collaboration and discussion

For example, allowing learners to choose a problem or issue that is personally meaningful to them to address through technology can encourage them to persist when facing difficulties. Collaboration and discussion allow learners to bring different expertise and knowledge to tasks, which can help challenge stereotypes about computing as a career.

Ensure your learning materials are accessible and inclusive

Consider the videos you use to introduce a topic, for example: are there captions that can support those with English as

SUMMARY

Culturally relevant pedagogy emphasises valuing all learners':

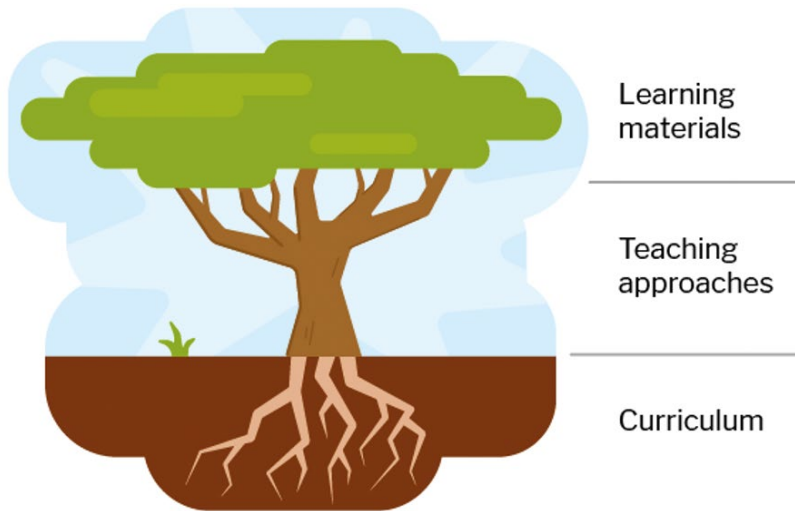
- Knowledge
- Heritage
- Ways of learning

Culturally responsive teaching includes:

- Opportunities for personally meaningful projects
- Curricula that draw on learners' cultural knowledge and experience
- Exploration of ethics, social justice, and bias

Benefits of culturally responsive teaching include:

- Improving learners' attitudes towards the subject
- Encouraging more learners to select computer science as a qualification
- Improving understanding in core computing concepts



■ The focus areas of culturally relevant and responsive teaching can be represented by a tree

an additional language? Can the captions be translated into other languages? Do the videos represent computing in a stereotypical way, or do they feature diverse groups of people? Ensuring that everyone can access and feel that they belong to computing is important, as it helps to engage and inspire learners.

You can think about these focus areas like a tree, with the curriculum forming the roots of the approach, and the branches representing a number of different teaching approaches you can take to deliver the curriculum. The leaves represent the learning materials you use in your computing lessons. Beginning with the curriculum and working your way up will give you the strongest basis from which to implement culturally relevant pedagogy in your classroom.

Benefits

The aims of culturally relevant pedagogy and culturally responsive teaching are to set high standards for all learners in terms of academic success, and to engage and retain more diverse learners in a range of subjects.

In computing, taking this approach has:

- Improved learners' attitudes towards the subject, increasing engagement, confidence, and feelings of belonging³

- Encouraged more learners to select computer science as a qualification
- Led to learning gains in computational thinking and core computing concepts^{4 5}

Providing authentic and meaningful contexts for learning computing and identifying different applications of computer science outside of school can help more learners see the relevance of computing to their lives and their communities.

Considerations

In order to successfully design and implement culturally responsive teaching, educators must first understand the approaches and reflect on their own unconscious biases. Adding a few activities as an add-on to regular teaching will not have the same impact on learners as incorporating the approach throughout all lessons. You can improve your understanding of the approach by reading the RPF guidelines and by looking up some of the resources suggested for professional development. These guidelines also encourage you to reflect on how your own cultural identity may affect the way you experience the world, and computing as a subject.

It is also vital that teachers identify areas in their current teaching where changes could be made. It is useful to work on

REFERENCES

- ¹ Ladson-Billings, G. (1995). Toward a theory of culturally relevant pedagogy. *American Educational Research Journal*, 32(3), 465-491 helloworld.cc/culture1
- ² Gay, G. (2000). *Culturally responsive teaching: Theory, research, and practice*. Teachers College Press helloworld.cc/culture2
- ³ Eglash, R., Gilbert, J. E., Taylor, V., & Geier, S. R. (2013). Culturally responsive computing in urban, after-school contexts: Two approaches. *Urban Education*, 48(5), 629-656 helloworld.cc/culture3
- ⁴ Davis, J., Lachney, M., Zatz, Z., Babbitt, W., & Eglash, R. (2019). A Cultural Computing Curriculum. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 1171-1175 helloworld.cc/culture4
- ⁵ McGee, S., et al. (2018). Equal Outcomes 4 All: A Study of Student Learning in ECS. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 50-55 helloworld.cc/culture5

this activity with a team of teachers, to bring together different ideas and cultural identities and ensure that culturally relevant pedagogy is being implemented similarly across different classes. It is even better if teachers can work across disciplines to incorporate culturally relevant pedagogy in a cross-curricular way to help embed the approach within the school.

Finally, uncovering unconscious biases and discussing meaningful, complex topics will include negotiating some uncomfortable conversations with colleagues and learners. It is important to model how to deal with these conversations sensitively. It is vital that everyone is able to speak openly and feel that their opinions and experiences are being heard and valued.

Making your computing teaching culturally relevant will include challenging conversations and difficulties at times. The reward, though, is a classroom where everyone feels welcome and represented, and feels that computing is most definitely for them. [\(HAW\)](#)



Credit: lo crifa/stock.adobe.com

LEARNING THROUGH MAKING

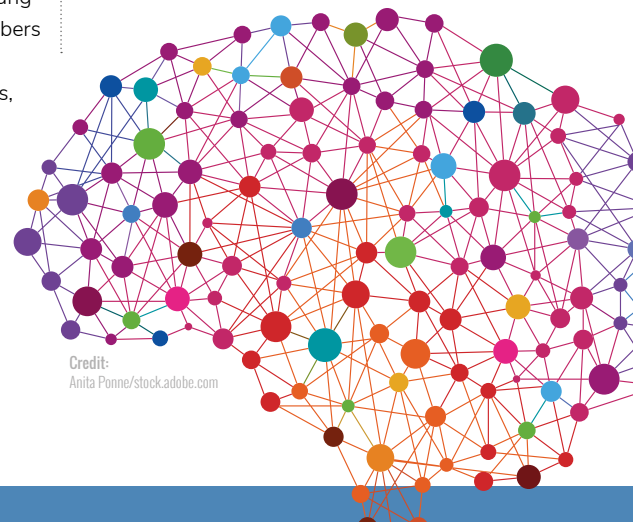
Making is so much more than just a tool for engagement; its power can be harnessed for a very different way of learning and understanding the world

It's pretty clear, if you know young people, that making is something that's going to engage them. Active lessons always get the popular vote from classes, especially if they let students make choices about what they work on. The sense of achievement you get from making something and sharing it with others or taking it home is pretty motivating too. There are always a few who would rather have a theory lesson, but the engagement you get from making is usually a powerful motivator. It's hugely important to get people engaged with learning in order for it to be successful, but learning is more complicated than simply paying attention to something. Seeing learning through making only as a way to engage people would be missing something much deeper. For proponents of constructionism, it's also about how making interacts with the way we develop understanding.

From concrete to abstract

The culture of education in the West can often be very focused on the cognitive — the abstract thinking that can be clearly defined in learning objectives, exams, and books. It tends to think of formal education as the process of coming to understand abstract ideas, with abstract ideas being the most important level of understanding that can then be applied to our everyday lives. Young children usually start learning about numbers through physically playing with concrete objects such as blocks, counters, and toys, but the aim is for them to move on to being able to discuss and manipulate numbers as abstract ideas. Dealing with concepts on a totally abstract level is hard, and children often have to return to these concrete methods to support their understanding. It takes time before children can add and subtract without the convenient aid

of fingers to count on; even when this is mastered, they often return to counters when learning about the more complex concept of division. This trajectory from understanding concepts in concrete, real-life terms towards being able to explore them in the abstract is explored well in the work of Jean Piaget, almost universally taught in teacher education courses across the Western world.



Credit: Anita Ponne/stock.adobe.com



Credit: ian855/stock.adobe.com

Affective learning

While we see the cognitive side of learning as key to understanding, we tend to see the affective, or experiential and feelings-based, side as useful for making learning engaging and memorable, but not as a fundamental part of it. Computer scientist and pioneer of the constructionist movement Seymour Papert saw it differently. In his seminal text *Mindstorms*, he vividly relates the affective experience of playing with cogs and gears as a child, and how he came to an understanding that machines could be both very structured and creative ways of interacting with the world.

was made by a person, and that with the right learning, that person could be you.

Learning as becoming

Such a change in understanding is quite a shift from the way educators are often encouraged to see learning; it's a different metaphor for the process. Much of the time, our language about learning is based on what Professor Anna Sfard calls the 'learning as acquisition' metaphor, in which learning is seen as discrete blocks of content that can be gradually acquired. There are other metaphors; when exploring

simply can't in the physical world. Logo may seem like primitive software to us in 2021, but Papert saw its potential to allow children to actively manipulate concepts such as angles and geometry. This made abstract concepts accessible for children to manipulate and understand by feel, much as sand and water trays in the early years allows children to explore their understanding of basic physics. We expect children to move on from this playful, exploratory approach to learning as they get older, but perhaps this is only because we lack the tools to make more sophisticated concepts concrete and accessible to them to manipulate. The power of computers for learning is described in Papert's writing not as being a way to deliver content to children, but as a tool they can use to explore and manipulate previously abstract concepts in a concrete way.

“ CHILDREN CAN USE COMPUTERS TO EXPLORE AND MANIPULATE ABSTRACT CONCEPTS IN A CONCRETE WAY

Papert writes about changing his world view not only in terms of gaining knowledge, but in terms of gaining a new relationship with knowledge. Manipulating and exploring the concrete objects of gears allowed him to develop an affective understanding of how machines work, and realise that these complex constructions are knowable and understandable. Mark Surman, the executive director of the Mozilla Foundation, describes this memorably as seeing the 'Lego lines' in the world; the visible joins that help you understand that something

the potential of learning through making, it helps to think about the 'learning as becoming' metaphor, the idea that we learn in order to explore and develop who we are as a person, and the way we see our identity fitting in to the world.

New tools for learning

Much of this could be an argument for learning through experience, but for Papert it was using computers that was incredibly powerful. Why? Computers allow us to manipulate abstract concepts in a way we

Harnessing the tools

Making is often a fun and engaging way to learn, yet its power can go beyond engagement and towards a very different way of learning and understanding the world. It takes a shift in how we think about learning and in the way we encourage young people to use computers to understand the world. These days, we certainly have more powerful and sophisticated tools accessible to young learners; perhaps the biggest challenge is understanding how they can be used not only to engage, but to learn in new ways that are both effective and affective. (HW)

SCRATCHMATHS: INTEGRATING COMPUTING AND MATHEMATICS

STORY BY Jonathan Dickens

Evidence has shown that learning programming can help students to develop higher levels of understanding of certain mathematical concepts. A project from UCL Knowledge Lab at University College London has been testing a fresh approach involving learning computing alongside mathematical concepts. They have designed an integrated curriculum using the programming environment Scratch to teach mathematical ideas. A recent evaluation has found that using this approach improves computational

thinking. There was no evidence, however, of an improvement in maths attainment, as measured by Key Stage test scores.

ScratchMaths is a two-year computing- and maths-based curriculum for pupils aged nine to eleven. Scratch is used to integrate coding activities into mathematical learning, to address a key problem that students have in learning mathematics: expressing mathematical concepts in formal language. The programme focuses on Scratch programming skills and computational thinking, with explicit links made to areas of mathematics. ScratchMaths provides teacher professional development to help educators deliver the curriculum, along with a range of teaching resources. John Morris, head teacher at Ardleigh Green Junior School in Essex, says, "For us, ScratchMaths is a breath of fresh air because it allows us to approach the teaching of mathematics in a new, exciting, and engaging way."

The Education Endowment Foundation (EEF) funded the programme. It has recently released a report produced by Sheffield Hallam University evaluating the impact of ScratchMaths on a sample of schools that worked with the curriculum over two years.

More than 100 schools, with over 6,000 pupils, were involved at the beginning of the trial. Around half were in the

intervention group in which ScratchMaths was taught for at least one hour every fortnight. In a randomised control trial, they were compared to the other half of schools, which did not use ScratchMaths. Teachers in participating schools received professional development for teaching ScratchMaths, with teacher mediation seen as a crucial factor in helping pupils build links between computing and mathematics.

Participating schools were located across England, including schools in deprived areas. Piers Saunders, a lecturer in mathematics education at UCL, explains, "The curriculum was designed to meet the needs of all children, not just for high-attaining children, or those who often attend clubs. It really was for all children, as evidenced by the large range of national schools that we had involved."

Evaluation of impact

The independent team at Sheffield Hallam University evaluated the computational thinking scores of pupils after one year of intervention, and the maths scores of pupils after two years. They also gathered data from teachers via surveys and telephone interviews, and assessed how schools had implemented ScratchMaths.

They found no evidence that ScratchMaths impacted pupils' Key

THE FIVE Es OF SCRATCHMATHS

- **Explore:** Moving from direct control to planning or building behaviours
- **Explain:** Experiencing concepts before defining them through discussions
- **Envisage:** Including unplugged activities and encouraging identification with the object being programmed
- **Exchange:** Providing the need to share or build on others' ideas
- **bridgE:** Explicit links to mathematics national curriculum

Stage 2 (ages seven to eleven) maths attainment, as measured by Key Stage scores. However, pupils in schools using ScratchMaths made more progress in computational thinking relative to schools that did not implement ScratchMaths. Teachers reported that ScratchMaths was useful for addressing certain aspects of the primary computing curriculum, and good for improving Scratch skills.

This positive effect on computational thinking skills did not differ between boys and girls who took part in the evaluation, which is a noteworthy finding, as previous interventions in programming education often highlight gender differences in how children benefit. The evaluation also found that progress was highest for children who were, or had been, eligible for free school meals, suggesting that ScratchMaths is an accessible curriculum for children of different socio-economic backgrounds.

The report also found that many schools did not fully implement ScratchMaths during the trial, with attendance of training sessions, use of materials, and time spent teaching ScratchMaths decreasing

“ SCRATCHMATHS IS A BREATH OF FRESH AIR

between years one and two of the trial. Pressures around the national standardised assessment tests (SATs) for ten- and eleven-year-old students in the second year were reported as a barrier to implementation for many teachers. Some students involved in the trial also experienced a change of teacher between years one and two of the trial, with incoming teachers not receiving professional development from ScratchMaths, and perhaps being less proficient in Scratch.

Given that the second year of the trial was the year in which computing concepts were to be more explicitly linked with mathematics ideas, it could be that these pressures resulted in the lack of

MODULE 1 • POSTER 1

TWO BASIC ALGORITHMS FOR CREATING CIRCULAR PATTERNS

▶ Algorithm **move + turn + stamp** (used in Module 1: Investigations 1 and 2).

▶ Algorithm **move forward + stamp + move backward + turn** (used in Module 1: Investigations 3 and 4).

■ A resource from the ScratchMaths 'Tiling patterns' module

MODULE 2 • POSTER 4

ANGLES AND REGULAR POLYGONS: HOW MUCH TO TURN?

repeat 6
move 60 steps
turn 60 degrees

exterior angle **interior angle** **external supplementary angle**

supplementary angles are pairs of angles which add up to 180°

To draw a polygon the Beetle needs to turn by the external supplementary angle. You can calculate this by calculating the difference between the interior angle of your polygon and 180°

total turn 360° is the sum of all external supplementary angles when drawing a regular polygon

■ All ScratchMaths resources are freely available at helloworld.cc/scratchmaths

improvement in mathematics attainment, and that application of the curriculum as intended could see the predicted effect on mathematics attainment. It is also possible that the timescale of the study wasn't long enough to observe this effect.

As the ScratchMaths resources are free and online, schools from various regions are exploring implementing the curriculum in their settings. There is a trial underway in Spain measuring the impact on students' computational thinking skills, and some educators have localised the resources to make them relevant to their existing computing and mathematics curriculum. In

the UK, local coordinators are in place to advise on implementing ScratchMaths.

While ScratchMaths has not raised mathematical attainment in this trial, the report suggests that where there is a need to develop teachers' computing skills, ScratchMaths could act as a low-cost form of professional development, helping teachers to develop computational thinking skills in pupils.

You can find out more about ScratchMaths and access resources at helloworld.cc/scratchmaths. If you're using ScratchMaths in your classroom, we'd love to hear your thoughts! (HW)

SCRATCH ENCORE: A CULTURALLY RELEVANT SCRATCH CURRICULUM

STORY BY Katharine Childs

Young people from minority groups can sometimes feel excluded in the computing classroom because learning resources use real-world examples that they cannot relate to. Educators can make use of culturally relevant materials to foster a sense of belonging in the classroom, and this can help learners to see computing as a subject where they fit in. One resource that educators can use is *Scratch Encore*, a US-designed curriculum for students aged 10 to 14 in which equity is valued as much as learning outcomes. This equitable ethos translates into a set of culturally responsive themes and topics with which students can create Scratch projects and learn computing concepts.

How does the curriculum work?

Scratch Encore consists of 15 different modules covering a variety of different computing topics. Educators can teach each module using one of three strands: Multicultural, Youth Culture, and Gaming. The strands and themes within them were developed through participatory design sessions with educators, students, and parents. Teachers can choose the strand that they think will best resonate with their learners. For example, they might choose to teach events in Scratch using a multicultural context, and then teach conditional loops using projects themed around youth culture.

The resources are written around the Use–Modify–Create approach, in which learners begin by investigating existing projects and then move on to applying what they have learnt by creating their own project. The resources were piloted with a small group of teachers, who suggested that they were useful in terms of student engagement. An automated assessment

tool analysed the projects made in the Create section and compared the frequency of blocks used by students in their projects against the blocks that had been taught in the Use and Modify sections. The initial results indicate that the *Scratch Encore* curriculum may indeed be balancing equity with learning outcomes.

Overcoming barriers to equity

The structure, content, and materials used in *Scratch Encore* have been designed to overcome several barriers to equity in the computing classroom. The Use–Modify–Create approach means that the beginning of each module is heavily scaffolded, ensuring that learners feel confident before moving on to a more open-ended approach. This approach also recognises that teachers have varying levels of experience and confidence. The curriculum makes provisions for the diverse prior experiences



■ The *Scratch Encore* curriculum offers teachers a choice of contexts for teaching computing concepts

a curriculum. It does this by carefully considering the inequities and barriers to learning computing and putting in place strategies to overcome them. Teachers who are preparing their own schemes of work for learners may benefit from considering the learning materials from an equity perspective and identifying ways in which they can engage learners by using culturally relevant contexts. [\(HW\)](#)

“ SCRATCH ENCORE HAS BEEN DESIGNED TO OVERCOME BARRIERS TO EQUITY

that students may have of using Scratch, and provides a review of introductory computing concepts in the first three modules. Learners' cultural backgrounds are represented through the diverse themes present in the three different strands; furthermore, learners are given autonomy in the Create section of each module. Finally, the resources are available for free, so that schools working within stretched budgets are not excluded.

Although it's still in the early stages of research and development, *Scratch Encore* provides a new approach to creating

FURTHER READING

- ✓ Franklin, D. et al. (2020). *Scratch Encore: The Design and Pilot of a Culturally-Relevant Intermediate Scratch Curriculum*. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE '20. New York. 794-800. helloworld.cc/scratchencorepaper
- ✓ The *Scratch Encore* curriculum: helloworld.cc/scratch-encore

“ENGINEERS MAKE THINGS THAT HELP PEOPLE”

Katie Henry examines ways to cultivate engineering skills in all students

“Let’s get a fan and see if it works,” says Edwina. Edwina, a student in my grade class, had just finished building a model of Theo Jansen’s *Strandbeest* — a kinetic sculpture that walks in the wind. As the fan came on, her sculpture began walking across the table, and a few children cheered. I had never seen Edwina with such a wide smile. Then one of the legs fell off. Edwina wrangled it back into place, explaining that she’d need more time to “really fix the leg”.

As she and a few other students tinkered with her sculpture, I reflected on what I’d seen: a nine-year-old girl confidently dealing with complexity, persistent in working on challenging problems, flexible, and tolerant of ambiguity. (Others may have noticed that she has dyslexia, reads below grade level, and struggles with a severe speech impediment.)

tools, programming, or robotics in your classroom. It’s already happening on school buses, at lockers, and — whether you realise it or not — in your classroom. Introducing young people to engineering starts with learning to recognise engineering skill, and helping students to recognise it. The next sentence you speak can introduce a young person to engineering. Below are five types of engineering skill, what they might look like in your classroom, and what you can say to help students develop these skills.

- **Intentional design** Look for preplanning or evidence of thinking ahead.
What you can say: “Tell me more about your planning process.” Encourage intentional action and allow students to make their own decisions.

weaknesses of each solution. Engineers inspect and adapt.

- **Prototyping** Look for evidence of a student modelling an idea to reach a goal.
What you can say: “In what ways does this model represent your thinking?” Prototyping takes many forms: ‘works-like’ prototypes are working models; ‘looks-like’ prototypes are non-working. Engineers use a variety of models, tools, and strategies to better understand their ideas.
- **Communicating design** Look for students sharing ideas about something they’re planning or are creating.
What you can say: “Who most needs to hear your idea? What would be the best way to share it with them?” Encourage students to communicate their ideas in multiple ways, for multiple audiences. Engineers share what they discover and make with others, in order to make their ideas better. [\(HW\)](#)

“ENGINEERING DOESN’T HAVE TO BEGIN WITH COSTLY TOOLS OR ROBOTICS

This is a public school. “You’re growing as an engineer,” I say. She was puzzled: “What’s an engineer?” Max, another student, gave a definition that I still use to this day: “Engineers make things that help people.”

Recognising engineering skill

Jennifer Cross, author of *Creative Robotic Systems for Talent-Based Learning* (helloworld.cc/roboticsystems) writes, “Engineering design is the process of developing a concrete solution for an ill-defined problem within technical feasibility constraints.” The good news is that engineering doesn’t have to begin with costly

- **Innovating** Look for novel or risky efforts.
What you can say: “How did you decide to try it this way?” Try to understand why the student created something new, and help them to consider their efforts from another person’s point of view. Engineers make things that help people!
- **Refining and testing** Look for efforts that repeat and improve each time, in order to reach a goal.
What you can say: “Can you share more about your goal?” Encourage the student to focus on their goal, generate more solutions, and consider the strengths and



Katie is a former classroom and STEAM teacher. She now works as Head of Partnerships, North America, for the Micro-bit Educational Foundation (@KatieHenryDays).

DON'T TOUCH THE PLUGS OR BREAK ANYTHING!

How can getting hands-on with computers (and wires!) help us to break down abstract concepts in primary school computing?

In an early article in Hello World issue 5, I asked the question "Is a Bee-Bot a computer?" and considered children's existing perceptions about what a computer is and isn't. However, acknowledging the problem is one thing, and finding new ways to teach it is another. When I was a computing teacher, I was looking for an exciting, discovery-based learning activity that would allow children to test their assumptions without just being lectured about what makes a computer a computer or what makes a peripheral or an input device.

To do this, I looked to other subjects, and in particular to one area where my school was doing this really well: maths. In maths, when we got to the tricky concepts, we got out the cubes, the pizza-shaded fraction sets, or the place value number cards, and they allowed us to demonstrate these concepts. Children could then be left to explore, by moving their manipulatives around and seeing what they could discover. Could we do this in computing?

Manipulatives in computing

Computing often seems a very hands-on subject, because learners in computing use technology. However, we know that this isn't enough for them to understand how it works. Poking mindlessly at a device ultimately just becomes frustrating, and when you do manage to achieve what you wanted, you've often forgotten how you did it, so you can't repeat it.

And then I saw the wires ...

Desktop computers have many wires plugged into them. If you have an ICT suite, these wires may even be cable-tied into the base unit, to prevent small fingers from poking them, but this is where discovery learning happens! This is where you can explore what a computer needs to work, and what are just useful extras.

Now what?

Firstly, my school didn't have any desktop computers the children could use (as lovely as the office staff were, they'd probably have drawn the line at us dismantling their computers for the children to explore in the middle of a work day). To get around this, I used some Raspberry Pi computers, as these are cheaper to get hold of, and a good size for young children to explore. However, if you have desktop computers, use those!

I started by showing the children the Raspberry Pi under the visualiser with no

case, so you could see everything inside (you can take the side off if you're using a regular desktop computer) and said, "This is a computer." This may seem counter-intuitive for discovery learning, as I told the children the answer, but there's much research that suggests children will often conclude incorrectly based on their own discoveries. I then got them to explore in their groups and discover for themselves what made it a computer, with each table having their own computer. They had the answer, but they had to make the journey. This journey was documented using a 'statements, questions, ideas' model. Pupils used big paper and discussed in table groups what they could see on a Raspberry Pi that suggested it was a computer (statements), what questions they had or would like to explore further to show that Raspberry Pi is a computer (questions), or any ideas that they had for what they would like to do with this computer (ideas).

Statements	Questions	Ideas
It has holes on the back	What are the gold stick things for?	Could this be so that you can plug in a keyboard?
There are square things drawn on	Why aren't all the holes the same?	This says power - is it like a phone charger?
Everything is connected with wires	Why is there so much empty space inside?	Is this what's inside things like traffic lights?
This is where you plug in the charger		

■ Students' thoughts about what makes a Raspberry Pi a computer - yes, there are some misconceptions!

We shared this discussion as a class and then I plugged everything in. We proved that it is a computer, and that it works!

This is a very initial exploration, but it's the beginnings of the children recognising inputs and outputs, as well as which parts of a computer are required, and which part are extra peripherals. With seven- to eleven-year-olds, if I jumped straight to asking the children to plug in a Raspberry Pi and turn it on, they would miss a lot of the dialogue that helps them to assimilate their new understanding.

Playing with plugs

In the next lesson, I began by getting the children to connect the peripherals to the computer and giving them the chance to turn it on in groups. There's not much that can go wrong here: it's like a jigsaw puzzle where each wire only connects in one place. For example, if you tried to plug an HDMI cable into the USB port, it simply



■ You can use Raspberry Pis as manipulatives in computing

wouldn't fit. The children were given time to plug everything into an extension lead. This is something the children had probably never been allowed to do, and it was both exciting and scary at the same time. They've often been told that electricity is dangerous, and that they shouldn't touch things, as they would hurt themselves, or break them. While safety messages and lessons about electricity are important in children's upbringings, there's very little damage the children could do to a computer, particularly when it's not even plugged into the mains.

After they'd achieved this, I started to challenge their misconceptions further. As a class, they watched me with my nicely connected computer, using the board as my monitor. I asked, "What would happen

if I unplugged the keyboard?" This was met by mostly horrified faces of learners who thought the computer would crash and never turn on again. So I did it. We held our breaths. Nothing happened. It was all very anticlimactic. I asked again, "What happens?" This time, we concluded that you can't type anything. This didn't seem that important to most of the children, who preferred clicking on things anyway. I then repeated this process with the mouse and the monitor, and then suggested it with the power supply.

I didn't actually disconnect the power, as this is more likely to confuse or corrupt the computer, but did the children recognise that a computer needs a power source? This is not a peripheral; it's a vital component. Now, you could discuss whether the computer is still a computer if it doesn't have a keyboard or mouse. You just can't use the keyboard or mouse. How else could you control it? What about voice control, such as Alexa or Siri? Once again, you're back to exploring what inputs and outputs are, and what parts are necessary for the computer to work, and the children are refining their own definition of what a computer is.

Using Raspberry Pis as a manipulative in computing helped the students to grasp concepts of an abstract nature. By starting the exploration with the device unplugged, the children were able to recognise their own understanding, which led to a more focused testing of these beliefs in subsequent lessons.

These were the definitions I was given when I first asked: "What is a computer?"

- A keyboard and a screen
- A machine with a keyboard
- A search engine
- A machine used for work

After these lessons, I was told:

- A computer has lots of switches and plugs to plug things into; it doesn't have to have a screen
- A computer needs code on a microchip to make it work; without that, pressing a letter would make nothing happen
- Not all computers look like a computer; they have different shapes and designs and are used to meet different needs

THE NEXT STEP

For older children, it can be fun to give them working projects that don't use a keyboard and mouse, to get them to figure out how the computers work. Using simple online tutorials, in the past I've set up:

- A motion-sensor camera to take photos when you're nearby
- A warning system for when a plant needs watering, using a micro:bit
- A simple robot using a controller to make motors turn

This is a chance for children to explore what has a computer inside it, and the various inputs and outputs that are fit for purpose.

While these answers still showed some conceptual misunderstandings, I think there's noticeable progress in the complexity of their answers and the larger breadth with which they were trying to draw their conclusions. Their reliance on a computer's functionality to define it is reduced, and their explanations of what a computer needs to be defined as such is more dense.

Have you ever asked your class what a computer is? You might be surprised at the answers you get! [\(HW\)](#)



Sway is a senior learning manager at the Raspberry Pi Foundation, where she leads a team developing computing resources for primary teachers (@SwayGraham).



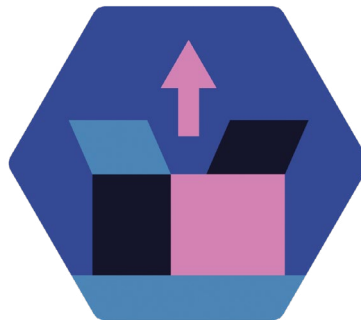
UNPLUG, UNPACK, REPACK

- 46** SEMANTIC WAVES
 - 49** GO UNPLUGGED FOR BETTER COMPUTATIONAL THINKING
 - 50** SEMANTIC WAVES AND CRAZY CHARACTERS
- 

Building on the idea of making concepts concrete is the idea that you should unplug complex terms and concepts from the context of computing. The practice of unplugged computing involves stepping away from the context of computing, and often from the computers themselves, to explore concepts in familiar settings. (This shouldn't be confused with simple offline activities in which learners explore computing without the computer.)

An important requirement of any unplugged activity is that learners make connections back to the computing context. This is where you should apply a semantic wave approach. To begin, you take the terminology used by experts and unpack the meanings, making them simpler and more relevant to your learners. After this, pupils can repack those simple meanings into the expert terminology, ensuring that they understand their nuances and can use them appropriately.

By following this wave from the original meaning down to something familiar and then back up, you can build pupils' understanding and prevent them from misunderstanding key terms or being limited to overly simplistic language.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:**
semantic waves
- **What the research says:**
unplugged activities for
computational thinking
- **Reviewing a lesson activity**
using semantic waves



SEMANTIC WAVES

Educators can improve explanations in computing using semantic waves, unpacking and repackaging abstract concepts, and technical language

Educators can improve explanations and learning activities in computing by using semantic waves. Semantic waves describe an ideal conceptual journey for novice learners to follow, shifting between expert and novice understanding, abstract and concrete context, and technical and simple meanings. It is part of Legitimation Code Theory (helloworld.cc/lct) or LCT (Maton, 2013).¹ Semantic waves have been successfully applied by educators across many disciplines, including computing, to plan and evaluate learning experiences. The theory also helps explain when and why metaphors and unplugged teaching work (and why, sometimes, they might not).

Following a semantic wave

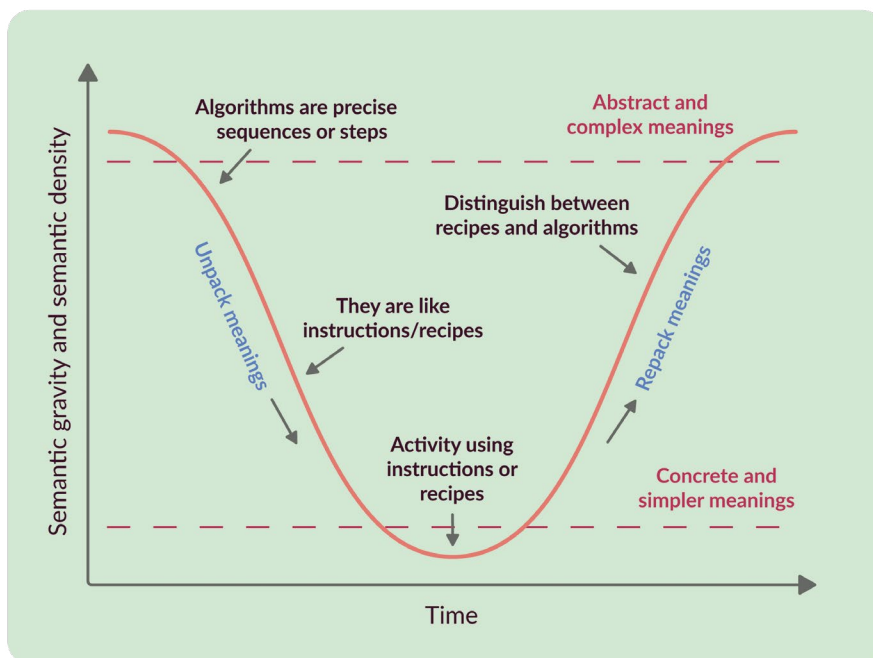
Computing, and especially programming, is a subject with lots of technical terms that have precise technical meanings. To succeed, learners have to master the terminology while simultaneously developing a firm understanding of the concepts. A great strategy for supporting learners is to make your learning experiences follow a semantic wave.¹ This involves introducing abstract concepts (with the associated terminology), but then using simpler language to explain their meaning. This is why metaphors, analogies, and unplugged computing are powerful ways to teach, provided they are used well.² However, it is important to then help

students link those simpler meanings directly back to the abstract concepts and associated technical language.

For example, when an educator introduces variables and assignment (using technical words and abstract concepts), learners are at the top of a semantic wave. To help learners descend the semantic wave, the educator might explain variables using boxes (helloworld.cc/boxvariable). To help them descend further, the educator might then illustrate the explanation with physical props. However, the educator shouldn't leave learners thinking that it is just about boxes, by talking only about moving values between boxes; they must help learners link this back to the technical and abstract, so that learners can surf back up the semantic wave. For example, the educator might do a step-by-step demonstration of a sequence of assignments in Python by putting values in boxes, or they might have learners follow a program fragment, to help learners repack the meanings. In traversing this wave, educators can support their learners in understanding complex, abstract concepts that are underpinned by concrete and familiar ideas.

Language and context

Experts and novices understand and describe concepts differently. While novices are more comfortable using concrete contexts to express concepts in simple language, experts are far more likely to describe the same concepts in the abstract, and to use precise technical language. Unpacking and repacking concepts is achieved by adjusting either of these two aspects.



■ This is a semantic wave for a lesson about algorithms



RESEARCH

■ Educators can use simpler language to convey meaning, but they should always return to the technical language

Credit: stock.adobe.com/Gorodenkoff

By decreasing the complexity and precision of the terminology (the semantic density),¹ educators can make ideas more accessible to learners. Educators may start with precise terminology such as 'iteration' or 'selection', but then use less precise terms for novices (such as 'repeating' or 'decision'). An important final step is to return to the original and precise terms that were used to introduce the concept.

The other approach to unpacking and repacking concepts revolves around the context through which they are presented

they break down. They move their understanding from the specific and concrete to the general and abstract.

Semantic profiles

Semantic profiles are visual representations in a learning activity of changes in language and context, and allow educators to critique those experiences. Studies have identified some common teaching patterns that have poor semantic profiles, and therefore lead to poor explanations, and make it harder for students to learn, as follows:³

“ SEMANTIC WAVES DESCRIBE AN IDEAL CONCEPTUAL JOURNEY FOR NOVICE LEARNERS TO FOLLOW

(their semantic gravity).¹ Educators do this all the time through analogy, unplugged activities, physical computing, and so on. A more contextualised exploration of a concept gives learners a concrete example to build their understanding. However, if learners don't then step back from their concrete examples and view the concepts in the abstract, their understanding may become limited to the single context.

For both language and context, the repacking phase in the semantic wave is crucial: during this phase, learners explore the nuance of technical terms such as 'algorithm', as well as where analogies work, and where

High flatlining: The educator might only explain and discuss concepts in technical language and abstract contexts. This is what experts do when talking together. They do not unpack the meanings at all, assuming that the other has mastery of the language



■ A high flatlining semantic profile

SUMMARY

Following a semantic wave structure:

- Helps make expert knowledge accessible to novices
- Varies the context of the concept to build links with concrete examples
- Connects the technical terminology used in activities with simpler meanings
- Helps learners to unpack new concepts and repack them into more complex contexts, to encourage the acquisition of new knowledge
- Promotes achieving a secure knowledge of one concept before progressing to the next
- Helps novices develop both understanding of abstract concepts and mastery of technical meanings

Considerations:

- Plan your lessons around a semantic wave structure; look for opportunities to unpack and repack concepts
- Evaluate your lesson plans and explanations in detail, drawing the semantic profile
- Make sure that across your learning resources you use both routes to expertise, varying either the language or the context
- Avoid semantic flatlines (never unpacking or repacking concepts)
- Complete each semantic wave and avoid down escalators
- Encourage learners to write their own explanations using semantic waves



■ Educators should link the concrete example (e.g. recipes) back to the abstract concept (e.g. algorithms) or they will be 'low flatlining'

Credit: stock.adobe.com/baibaz



■ A low flatlining semantic profile

► and concepts. Such an explanation is incomprehensible to a novice learner, as they do not understand the terminology.

Low flatlining: The educator might only use simpler examples and language, and never make the links to the concepts they are trying to explain, or move out of specific contexts. For example, in a lesson about algorithms, if the educator just talks about recipes, learners may understand the explanation, but never understand how recipes are like algorithms, or how they are not.

Down escalators: The educator may structure an explanation to take learners down the semantic wave, but not back up. The educator makes a link from a technical concept, but learners do not repack the ideas during the activity. The class moves on to

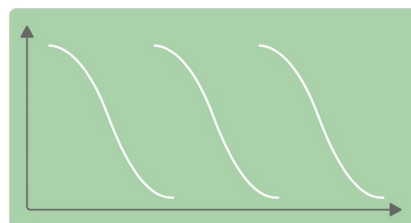
the next concept before having repacked the simpler meanings into the technical meanings.

Reviewing learning activities

An important use of semantic waves, and particularly semantic profiles, is as a basis for reviewing learning activities. In a recent paper, Waite et al.⁴ used this methodology to review the Barefoot activity Crazy Characters (helloworld.cc/crazycharacters). You can read a summary of the paper on page 50.

Try this in your next lesson! By using semantic waves and profiles, you can predict and monitor learners' challenges and improve their learning experiences. [\(HW\)](#)

This article was adapted from Paul Curzon's blog,⁵ which is based on the work of Karl Maton¹ applied to a computing context. We would like to thank them both for their input.



■ A down escalator semantic profile

REFERENCES

- ¹ Maton, K. (2013). Making semantic waves: A key to cumulative knowledge-building. *Linguistics and Education*, 24(1), 8-22. helloworld.cc/semantic1
- ² Curzon, P. et al. (2018). Teaching of concepts. In: Sentance, S., Barendsen, E., & Schulte, C. (eds.), *Computer Science Education: Perspectives on Teaching and Learning in School*. London, Bloomsbury Publishing, (pp. 91-108). helloworld.cc/semantic2
- ³ Maton, K. (2019). Semantic waves: Context, complexity and academic discourse. In: Martin, J. R., Maton, K., & Doran, Y. J. (eds.), *Accessing Academic Discourse: Systemic Functional Linguistics and Legitimation Code Theory*. London, Routledge, (pp. 59-85). helloworld.cc/semantic3
- ⁴ Waite, J., Maton, K., Curzon, P., & Tuttiatt, L. (2019). Unplugged Computing and Semantic Waves: Analysing Crazy Characters. *UKICER: Proceedings of the 1st UK & Ireland Computing Education Research Conference*. New York, Association for Computing Machinery. helloworld.cc/semantic4
- ⁵ Curzon, P. (2019). *Tip 9: Follow Semantic Waves - Learning To Learn (To Program)*. helloworld.cc/semantic5

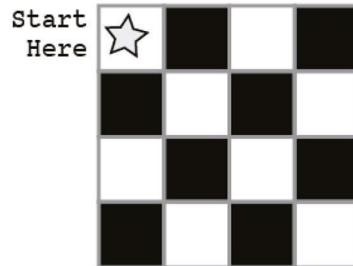
GO UNPLUGGED FOR BETTER COMPUTATIONAL THINKING

STORY BY Thom Kunkeler

Unplugged activities are a great option for students who do not have access to computers at home, and new research shows the benefits of such activities for computational thinking. Many of these activities can be done with only a pen and piece of paper, an instruction sheet, and a partner — such as a parent or guardian, sibling, or classmate.

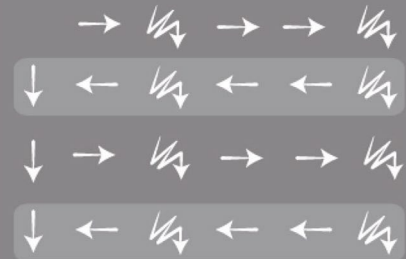
A group of researchers from the University of Castilla-La Mancha in Spain, led by Javier del Olmo-Muñoz, were interested in the relationship between unplugged activities and computational thinking. They found that students who start out in computing education with unplugged activities before switching over to computer-based activities show a significant advantage in computational thinking skills compared to their peers.

Jeannette Wing defined computational thinking, widely considered an important aspect of computing education, as “the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer — human



Follow along with your finger and see if you can figure out how to get this image from the program to the right.

Now, our entire program looks like this:



■ An example of an unplugged activity from Code.org



thinking, decomposition, evaluation, and generalisation.

For this study, the researchers designed eight 45-minute lessons for an unplugged classroom, as well as lesson plans for a plugged-in group. Unplugged activities were chosen from the computing education resources website Code.org. In the first

of an unplugged activity is My Robotic Friends, in which students adapt the set of symbols from the first exercise and take turns participating as a robot, acting out the algorithm defined by their partner.

Throughout the research intervention, students from one group engaged in three unplugged activities, followed by two plugged-in ones, and another group engaged in only plugged-in activities. Using tests before, during, and after the activities, students were assessed on their computational thinking skills, and they also completed a motivational survey for the activities. As well as the higher learning gains of the unplugged group, the findings indicate that these students were more motivated about their instruction. For parents, guardians, and educators, these findings illustrate the benefits of experimenting with unplugged activities from home. Code.org activities are free to access. [\(HW\)](#)

“ COMPUTATIONAL THINKING DOESN'T HAVE TO BE TAUGHT IN A CLASSROOM

or machine — can effectively carry [them] out”. However, it may not be obvious to educators that developing computational thinking doesn't need to take place in a classroom, and that it can be taught without the use of computers. This will be particularly useful for students who fall ill for long periods of time, or need to stay at home for remote learning, such as during the coronavirus pandemic. In addition to the advantage of not requiring computers, these activities have been found to be great for learning computational thinking skills such as abstraction, algorithmic

session, students were asked to pair up for the Graph Paper Programming exercise, in which students ‘program’ one another to draw pictures by giving each other instructions for drawing on a 4x4 grid, such as ‘move a square to the right’ or ‘fill the square with colour’. Another example

FURTHER READING

- ✓ Del Olmo-Muñoz, J., Cózar-Gutiérrez R., & González-Calero, J.A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education*, 150. helloworld.cc/unpluggedct

SEMANTIC WAVES AND CRAZY CHARACTERS

Reviewing a lesson activity using semantic waves

Crazy Characters is a free online lesson plan which introduces algorithms to primary pupils using an unplugged activity. It is one of the free resources available from the Barefoot website (helloworld.cc/crazycharacters). In the activity, learners are asked to follow verbal instructions (see **Figure 1**) to draw a crazy, made-up character. The instructions are not very precise, so that learners can then improve the algorithm.

JANE WAITE

Jane is a research scientist at the Raspberry Pi Foundation. Her interests include using design in primary programming, semantic waves, PRIMM, and migrating to online teaching using ABC.

KARL MATON

Karl is the director of the LCT Centre for Knowledge-Building at the University of Sydney. Karl is the creator of Legitimation Code Theory (LCT), which is being widely used to shape research and practice in education, sociology, and linguistics.

LUCINDA TUTTIETT

Lucinda is the Barefoot Education and Liaison Manager with the South West Grid for Learning. She taught in primary schools for 18 years before moving into the advisory services in Bristol and Somerset, supporting schools with ICT, and then the computing curriculum.

Following a whole-class activity, learners then design their own algorithm in order to draw their own crazy character.

Where did Crazy Characters come from?

In 2012, Michael Gove disapplied the English ICT Curriculum, creating a two-year hiatus while we, primary teachers, awaited a new statutory ICT curriculum.

In the meantime, we were still required to deliver the old curriculum, or to start to teach what we thought might come next. I [Jane] recall being at BETT, the big computing education trade show, as the announcement was made, and then frantically searching for resources and people who could help me rewrite my school's ICT subject planning. I recently found my first revised scheme of work, which I created for September 2012 – there was no mention of algorithms, but there were learning objectives such as, 'I can predict the results of someone else's instructions.'

Many computing lesson plan versions later, in spring 2014, I applied for a secondment from my school and for a

job as a content author on the Barefoot Computing Programme. Managed by the British Computer Society (BCS), the initiative was funded by the DfE and BT, and was one of the first of many successful, innovative, and crucial Computing at School (CAS) programmes to support teachers in their delivery of computer science in school. Very luckily, I got the job and my life changed completely.

Over the next year, the Barefoot team developed resources that demystified the computer science elements of the new computing curriculum. Using an iterative approach, we wrote concept documents and their associated classroom activities, publishing as we went along. The algorithms concept was first, and I was tasked with thinking of an introductory unplugged activity. In June 2014, Crazy Characters was born.

Crazy Characters was one of the first resources on the new Barefoot website, part of the very first continuing professional development (CPD) presentation, and is still a staple of the Barefoot volunteer workshop delivered to teachers in schools. When writing the activity, I was keen to make sure

SEMANTIC GRAVITY EXAMPLES

An activity with weaker semantic gravity would be to ask learners to memorise a definition of an algorithm without any context, such as 'an algorithm is a set of precise rules or steps to solve a problem'. Semantic gravity would become stronger by adding an example, such as 'an algorithm is a set of precise rules or steps to solve a problem, such as

an unambiguous set of steps to draw a square'. This activity has now shifted from weaker to stronger semantic gravity. It would be strengthened further if learners then engaged in a practical activity of creating algorithms to draw squares in which the need for sides of equal length was explored, to highlight the importance of precision.

that it was easy to run in class, fun, and, most importantly, gently introduced this new word 'algorithm' by doing, rather than telling.

I built on the way in which I normally taught instruction writing in literacy, which included a spot of curiosity, teachers getting things wrong, humour, and peer review. I had no idea that five years later I would be asking Professor Karl Maton, a leading education researcher, to review the lesson plan and to investigate my planning in terms of semantic waves.

Semantic waves – what are they?

I was introduced to semantic waves by Professor Paul Curzon, who has written about their potential for teaching programming (helloworld.cc/curzonblog). The notion of semantic waves is part of a wider theory called Legitimation Code Theory or LCT, created and developed by Karl Maton (helloworld.cc/maton2013). Very simply put, we can use semantic waves to review learning activities, and

SEMANTIC DENSITY

An activity asking learners to 'follow the instructions to draw a square' would have weaker semantic density than one requiring learners to 'follow the algorithm to draw a square'. This is because the first activity is less complex, as the term 'instruction' has a less complex meaning than the term 'algorithm'.

Semantic density explores the complexity of meanings. Where meanings are relatively simple, such as something that is described in everyday language, semantic density is weaker; where meanings are more complex, such as in the case of technical concepts, semantic density is stronger.

We can depict changes in semantic gravity and semantic density as a semantic profile; an example is shown in **Figure 2**.

How to draw a crazy character algorithm



Figure 1 Teachers read out their algorithm for how to draw a Crazy Character

Credit: Jane Waite

feeding into teacher training, curriculum planning, and classroom practice.

Enough of the theory — we now need a concrete example, so we are going to strengthen our own semantic gravity!

Creating the semantic profile for Crazy Characters

I contacted the creator of semantic waves, Karl Maton, asked him if he could help me create a semantic profile for a resource, and suggested Crazy Characters, as it is very familiar to me, is still very popular with teachers, and was also due for a review.

In an online hang-out, Karl read the lesson plan, and together we walked very carefully through each step of it and drew up the semantic profile. We profiled the plan as though a teacher was following the plan to the letter.

What does the semantic profile for Crazy Characters look like?

The semantic profile for just the introductory part of the Crazy Characters lesson is

SEMANTIC GRAVITY EXPLORES THE CONTEXT OF MEANINGS AND LOOKS AT HOW MUCH MEANING DEPENDS ON THE SOCIAL CONTEXT TO MAKE SENSE

abstract the process of learning to better think about how learners develop an understanding of knowledge. The overall aim is that, by doing this, we can reflect on and improve teaching experiences for our students. To explain these ideas, I need to briefly outline two concepts introduced on pages 46–47: semantic gravity and semantic density.

Semantic gravity explores the context of meanings, and how much of meaning depends on the social context to make sense. So where meanings are more concrete (such as practical examples, or those from personal experience), semantic gravity is stronger; where meanings are more abstract (such as theory), semantic gravity is weaker. Changes in semantic gravity can be shown over time, such as when teachers or students move from theory to examples, or from practical activities to a concept.

In this example from teaching biology, the teacher begins by discussing a scientific concept in abstract and technical terms. The teacher and students then unpack some of its meanings in everyday language, through practical and concrete examples. Finally, the students repack those examples into technical terms by completing a table of concepts. This moves from abstract and complex meanings down to more grounded and simpler meanings, and then back up to abstract and complex meanings. These movements up and down are called semantic waves, and a rapidly growing body of research is showing that they are crucial for knowledge-building in classrooms. Study after study is showing that waves enable knowledge to be built, while flatlines (such as continuous description or incessant theorising) hinder knowledge building. These insights are now

A USEFUL LINK

To get the most out of this article, download the Crazy Character lesson plan and walk through it with us: helloworld.cc/crazycharacters.

FEATURE

➤ shown in **Figure 3**. It is broadly a U-shape, but with steps coming out of the U. We will now go through each of the lesson plan steps and explain the wave.

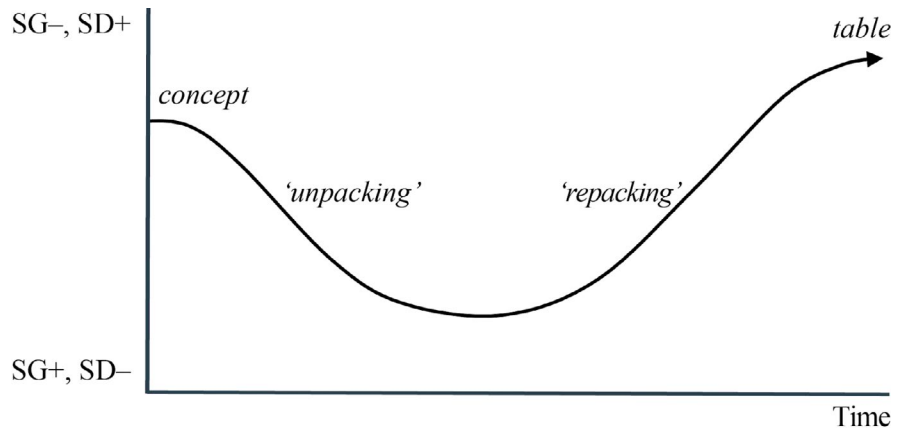
SIGNALLING To start with, the teacher is asked to explain to students that a special new word is going to be used. Learners are signalled that something important is coming, that a concept high up the semantic profile is on the way. Learners are NOT provided with a definition at this stage. Instead, curiosity and expectancy are kept high, so they can form their own understanding of the term later through the practical experience. There is no practical concrete activity going on here, so semantic gravity is weaker.

CONCEPT INTRODUCTION The term 'algorithm' is introduced as the teacher starts to use the word. The teacher should NOT explain what the word means at this point. There is no practical activity here (weaker semantic gravity), but it is clear that the term is a complex and technical one (stronger semantic density).

CONNECTING In the plan, the teacher is instructed to say that they are going to use the algorithm now. This clear connection of the concept to the activity is very important. The connection enables learners to add the knowledge they gain during the practical activity to their emerging understanding of the meaning of the concept. As shown in **Figure 3**, the semantic profile line drops, like a bungee rope, as we connect the theory

ADVANTAGE

Some research indicates that learners from more socially advantaged homes may be more comfortable with semantic waves than students from less advantaged homes, who may experience less semantic waving. The rationale is that some learners are more likely to have generalised and complex meanings explained to them, from a very young age. In other words, the 'why' question gets answered, and experiences are provided that exemplify the 'why'.



■ **Figure 2** This example of a semantic wave comes from biology teaching (Maton, 2013)

Credit: Karl Maton & Jane Waite

to the practical activity (strengthening the semantic gravity as the context is introduced). If there was no connection, the line on the profile would break.

CONCRETE ACTIVITY Next, the teacher is asked to read out the steps to enable the learners to draw the crazy character. The wave is low on the profile: it is a concrete activity (stronger semantic gravity) and likely to be expressed through relatively simple meanings (weaker semantic density), unless learners start to use the term 'algorithm'. If this were the case, there would be little spikes of semantic density.

COUNTER-EXPECTANCY The teacher is asked to be very vague with the instructions given to learners. The aim is that when they ask the pupils to share their drawings, the images will be very different, and they can say that they did not expect this to be the case, and ask why. This is called counter-expectancy. This means that the context in which the learners are developing their understanding is challenged and alternative options are raised. This increases the meaning of the concept. On the semantic profile, this is shown as a step up (widening the context weakens semantic gravity; adding meaning strengthens semantic density).

STAGED RETURN Next, the teacher is instructed to ask the learners how they could improve the algorithm. Learners start to think about making the algorithm more precise, but this is still in a relatively

specific context. On the graph, this shows as another step upwards (adding meaning strengthens semantic density).

PACKING Finally, the lesson plan instructs the teacher to ask a generic question of 'What was the algorithm?' This is a more general view of the activity, requiring the learner to 'repack' their accumulated understanding from the practical activity. Again, this is moving up the profile, further away from a specific context, and adding more meanings (reducing context weakens semantic gravity).

THE REST OF THE LESSON We have not profiled the rest of the lesson for this article. Broadly, it follows a similar set of patterns. However, the highly prescriptive nature of the introduction is loosened as the learners create their own Crazy Character algorithms. Included in this is the introduction of a further concept, that of debugging, as they ask their friends to implement their algorithms as drawings and then, together, debug the algorithm in order to produce the same imagined character.

How has creating the semantic profile been useful?

By drawing the semantic profile for Crazy Characters with Karl Maton, I have had the opportunity to apply semantic waves from theory to practice. This experience has provided a number of useful outcomes.

Firstly, it has introduced me to a language that has helped me describe

the lesson plan. Secondly, and more importantly, semantic profiling has enabled me to analyse and reveal why the learning activity worked. It showed how ideas were introduced in a concrete way and more complex meanings were gradually added, stepwise, to develop a more general and abstract understanding. Thirdly, the process has supported my review of the activity, helping me think of ideas to improve and build upon the lesson plan. Finally, I have concluded that semantic profiling is a practical and useful approach that I will continue to explore and use when designing teacher professional development, and in creating lesson planning material.

Would you like to change Crazy Characters?

To maintain the semantic wave I would like to add a follow-on lesson that applies in a programming context what was learnt in this unplugged lesson. I would also like to reorder the learning intentions to match the Use-Modify-Create theory ([helloworld](#)).

cc/umc) and I would increase the use of the term 'design', following my own research area.

However, overall, I would not change the main steps of the Crazy Character lesson plan. By creating the semantic profile, I have revealed how the plan provides a carefully scaffolded learning experience to help learners develop an understanding of the meaning of the algorithm concept. As shown in **Figure 3**, the lesson plan includes a signal that a new concept is to be taught, introduces the concept, connects theory to a concrete activity, incorporates a concrete activity, and increases the meanings condensed within the concept to reveal a 'packed' (complex) definition of the concept.

Working with Karl Maton on reviewing Crazy Characters has been thoroughly enjoyable. I am indebted for the time he has kindly spent supporting me in writing this article. I would also like to thank Lucinda Tuttiett, who read through the article and helped keep it real!

SEMANTIC PROFILES

Does all this mean that the profile for Crazy Characters will always be the same?

No. The profile is likely to be different each time it is delivered. We have analysed the lesson plan as though it is delivered to the letter of the plan. Teachers are likely to change how they deliver the lesson, so the semantic profile will be different each time they deliver it. Similarly, different learners will engage in an activity in different ways. This will mean that each learner experiences a different personal semantic profile based on their own knowledge-building event.

I now have a fledgling understanding of how semantic waves can be used to reflect on and develop teaching activities, and we hope that by sharing our semantic profile of a popular lesson plan, we will help others learn about this approach. [\(LW\)](#)

Semantic profile for the affordances of the Crazy Characters lesson plan – introduction steps

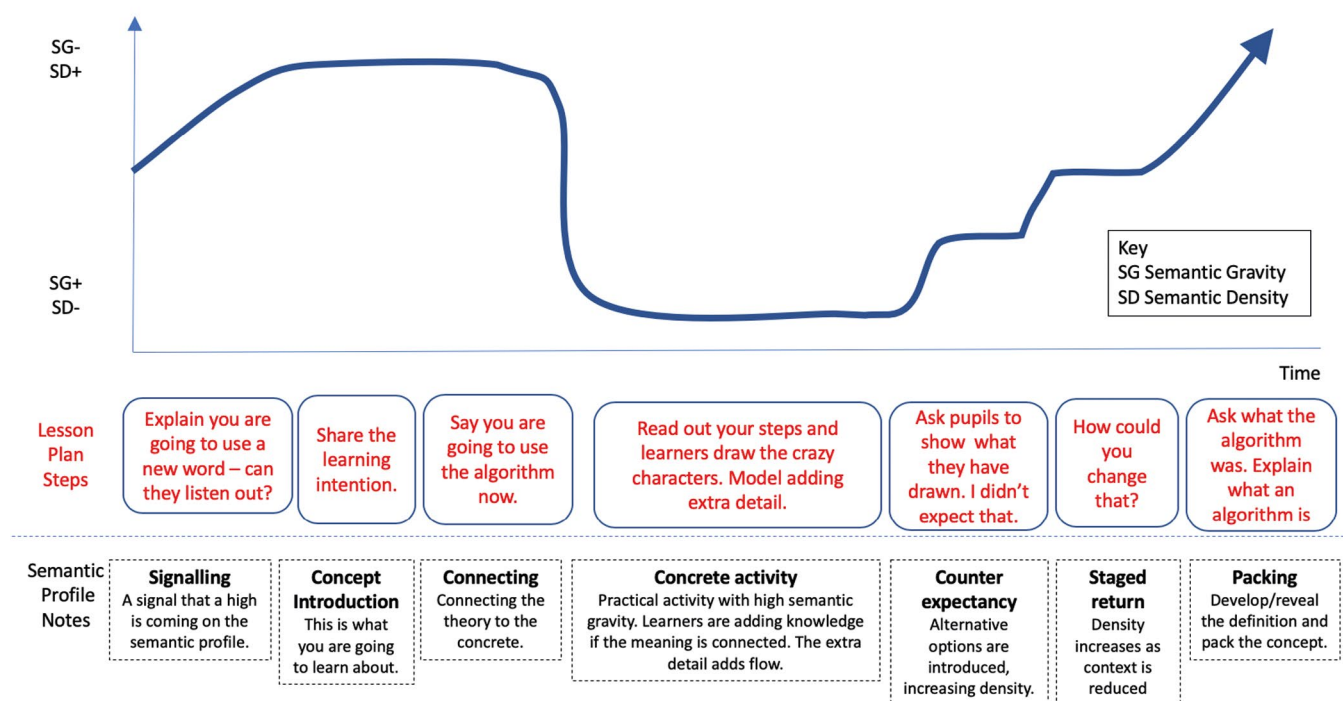



Figure 3 This is the semantic profile for the Crazy Characters lesson plan (introduction only)

Credit: Karl Maton & Jane Waito



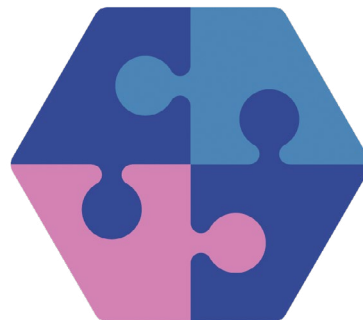
WORK TOGETHER

- 56** PEER INSTRUCTION
 - 58** PAIR PROGRAMMING
 - 60** DISCOVERING COLLABORATIVE PROBLEM-SOLVING
 - 62** COLLABORATION AND COMMUNICATION IN THE COMPUTING CURRICULUM
 - 64** COLLABORATION IN PROGRAMMING PROJECTS
- 

Collaboration is crucial. Not only is it highly prevalent in modern computing professions, but it is also a valuable way for individual pupils to learn from their peers. Working together stimulates classroom dialogue, the articulation of concepts, and the development of shared understanding. Specifically, you can use activities such as pair programming and peer instruction.

Pair programming is an effective approach to programming in which pupils share the cognitive load placed upon them. They work in pairs, with one pupil focusing on the wider problem and the other focusing on the implementation of the code they are working on. Research demonstrates that this approach supports learners in developing their programming confidence and ability.

Peer instruction is another approach to working together. It is a great way of building a shared understanding between your pupils, while identifying and challenging misconceptions. You can support your pupils' development by providing opportunities for conversation through structured group tasks, to enable them to articulate concepts and develop their own shared understanding.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:**
peer instruction
- **What the research says:**
pair programming
- Collaborative problem-solving
- Encouraging communication and collaboration in the classroom
- Version control in programming projects

PEER INSTRUCTION

Using peer instruction in lessons helps students learn, retain, and discuss computing concepts

Peer instruction (PI) is an instructional technique first proposed in the 1990s by Eric Mazur,¹ whose research demonstrated the benefits of focused discussion for pupils' understanding and retention in physics. Subsequent studies have highlighted similar benefits of using PI in teaching

other subjects, including computing.^{2,3} Here, we explore PI and its benefits, look at constructing a good multiple-choice question (MCQ), and give some advice on bringing PI to your classroom.

What is peer instruction?

PI is a teaching approach that combines pre-instruction, MCQs, and peer discussion. Pre-instruction includes reading, videos, and so on, which learners can use to study and become familiar with the material in question before the class. The educator will carefully construct MCQs based on the pre-instruction material. In class, those MCQs are combined with peer discussion to explore and challenge student understanding.

PI is carried out as follows:

1. Learners complete a pre-instruction task (ideally outside class) to help them become familiar with the relevant concepts and knowledge.
2. The teacher poses a carefully selected MCQ. Learners have limited time to individually vote for their answer, using a method such as voting cards, clickers, or raising their hands.
3. Learners then discuss the question and their answers in small groups, aiming for a consensus.

Read the Python program below:

```
a = 1
b = a+1
print(b)
```

What will be the output of the program?

b a+1 2 11

■ An example of a multiple-choice question

4. The teacher displays the same question, and now, learners vote according to their group consensus.
5. Optionally, the teacher shares the results of both votes to highlight where responses have changed.
6. Finally, the teacher leads a class discussion about the question, sharing the correct answer and exploring the incorrect answer options (distractors).

“ MOST PUPILS RECOMMEND PI TO THEIR OTHER TEACHERS

Benefits

While most studies examining PI have so far focused on its use in higher education, the practice offers many benefits which should transfer to other settings:

- Mazur¹ demonstrated that PI leads to significant learning gains for learners: those engaged with PI made up to twice as much progress as other learners. Similar effects have been found in subsequent studies,² which also highlight the importance of the discussion element of PI.
- The same studies indicate that using PI in teaching helps students to retain knowledge.
- Once PI is part of the regular teaching practice, most students value the approach, recognise its benefits, value the discussion, and would recommend PI to their other teachers.²

SUMMARY

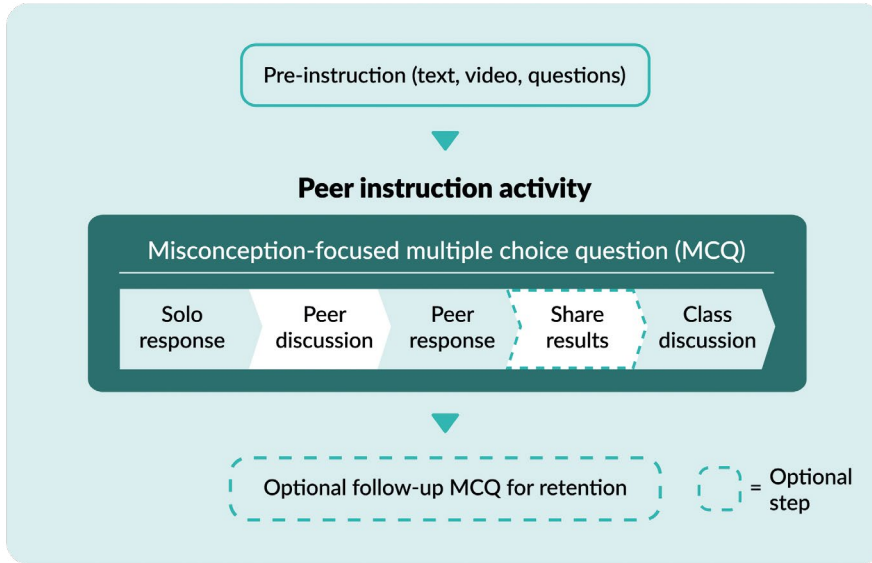
Peer instruction (PI) can replace a traditional teaching approach by combining pre-instruction, multiple-choice questions (MCQs), and peer discussion, to encourage deeper engagement with the content in question.

Benefits

- It is a straightforward approach for educators to apply in their classrooms
- It leads to roughly double the learning gains when compared with no PI
- Learners value the PI approach, especially the discussion element
- Learners are more likely to retain key concepts and knowledge taught using PI
- Peer-led discussion promotes learning

Considerations

- PI should follow some pre-instruction stimulus, ideally before the lesson
- Make sure that learners understand the rationale and benefits of PI
- Always encourage participation over accuracy: PI is a tool for learning, not assessment
- Give learners challenging questions and enough time to discuss them
- Decide whether you want to collect response data, and if so, how



■ The peer instruction process

- PI is fairly straightforward to implement, and evidence shows that even teachers who are new to the practice can quickly see its positive effects.²
- Some researchers cite anecdotal evidence that PI may encourage learners to develop a growth mindset.⁴

What makes a good MCQ?

Carefully constructed MCQs are a key aspect of PI. Good-quality MCQs are harder to write than you would think, as teachers have to predict the misconceptions their learners are likely to hold. For some topic areas, there are lists of known misconceptions; for others, teachers need to rely on their experience. While there are no definitive rules for developing MCQs, these are some guidelines:⁵

- Questions should be clear and unambiguous
- Each question should test only one concept
- Learners should be able to answer questions quickly
- Teachers should learn something from each incorrect response
- It shouldn't be possible to answer correctly while still holding on to a misconception

The image on page 56 shows an example of an MCQ. Can you identify the correct response and explain what might lead learners to select the incorrect responses?

Considerations for applying PI

- For many teachers and learners, classroom PI represents a change in practice. It is important to be clear about the purpose of this approach and how it can benefit learners.
- PI isn't an assessment tool, but a means of instruction. Educators should shift the focus away from getting the correct answers, and instead, promote the participation and discussion aspects of the technique.
- A PI activity should be given as much time as possible. This is especially important for the discussion step, which should last at least two to four minutes.¹ This can feel like a long time, but it is time well spent.
- If using an online voting system — such as handheld clickers or web-based quizzes — the recorded data can be helpful in predicting which learners may require extra interventions.
- Questions should be challenging enough to promote discussion. Mazur suggests that the best results are seen where 50 percent of learners get the initial question wrong.¹
- Pre-instruction is important. With older learners, a flipped approach is best, in which students are introduced to the learning material before the class. They prepare by reading, watching a video, or similar. Where home learning is not possible, PI activities should build on

REFERENCES

- ¹ Crouch, C. H., & Mazur, E. (2001). Peer instruction: Ten years of experience and results. *American Journal of Physics*, 69(9), 970-977. helloworld.cc/peer1
- ² Porter, L. et al. (2016). A Multi-institutional Study of Peer Instruction in Introductory Computing. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. New York, ACM. 358-363. helloworld.cc/peer2
- ³ Simon, B., & Cutts, Q. I. (2012). Peer instruction: a teaching method to foster deep understanding. *Communications of the ACM*, 55(2), 21-29. helloworld.cc/peer3
- ⁴ Simon, B., Hundhausen, C., McDowell, C., Werner, L., Hu, H., & Kussmaul, C. (2019). Students As Teachers and Communicators. In: Fincher, S. & Robins, A. (eds.) *The Cambridge Handbook of Computing Education Research*. Cambridge Handbooks in Psychology. Cambridge, Cambridge University Press, 827-858. helloworld.cc/peer4
- ⁵ Barton, C. (2017, November 29). What makes a good diagnostic question? *Medium*. Available from: helloworld.cc/peer5. helloworld.cc/peer5

previous lessons, or even on content studied earlier in the lesson.

Where to start

If you'd like to try PI in your classroom, consider the following tips:

- Review your content and highlight opportunities for pre-instruction. Consider what learning can be moved outside the classroom to allow for discussion time during the lesson.
- Review and trial some existing MCQs (helloworld.cc/computingMCQs) using PI to diagnose some of your learners' misconceptions.
- Write your own MCQs, describe the misconception that each answer addresses, and share the questions with other educators.
- Encourage learners to deepen their understanding of a topic by writing their own MCQs.
- Visit peerinstruction4cs.com for more guidance and resources. (HW)

PAIR PROGRAMMING

Educators can use pair programming to support learners in producing better solutions to complex programming problems

Pair programming is a pedagogical approach that you can use in your classroom; it involves two learners working together on a problem to develop programs. Pair programming was first used in the software industry and later came to education as the observed benefits became clear.

What is pair programming?

The application of this concept is more structured than simply asking two learners to work together. It involves specific driver and navigator roles. Pairing learners without giving guidance as to how you want them to work together can often lead to one, or both, learners quickly losing focus. Ideally, both learners should be engaged and contribute

equally to the task. Poor communication can be detrimental to the pair's collaboration and can cancel out the benefits of pair programming. Therefore, an essential part of making pair programming a success is ensuring that learners have a good understanding of the driver and navigator roles that they will fulfil during the task.

The driver will control the keyboard, mouse, or pen, depending on the task. They will type the code, or write out the algorithm as instructed by the navigator. These tasks have a low-level cognitive demand for the learner and allow them to concentrate on writing code accurately, rather than also having to focus on tasks such as problem-solving, deciphering the instructions, and algorithm development.

The navigator will support the driver, watching with a keen eye for any errors being made. The navigator will also play a strategic role by thinking of alternative solutions to problems, reading the notes from the teacher, or even walking around the class to look at what others are doing. These tasks have a higher cognitive demand than the tasks of the driver, but as the navigator doesn't have the responsibility of having to write the code, the load on each member of the pair is reduced.

Learners choose, or are assigned, an initial role, and once the task has started they swap roles regularly — approximately every five to ten minutes, depending on the activity. This will make sure that everyone is playing an equal and active role, and that learners are encouraged to both take ownership of the problem that they are solving, and to think in different ways.

SUMMARY

Driver/navigator

- Learners take turns playing the role of the driver and the navigator, swapping roles at regular intervals
- The driver controls the keyboard and mouse and writes the code
- The navigator focuses on the wider aims of the task, spots errors, problem-solves, and reads out instructions to the driver

Benefits

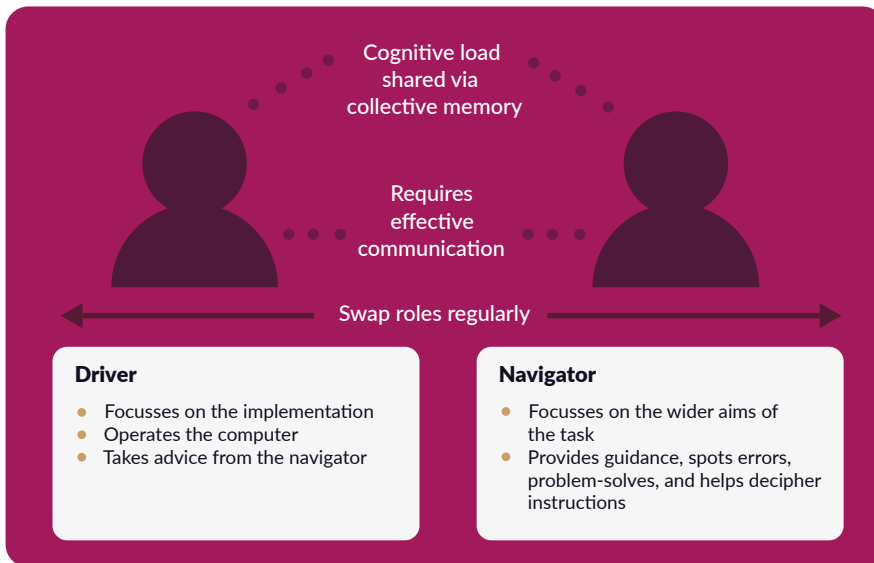
- Reduction in individual cognitive load via the collective working memory effect (see page 20)
- Improved confidence in finding solutions, particularly among female students
- Improved quality of programs (fewer errors, more efficient and elegant code)
- Retention of learners' interest in the activities, lessons, and subject

Key considerations

- Communication between driver and navigator is key: spend time modelling, emphasising, and rewarding these skills
- Spend time ahead of the lesson carefully planning the pairings based on skills, personalities, or friendships
- Ensure that the driver and navigator are always working on the same task at the same time
- Experiment with length of time spent in each role to suit your learners' needs
- Ensure that summative assessment is based on paired and individual work/tests, with a greater weighting given to individual work
- Check that both members of the pair are fulfilling their roles, and do not allow one person to dominate

Benefits

Several benefits of pair programming have been observed in a range of studies. Through pair programming, the individual cognitive load of both the learners is reduced, because the tasks to complete are shared between them. This is known as the collective working memory effect (see page 20). Pair programming "separates tasks with low-level demands (typing, computer management, and navigation) from tasks with higher cognitive demands (syntax analysis, algorithm development and problem search)".¹ However, poor communication between learners can create additional cognitive load, which could eliminate the benefits of this effect. Modelling and developing strong communication skills is therefore key (see the 'Pairing learners' section).



■ Pair programming requires specific driver and navigator roles

Another benefit of pair programming is the likely improvement of the quality of the programs produced by the learners. The learners support each other by debugging, spotting syntax errors as they occur, and making their code more elegant and efficient.

Although most studies conducted so far have been with university students, the results suggest that pair programming has its biggest impact with learners who have less advanced skills and lower confidence, or with groups of learners who are studying introductory courses in programming.²

Research shows that pair programming benefits all learners. However, there is some evidence that suggests that the technique has a greater impact on girls. In studies conducted on learners taking foundation programming courses in higher education, Werner et al. reported a significant increase in confidence levels reported by the women who were paired, compared with those who worked independently.³ Similar findings by Braught et al. showed that women who worked alone were more frustrated than women who worked in pairs.⁴

While evidence shows that pair programming can benefit girls in terms of results and their perception of the subject,

there is no evidence to suggest that it has a negative impact on boys. Hanks et al. found that female students had more positive impressions of pair programming than their male counterparts, but the differences were not statistically significant.² Allowing female learners to work together might help maximise some of the benefits of this approach.

Practical considerations

Pairing learners: As an educator, you will need to use your professional judgement to choose the best pairings in order to optimise the benefits of the collective working memory effect.¹ Key factors that could be considered when creating pairs include the following:

- Learners' personalities and their degree of comfort working together should be considered for sustained or complex tasks, as the pair will benefit from their established relationship.⁵
- Many studies advocate focusing on the skill sets of the learners when pairing. While there is no consensus from the research as to which skill-based pairings are most successful, it is good to start by pairing learners with more advanced skills with those with less advanced skills.

REFERENCES

- ¹ Sands, P. (2019). Addressing cognitive load in the computer science classroom. *ACM Inroads*, 10(1), 44-51. helloworld.cc/pair1
- ² Hanks, B. et al. (2011). Pair programming in education: a literature review. *Computer Science Education*, 2(2), 135-173. helloworld.cc/pair2
- ³ Werner, L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *ACM Journal of Educational Resources in Computing*, 4(1). helloworld.cc/pair3
- ⁴ Braught, G., Wahls, T., & Eby, L.M. (2011). The Case for Pair Programming in the Computer Science Classroom. *ACM Transactions on Computing Education*, 11(1), 1-21. helloworld.cc/pair4
- ⁵ Preston, D. (2005). PAIR programming as a model of collaborative learning: a review of the research. *Journal of Computing Sciences in Colleges*, 20(4), 39-45. helloworld.cc/pair5
- ⁶ Werner, L., & Denning, J. (2009). Pair Programming in Middle School: What Does It Look Like? *Journal of Research on Technology in Education*, 42(1), 29-49. helloworld.cc/pair6

Whichever method of pairing you opt for, it is important to check in regularly with pairs to ensure that they are working well.

Assessment: Learners should be assessed on both their paired work and their individual work. It is not recommended that any summative assessment be based solely on the work that they complete as a pair. Preston⁵ makes two recommendations for assessment to encourage individual accountability with pair programming:

- Assessment should require students to develop code, interpret code, or both
- Assessment scores for individuals should be weighted more heavily than the joint project score when determining the final grade

Further advice and guidance on pair programming can be found in a paper focused on middle schools, by Werner and Denning.⁶ [\[HW\]](https://helloworld.cc/pair6)

SAMANTHA BALORO ASSISTANT EDUCATIONAL PSYCHOLOGIST

DISCOVERING COLLABORATIVE PROBLEM-SOLVING

Could the future of learning be working together?

Instead of teaching in routine ways, imagine basing lessons on open-ended questions such as “How much paint is needed to paint a classroom?”, with the aim of developing collaborative skills and problem-solving. Collaborative problem-solving (CPS) is loosely defined as a group of people working together on a shared problem. A recent report commissioned by Nesta, and written by academics at UCL, argues for a greater use of CPS in education.

CPS brings together individual problem-solving and the process of collaboration, and is one of the most important skills needed by this generation. However, it can be very easy for collaboration to become a one-sided affair, with one person taking over. The success of the collaborative activity depends on the skills and attitudes of learners in relation to each other, as well as on the type of activity.

The Nesta report states that there are five essential features of successful collaborative learning:

- **Positive interdependence**
Learners work harmoniously together, without one member taking over
- **Promotive interaction**
Learners support each other when completing tasks
- **Individual accountability**
Learners are committed to their section of work, and feel responsible for the group’s success in the task
- **Interpersonal and group skills need to be developed**
Learners won’t necessarily have or use high-level collaboration skills
- **Group processing**
Learners reflect on their working relationship, and consider how it can be improved jointly and individually

Studies have found that engaging in collaborative group-based learning promotes positive attitudes to schooling, and improves the social climate of classrooms. It also results in learners being more actively engaged in their learning and having higher levels of motivation. The reason for this change in students’ attitudes to learning may be because engaging in CPS involves students being able to:

- 1 Articulate, clarify, and explain their thinking
- 2 Listen to ideas from others, which in turn leads to developing understanding in areas that were previously unclear to them
- 3 Resolve conflicts by presenting counter-explanations, evidence, and arguments to others, as well as actively engaging in the construction of ideas and thinking in order to co-construct knowledge



Credit: stock.adobe.com/New Africa



Current research in UK schools has found that, although learners are often seated in pairs or groups during school activities, it is rare that active collaboration occurs in ways that are cognitively beneficial. When examining the amount of collaboration occurring in different subjects, it was also found that maths and humanities students were less likely to participate in collaborative work than science students.

There might be a number of reasons for the scarcity of meaningful collaborative problem-solving. Here's a shortlist of the potential issues that have been considered in the research:

- The gap between CPS and the current national curriculum in England, which focuses on exams
- The high workloads teachers are faced with
- Teachers being hesitant to practise CPS because they will have less control over learners
- Teachers not having enough training or confidence to teach CPS
- Students not enjoying working in groups

The success of CPS also depends on how teachers organise, engage, and set up tasks, as well as how they support groups. This is very often difficult to get right, as nobody can assume that just by putting people in groups, collaboration is naturally going to occur. It is important to avoid being too directive, as this can negatively affect group collaboration. When facilitating CPS, teachers should monitor group interactions while asking open-ended questions to challenge students and encourage them to reflect on their views.

Behaviour management is another aspect that needs to be considered. CPS can lead to increased noise and disagreements among learners. It is important to introduce CPS activities in the classroom gradually and frequently, so that learners and teachers can develop the skills needed for productive CPS, such as self-control and productive conflict

“

ENGAGING IN COLLABORATIVE GROUP ACTIVITIES PROMOTES POSITIVE ATTITUDES TO LEARNING

resolution. Most importantly, it is not possible to implement collaborative problem-solving without the active support of a school's senior leadership team. This involves ensuring that they understand the importance of CPS and allow teachers sufficient time to attend training and embed it into practice. In short, it's not an approach that can be rolled out without consideration.

But it's worth bearing in mind that the research shows that when done right, CPS can significantly benefit children's cognitive attitude to learning, and can help considerably when it comes to their development of crucial skills such as teamwork and problem-solving. Implementing it involves a tricky balancing act, and more research is needed on collaborative problem-solving and its application in the classroom.

The full report is available at helloworld.cc/CPS. Nesta has also completed an exploratory pilot looking at collaborative problem-solving discussion in the classroom, available at helloworld.cc/CPSdiscussion. (HW)

Samantha Baloro is an assistant educational psychologist. She completed the Raspberry Pi Foundation research internship in 2017.

COLLABORATION AND COMMUNICATION IN THE COMPUTING CURRICULUM

Neil Rickus examines ways to provide opportunities for talk, and to enable pupils to collaborate effectively in computing lessons

The English computing curriculum states that pupils must be able to create content and express themselves. This can lead to a range of collaboration and communication opportunities. In particular, Reid and his colleagues outline that having technology available enables pupils to develop a variety of social skills, such as communication,

negotiation, problem-solving, and collaboration (helloworld.cc/reid02). To engage pupils in computing lessons and ensure coverage across the curriculum, Computing at School says there's a need to undertake creative projects in a "fun and collaborative environment" and, when tasks undertaken are relevant to the pupils' interests, they can

undertake "exploratory talk" (helloworld.cc/mercer08) in the safe and secure environment of the classroom. So, how can we provide opportunities for collaboration when teaching computing?

Computational thinking and program development

Barefoot Computing defines computational thinking, which underpins the computing curriculum, as a combination of six concepts and five approaches, with each area providing opportunities for pupils to work together. Barefoot's approach to collaboration could involve a range of pedagogical techniques, such as paired programming, which allows pupils to develop their programs together, with one pupil solely using the input devices, while the other pupil focuses on the required instructions. Collaboration also complements a number of other computational thinking approaches, such as debugging and persevering, which may not be as effective if approached independently. Teachers have also noted the "positive motivational impact" that collaborating on programming tasks has on individuals and on the class as a whole (helloworld.cc/sentence15).

In DT, the "engineering design process" (helloworld.cc/bers10) is often followed,



©Image is licensed under CC0 and was taken by Lucifera Ribeiro

■ Pupils discuss bugs, or errors in their program's code, while working together at the computer

with the Imagine and Plan stages allowing children to articulate their ideas and share their thoughts with others. Such a process is increasingly being used when implementing physical computing projects, which ensures pupils have carefully thought about both the physical and programming elements of their project before beginning work. The process also allows children to regularly evaluate and improve their work, which, as Dawes and her colleagues state, lets them “talk [the outcome] into existence” through the various iterations of their project.

Teacher input

When teaching programming, teachers need to ensure they focus on the concepts being taught, rather than on the hardware or software being used, as pupils can become disengaged if they keep using the same technology on an ongoing basis. By using technical language, pupils become more familiar with technical terms, which is essential as they produce more complex projects and move to using text-based programming environments (helloworld.cc/kolling15). When working with others, this can also allow pupils to articulate clearly how their programs function, and it can also facilitate debugging.

When using paired programming and other collaborative teaching approaches, (such as C3B4ME, in which pupils have to ask three peers for assistance before asking the teacher), teacher modelling is needed, to demonstrate to pupils how they should interact (helloworld.cc/bird14). This enables children to ensure they empathise, listen, and potentially continue their conversations in more depth, rather than the exchange ending in a dispute (helloworld.cc/littleton13). Teachers with limited experience of delivering computing lessons can use these pedagogical techniques to act more as a facilitator, rather than a knowledge bearer, and can help to avoid them being inundated with “lazy questions” (helloworld.cc/bird14). However, this process needs to be carefully managed, to ensure more able pupils aren’t continuously disturbed during lessons, and that the lesson is challenging for all pupils.

Block-based programming environments can facilitate further speaking and listening

TALK DURING COMPUTING

Speaking and listening activities in computing lessons need to be carefully managed. Giving your learners too much freedom can lead to off-task behaviour, while limiting collaboration reduces opportunities for pupils to learn from each other. We work carefully with pupils to model the appropriate language for talk, and encourage them to use technical vocabulary where possible.



activities. For example, most environments allow sound to be recorded for inclusion in pupils’ programs, and a spoken narrative can often be recorded while the program is on the screen.

Introducing IT elements

Unplugged computing activities enable pupils to develop their understanding of computational thinking without the need for technology. Many unplugged activities, such as Phil Bagge’s Sandwich Robot (helloworld.cc/sandwich), require pupils to limit their vocabulary choices and give precise, unambiguous instructions, while developing their understanding of the computational thinking areas of abstraction (removing unnecessary detail), evaluating, and collaborating. These activities also provide opportunities to introduce other technologies into lessons, such as the camera on a tablet device, which pupils could use to film their peers in role to assess the quality of their instructions, and to aid debugging.

The use of IT to record audio and video can also enhance other areas of the computing programme of study. E-books containing multimedia content created by the pupils provide opportunities to share their work with an audience beyond the classroom and, when considered in conjunction with Papert’s theory of constructionism, this use of technology is likely to enhance pupil outcomes by providing an audience for their work. When pupils are including information gained from their own online research, the most

successful pupils use “exploratory talk” (helloworld.cc/knight15) to assist with “sorting out his or her own thoughts” (helloworld.cc/mercer08).

Other recording technologies, such as talking picture apps, which allow recordings to be synchronised with the movement of a character’s lips, can enable pupils to demonstrate their understanding of curriculum areas without the barrier of having to produce written content. For certain pupils with special educational needs and disabilities, particularly those with autism, the use of a computer can even help to reduce anxiety and effectively aid communication, according to the Autism Education Trust.

So, how can you provide more opportunities for pupils to talk and collaborate when teaching computing? Let me know your thoughts via Twitter [@computingchamps](https://twitter.com/computingchamps). (HW)



NEIL RICKUS

Neil is a senior lecturer in computing education at the University of Hertfordshire. He is a CAS community leader and a Raspberry Pi, Google, and Microsoft Certified Educator ([@computingchamps](https://twitter.com/computingchamps)).

COLLABORATION IN PROGRAMMING PROJECTS

Stefan Seegerer, Tilman Michaeli, and Jane Waite introduce Smerge, a free version-control system for the block-based programming language Snap!

Getting pupils to collaborate effectively on programming projects is tricky. We often can't see how much work each pupil has done, and it's hard to interrupt the flow of work to provide constructive feedback. It's also difficult to bring the reality of the world of work, and how programmers work in industry, into the classroom. Smerge is a free Snap! add-on developed by a research team in Germany that helps tackle these issues.

What is a version-control system?

Version control is crucial in real-world software development. Whether you are working in a small or large team, collaborating face to face or remotely, knowing where you

are up to in the creation and modification of your code is essential. Learning about version control and its underpinning concepts as a core computational practice is important, as is learning how to use it. However, professional tools are extremely complex, and even professional software developers can have problems with them.

Version control for Snap!

Smerge is an easy-to-use, beginner-friendly online version-control system for the Snap! block-based language. Smerge enables students to work together collaboratively to develop programs. Teachers and peers can provide feedback on specific versions of a project, giving hints, tips, and suggestions for improvement, and they can do this in class or at home.

The design of Smerge was based on a review of professional version-control systems and how they were used in computer science classrooms. While being tailored to novice programmers, Smerge still includes all core concepts of professional version-control systems, which are:

- **Project history:** In Smerge, the project and its history are visualised in a project history diagram (a graph).
- **Committing:** Changes are added to a project by committing them to the version-control system. Changes are committed to Smerge by simply clicking on an add-on Snap! block (**Post to smerge**).
- **Branching:** Branching opens an alternative path so that changes can be made in parallel. A branch might be used for developing new features or fixing bugs, without impacting the current state of the project. This is where Smerge differs from many existing systems. Each person

editing the shared program is provided with their branch automatically.

- **Merging:** A version-control system simplifies combining changes and the resolution of conflicts. For merging, users select which nodes they want to merge in the project history diagram and confirm their selection. When possible, conflicts are resolved automatically by Smerge. If they cannot be resolved automatically, the conflicting code fragments are shown in a merge view in Snap! for the person editing the program to sort out.
- **Data backup:** Having all files backed up to version control allows for returning to every (older) version easily, and thus enables risk-free trial and exploration of different ideas. In Smerge, old versions can be accessed using the graph.

How to Smerge

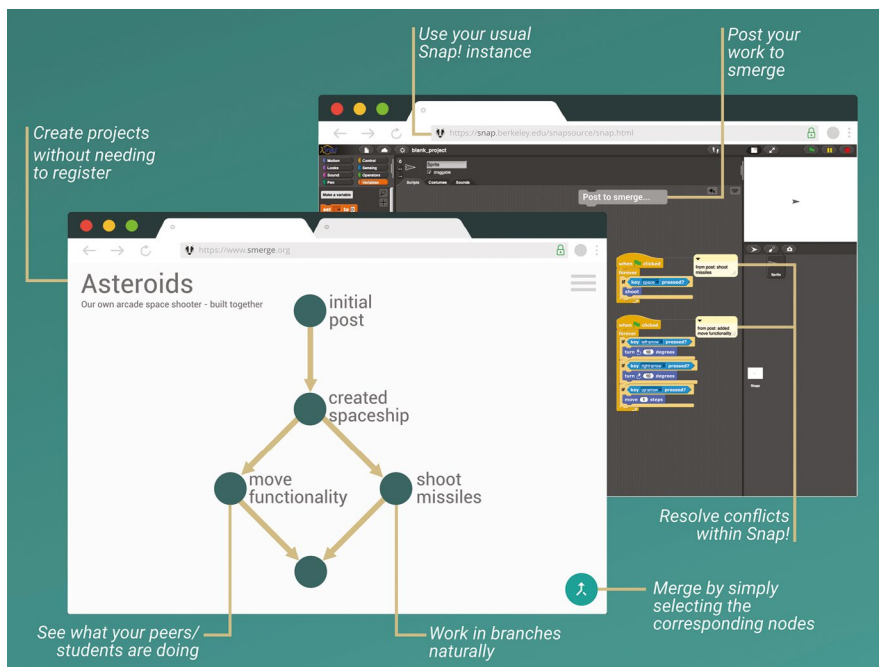
To get started with Smerge, the teacher or students create a project on smerge.org. At no point do they need to register on the Smerge system. Students might start from a given template, or a plain Snap! project. Imagine that a team wants to create a game such as Asteroids, the classic arcade shooter game. One student can develop the functionality to move the spaceship, and another to program the feature to shoot missiles.

To start implementing this in Snap!, each of the students just double-clicks on the node they would like to edit. Every student is automatically provided with their own branch to work on. Then they just use their usual Snap! window for programming. When they have finished adding their feature, they post their program back to Smerge by using the custom **Post to smerge** Snap! block, found on the variables palette.

USEFUL SMERGE FEATURES

Smerge was created for classrooms and offers teaching and learning features:

- Enables collaboration on project-based learning with Snap!
- Teachers can supply custom sample code
- Teachers and students can provide feedback or marking and suggest alternative ways forward
- Helps teach industry version-control skills
- Helps students think about the development process by making it more transparent
- Students can see the version history, revert changes, and go back to old versions
- Teachers and students can easily share projects with no need to register and no personal information stored



■ Smerge makes version control simple

One of the students then activates the 'merge' mode in Smerge. This is done by clicking the 'merge' button. The student selects the two nodes to be merged and confirms the selection. The code will be automatically merged, resulting in a new version. If more than one student changed the code for the same script in the same sprite, both versions of that changed script are retained by Smerge and both are shown in the new merged code. This enables the students to discuss the conflict, test the two options, and choose which version to keep.

Project ideas

You can use Smerge for any Snap! project. Here are some ideas to foster collaboration:


- **Animal dance party:** The teacher creates a starter program, with one dancer and music. Every student then adds their own dancer to the party.
- **Celebration card:** In this project, students collaboratively create a celebration card, such as a birthday card. Every student creates one letter, and the letters are shown, one after the other. This involves students working out how to coordinate the appearance of each sprite.
- **Quizzes:** The teacher provides a starter quiz and students add extra questions. Lots of testing might be needed here to check that

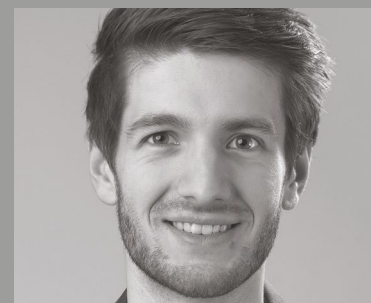
shared variables, such as a score, are used in the same way by all contributors.

- **Animated story:** Students storyboard the animation of a story, figure out what work needs to be done, and share the work between group members.
- **Games:** Last but not least, students can collaboratively create games, as each student focuses on its different parts.

Pedagogy ideas

As well as modelling how Smerge works, try a Use-Modify-Create approach. Create an example Smerge project and ask students to explore the project history. Then ask them to take one branch to make their own version. Ask the students to work in pairs to compare their versions and discuss a potential combined version. Encourage using the 'merge' mode, and compare the results with their expectations. You could start more simply, introducing collaboration later on.

Using Smerge in the classroom provides many valuable opportunities to address important aspects of collaboration and program design. When using version-control systems, students learn to decompose functionality sensibly, create reusable functions, think about interfaces or test data, and meaningfully name sprites, assets, or descriptions of what has been changed. Smerge is free to access at smerge.org. 



STEFAN SEEGERER

Stefan is a researcher, developer, speaker and educator. He works at the Free University of Berlin, Germany, exploring ways to make computing accessible to everyone (@StefanSeegerer).



TILMAN MICHAELI

Tilman is a researcher at the Computing Education Research Group at the FAU in Germany. He works on projects such as developing concepts for the classroom supporting debugging skills, and fostering collaboration for programming projects (@TilmanMichaeli).




JANE WAITE

Jane is a research scientist at the Raspberry Pi Foundation. Her current interests include using design in primary programming, semantic waves, PRIMM, and migrating to online teaching using ABC (@janewaite).



READ AND EXPLORE CODE FIRST

- 68** CODE TRACING
 - 70** READ BEFORE YOU WRITE
 - 72** ASSEMBLY LANGUAGE ON
THE PI: “LEARNING HOW TO
WALK AGAIN”
- 

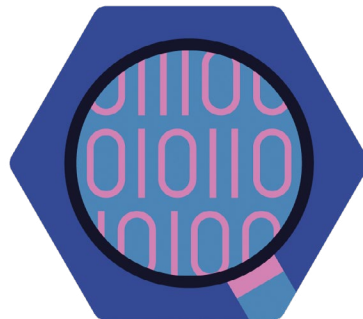
T

here is no doubt that programming can be a highly rewarding and satisfying experience for everyone.

However, you should be in no rush to have your students write their first independent program; in doing so, you may miss out some important steps.

We typically wouldn't ask pupils to write before they had learnt the basics of reading, or encourage them to write number sentences before they had learnt to count. Likewise, there is a body of evidence that suggests that engaging pupils in reading code that other people have written before they create their own code can enhance and improve their ability to write better code later on.

Students should be able to review, interpret, understand, and manipulate code. This approach applies to all sorts of programming experiences, whether they are text-based (for example, Python) or block-based (for example, Scratch).



IN THIS SECTION, YOU WILL FIND:

- **What the research says:**
code tracing
- Evidence-based approaches
for helping pupils to read code
- Teaching assembly language
on Raspberry Pi to older
pupils

CODE TRACING

Reading code before writing it is vital, and code tracing is an ideal activity to develop learners' program comprehension

Developed in the early 2000s, code tracing is a well-established approach to help learners build their program comprehension. Put simply, it involves reading and analysing code before running it, to predict its outcome. Novice programmers should be competent in code tracing before they can confidently write programs of their own. Here, we explore code tracing, its benefits, how it fits in with the concept of the notional machine, and how you can use code tracing in the classroom.

What is code tracing?

It is widely understood that young learners should ideally have developed some reading skills before they begin learning to write. Similarly, in computing, there is a body of evidence to suggest that code tracing, a form of code reading, is an

effective precursor to code writing and independent programming.

When tracing code, learners review chunks of code, or whole programs, and record their expected behaviour and execution flow at various stages. They can capture this through annotation as well as by recording the program output at each stage. You may ask learners to trace a piece of code, predict the outcome, and then guide them through the code, line by line, to test their prediction. Typically, you'll ask learners to predict away from the computer, to ensure they focus on reading rather than executing the code. You could also give

basic level, followed by explaining the code, and finally progressing to writing. Many other studies have been completed based on this theory. Hertz and Jump, who developed the trace-based teaching model, found that starting a class with 20–30 minutes of tracing increased attainment and decreased dropout rates.³ A 2004 study found that learners who could trace effectively less than 50 percent of the time could not explain the code effectively.⁴ Code tracing can help to reduce the cognitive load placed on learners. By focusing learners' efforts on existing and working programs, and by answering specific

STARTING A CLASS WITH 20–30 MINS OF TRACING INCREASES ATTAINMENT AND DECREASES DROPOUT RATES

learners short sections of code in the form of worked examples (see page 90), or ask them to complete trace tables where some values are provided, and they can use code tracing to record the missing values. Then, with this secure understanding, you can allow learners to create their own programs featuring the concepts they have traced. While there is no single approach to tracing, there are some clearly defined methods, such as TRACS¹ tracing, which may be helpful for learners to follow.

Benefits

Harrington² identified that when learning to program, learners build their understanding in a hierarchical way. Tracing is at the most

questions, educators can avoid unnecessary extraneous load being placed on learners. If learners have the opportunity to trace code, they can comprehend the code and its function before seeing it in action. Many other subjects explore similar ideas, such as the Talk for Writing framework in literacy, and progressing from concrete objects to abstract numerals in mathematics. This approach also helps develop learners' understanding of the notional machine — how the code is executed.

If we accept that there is a broad consensus advocating code tracing as an effective strategy with a range of evidence to support the claim, what should we consider when using it in the classroom?

SUMMARY

Tracing involves:

- Reading the code
- Interpreting the meaning
- Recording the flow and/or outputs

Benefits of tracing:

- Fosters program comprehension
- Improves code writing
- Supports learners in analysing and explaining code
- Exposes misconceptions
- Reduces cognitive load
- Helps learners develop a consistent notional machine

Variables table

counter	
1	
2	
3	
4	

What is output?

Happy days
Happy days
Happy days
End of program

▲ In the example above, elements of the program flow and code are provided. By completing the variables table, learners can demonstrate their understanding of the program through code tracing.

The notional machine

When tracing code, learners apply their current understanding of how a machine works: their notional machine. This concept was first introduced by Benedict du Boulay and describes the conceptual model that learners have about how a computer processes instructions and data.⁵

The notional machine can look very different depending on the type of programming language being used. In Scratch, it is simple to run more than one process concurrently (threading), whereas in most text-based languages (including Python), this is more complex. This has implications for how we begin to teach programming in Scratch. Learners may demonstrate that they can use threads, but may not understand how the machine handles them. This gap in their notional machine understanding can lead to gaps

in their knowledge, or to misconceptions. If you encourage learners to use threads in Scratch without addressing the notional machine, it may lead to problems later when learners find threading more difficult to achieve in Python.

In-context application

You can incorporate code tracing in the classroom as a standalone activity, or as part of a broader approach:

- The PRIMM (Predict–Run–Investigate–Modify–Make) approach is ideally suited to tracing, as PRIMM requires learners to Predict as its first step, which involves reading and tracing (see page 22).
- You can begin a programming activity or project by providing learners with an existing project or snippet of code for them to trace.

REFERENCES

- ¹ Donaldson, P., & Cutts, Q. (2018). Flexible low-cost activities to develop novice code comprehension skills in schools. *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*, 1-4. helloworld.cc/tracing1
- ² Harrington, B., & Cheng, N. (2018). Tracing vs. Writing Code: Beyond the Learning Hierarchy. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 423-428. helloworld.cc/tracing2
- ³ Hertz, M., & Jump, M. (2013). Trace-based teaching in early programming courses. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 561-566. helloworld.cc/tracing3
- ⁴ Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4), 119-150. helloworld.cc/tracing4
- ⁵ Du Boulay, B. (1986). Some Difficulties of Learning to Program. *Journal of Educational Computing Research*, 2(1), 57-73. helloworld.cc/tracing5

- Tracing is also a great way to check learners' understanding of the capabilities of the notional machine. Using examples in which specific misconceptions may lead to an incorrect solution, tracing can expose and help address this misconception.

If you would like to try some code tracing activities, the Teach Computing Curriculum 'Programming' units include examples (helloworld.cc/tcctracing). (HAW)



■ It's important to give students the ability to read and understand code before they start writing it

READ BEFORE YOU WRITE

Will Grey shares some evidence-based approaches for helping pupils to read code

Before students can write code, they need to be able to read code. Computer science pedagogy is often based around the ideas of Piaget's constructivism (in which pupils develop their knowledge through exploration) and Papert's constructionism (in which pupils learn through creating artefacts). However, learners need guidance to gain useful knowledge efficiently, and to organise that knowledge in a clear and logical way. They need to be able to break a problem down, remove the unnecessary detail, find patterns, and think algorithmically before they can start to write programs for solving problems.

Just as we wouldn't expect a young child to write prose before they can read, we need to provide guided approaches that

use direct instruction and scaffolding to help our students read code before they can be expected to write code themselves. These guided approaches are needed just as much as, if not more than, creative discovery activities.

Explain the code

My first approach to improving code comprehension is to ask my pupils to explain in plain English what a piece of code does. There are many variations on this activity, but I find that it works well when pupils explain to each other what the code does. That way, pupils have valuable contributions from others that they can then incorporate into their own written explanations.

We could take this further and ask students to carry out various explorations of

the code. For instance, pupils could annotate or add comments to code, list and explain the purpose of the variables and functions, and create structure diagrams. I particularly enjoy doing this with the UK's AQA A level Computer Science prerelease material — a reasonably substantial piece of code that enables a rich investigatory experience (helloworld.cc/aqaprerelease).

Worked examples

Another approach is to use worked examples (see page 90). They can be delivered by modelling with live coding, or by using tutorial guides. In a live-coding demonstration the teacher explains each step of the code as it is being written. This is beneficial because students can see how teachers tackle problems. Pupils also see that

FURTHER READING

- ✓ Lopez et al. (2008). Relationships between reading, tracing and writing skills in introductory programming. In: *Proceedings of the Fourth international Workshop on Computing Education Research*. New York. 101-112. helloworld.cc/lopez2008

making errors is a normal part of the coding process. Another benefit of live coding is that the pace of delivery is generally slow, and this may help some groups of pupils — I have found it particularly helpful with underachieving boys.

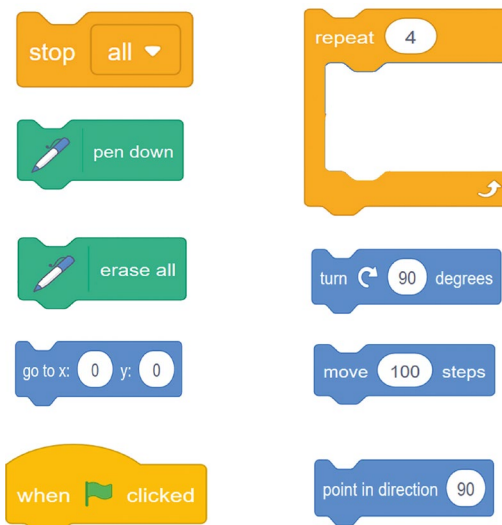
Nevertheless, demonstrations need to be kept short and progress only a few steps at a time, so as not to overload working memory. Lessons should have several shorter demonstrations spaced throughout, rather than fewer longer demonstrations. Perhaps live-coding demonstrations could be prerecorded as screencasts with audio and made available to pupils to watch, pause, and replay in their own time, thereby helping pupils who did not follow the first time around. I tend to couple the demonstrations with tutorial guides because it gives my pupils greater autonomy. I have used worked examples across the age ranges to deliver lessons on Scratch to lower secondary pupils, all the way through to advanced concepts in Python such as server-side scripting with sixth-form students.

Sub-goals

The third approach is to group individual steps under meaningful sub-goal labels. In addition to helping students organise and structure the code in a coherent manner, this reduces the load on working memory because individual steps are chunked together. Pupils could be asked to do this themselves, or sub-goal labels could be provided by the teacher, according to the specific needs of the student.

Trace the code

Tracing code by hand to simulate the outcome of each instruction in an algorithm is another method (see previous article).



■ An example of a Parson's Problem: can you rearrange the code to draw a square?

This is where the student replicates the task of the computer by keeping track and recording the values of the variables at each step, and performing the various arithmetic and logical operations. Pupils find tracing code tricky, as it requires considerable patience and concentration to not lose their train of thought. It takes practice to master, especially when tracing recursive

starter activity, and can come in a variety of forms, from those on a computer to paper-based and jigsaw-type puzzles with snippets of code that can be pieced together —but these do require quite a bit of preparation.

Putting it all together

Once pupils have gained comprehension of the various programming constructs and seen them applied in different contexts, they will be able to write better code. There are many scaffolded techniques that can help students write code by taking away the challenges of code design and abstraction.

These ideas for reading code are generally very simple to implement, and can be used together and alongside methods for writing code and more general teaching pedagogies, such as flipped learning, direct instruction, questioning, and verbal feedback, to deliver effective lessons on coding in visual and textual languages to schoolchildren of all ages. (HW)

“ THERE ARE MANY SCAFFOLDED TECHNIQUES THAT CAN HELP STUDENTS LEARN TO WRITE CODE

algorithms like merge sort. I tend to practise these regularly with my GCSE and A level groups — both with the familiar algorithms for searching, sorting, and traversing, and with algorithms they have not seen before. There are plenty of past paper questions across the exam boards that can be used support this activity (see the previous article for more).

Parson's Problems

Parson's Problems bridge the gap between reading and writing code (see page 80). Here, code is presented to pupils jumbled up. The task is to put the instructions into the correct order so that the whole code performs a predefined task. Students do not need to be concerned with remembering syntax, and can focus on logic and sequencing. They make a lovely



WILL GREY

Will is head of computing at Comberton Village College in the UK, and coordinator of the CAS South Cambridgeshire Community of Practice. He has developed a comprehensive set of resources for A level and GCSE computer science at helloworld.cc/grey.



ASSEMBLY LANGUAGE ON THE PI: “LEARNING HOW TO WALK AGAIN”

Simon Humphreys shares how Raspberry Pi’s ARM processor makes it a great tool for A level students to get their hands dirty with assembly language

A ssembly language programming is on every university computer science course and also in numerous advanced level courses — so it must be important! The instruction set for the UK’s AQA A level is akin to the ARM instruction set used on Raspberry Pi, making it a great tool for bare-metal programming. In this article, I discuss taking a PRIMM (Predict–Run–Investigate–Modify–Make) approach as one way of introducing the topic, and look at how to encourage students to use a real processor to build programs using ARM assembly.

Why learn assembly language?

Programming in assembly is the closest we ever come to the CPU and its architecture, as each assembly instruction is shorthand for the binary code the CPU executes. There is no better way of understanding the role

of the processor, how it executes programs and manages memory, than by attempting some low-level programming using a given instruction set. It helps us to answer questions such as:

- How is an integer stored in memory?
- How does the computer carry out an ‘if-then-else’ statement?
- What happens when one function calls another function?
- How does the computer know where to return to after running a function?

There are many brilliant simulators, especially the new ARMLite Assembly Language Simulator by Peter Higginson, that can help students to take what they learn in that sandbox to a real processor (helloworld.cc/ARMLite). I highly recommend using it, along with Richard

Pawson’s book *Computer Science From the Metal Up* (helloworld.cc/pawsonbook).

If your school has a set of Raspberry Pis, it’s easy to make a start with real-world assembly and give your students insights into how their high-level code is ultimately made to work on a given architecture. Putting into practice some of the fundamental concepts of computer science (for example, binary arithmetic, allocation of memory, working with the stack, character set encoding, or the handling of interrupts) should help to make them better high-level programmers.

We’ll now look at a simple example, following the PRIMM approach to teaching and learning programming, which can be equally applied to low-level languages as high-level. The following overview should take one lesson to cover, with additional time for extensions.

A SIMPLE EXAMPLE (CONTINUED OVERLEAF)

PREDICT

Ask students if they can work out what the following ARM assembly language program does.

```

1.  .data
2.  first: .word 5
3.  second: .word 8
4.
5.  .text
6.  .global _start
7.
8.  _start:
9.      ldr    r0, =first    @ get address of first value
10.     ldr    r0, [r0]      @ get value at that address in r0
11.     ldr    r1, =second   @ get address of second value
12.     ldr    r1, [r1]      @ get value at that address in r1
13.
14.     cmp    r1, r0        @ compare: is r1 > r2
15.     bgt    _greater     @ if true, branch
16.     b      _exit        @ jump over to exit
17. _greater:
18.     mov    r0, r1        @ true, put return code of 1
19. _exit:
20.     mov    r7, #1
21.     swi    0

```

OK, perhaps it's an unfair question if they've never encountered ARM assembly instructions before, but the comments have been left in place for students, and should provide some clues!

RUN

Ask students to enter the code in an editor of their choice and save it as `max.s`, then in a terminal window, enter the following commands (the dollar is the terminal prompt):

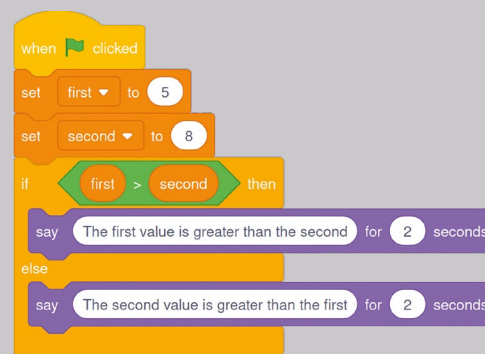
```

$ as max.s -o max.o
$ ld max.o -o max
$ max ; echo $?

```

If all goes well, the value 8 will be displayed (note: the command `echo $?` displays the value in register 0 on the screen).

Does the answer provide any further clue as to the program's purpose? If in doubt, the Scratch example pictured should make it clear.



■ A classic max program using Scratch

```

int first = 5;
int second = 13;

```

Lines 5-6

This is the code section. Our instructions start here and the entry point, `_start`, is given by another label.

Lines 8-21

This is the code for our program. Each line has either a label or an instruction, with labels placed on their own line to help with readability. Each instruction is an operation, or opcode (operation code), followed by the operation's operands.

Lines 9-12

These lines fetch the data values from memory. We start by getting the address of our first value and copying it into register 0, then we fetch the item of data at that address and copy it into register 0 (overwriting the address). The next two lines do the same thing for our second value, using register 1. Note that the direction of operation, `ldr r0, =first`, will copy the address of our data item into register 0, and not the other way around.

The CPU has a number of registers. Some are general-purpose, and others are set aside as special-purpose, for example, the program counter. The ARM CPU has 16 of these, with `r0` and `r1` being two of the general-purpose registers.

Now we have our two values in two registers, where they can be processed by further instructions.

INVESTIGATE

There may be several activities we can do with our students to dig into the code, for example reminding them about the role of the CPU in fetching, decoding, and executing instructions. Here, we'll just deconstruct what is happening in the code from the Predict stage, section by section, line by line.

Lines 1-3

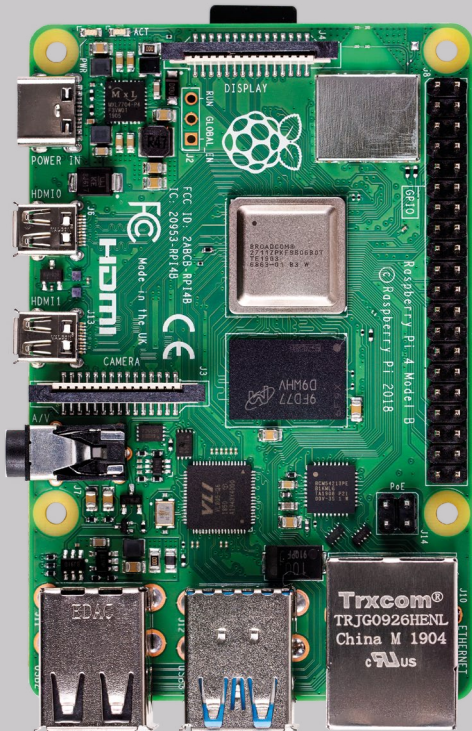
This is the `_data`-section, and here we declare our data. Each is given a label, a size, and a value. The first label, `first:`, will allocate 4 bytes of memory, and assigns the value 5 to that address. The label is a symbolic address, meaning it points to a memory address. The second label does the same thing, but with a different value.

This is equivalent in a high-level language to:

A SIMPLE EXAMPLE (CONTINUED)

Line 14

The two values are compared using the `cmp` instruction. Internally, the processor will subtract one value from the other and update another register, the flags register, based on the outcome of that operation. For example, if the result is negative, the negative flag will be set to 1. Other flags include the zero flag and the carry and overflow flags.



Line 15

The instruction `bgt` means 'branch if greater than'. As high-level language programmers, we've been taught to avoid using the GOTO instruction, but we cannot avoid it in assembly! It's the only way we can force the CPU to jump to a new address and fetch data/instructions from that address. In this case, we jump to the address defined by the label `_greater` if the second value is greater than the first.

Line 16

Line 16 contains another branch instruction. This is unconditional: just jump or GOTO the address specified, `_exit`.

The branch instructions can take a number of conditional commands, e.g. `eq` branch if equal, `ne` branch if not equal, `lt` branch if less than, etc.

The combination of the compare instruction, `cmp`, and a branch, `bxx`, is the equivalent of an 'if

then-else' statement, so either line 16 or line 18 will be executed.

Line 19

We then fall through to the `_exit` section, which ensures we exit gracefully from the program.

Thus, the program is a classic introductory programming problem, returning the larger of two given values.

MODIFY

We can then invite students to carry out a number of potential activities, such as returning the smaller of the two values, or providing three values to compare against each other.

MAKE

Once the students know how to write an equivalent 'if-then-else' construct and how to branch, it becomes relatively trivial to introduce loop constructs such as 'do-while', which have a conditional expression at the bottom of the loop and need to jump back to the top of the loop.



PROGRAMMING IN ASSEMBLY IS THE CLOSEST WE EVER COME TO THE CPU AND ITS ARCHITECTURE

➤ The above exercise does need to come with a health warning! One of my students described programming in assembly as learning how to walk all over again. Assembly does not come with nicely prepared features such as variables, loops, selection constructs, or functions, and even the data types and manipulation of memory are left to us to manage. But stick with it — the pay-off is rewarding.

Where next?

In a short article, all we can do is scratch the surface of such a fascinating area. This example provides many insights into the role of the CPU; how a compiler would take source code and translate it into machine code; how

memory is allocated; and that a variable is just an address of a fixed size. The registers can be inspected using a debugging tool such as GDB (helloworld.cc/gdb), and students can also see the role of the program counter and how it is updated as the program progresses.

Assembly language programming is fascinating, tricky, and irritating in equal measure. For A level, the problems set are usually no more difficult than rewriting a simple high-level algorithm that uses one or more of the constructs such as our example. Taking students further, to build functions or manipulate the stack, provides genuine insights and will make them better high-level language programmers — but perhaps that's for another article! (HW)



SIMON HUMPHREYS

Simon is the vice chair of Computing at School (CAS). He is also a computer science teacher at Hills Road Sixth Form College in Cambridge, UK.

(HW)

SUBSCRIBE TODAY

**FREE
IN PRINT**
for UK-based
educators



**FREE
PDF**

for anyone,
anywhere

Why subscribe?

- Teaching resources and ideas used by over 90 percent of our readers
- Exclusive news, research findings, and in-depth features
- Delivered to your door three times a year

Not a UK-based educator?

- Subscribe to receive the free PDF on the day it is released
- Read features and news at helloworld.cc

TO SUBSCRIBE VISIT:
helloworld.cc/subscribe



FOSTER PROGRAM COMPREHENSION

78 PROGRAM COMPREHENSION
AND THE BLOCK MODEL

80 PARSON'S PROBLEMS

82 HOW TO SUPPORT YOUR
STUDENTS TO WRITE CODE

84 THE I IN PRIMM



W

hen teaching programming, it is important for learners to understand a program from multiple perspectives, including how it is written (syntax and symbols) and how it executes, as well as its function or purpose.

There are many ways in which you might support program comprehension, but particular examples include activities that promote debugging and tracing, and the use of Parson's Problems. Additionally, tasks that ask pupils to consider the purpose of a program (or a small part of a program) are very helpful. Other beneficial tasks include selecting appropriate names for variables, functions, or entire programs; predicting the output of a program; or matching programs to their purpose.

The Block Model is a helpful framework through which to view this notion of program comprehension, as it captures twelve perspectives through which a programmer might view a program. Each of these perspectives is useful, and you should encourage your pupils to be able to use each perspective and move between them.

Embed these comprehension activities into your practice on a regular basis to secure understanding and build stronger connections with new knowledge.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:** the Block Model
- **What the research says:** Parson's Problems
- Teaching students to write code
- The Investigate stage of PRIMM and the Block Model



PROGRAM COMPREHENSION AND THE BLOCK MODEL

Understanding programs before we write them is vital, and the Block Model can help to foster program comprehension

As the name suggests, program comprehension concerns understanding programs and includes not only understanding the program as written, but how the program works, and its purpose. In recent years, program comprehension has been recognised as an important part of learning to program. It is something that can be easily missed, as learners dive straight into writing programs before they have learnt to read them. So, what exactly is program comprehension, why is it so important, and how can educators develop these skills with their learners?

The challenges of programming

Although programming is a valuable and rewarding skill to learn, many learners find the process challenging:

- Even simple programs are rich in concepts that can cause cognitive overload in learners
- Learners may rush to write programs too soon, before they have read and understood the relevant concepts
- Programs often don't work first time, demanding resilience and persistence
- Learners need to switch between different abstractions, the problem, the program text, and its execution,

constantly moving from single lines to the program as a whole

- Learners also need a mental model (a notional machine) for how the computer works and will execute the program

These challenges do not mean that programming is intrinsically difficult, and recognising them can help educators identify where they can support their learners.

Program comprehension

Experienced programmers demonstrate a high degree of program comprehension. As well as having a robust notional machine, they can develop programming 'plans' (chunks of code that perform a specific task), based on common features in programs that they have seen. They can then use these plans or patterns to interpret, explain, adapt, debug, and create programs.

Novice programmers know of very few programming plans, and have limited awareness of how programs are executed. Their focus may be limited to decoding individual words in a program, rather than comprehending their meaning, or the meaning of the wider program. We must understand how to bridge this gap.

Program comprehension tasks

There are many great examples of activities that promote program comprehension, including code tracing, Parson's Problems, PRIMM, and tasks in which learners 'explain the purpose'. A teacher-focused study¹ identified more than 60 different activities that could support learners in developing program comprehension skills. It also highlighted that many of these activities

Macro structure (M)	<ul style="list-style-type: none"> • Annotate code or draw a diagram to show the overall structure • Restructure an 'untidy' program 	<ul style="list-style-type: none"> • Identify inputs needed to test all program branches • Will line X ever be executed? 	<ul style="list-style-type: none"> • Choose a name for a given program • Select/write a sentence that describes a program's purpose
Relationships (R)	<ul style="list-style-type: none"> • Identify variable scope • Highlight function calls 	<ul style="list-style-type: none"> • Draw the flow of control • Find redundant conditional branches 	<ul style="list-style-type: none"> • Choose a name for a variable/function • Are two programs/segments functionally equivalent?
Blocks (B)	<ul style="list-style-type: none"> • Identify block types, such as finite loops, 'else' conditions, function definitions, etc 	<ul style="list-style-type: none"> • Reordering lines of code • Parson's Problems 	<ul style="list-style-type: none"> • Explain the purpose of a block of code
Atoms (A)	<ul style="list-style-type: none"> • Identify statement types, such as assignments and conditions 	<ul style="list-style-type: none"> • Trace values through a program 	<ul style="list-style-type: none"> • Explain the purpose of a single line
	Text surface (T)	Program execution (P)	Function/purpose (F)

■ The Block Model

were already used to assess program comprehension, rather than support its development. As program comprehension is broad and there are many activities available, it can be difficult to know which activities to use in which circumstances.

The Block Model

One tool for understanding and categorising aspects of program comprehension is the Block Model², which consists of twelve areas of comprehension where four levels and three dimensions interact. The framework defines four distinct levels at which the learner may focus on a program:

- **Atoms**, the smallest elements, are the keywords, symbols, and syntax, or a single line of code
- **Blocks** are small chunks of code that perform a task — for example, single lines, loops, or selection statements
- **Relationships** are the connections between blocks, and the manner in which they work together, such as function calls and return values
- **Macro structure** refers to the program as a whole

The framework also considers the 'dimension' of the program, or how the learner is viewing it:

- The program exists as a static piece of text. This dimension is the 'text surface' and is where learners need to consider the program's grammar and syntax.
- When the program is executed, it becomes a dynamic object that may behave differently depending on its inputs. This dimension is known as the 'program execution'.
- The 'function' dimension is where learners consider the intended purpose of the code, whether that be an individual atom, block, or full program.

The Block Model therefore comprises twelve areas of program comprehension that learners should be able to move between as they develop their understanding. The related 'holey quilt' theory³ suggests that learners begin with varying levels of knowledge in each zone,

ranging from fragile to deep. Knowledge is deepened and supported over time by learning activities targeting each zone.

Mapping tasks to the Block Model

It is important to devise activities that develop comprehension in each of these zones. By considering each dimension in turn, we can identify tasks that may foster comprehension at each level of focus.

Comprehending the text surface can be tricky, as learners need to discern the meaning of the program from text with unfamiliar terms, structures, and syntax. Without support, they may get stuck focusing on the program at the level of atoms. A simple strategy is to identify aspects of the code within the text. By identifying examples of variables, conditions, functions, and so on, educators can help learners make sense of the text, and connect it to underlying concepts.

When considering program execution, several approaches could be used to help develop understanding. Learners could trace simple programs, determining the end state of variables or the inputs required to reach a specific state. They could complete Parson's Problems, which transcend the text surface and enable learners to focus on the correct sequence of instructions for a specified goal. Similarly, they could investigate the effect of swapping two lines of code, or try to find lines that can never run.

Learners can also benefit from exploring function. Asking learners to explain the function of a line, snippet, or entire program is a great place to start. They will have to use clues within the text and observe the execution. Educators can vary the degree of challenge by the clues they leave in the programs. Educators can also connect function back to text by asking learners to provide meaningful names for variables, functions, or entire programs. Alternatively, learners could be given a description of the purpose and identify a program that matches, or compare multiple programs to find which are functionally equivalent.

There are many options to choose from, but the most important step is to review our own practice, to find and fill those gaps in learners' program comprehension. ^(HW)

SUMMARY

Programming has several challenges:

- It is concept-rich, leading to cognitive overload
- It balances comprehension with coding experience
- It demands persistence and resilience
- Learners need a secure mental model of computation

Program comprehension:

- Allows learners to interpret, explain, adapt, debug, and create programs
- Supports learners to develop programming patterns or plans
- Can be divided into twelve zones using the Block Model
- Learners should develop knowledge in each zone and be able to move between them

Comprehension tasks:

- Educators can use the Block Model framework to categorise tasks and identify gaps where students need support
- A range of strategies already exist that have been mapped to the Block Model

REFERENCES

¹Izu, C. et al. (2019). Program Comprehension: Identifying Learning Trajectories for Novice Programmers. *ITICSE '19: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. New York, ACM. 261-262. helloworld.cc/comprehension1

²Schulte, C. et al. (2010). An introduction to program comprehension for computer science educators. In Clear, A. & Russell Dag, L. (eds.) *ITICSE-WGR '10: Proceedings of the 2010 ITICSE working group reports*. New York, ACM (pp. 65-86). helloworld.cc/comprehension2

³Clear, T. (2012). The hermeneutics of program comprehension: a 'holey quilt' theory. *ACM Inroads*. 3(2), 6-7. helloworld.cc/comprehension3

PARSON'S PROBLEMS

Educators can foster program comprehension by using Parson's Problems to reorganise jumbled lines of code

An important precursor to learning how to write computer programs is having the necessary program comprehension to interpret the function and structure of existing programs. One tool that can help learners develop program comprehension is Parson's Problems. Here, we explore Parson's

Problems, their benefits, the concept of distractors, and how to write a Parson's Problem task.

What is a Parson's Problem?

A Parson's Problem is a task in which learners are given all of the blocks or lines of code needed to solve a problem. The lines of code, though, have been jumbled so that they are no longer in the correct order. Learners are asked to reorganise the code into the correct order to perform a specific task.

The short example in **Figure 1** shows some jumbled lines of code (in Python and Scratch), and sets out the task that needs to be completed. Why not see if you can solve the problems in the example?

Parson's Problems can be applied to both text- and block-based programming and can vary in difficulty, to accommodate learners' existing understanding. For example, when you feel that learners are ready, you could provide them with lines of code and ask them to work out the indentation themselves (known as 2D Parson's Problems).

There are many ways in which Parson's Problems can be presented to learners. They make for excellent offline or paper-based activities that could be done individually, in pairs, or in small groups. You may choose to create problems directly in the development environment, to allow learners to immediately test their solutions. Alternatively, there are online tools such as [js-parsons](http://js-parsons.com) that allow you to create your own interactive problems (helloworld.cc/jsparsons).

Parson's Problems can be used to support formative assessment, as classroom discussion following the

activity plays an important part in learners' development. Immediate feedback also avoids any misconceptions being committed to long-term memory.

Benefits

The main benefit of Parson's Problems is that the learner is focusing on the structure and logic of blocks of code, rather than on the syntax of individual text elements (the 'atoms' of the Block Model). The process reduces the cognitive load experienced by learners, allowing them to practise sequencing and problem-solving with code. This experience is particularly helpful in the early stages of learning to program, when learners may be easily frustrated and put off by repeated unsuccessful attempts to solve a problem. Parson's Problems also expose learners to logic and syntax that they may not be fully familiar with.

Denny et al.¹ suggest that learners' solutions to a Parson's Problem "make clear what students don't know (specifically in both syntax and logic)". These solutions can allow for an easier analysis of the common errors that learners make, whereas "the open-ended nature of code-writing questions makes identifying such errors difficult". For example, when using a Parson's Problem, we can be sure that an error was not caused by a typing mistake.

Parson's Problems can promote some higher-order thinking in learners than simple code tracing (reading code and identifying its purpose or output). Parson's Problems can act as a stepping stone between the lowest and highest categories — being able to read and interpret code, and being able to write original code, which involves evaluation and creation (the highest categories in Bloom's taxonomy).

SUMMARY

Parson's Problems support learners by:

- Developing their understanding of how the program is executed (their 'notional machine' - see page 69)
- Reducing cognitive load
- Focusing on blocks of code, rather than on syntax
- Providing all the correct code in an engaging challenge
- Promoting dialogue and discussion about code

Benefits of Parson's Problems:

- Constrain the logic
- Avoid common syntax errors that can be barriers to learning to code
- Model good programming practices
- Provide the potential for immediate feedback
- Make it easier to identify common misconceptions
- Increase engagement of learners

Advice for writing Parson's Problems:

- Share problems with only a single solution
- Allow learners to manipulate actual code blocks
- Provide a clear description of the problem
- Clearly show the desired logic
- Share multiple similar problems over time

REFERENCES

- ¹ Denny, P., Luxton-Reilly, A., & Simon, B. (2008). Evaluating a New Exam Question: Parsons Problems. *ICER '08: Proceedings of the Fourth International Workshop on Computing Education Research*. New York, ACM. 113-124. helloworld.cc/parsons1
- ² Izu, C. et al. (2019). Fostering Program Comprehension in Novice Programmers - Learning Activities and Learning Trajectories. *ITICSE-WGR '19: Proceedings of the Working Group Reports on Innovation & Technology in Computer Science Education*. New York, ACM. 27-52. helloworld.cc/parsons2
- ³ Harms, K. J., Chen, J., & Kelleher, C. L. (2016). Distractors in Parsons Problems Decrease Learning Efficiency for Young Novice Programmers. *ICER 16: Proceedings of the 2016 ACM Conference on International Computing Education Research*, New York, ACM. 241-250. helloworld.cc/parsons3

Izu et al.² place Parson's Problems in the 'Blocks' row of the Block Model proposed by Schulte (see page 78). They state that "novice programmers should develop program comprehension skills as they learn to code so that they are able both to read and reason about code created by others, and to reflect on their code when writing, debugging or extending it". They also state that Parson's Problems support learners in developing their understanding of the 'notional machine' (see page 69).

Distractors

Some Parson's Problems include distractors. These are incorrect blocks or lines of code that are included in the set of provided code, meaning that learners need to be selective about which blocks they use; see the example below.

The inclusion of distractors can add an additional level of challenge³ for more confident learners. However, care should be taken, as they may unnecessarily increase

EXAMPLE

Rearrange the lines of code to create a program that outputs the total cost to the customer. Be aware that there are two lines of code that will cause errors in your program if used.

```
price = 3.50
quantity = 5
total = price * Quantity
total = price * quantity
print(total)
print("total")
```

Rearrange the lines or blocks of code below to create a program that asks the user for their name, then for their favourite food, before telling them that their food choice is a good choice.

Python

```
print("Hi " + name + ". What is your favourite food?")
print(food + " is a good choice " + name)
food = input()
name = input()
print("Please enter your name: ")
```

Scratch



Figure 1 Examples of Parson's Problems in Python and in Scratch

the cognitive load or the time spent on a task, or even result in a misconception or error being committed to long-term memory.

Advice for writing Parson's Problems

Explain clearly to your learners what the program should do when correctly sequenced; this reduces their cognitive load. Additionally, Denny et al.¹ recommend making sure that there is a unique answer for each question; that is, there should only be one ordering of the lines that achieves the goal.

Ensure that learners manipulate the actual lines of code, rather than using letters or numbers as a shorthand. Working with real lines of code helps to develop

“ WORKING WITH REAL CODE HELPS DEVELOP FAMILIARITY WITH SYNTAX ”

their familiarity with the syntax and the construction of the code.

In theory, it is possible for learners to guess the correct answer to a simple Parson's Problem without fully understanding the construct or logic being tested. Asking more than one question over time that tests the same logic or construct can reduce this concern.

Providing structure (such as braces, colons, or indentation) can make a question more accessible, as learners can use these visual clues to develop their solution. Providing this structure can also make it possible to tackle problems including more complex programming concepts. (HW)



■ Learning to write code can be a challenge for students, but there are evidence-based approaches to support them

Credit: Shutterstock.com/Gorodenkoff

HOW TO SUPPORT YOUR STUDENTS TO WRITE CODE

Will Grey shares some evidence-based approaches for teaching students to write code

For many children, writing code can be a daunting prospect. To help our students learn to write code, we can use a range of scaffolded pedagogies. Initially, these approaches take ownership of the code away from the students, giving them confidence to explore and experiment. Gradually, as the students gain confidence, we can reduce the amount of support until they are able to write their own programs independently.

On page 70 of this special edition, I shared approaches to support pupils learning to read code. These included activities such as explaining, predicting, and tracing code, as well as live demonstrations with worked examples. Now, I'll share some evidence-based approaches to support pupils who are learning to write their own code.

Fixing broken code

Children can find and fix common syntax, runtime, and logical errors in a piece of code. These errors might include missing brackets, missing speech marks, spelling mistakes, or missing variable declarations. Pupils can use the programming language's translator to help find the errors by making the fixes and then running the code to test that it works.

Pupils will need to be made aware of how the errors present themselves – this is unique to each translator and language and is not always obvious. For instance, in the default IDLE Python interpreter, it can be unclear where each error is located. Once pupils are familiar with a variety of common error messages, their causes, and how to fix them, they will be able to debug their own code with greater confidence

and resilience. For extra challenge, debugging exercises can be carried out on paper, where there is no feedback from the translator.

Completion problems and pseudocode

We can give children partial solutions to complete. This has scope for a variety of levels of differentiation: as students progress, we can gradually leave out more code. Initially, students might be given most of the code but with parts of some statements missing. Then, we might remove statements completely. Next, we could remove several lines or complete blocks of code. Finally, we could remove all code and only provide comments to the pupils. If these comments are detailed enough, then this is like converting from pseudocode into working code.

At some point, children will need to plan their own code with pseudocode. It is difficult and does require the application of high-level computational thinking, where pupils perform abstraction and code design. A good approach is to get pupils to write the comments for each line of their program before they start coding.

Parson's Problems

As introduced in my previous article, Parson's Problems are problems in which the code is presented to children all jumbled up. The aim of the activity is to put the instructions into the correct order so that the whole code performs a specified task. Students do not need to be concerned with remembering syntax and can focus on logic and sequencing. These problems will help develop students' program comprehension, preparing them for writing their own code.

Modifying code and tinkering

Pupils can make modifications to a piece of code to adapt it for a slightly different purpose. Suppose we have a Scratch program that draws a square. We could ask pupils to modify the code to draw a triangle, pentagon and circle, for instance. We could extend pupils' understanding and get them to find patterns in the relation between internal angles of a shape and the number of sides – we know finding patterns is a key aspect of computational thinking.

When pupils are modifying and completing code, a crib sheet that provides

two students work together on a single computer, can support students who are learning to write code. Within each pair, one student does the coding and the other observes, reviews and gives feedback. Students in each pair should regularly switch roles (see page 58 for more).

A tactical seating plan can be employed – with mixed attainment pairs, or pairs who can challenge each other. Even if each student is working independently on their own code, pupils should still be encouraged to collaborate with their neighbours. Collaborative approaches are often more effective than coding activities where pupils work in isolation without the support of their peers. Moreover, the cooperative and social nature of this approach has been shown to be particularly successful at engaging girls.

Planning lessons

We cannot expect children to be able to write code before they have learned how to read code. Through careful planning of lessons and sequences of lessons, we can slowly remove the scaffolding as pupils

Python Cheat Sheet

Printing strings	Output	Comments
<code>print ("Hello World")</code>	Hello World	<code>#This is a comment.</code>
<code>print ("Hello", "World")</code>	Hello World	<code># It helps you or others to</code>
<code>print ("Hello" + "World")</code>	HelloWorld	<code># understand the code better.</code> <code># It does not affect how the program works.</code>

Variables	Print variables
<code>apples=10</code>	<code>apples=10</code>
<code>name="Frodo Baggins"</code>	<code>oranges=3</code>
Input variables	<code>print (apples+oranges)</code>
<code>Name = input ("what is your name?")</code>	<code>output: 13</code>
<code>Age=int(input(How old are you?))</code>	

If ... else ...	Operators
<code>if (score < 12):</code>	<code><</code> Less than
<code>print "Grade U"</code>	<code>></code> Greater than
<code>elif (score >= 12 and score < 16):</code>	<code>>=</code> Greater than or equal to
<code>print "Grade Pass"</code>	<code><=</code> Less than or equal to
<code>else:</code>	<code>!=</code> Does not equal
<code>print "Grade Distinction"</code>	<code>==</code> Equals

For loops	Output	While loops
<code>for i in range(4):</code>	0	<code>x=0</code>
<code>print (i)</code>	1	<code>while x<4:</code>
	2	<code>print (x)</code>
	3	<code>x=x+1</code>

■ A Python crib sheet that students can refer to when writing code

code and gain increasing ownership of the code as they progress (see page 22).

If we consistently apply scaffolded approaches for reading and writing code in our lessons, we can make the experience of learning to code for children a lot less intimidating, and a good deal more enjoyable. [\(H/W\)](#)

“ WE CANNOT EXPECT CHILDREN TO BE ABLE TO WRITE CODE BEFORE THEY HAVE LEARNED HOW TO READ CODE ”

the basic syntax of constructs with some examples is a helpful aid. Pupils should be encouraged to organise any code that they write so that they can reuse the code in the future. The ability to recognise code that can be adapted to a new context is an important skill.

Pair programming and collaboration

Social constructivist approaches that use collaboration and pair programming, where

gain greater independence as they progress from reading and using code, through modifying code to ultimately writing their own code to solve problems.

The great news is that the PRIMM (Predict, Run, Investigate, Modify, Make) model offers a framework around which we can plan our lessons. For each lesson episode within this structure, we can pick and mix from our assortment of techniques where pupils read code before they write



WILL GREY

Will is head of computing at Comberton Village College, UK, and coordinator of the CAS South Cambridgeshire Community of Practice. He has developed a comprehensive set of resources for A level and GCSE computer science at helloworld.cc/grey.



THE I IN PRIMM

Sue Sentance explores how the Block Model – a framework for program comprehension – can create opportunities for students to investigate code fully

There are several reasons why some students find programming difficult to grasp. Combining understanding the syntax of a programming language with the need to work out logical solutions can lead to high cognitive load. Students can also become easily disheartened when faced with errors or programs that don't work. And copying code without understanding it can lead to students developing misconceptions.

To address some of these issues, I developed the PRIMM approach to teaching programming. Following this approach, teachers can structure programming lessons with five elements as follows: Predict, Run, Investigate, Modify, and Make. The approach is outlined on page 22, but here is a refresher.

- **Predict:** Students discuss a program and predict what it might do, drawing or writing out what they think will be the output. At this level, the focus is on the function of the code.
- **Run:** Students run the program so that they can test their prediction and discuss it in class.
- **Investigate:** The teacher provides a range of activities to explore the structure of the code; this involves activities such as tracing, explaining, annotating, and debugging.
- **Modify:** Students edit the program to change its functionality via a sequence of increasingly challenging exercises; the transfer of ownership moves from the code being 'not mine' to 'partly mine' as students gain confidence by extending the function of the code.
- **Make:** Students design a new program that uses the same structures but solves a new problem (e.g. has a new function). There are three key principles underlying PRIMM, all emerging from research in computer programming education. The first is that students should learn to read before they write. The excitement of writing a new program and creating something that works means we don't spend enough time reading and learning from simple, well-written programs first. In literacy, we learn to read first, and at a level beyond what we can write. We learn from reading examples of the written word. The same can apply to programming. There is a substantial body of research around tracing and reading programs that has shown that reading first is beneficial for novice programmers.



■ One of the principles underlying PRIMM is that students should talk about their programs

Credit: stock.adobe.com/sutlaff

The second principle is that students should talk about their programs. This works at three levels: we need to find the right language or terminology to use to articulate our understanding; we are helped by verbalising what may be a complete mishmash in our head; and we also share in the creation of understanding through dialogue with others.

The final principle is that students should start with code that isn't their own. Using a starter program, written by somebody else who takes responsibility for any bugs, reduces the emotional strain caused by our programs failing. We often hear that students in computing should build up resilience, because their programs will always fail. However, I believe that by providing code written by somebody else first, we protect our students from the disappointment, anxiety, and frustration that we've all felt when our programs fail. This is especially important for young learners, but applies to anyone learning to code.

The Investigate phase of PRIMM gives us the opportunity to devise creative activities and ask insightful questions about a program. In this way, we are moving from thinking about what the code does to how it does it. Armed with this knowledge, learners should be more confident to tackle their own programs.

“ STUDENTS SHOULD START WITH CODE THAT ISN'T THEIRS — THIS REDUCES THE EMOTIONAL STRAIN CAUSED BY FAILING

In the Investigate phase, we often use a range of activities. Here are some examples:

1. Ask, “What would happen if those two lines were the other way round?”
2. Ask, “What would happen if the input to the program were ____?”
3. Ask students to draw on the program to identify blocks of code or types of construct.
4. Draw the flow of control on the program showing what line is executed when a loop is exited, for example.
5. Ask students to identify the scope of a variable.
6. Identify the purpose of a single statement. ▶

MORE ABOUT PRIMM

PRIMM was established from a combination of experience and research, so not surprisingly, many experienced teachers will recognise elements from their own practice. It's very useful if you are new to teaching programming, or are struggling to get your students to understand a particular concept. If you've never heard of PRIMM, here are some resources to have a look at:

- PRIMM: A research-based article on p.22 of this special edition of Hello World
- PRIMM: Something for your programming pedagogy toolkit? Sue Sentance, Hello World issue 4, pp.62-63
- Programming pedagogies: PRIMM. Oliver Quinlan, Hello World issue 5, p.25
- Reusing familiar techniques. Jane Waite, Hello World issue 6, pp.74-75
- An overview of PRIMM: primmportal.com
- Sentance, S., Waite, J., & Kallia, M. (2019). Teachers' Experiences of using PRIMM to Teach Programming in School, *SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 476-482. helloworld.cc/primmexperiences



➤ These activities can be varied and added to, and many teachers have been very innovative in how they ask students questions that really get them thinking. However, we can fall into the trap of asking the same questions about code, and discover that we are not really supporting understanding as much as we could.

The Block Model

This is where the Block Model comes into play. The Block Model was created by Carsten Schulte and gives us some insight into the various granularities of skills that new programmers need. As outlined on page 78, the Block Model is a grid with two axes — one showing the size of the programming element under consideration, and the other the distinction between the structure of the program, the execution of the program, and the function of the program. In 2019, Cruz Izu and Carsten Schulte led research that mapped classroom activities using the model, leading me to link the idea to PRIMM. The Block Model ensures that the Investigation stage of PRIMM addresses the range of skills required to thoroughly understand a program.

“ IN THE INVESTIGATE PHASE WE DEVISE ACTIVITIES AND ASK INSIGHTFUL QUESTIONS ABOUT A PROGRAM

From atoms to macro structure

At the bottom of the Block Model is the atom: a single language element such as a variable, or the word 'if'. The top is the least granular end of the scale, where we have the macro structure: the whole program. Between these two extremes we have blocks of code, and relationships between blocks of code. Using the Block Model enables us to reflect on whether our questioning spans all these four levels.

Structure versus function

When we ask students to predict, we ask them what a program might do. This relates to the function of the program or piece of code. In contrast, the structure of the program relates to the syntax and how the program is constructed. For example, we might be using the keyword 'for' in Python to indicate the beginning of a loop. There is one more column:

program execution. This relates to the flow of control and values of variables when the program runs, or what code is called when.

FURTHER READING

- Schulte, C. (2008). Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching. *Proceedings of the Fourth International Workshop on Computing Education Research*, 149-160. helloworld.cc/iinprimm1
- Izu, C. et al. (2019). Fostering Program Comprehension in Novice Programmers - Learning Activities and Learning Trajectories. *ITICSE-WGR '19 Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, 27-52. helloworld.cc/iinprimm2

THE BLOCK MODEL

(M) Macro structure			2. Ask, "What would happen if the input to the program were ___?"
(R) Relationships	5. Ask students to identify the scope of a variable	4. Draw the flow of control on the program	
(B) Blocks	3. Ask students to draw on the program to identify blocks of code or types of construct	1. Ask, "What would happen if those two lines were the other way round?" 4. Draw the flow of control	
(A) Atoms			6. Identify the purpose of a single statement
	(T) Text surface	(P) Program execution	(F) Function
	Architecture/Structure		Relevance/Intention

AN AWARENESS OF THE TYPES OF ACTIVITIES WE USE WILL HELP US UNDERSTAND STUDENTS' DIFFICULTIES

For example, if we have a 'for' loop in Python, the program execution level helps us to understand the flow of control back through the loop code a given number of times, and the impact this has on the value of the stepper variable.

Twelve zones of understanding

Looking at the Block Model in its entirety, we can see that it gives us twelve zones of program comprehension. So the question is: can we devise activities or ask questions about a program that develops knowledge in each of these twelve zones? It's a challenge to do this, but if we don't, we may not totally support students' full understanding of the program.

If we take, as an example, some of the questions and activities I outlined earlier, we could start to put some of them into the Block Model and identify the gaps. The


exact placement of each task in the table above may vary depending on the views of the teacher, but in this case, it's clear that there are gaps. For example, I can see that there is a gap at the bottom left, at Atoms/Text surface. To address this, I could add the following — "Can you locate a function call in the program?" — to my list of questions to ask the students.


What I've outlined here would take a lot of time and effort to do in every lesson, but an enhanced awareness of the types of questions and activities we use will help us understand our students' difficulties with programming. I haven't tested this widely, but I believe that it may help us develop a language we can use to understand our own practice as programming teachers. The approach adds to our pedagogical content knowledge and increases metacognition relating to the teaching of programming.




SUE SENTANCE

Sue is chief learning officer at the Raspberry Pi Foundation. An ex-teacher, a teacher-trainer, and an academic, she is passionate about research in computing education.

I'm keen to hear from any teachers who try to apply the Block Model in their PRIMM lessons! Do contact me via Hello World at contact@helloworld.cc and let me know how you get on! 



MODEL EVERYTHING

- 90** WORKED EXAMPLES
 - 92** LIVE CODING
 - 94** LEARNING TO PROGRAM VIA
VIDEOS AND SELF-EXPLANATION
 - 96** HOW MODELLING CAN SUPPORT
LEARNERS
 - 98** WATCH AND LEARN
 - 100** SMELLY CODE
- 

T here are many practices and processes in computing that can easily be taken for granted: for example, program debugging, tracing programs and algorithms, and compressing data. These are all skills pupils need to learn and practise, and as an educator, your role is to model them. Your learners will benefit from modelled techniques, as they can demonstrate a correct method, highlight best practice, or surface an otherwise hidden thought process. There are many ways to model things for your learners, including worked examples and live coding.

A worked example provides a complete (or partially complete) model process for solving a problem or achieving a task. This scaffolded approach can be particularly beneficial for novice learners, helping them to complete similar tasks before you gradually remove support.

In programming specifically, educators can model the programming and debugging process through live coding, in which teachers write programs live with their students.

The teacher explains their thought processes, modelling how to construct the program, making mistakes, and showing what to do when things go wrong. Taking this approach can help demonstrate the thought processes and methods of an expert, which pupils can then apply in their own programming practice.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:** worked examples
- **What the research says:** live coding
- **What the research says:** learning to program via videos
- How modelling can support learners
- How video is changing the way we teach computing
- Smelly code: best practice in teaching block-based programming



WORKED EXAMPLES

Educators can use worked examples to support novices to develop their programming practice

Worked examples demonstrate an 'expert' solution to a problem. They are used in many subjects to support novices, who use the examples as blueprints for solving new but related

SUMMARY

Worked examples can:

- Help reduce extraneous cognitive load
- Aid learners in assimilating new knowledge into their existing understanding
- Be especially useful for novices during the early stages of learning

Good worked examples:

- Include sub-goal labelling to highlight structure and common programming 'patterns'
- Present relevant information in an integrated manner
- Combine multiple modes of delivery, such as visual and aural explanations
- May only be partial and require learners to complete them as part of exploration

In a learning sequence:

- Combine worked examples with similar practice problems
- Alternate worked examples and practice problems to keep the example in mind
- Use at least two examples for each concept or 'pattern' explored
- Fade the use of worked examples over time
- Focus on examples that emphasise program structure over surface details

Illustrating process:

- Educators should explicitly model their approach to solving a problem
- Process-oriented worked examples emphasise how a solution was reached
- Product-oriented worked examples provide a possible solution

problems. They can be used in many areas of computing and are particularly useful for supporting programming practice. Learners who encounter worked examples in conjunction with practice problems are more likely to develop and assimilate strategies for solving similar problems.¹ Here, we'll discuss how worked examples can reduce cognitive load, the different types of worked examples, and how to design and integrate them into your lessons.

Reducing cognitive load

Worked examples help reduce the extraneous cognitive load placed on a learner's working memory by providing a model solution for a problem, which the learner can read, understand, and adapt to solve similar problems.

When a learner is given a partial or complete solution, they do not need to recall as much from their long-term

memory.² If a concept is included in the solution, the learner can quickly retrieve and apply their existing understanding relating to that concept. Partially complete problems help focus learners on particular concepts, because learners only need to focus on the missing aspects.

In focusing learners' attention on structural elements of problems, worked examples support students to organise their new knowledge into 'schemas' (clusters of connected ideas) within long-term memory³ (see page 20).

Product versus process

There are typically two types of worked examples found in literature. Both support the learner by modelling solutions to problems:

- **Process-oriented examples** model the steps taken to reach a particular solution. They may be written down (for example,

Worked example (partial)

```
#Use a turtle object and a finite loop to
#draw a square with two sides 100 pixels long.

#Add required functions
from turtle import Turtle

# Initialise a turtle object
t = Turtle()

# Move/turn once per side
for side in range ( _ ): # <- What value here?
    t.forward( _ )
    t.right(90)
```

Integrated instructions

Sub-goals labels highlighting stages in the solution

Comments used as questions and reflection prompts

Incomplete elements for learners to resolve

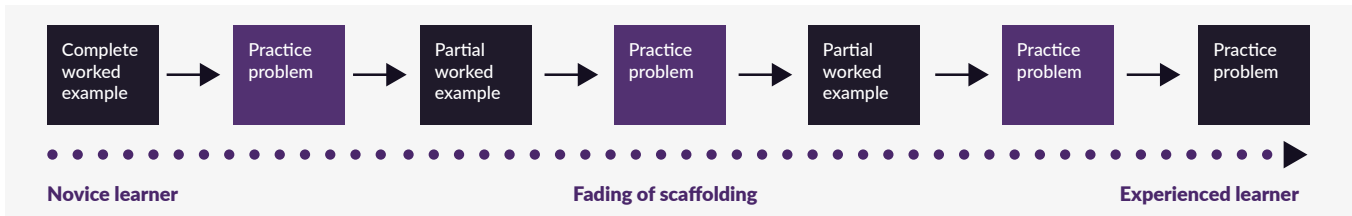
Possible problems focussed on finite iteration

Write a program that uses a loop to draw another regular polygon.

Make a program that prints the message "Hip Hip Hooray!" three times.

Product-oriented worked example written in Python

■ A product-oriented worked example written in Python



■ Educators should use worked examples while a concept is new, then fade this support over time

helloworld.cc/poewritten), demonstrated by an expert, or captured on video (for example, helloworld.cc/poevideo).

- **Product-oriented examples** model one possible solution and allow learners to examine and apply the solution to a new context.

There is evidence that complete novices benefit more from process-oriented worked examples, as they provide rationale for each aspect of the solution. Learners with some experience will then benefit more from product-oriented examples, from which they can infer the rationale.⁴

Designing worked examples

When designing worked examples, educators should consider the following effects that may affect learners' cognitive load and ability to follow an example.

The split attention effect occurs when information about a problem or example is presented separately. To follow the example or solve the problem, learners must first combine the separate sources of information in working memory. If possible, educators should integrate all of the information into one clear representation.

Similarly, the redundancy effect occurs when information is duplicated within a problem unnecessarily, or other redundant information is included. Learners may still process this information as they try to understand the problem, which results in an unnecessary cognitive burden.

Take advantage of the multimodal effect by presenting key information both visually and aurally, as the brain will process these separately. Studies have shown that presenting the same information — and

more specifically, worked examples — in another mode, either simultaneously or sequentially, can support learners in their comprehension, and therefore, their ability to solve future problems.

It is broadly accepted that novices (in many fields) tend to focus on the context of a problem and the surface details, rather than the underlying structure and common elements to solutions. Educators can use sub-goal labelling to identify the important components or steps in a solution and highlight them to learners. To do this, educators could use explanatory comments or annotations, visual labels or highlights, or white space to group related instructions into 'chunks'.¹

The learning sequence

Educators should also consider how to combine worked examples with practice problems and other worked examples.

It is important to consider variety: presenting the same concept or programming pattern across multiple examples and problems within varied contexts. This variety helps learners to focus on structural connections between the solutions and therefore focus on the general concept, rather than the surface details of the problem.

Research suggests that the more worked examples learners experience, the more they benefit, and that educators should expose learners to at least two worked examples for each concept or pattern. Some studies also suggest that learners should be presented with example and practice pairs⁵, which require them to understand an example, then apply it in practice. Alternatively, learners could review one example problem, then

REFERENCES

- ¹ Atkinson, R. K., Derry, S. J., Renkl, A. & Wortham, D. (2000). Learning from Examples: Instructional Principles from the Worked Examples Research. *Review of Educational Research*. 70(2), 181-214. helloworld.cc/workedex1
- ² Sweller, J., Ayres, P. & Kalyuga, S. (2011). *Cognitive Load Theory*. New York, Springer. helloworld.cc/workedex2
- ³ Sweller, J., van Merriënboer, J. J. G. & Paas, F. (2019). Cognitive Architecture and Instructional Design: 20 Years Later. *Educational Psychology Review*. 31(2), 261-292. helloworld.cc/workedex3
- ⁴ van Gog, T., Paas, F. & van Merriënboer, J. J. (2008). Effects of Studying Sequences of Process-Oriented and Product-Oriented Worked Examples on Troubleshooting Transfer Efficiency. *Learning and Instruction*. 18(3), 211-222. helloworld.cc/workedex4
- ⁵ Abdul-Rahman, S. S. and du Boulay, B. (2010). Learning Programming via Worked-examples. In: *Proceedings of PPIG-WIP 2010, 7-8 January 2010, Dundee*. helloworld.cc/workedex5

complete several practice problems, which would require them to hold the example in their working memory for longer.

Worked examples are highly beneficial for novices, because they support them to build patterns for programs and procedures. However, as learners develop their expertise, they benefit more from solving new problems than from working from examples. Educators should use worked examples while a concept is new and gradually fade this support.³ In doing so, they can support learners in developing a useful collection of common programming patterns which they can apply, adapt and build on. (HW)

LIVE CODING

Live coding brings coding to life for learners by demonstrating the thought processes of a programmer

When learners read static, completed programs, they aren't exposed to the troubleshooting that has already taken place to get to that end product. This is known as being product-focused. Live coding is when an educator develops the solution to a problem in front of the class for learners to follow, which is known as being process-focused.

Bringing programming to life

Novice programmers can often look at a finished program and have the misconception that it has been written from top to bottom, and that a skilled programmer always knows exactly what they are doing and doesn't make any mistakes. As any programmer knows, this is not the case.

Live coding demonstrates to learners the incremental nature of programming. It shows that problems are decomposed into small sections that are programmed, tested, and debugged, before the next stage is worked upon. It models good programming practice and shows learners that a plan for a program is formulated and followed, rather than a solution being formed on an ad hoc basis.

Bringing programming to life is essential to show learners that program development is non-linear. Code moves around and changes as a solution is developed. Live coding models how programs should be tested frequently to debug them. It also shows learners how to solve common errors that may occur when using a new concept.

Cognitive apprenticeships

Allowing learners an insight into the thought processes of an expert programmer follows the teaching approach of cognitive apprenticeships: employing methods traditionally used in apprenticeships in the teaching and learning of cognitive skills. The idea of cognitive apprenticeships was introduced by Collins et al.¹ in 1987. They believed that "teaching methods should be designed to give students the opportunity to observe, engage in, and invent or discover expert strategies in context". At its core, a cognitive apprenticeship involves modelling, coaching, and scaffolding.

Modelling involves an expert showing learners how to carry out a task, which "requires the externalization of usually internal (cognitive) processes and activities" by the educator.¹ In live coding, an educator develops a program in front of a class while highlighting their choices, decisions, mistakes, and debugging strategies.

Coaching is where educators give learners a challenge that is slightly too difficult for them and support them towards finding the solution through feedback and further modelling. Live coding is a great example of a coaching strategy, and is a way of guiding learners through a task that would usually be unattainable.

Through live coding, educators can provide scaffolding for learners in how they break down a problem and highlight milestones or sub-goals. As learners attempt their own programming tasks, they can work towards these sub-goals, or apply the same technique to novel problems.

Slowing down

Live coding is very different to reading solutions on a worksheet or in a textbook.

SUMMARY

According to the literature, the key benefits of live coding are that it:

- Reduces cognitive load through collaboration
- Makes the process of learning programming easier to understand for novices
- Helps learners understand the process of debugging
- Exposes learners to good programming practices

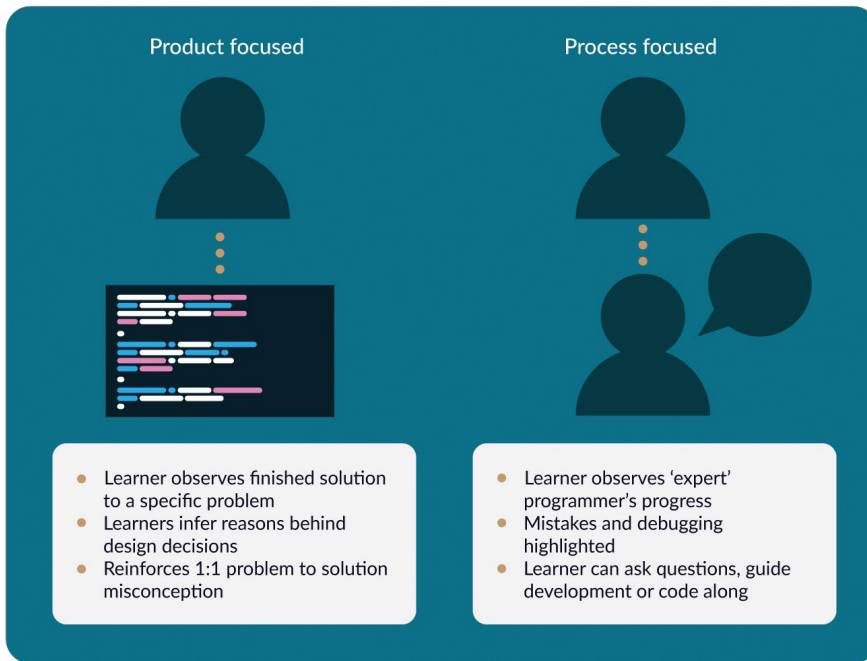
Good practice when live coding:

- Select an appropriate programming challenge to teach a new concept, consolidate learning, or address misconceptions
- Talk to your learners and ask them questions
- Narrate your inner monologue
- Make and fix mistakes, either planned or accidental
- Slow down to give your learners time to process

- Show learners that code isn't written from top to bottom in a linear way; it moves around as it is developed
- Be visible: let learners see your face, and don't turn your back for too long
- Pause to write things on the board, draw diagrams, and work things out
- Use the largest font possible without losing view of the full line of code
- Break the code into small chunks (decompose) and use sub-goal labelling while forming the solution

Strong links with worked examples:

Live coding helps novices learn by observing an expert programmer working through a problem, and so it has strong links to the concept and benefits of worked examples (see page 90).



“ IN LIVE CODING, AN EDUCATOR DEVELOPS A PROGRAM IN FRONT OF A CLASS, WHILE HIGHLIGHTING THEIR CHOICES AND DECISIONS

Those examples show a final, polished solution without any insight into how the programmer has made decisions about their code. The *Role of Live-Coding*² paper states that “when students begin to learn programming, usually they don’t have a good idea about where to start”.

If you write your solution in front of learners it forces you to slow down, which helps you think about what you are doing, and enables learners to follow your process. It is important that you don’t simply copy and paste the solution from one tab into a new window; this defeats the purpose, and your learners may get lost very quickly. You could write some notes about how you solved the problem and keep these on your desk as a prompt.

Learners benefit from following the process of your work, as it keeps them engaged in finding the solution. This is another reason why slowing down is important. You can chunk the demonstration and have sections where learners watch, and sections where

they code. It is important that they don’t miss key things while they are typing, so monitor their progress as you carry out your session. You can also provide video recordings of your sessions, to help learners who may need a recap or who learn at a different pace. If you decide to record your live-coding session, make sure you stick to the live-coding principles.

Predicting, testing, and debugging

When carrying out a live-coding session, it is important that it doesn’t become a tutorial that leads learners to the perfect solution on their first attempt. The learners are part of the journey. The best way to engage them is to ask them to make predictions about the program before it is run.

Wilson’s *Teaching Tech Together*³ emphasises the importance of making mistakes while live coding. Mistakes should be embraced because they allow learners to see that programmers don’t get it right first time, and often have to review and fix their work to find a solution. When

REFERENCES

- ¹ Collins, A., Brown, J.S., & Newman, S. E. (1987). *Cognitive apprenticeship: teaching the craft of reading, writing, and mathematics*. BBN Laboratories, Cambridge, MA., Centre for the Study of Reading, University of Illinois. Report number: 403. helloworld.cc/livecode1
- ² Raj, A. G. S., Patel, J. M., Halverson, R., & Halverson, E. R. (2018). Role of Live-coding in Learning Introductory Programming. *ACM*, 13, 1-8. helloworld.cc/livecode2
- ³ Wilson, G. (2009). *Teaching tech together: how to create and deliver lessons that work and build a teaching community around them*. Taylor & Francis. helloworld.cc/livecode3
- ⁴ Miller, K., Lasry, N., Chu, K., & Mazur, E. (2013). Role of physics lecture demonstrations in conceptual learning. *Physical review physics education research*, 9(2), 1-5. helloworld.cc/livecode4

live coding, you should plan intentional mistakes, but should also be confident when making unintentional mistakes. Intentional mistakes should link to common errors or learner misconceptions in order to target and alleviate them. You should also test your program continually. This helps learners to see this as a natural part of programming and teaches them to test their own work frequently.

When making intentional mistakes, encourage learners to predict what will happen, before running the code. Doing this will help learners suggest strategies to fix those errors. Miller et al.⁴ discovered that “students who predict are significantly more likely to correctly report the outcome of a demonstration”. Outcomes were improved whether their prediction was correct or incorrect. Therefore, asking prediction-focused questions while live coding is an important part of the process.

The benefits of live coding are evident. Try bringing programming to life in your next class and help learners understand the journey that a program takes. (HW)

LEARNING TO PROGRAM VIA VIDEOS AND SELF-EXPLANATION

STORY BY Lucia Flóriánová

Results from a small study of high-school students in Denmark suggest that an approach combining guided instructional videos with questions prompting students to reflect on their learning can help them learn to program. The researchers have named the approach 'stepwise self-explanation'.

Novice learners face various difficulties while acquiring programming skills. In particular, they struggle to combine and use basic structures to build a program, and to apply acquired knowledge to new situations.

Making programming easier

Aureliano, Tedesco, and Caspersen (2016) explored how to make learning programming easier for novices. They used an approach in which students answered questions prompting them to explain what they were learning as they watched instructional videos on how to build a program.

The video instructions were taken from *The Joy of Code* series, which was developed to teach students to program in Java via the Greenfoot development tool. The videos are structured according to the stepwise

improvement framework, providing step-by-step guidance. After watching video instructions about a piece of code, students answered questions about its purpose and functionality. For example, they explained which part of the code made an object move. The researchers compared this method with another approach in which students self-studied the same videos, but did not do any self-explanation. Although the research cannot confirm how significant the difference

STEPWISE IMPROVEMENT

Stepwise improvement is a framework that advises learning programming by developing small pieces of code systematically and incrementally. It encompasses three types of activities that programmers use to build a program: extension, refinement, and restructuring.

Stepwise self-explanation is an approach to learning programming that combines the stepwise improvement framework with self-explanation. In this study, this was achieved using instructional videos alongside a list of questions about them.



■ Students watched videos from *The Joy of Code* series

“ THE STEPWISE IMPROVEMENT FRAMEWORK IS LIKE TAKING STUDENTS FOR A GUIDED TOUR

between the two groups is, there was a positive correlation between students' self-explanations and their final results. Students also left positive feedback.

What makes the approach work?

Caspersen and Kölling (2009) explain that the stepwise improvement framework is like taking students for a guided tour instead of leaving them to walk around randomly — it draws attention to the most important aspects of programming education. It breaks down the content and gives the instruction materials a frame that guides learners through programming processes. However, the structure of the instruction materials itself is not enough to make novice programmers learn, as they do not yet know how to study for the needs of a programming course. This can be mitigated through the use of self-explanation, which guides students further through the instruction materials.

Previous studies have found that students who generated explanations of instructions or programs learnt more and scored better in problem-solving.

Students enjoyed this method of learning. They appreciated that the videos presented the content step-by-step, and found them easy to follow. They also liked that they could rewind or rewatch videos when needed. Groups using self-explanation commented that the question prompts helped them to remember things and summarise what they'd learnt.

In the classroom

Efficient teaching of programming takes students through the right level of detail when they are ready for it, but also prompts them to think about what the code is doing as they learn. Stepwise self-explanation is one way to provide a balance between teachers' guidance and students' cognitive

engagement. Asking students a small number of questions can direct their attention to the most important learning objectives and computing concepts. Using instructional videos that are built around small pieces of code is not only engaging, but also guides students through a new topic, giving them the opportunity to come back to the parts they missed or about which they are unsure. [\(HW\)](#)

FURTHER READING

- ✓ Aureliano, V. O., Tedesco, P., & Caspersen, M. E. (2016). Learning programming through stepwise self-explanations. *11th Iberian Conference on Information Systems and Technologies*, 1-6. helloworld.cc/aureliano2016
- ✓ Caspersen, M. E., & Kölling, M. (2009). STREAM: A First Programming Process. *ACM Transactions on Computing Education*, (9)1/4. helloworld.cc/caspersen2009
- ✓ The Joy of Code videos: helloworld.cc/joyofcode

Well-placed questions can direct students' attention



HOW MODELLING CAN SUPPORT LEARNERS

Josh Crossman explains the modelling approach: by demonstrating a new concept, teachers can both support their learners and develop their own understanding of the key skills and materials being introduced

When giving learners the opportunity to use new skills or software, it is important to show them how first. Teachers can demonstrate a new concept by using a recorded video, or by modelling it for the learners. Modelling is an instructional teaching strategy in which a teacher demonstrates a new skill or approach to learning. Teachers first model the task or skill for learners, and then learners begin the task and work through it at their own pace.

Schools use modelling across various disciplines, such as writing, handwriting, mathematical strategies, and science experiments. It's a powerful strategy that can be used across many different subjects, and computing is no different.

As we have been developing units for the Teach Computing Curriculum (TCC) (helloworld.cc/tcc), we have reflected on the

tried and tested techniques that help make computing lessons a success. In this article, I will share some of our top tips for modelling.

Improved confidence

Perhaps the biggest benefit of modelling is the confidence and competence that teachers gain from using the software. In the TCC Year 6 (ages 10 to 11) '3D modelling' unit, learners are shown how to create a 3D shape and change the viewing angle in the 3D modelling software Tinkercad (helloworld.cc/3Dmodelling). Teachers who are new to 3D modelling may not have come across changing the viewing angle before.

Through prelesson preparation, and of course, through using the software when modelling, teachers may encounter misconceptions, errors, and perhaps shortcuts that the learners might make in

their own use. For example, in Tinkercad, teachers can click on the viewing cube to jump to different views. As their own confidence improves, supporting the learners with their errors or misconceptions will become easier.

Thinking aloud

If teachers provide a monologue — or think aloud — as they model, learners get the opportunity to observe expert thinking that they wouldn't usually have access to. It allows learners to follow more closely what the teacher is doing and why they are doing it. More precisely, it can reduce the extra cognitive load of having to deconstruct each step for themselves (see page 20 for more about cognitive load theory).

As teachers know their own learners best, they can tailor the language they use when modelling, to make it as beneficial as possible for their particular learners. This ensures that teachers can meet their learners' needs in a more targeted way.

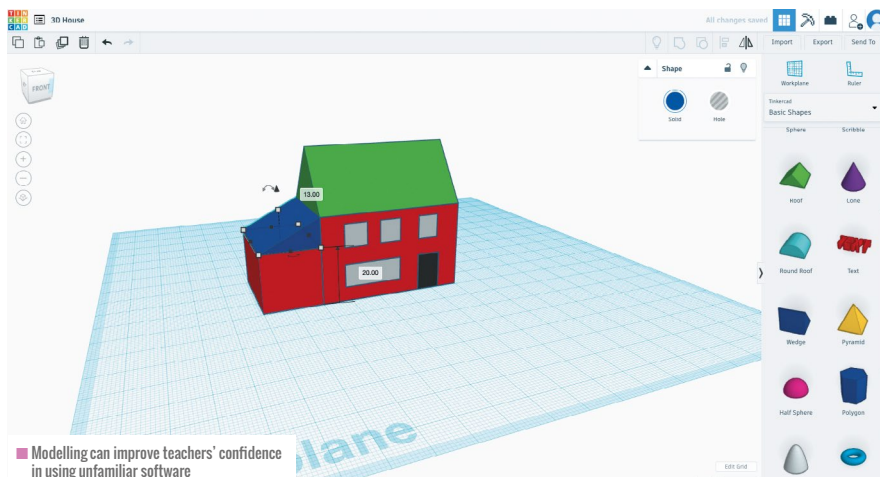
Greater freedom

Questioning is a key part of modelling — this includes both the teacher's questioning of their learners to draw out understanding, and the learners' questioning of the skills and processes the teacher is using. Modelling can enable teachers to give context to their answers to the questions, ensuring they are able to show alongside the tell. This will help to make their answers more visual and concrete.

Additionally, the freedom of modelling ensures that teachers are able to meet



Credit: stock.adobe.com/iusta_photos



the needs of all the learners in the room. This could mean revising an area that wasn't fully understood by some, or perhaps moving at a slower or quicker pace, depending on the learners' level of understanding. For example, in lesson one of the '3D modelling' unit of the TCC, teachers could break up the content in the video into smaller chunks, perhaps allowing the learners to get to grips with the viewing angle before introducing the zoom buttons.

Impromptu learning

In line with the benefit of greater freedom discussed previously, modelling also allows for more impromptu learning. As students ask questions, teachers may need to explore different areas of the software, or be challenged by questions about a skill they hadn't previously thought about. Using the freedom provided by modelling can lead to a deeper level of understanding. Equally, it can reinforce the skills used in previous lessons, giving learners the chance to consolidate their learning as they progress to the next stage.

As teachers narrate their modelling, it can also bring to the forefront things they do instinctively. Learners can pick up on these things, such as keyboard shortcuts (copy, paste, and duplicate come to mind) that teachers might use naturally. We do these things all the time when using new software or learning new things, but we might not always use the opportunity to share them. Modelling can also lead to exploration of a skill that might benefit learners, but wasn't originally planned for the lesson.

Making mistakes

By demonstrating computing experiences live with their learners, teachers can encourage the sharing of mistakes, both intentional and unintentional. Making and demonstrating mistakes is OK — in fact, it should be celebrated! Allowing learners to see that their teachers aren't immune to making mistakes can be encouraging for less confident learners and help them to build resilience. More importantly, it enables learners to see how to respond to mistakes, particularly when combined with the thinking aloud approach mentioned previously. This can be a powerful tool for encouraging perseverance.

By sharing mistakes and thinking aloud, teachers can guide their learners through strategies to overcome a range of obstacles. Use phrases such as: "When I first looked at this problem, I didn't know where to lead to start", or, "It's OK to feel frustrated at this point; I often do."

Video demonstrations vs live modelling

As well as live modelling, there is the option of demonstrating software using prerecorded videos. This means that you can show the video to learners and they can follow along and learn the skills you need them to know. However, many of the benefits to modelling, such as impromptu learning and improved confidence, will not be gained in this way.

Recorded video definitely has a place in the classroom — particularly in the context of the pandemic — but modelling these skills yourself, making mistakes, and

developing ideas as you go are invaluable ways to share learning experiences. If you do show your learners videos instead of using live modelling, here are some tips for including the modelling approaches:

- Watch the video before the lesson and practise the thinking aloud approach
- Share the video with learners and think aloud as the video is shown; allow the learners to attempt the task before asking them to share their own modelling of it
- Punctuate the video with your own questions, and don't be afraid to pause to answer questions posed by the learners

Throughout our work on the TCC, my colleagues and I recorded videos that could be used to model key skills. Our aim was to ensure that the content was as accessible as possible for the full range of teachers who may use it.

Though live modelling may take longer than displaying a prerecorded video, it is a great way to share the learning experience. In my view, it has many benefits over the more passive prerecorded approach. Hopefully, many teachers will watch these videos and gain the confidence to give modelling a try.

Modelling is a journey, and teachers inevitably won't get it right every time — but sharing that experience with the learners may turn out to be the most powerful part of the exercise. **(HW)**



JOSH CROSSMAN

Josh is a programme coordinator at the Raspberry Pi Foundation, working across programmes such as the Teach Computing Curriculum and Hello World. He is a Raspberry Pi Certified Educator and a former primary teacher.



CARRIE ANNE PHILBIN DIRECTOR OF EDUCATOR
SUPPORT AT THE RASPBERRY PI FOUNDATION

WATCH AND LEARN

How online video is changing the way we teach computer science

I remember writing my first computer program on a BBC Micro as if it were yesterday. It was of course during a maths lesson; I had already completed the set work, so I was allowed to use the computer! Logo was my favourite – typing commands to make pretty geometric pictures on a monitor was exciting.

The BBC Micro has a lot to answer for. It inspired a generation of computer enthusiasts and professionals like me, and led to the development of Raspberry Pi computers and other small form-factor programmable devices. But what was really revolutionary about the BBC Micro was its accompanying television series, called *The Computer Programme*, where you could learn to use it alongside the hosts, Chris Serle and Mac.

Before this TV programme, the most common way to learn about computers and how to program them was through magazines and books, or even the trusty computer manual. The availability of video tuition really helped me, and now, 30 or so years later, we have video-based platforms that provide content on every aspect of computing, computer science, and technology. They can help us both develop our subject knowledge as computer science specialists and support our students through all the stages of their learning.

Content for teachers

Continued professional development (CPD) can be tricky to fit in with the everyday demands of being a teacher. Trying to find time to be released from the timetable to attend courses

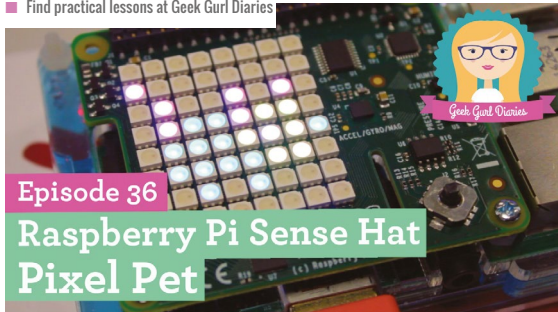
is always a cause of frustration. Thankfully, Computing at School (which spends every moment thinking about how it can support computer science teachers) created CAS TV (helloworld.cc/castv), a YouTube channel full to the brim with content for teachers at every stage of their computing journey. Videos include tips and tricks on teaching particular concepts at different learning stages, learning how to plan a scheme of work, and even how to engage underrepresented groups.

Some CAS TV videos also explore creativity and innovation with leading developers. Genevieve Smith-Nunes looks at how dance can represent data (helloworld.cc/dataanddance), and Andrew Fitzgibbon introduces computer vision and machine learning (helloworld.cc/computervision). Many of the videos are delivered by practising teachers in both primary and secondary schools, including Phil Bagge, as well as leading teacher trainers such as Miles Berry, Jane Waite,

■ Access bite-sized CPD from leading teacher trainers on the CAS TV YouTube channel



Find practical lessons at Geek Gurl Diaries



and Alan O'Donohoe. It is a one-stop shop for a short burst of professional development.

Learning with online video is now so popular that whole courses are developed for video delivery, for example on the Open University's FutureLearn platform, or Harvard and MIT's edX. In the last few years, this MOOC (Massive Open Online Course) structure has

“ VIDEOS CAN HELP US AND OUR STUDENTS THROUGH ALL STAGES OF LEARNING

moved towards developing educators as well as teaching students. The Raspberry Pi Foundation has a number of courses on FutureLearn, where you can learn online at your own pace (helloworld.cc/rpifuturelearn), including *Programming pedagogy in secondary schools*, *Teaching physical computing to 5-11-year-olds*, and *Object-oriented programming in Python*. The MOOC model also includes plenty of social interaction, with the FutureLearn platform promoting lots of discussion points and plenty of opportunities for comments.

Content for students

One of the best features of online video is that it can be played over and over, slowed down, or paused. There were days when I wished I could replay my teachers' explanations, especially when I was revising! Online videos can also be used to help supplement the teaching of difficult concepts.

It can sometimes be difficult to understand how the ALU works, for example, and then explain it to someone else, or answer an exam question on it. Thankfully, the team behind the YouTube Crash Course channel has created an entire series on computer science (helloworld.cc/crashcourse), supported by PBS Digital Studios. The videos break down concepts into ten-minute chunks of explanation, with animations to show what is happening. Another great channel is Computerphile (helloworld.cc/computerphile), which boasts videos “all about computers and computer stuff”, and is really fun to watch. Resources like this make great homework or revision exercises, and give students the extra support they might need.

Skills-based videos are a great way to supplement student and teacher development, especially in programming. When I'm trying to work out how to write a function in an unfamiliar language, I'm often drawn to find the answer through online video. I learnt lots about Python thanks to Trevor Payne's video tutorials. They start off with the basics of computer programming and expand to cover broader and deeper topics (helloworld.cc/pythonvideos).

Embracing video

While teaching in a secondary school in Dagenham, UK, I noticed two things: most of the students who had chosen my subject were boys, and they would often search for supplementary material — not on Google, but on YouTube. I decided that I would start to create online video content that demonstrated my passion for computer science and digital making, as well as what I was learning or teaching at the time. My channel, Geek Gurl Diaries

(helloworld.cc/geekgurl), was a real labour of love outside the classroom. Although I've taken a break from producing video content for Geek Gurl Diaries, I encourage you as a computing education practitioner to try creating some educational videos yourself. You'll help your students even more than you already do, and

without realising it, you'll also help many more learners around the world.

When I think back to how I learnt about computer science, I know that video played a key role. When I think about my continued learning in this field, I know that I turn to video first to find what I am looking for. When it comes to teaching, I make videos for others to learn from. Let's embrace online video — even though I'm sure we will never be able to compete with the original title: *The Computer Programme*. (HW)

Carrie Anne Philbin is the director of educator support at the Raspberry Pi Foundation, a Computing at School board member, author and YouTuber (@MissPhilbin).

Find free, high-quality educational videos for everyone on the Crash Course YouTube channel



SMELLY CODE

Are we passing on best practice when we teach block-based programming to primary school pupils?

When we teach literacy in primary school, we use high-quality texts to model the features we would like our pupils to learn. In poetry, we demonstrate alliteration and onomatopoeia through the work of popular children's poets. When teaching programming, do we do the same? Do we know who the good programmers are in the Scratch community? Are we even sure what good-quality programs look like?

What is smelly code?

Smelly code is code that displays some feature that indicates an underlying problem with the program. The program may do some things correctly, but good code should be of high quality, and readable, too! Unreadable or convoluted code is much more likely to contain subtle bugs, hiding in those whiffy places.

Several researchers have investigated programs from the Scratch community, and concluded that most are smelly! One

researcher looked at more than a million programs, while another checked 250,000 (see box for links). One of the research teams concluded that some of the things we want children to learn and make progress in for programming were lacking in the code sampled.

Don't worry — the researchers did not check all this code by hand! They have programs that do their checking for them. There is also the fabulous Dr. Scratch (drscratch.org) that you can use to analyse your own code.

Names matter

One example of a bad smell is a sprite that has not been renamed, and still uses its default name. Why is this important? Renaming a sprite might make our code easier to read, and indicates that we are thinking about what our program is doing. So, if the Scratch Cat is the narrator for an animation, rename it Narrator. If the Elephant asks questions in a quiz, call it Quizmaster.

Variables, broadcast messages, backgrounds, and costumes need renaming too. Just calling a variable 'a', or 'variable', makes our code harder to read and change. Children (and professional programmers) can work on a program over several weeks, or take an old program as the basis to start writing a new one, so this matters a lot.

Superfluous stuff

Another bad smell is caused by redundant stuff: code blocks, variables, sprites, and messages left lying around which don't do anything. If children added a casual full stop or a random word into a sentence, we might question it. Sometimes we do pull code apart as we are debugging, and leave code fragments at the side as we work out what the problem is, but when we save and share our code, we should tidy it up.

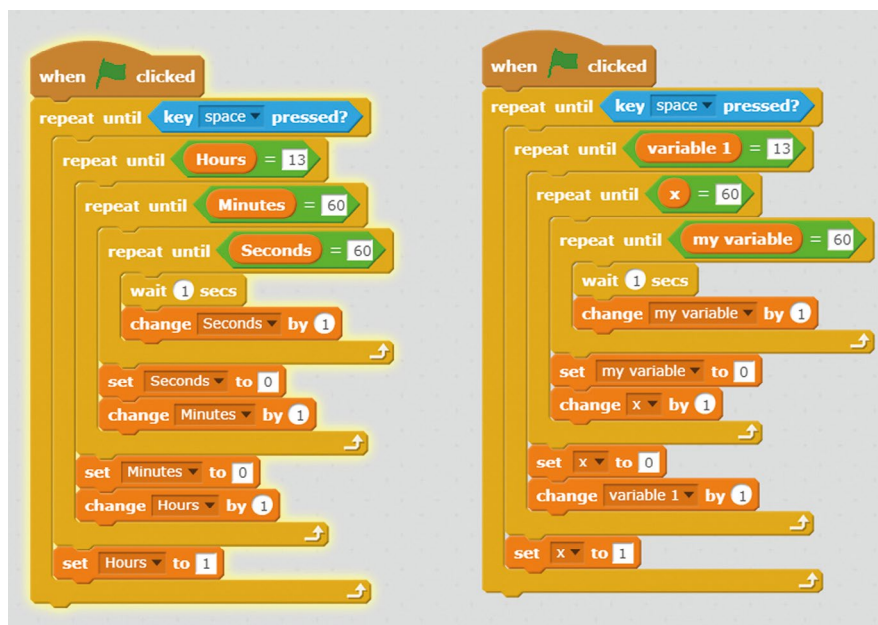
The researchers found other bad smells too, such as the same code being used for different sprites, and within the same sprite. Some repeated code for a single sprite can be implemented more elegantly, for example by using a loop. We can reduce repeated code by using clones, or by controlling the program using a main thread of commands, and synchronising activity through broadcasts and other techniques.

One of the smelliest aspects found by both researchers was very long scripts. Long scripts can be very hard to understand and debug, and often imply a lack of decomposition and design.

Make your own blocks

An aspect of Scratch that was missing from the sampled scripts was custom blocks. These purple blocks can be programmed to do a particular job for a sprite. Once a custom block is made, we can reuse it over and over again, and if it needs changing, we only need to change it in one place.

Using custom blocks demonstrates that we are breaking up our solutions into bite-



■ There is a bug in one of the scripts. Which is the easiest version to understand and debug? Thanks to Barefoot Computing for the original project. Code remixed by Jane Waite

■ Predict what this seaside script will draw. Code written by Jane Waite, inspired by the fantastic ScratchMaths

FURTHER READING

Dr. Scratch:
drscratch.org

ScratchMaths:
helloworld.cc/scratchmaths

Find your CAS regional centre:
helloworld.cc/casregional

Find your CAS Hub:
helloworld.cc/cashub

BCS Certificate in Computer Science Teaching:
helloworld.cc/bcscertificate

Scratch research:
helloworld.cc/scratchresearch1
helloworld.cc/scratchresearch2
helloworld.cc/scratchresearch3

sized chunks. Learning to spot potential custom blocks and work with them is a very important habit that gets children ready to learn more about procedures and functions in text-based languages.

Forever or until?

Another feature missing from the code was conditional repeats, or conditional loops. These commands allow us to control how a repeat ends based on another aspect of the program: for example, repeating an action until the space bar is clicked, or repeating something four times. Conditional repeats are important, as they show that we are

thinking carefully about why and how we are looping. Someone reading the code can see clearly how the loop will end.

Do you naturally model code using 'forever' commands, rather than 'repeat until', or 'repeat x times'? Does evolving rather than designing our programs contribute to lots of 'forevers' and not many 'untils'? The use of forever loops is seen by many experts as a bad habit.

Get ready before you go

Something else that is often missing from scripts is resetting stuff: setting variables to a start value, resetting sprites to a start position, clearing up any shapes we have drawn with the pen, and so on. This is sometimes called initialisation, and it is an important idea. If you do not include initialisation in your Scratch code, and you run it twice in the same sitting, the values and states from the end of your last run will become your subsequent start points. This can be very confusing, and you can't be sure what your program will do.

Creating programs that always do the same thing whenever you run them is a very good habit. In fact, some code needs initialising to work at all. Learning about initialisation is an important part of preparing children for the transition to text-based programming.

In your scheme of work, does the example code you use reset sprites to a start position? Do you explicitly teach resetting variables to a start value? Are there targeted tasks and guided exploration activities in your planning that reveal the importance of initialisation? Whatever form of scaffolding you use to teach programming, is initialisation covered?

So what, and what next?

Does bad-smelling code matter? There is plenty of research that says it does. If you give novice programmers smelly code, they will either have a poorer understanding of what the code does, or find it more difficult to modify the code, depending on the particular type of whiff!

I don't have all the answers, but there is emerging research in this area. I think that as a teaching community, we are ready to talk about what makes good-quality block-based programs, and to look at what kinds of habits we are exemplifying in our teaching.

How about investing in your CPD? Try a free online programming course, such as *Teaching programming to 5–11-year-olds*, at helloworld.cc/programmingcourse. If you're in the UK, ask your local CAS Regional Centre about programming courses. Find out more so you can help your pupils to write lovely, sweet-smelling code. (HW)




JANE WAITE

Jane is a research scientist at the Raspberry Pi Foundation. Her interests include using design in primary programming, semantic waves, PRIMM, and migrating to online teaching using ABC (@janewaite).

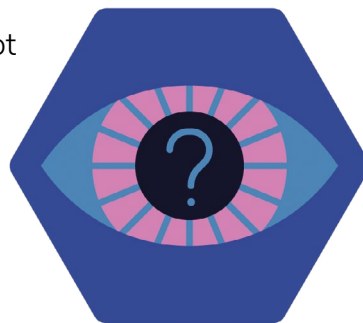


CHALLENGE MISCONCEPTIONS

- 104** ADDRESSING STUDENTS' ALTERNATE CONCEPTIONS IN COMPUTING
 - 106** ASSESSMENT FOR LEARNING
 - 108** WHAT'S IN THE BOX? METAPHORS AND MISCONCEPTIONS
 - 111** MULTIPLE CHOICE
 - 112** TACKLING NOVICE LEARNERS' NAIVE CONCEPTIONS IN INTRODUCTORY PROGRAMMING
- 

From their very earliest experiences of computing, pupils can easily develop alternate conceptions about a concept, including what it means and how it can be applied. This underdeveloped understanding, if left unchecked, can undermine students' understanding and their ability to grasp concepts in the future.

As an educator, it is therefore important for you to understand the common misconceptions that students are likely to encounter, many of which are well documented. Identification of misconceptions is an important step, as is understanding their impact; while many alternate conceptions can impede future understanding, others may be valid steps in a learning journey that can be addressed later on. There is a range of techniques you can use to help challenge or avoid alternate conceptions where needed. In particular, concept mapping can be used by pupils to connect and define concepts visually, while peer instruction can be used to challenge students and engage them in dialogue, which can help uncover and address misconceptions.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:** challenging misconceptions
- **What the research says:** asking the right questions
- Metaphors and misconceptions
- Using multiple-choice questions
- Misconceptions in introductory programming

ADDRESSING STUDENTS' ALTERNATIVE CONCEPTIONS

Addressing misconceptions in computing develops students' understanding, development and confidence

Alternative conceptions (often referred to as misconceptions) are beliefs commonly held about a concept by students that are overly simplified or inaccurate. Where these beliefs contradict with reality or accepted scientific understanding, they can cause confusion and affect students' efficacy and, ultimately, their performance.¹ It is therefore vital that computing educators understand commonly held alternative conceptions, how they arise, and how they can be addressed.

Although there is little research that has evidenced this, we suspect there are a number of alternative conceptions around computing. We must therefore also look to research in other subjects that form the traditions that underpin computing, in particular maths, science, and engineering. Some psychologists claim that alternative conceptions can be very persistent.²

In presenting learners with accurate conceptions that challenge their existing understanding, a state of "cognitive disequilibrium"³ is reached, in which

learners must reconcile the conflicting pieces of information. While this creates an opportunity to replace an alternative conception, learners may choose to discard accurate information that doesn't fit with their existing mental models. Educators therefore need to be aware of common alternative conceptions that their learners may hold. They should develop a range of strategies that support learners through their misconceptions, encouraging them to recognise them as wrong, but without labelling them in this way.

Sources of alternative conceptions

According to Piaget, learners build new understanding by combining experience with existing mental models. Educators help facilitate this learning by providing learning experiences and supporting learners to integrate the experience with their existing understanding. An alternative conception can arise in a number of ways when learners' experiences and their mental models interact in different ways. Research⁴ from science education proposes five categories of alternative conceptions:

- **Preconceived notions** involve learners making intuitive conceptual leaps based on their everyday experience. They do not have sufficient relevant experience of a concept or phenomenon, so they use their existing experience to fill in the gaps. For example, learners who are used to Scratch, which automatically handles concurrent execution of code, may expect similar behaviour from Python.

SUMMARY

Alternative conceptions (commonly known as misconceptions) can develop when new knowledge conflicts with a learner's existing mental models.

Alternative conceptions can be categorised as:

- **Preconceived notions** are intuitive but inaccurate leaps made about new ideas, based on existing knowledge
- **Non-scientific beliefs** can arise when a learner's mental models have been informed by non-authoritative sources and are counter to accepted science
- **Conceptual misunderstandings** occur when instruction fails to challenge existing mental models and learners attempt to resolve these independently with mixed success
- **Vernacular misconceptions** occur where new terminology is the same as that used in another context or everyday language, but with another meaning

- **Factual misconceptions** derive from false facts or information that have been assimilated into memory without being challenged

Become familiar with commonly occurring misconceptions:

- Review existing research into alternative conceptions in computing
- Reflect on your own experience
- Share common alternative conceptions with your peers and the community

Identify alternative conceptions through:

- Varied opportunities for classroom talk
- Diagnostic multiple-choice questions

Effective ways to address alternative conceptions include:

- Constructing individual or group concept maps
- Reach consensus around a concept using peer instruction

- **Non-scientific beliefs** can arise when a learner's mental models have been informed by information provided by a non-authoritative source, counter to accepted science. An example from programming might be the 'superbug'⁵ — a belief that a computer possesses

conceptions that their learners might develop. This reflection might include:

- Reviewing the research about common alternative conceptions in programming^{6,7} (there is limited research into wider areas of computing)

LEARNERS MAY DISCARD ACCURATE INFORMATION THAT DOESN'T FIT WITH THEIR EXISTING MENTAL MODELS

innate intelligence that can go beyond what it is told to do, which causes the learner to have unrealistic expectations of the machine.

- **Conceptual misunderstandings** occur when learners experience instruction of insufficient depth. The experience fails to challenge existing mental models and confront conflicts. Learners attempt to resolve these independently, with mixed success. In computing, we regularly use analogy to unpack and explain abstract concepts. Learners may be left equating the analogy and the concept, unless educators spend time distinguishing between them (see page 46).
- **Vernacular misconceptions** occur when terminology and symbols have different meanings in different contexts. Learners can project meaning from their existing mental models onto the new experience, causing confusion. For example, computing shares many terms with maths (variables, graphs, etc.) that have different meanings in the two subjects.
- **Factual misconceptions** derive from false facts that have been assimilated into memory without being challenged. An example is the common belief that Apple Macs are immune to viruses.

Identifying alternative conceptions

The first step in mitigating alternative conceptions is identification. Educators need an awareness of potential or common alternative conceptions, and the ability to recognise them in their learners. Before teaching new material, it is valuable for educators to reflect on alternative

- Reflecting on their own experience of teaching a topic, and any alternative conceptions that past learners exhibited
- Discussion with peers sharing their collective experience of alternative conceptions (as well as strategies to address them)

Educators also need strategies to spot misconceptions as they occur. Some techniques include (but aren't limited to):

- Classroom talk and discussion: whether this takes the form of questioning, whole-group or peer discussions, providing opportunities for learners to express their understanding and listening to them is crucial
- Carefully designed multiple-choice questions can be used to probe learners' understanding and highlight alternative conceptions

Challenging alternative conceptions

Many alternative conceptions can be addressed with relative ease during instruction, and may even be corrected by the learner. However, many will require more work, as the learner already holds a model that they will probably resist replacing. To address these persistent alternative conceptions, learners need the opportunity to construct (or reconstruct) an accurate model. This opportunity can be provided in a number of ways, including:

- Creating concept maps to help externalise learners' understanding and highlight their alternative conceptions

REFERENCES

- ¹ Kallia, M. & Sentance, S. (2019). Learning to use functions: The relationship between misconceptions and self-efficacy. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (752-758). helloworld.cc/misconception1
- ² Eggen, P. & Kauchak, D. (2001). *Educational Psychology: Windows on Classrooms*. Pearson. helloworld.cc/misconception2
- ³ Wadsworth, B. J. (1971). *Piaget's theory of cognitive development: An introduction for students of psychology and education*. McKay. helloworld.cc/misconception3
- ⁴ Davis, B. G. (1997). *Misconceptions as barriers to understanding science. Science teaching reconsidered: A handbook*. The National Academies Press, 27-32. helloworld.cc/misconception4
- ⁵ Pea, R. D. (1986). Language-independent conceptual "bugs" in novice programming. *Journal of educational computing research*, 2(1), 25-36. helloworld.cc/misconception5
- ⁶ Sorva, J. (2018). Misconceptions and the Beginner Programmer. In: *Computer Science Education: Perspectives on Teaching and Learning in School*, 171. helloworld.cc/misconception6
- ⁷ Swidan, A., Hermans, F., & Smit, M. (2018). Programming Misconceptions for School Students. *Proceedings of the 2018 ACM Conference on International Computing Education Research* (151-159). helloworld.cc/misconception7
- ⁸ Sadler, P. M. et al. (2013). The Influence of Teachers' Knowledge on Student Learning in Middle School Physical Science Classrooms. *American Educational Research Journal*, 50(5), 1020-1049. helloworld.cc/misconception8

before the accurate model can be re-assimilated (see page ten).

- Peer instruction, to help learners explore and challenge their own mental models as they collaborate to form a shared understanding around a concept (see page 56).

In approaching learners' alternative conceptions, there is no single approach that will work for all learners. However, educators who are able to identify their learners' common alternative conceptions are better equipped to support their learners' understanding⁸, development, and confidence. (HAW)

ASSESSMENT FOR LEARNING

Diagnose your students' learning needs by asking the right questions

STORY BY Oliver Quinlan

Mention assessment in schools and most people think of tests. Memories of creaky exam halls and regulation stationery may come first, but there are several developments in assessment that can change how we think about discovering what students have learned.

Back in the late nineties, Black and William challenged educators' views on assessment with their seminal work *Inside the Black Box*, which popularised the idea of 'assessment for learning'. They suggested an approach which brought assessment into the learning activities themselves. When this is executed well, students are given feedback as they learn and have the opportunity to act on this feedback immediately. It's about

the teacher not just delivering and then assessing later, but regularly checking for understanding and adapting their teaching. It's also about the learners getting regular insight into how they are learning and, crucially, having an opportunity to act on the feedback they get and ensure that they are making progress. The assessment is designed to serve the students' learning, and not to certify that they have achieved a set standard.

Assessment revolution

This work led to many developments in schools, with the government in the UK taking up 'assessment for learning', teachers being trained to regularly assess students' learning within lessons, and students being provided with feedback to act on as they

learn. More recently this approach has been supported by John Hattie's meta-analysis of influences on achievement in schools, which puts feedback at the very top in terms of the size of the measured effect (helloworld.cc/hattie).

Truly effective formative assessment is not just about finding out whether students have 'got it' yet. It's about understanding how they are thinking about a topic, what misconceptions or naive understandings they have, and how your teaching and activities can be adjusted to address them. Given the abstract nature of computing, the potential for misconceptions is very high.

Diagnostic questions

Edtech company Diagnostic Questions is seeking to address this with its online assessment platform. Its assessments look familiar at first: multiple-choice questions with four answers. Multiple-choice questions have been given a bad reputation by some educators, but their quality comes down to how the questions themselves are put together. If a question has a correct answer and three laughably implausible answers, it won't be a useful tool. However, if each answer represents a different level of understanding, or a common misconception, then the answer the student gives is useful, even if it is the wrong one. Imagine being told after an assessment not just who got the right answers, but why those who got it wrong did so, and potentially the misconception you need to address for each group of students.



■ Assessment for learning is designed to serve the students' learning - not to certify that they have achieved a set standard

In 2017, Diagnostic Questions started working with Computing at School, the Centre for Evaluation and Monitoring, and later, Cambridge Assessment to bring this approach to computer science. Project Quantum is a project exploring the potential for crowdsourcing diagnostic questions for the computing curriculum, and using them for formative assessment. They have been encouraging teachers to add questions to the platform, and use the questions already available with their students to help them better understand their learning. This project brings the potential of a relatively

have understood those things. However, real-life programming is often more about the elegance of the design of a solution. What if the most accomplished student doesn't use the things you have on your checklist? This is a particular problem when using wide briefs for tasks, or even open-ended projects.

Comparative judgement is a field relatively new to education practice, which offers huge potential for solving this problem. It's based on well-established research showing that humans are relatively poor at making

FURTHER INFORMATION

- Inside the Black Box: helloworld.cc/blackbox
- Diagnostic Questions: diagnosticquestions.com
- Project Quantum: helloworld.cc/quantum
- No More Marking: nomoremarking.com

“CREATIVITY AND PROBLEM-SOLVING ARE KEY TO COMPUTING, BUT DIFFICULT TO ASSESS USING TRADITIONAL APPROACHES

new approach to assessment to computer science teachers, and a chance to better understand how students make sense of difficult topics.

Comparative judgement

Creativity, problem-solving, and original approaches are key to computing, yet these things are very difficult to assess using traditional approaches. It's very common in education to use criteria to assess how well students are performing. We might set them a programming problem and then tick off whether they have used a loop or an 'if' statement, showing that they

objective judgements about individual objects, but very good at making comparisons. Play a musical note to most people and ask them what it is and they will struggle. Play them two notes and ask them which is higher and they are likely to be successful. Repeat this several times, with a clever algorithm to keep track, present them with the right combinations, and you can come up with a scale. These rankings have been shown to be very reliable — even more so if you involve several people as judges.

This method has been shown to work well even for judging things for which we

don't have clearly defined criteria, such as looking at workings in maths and asking, 'Who is the better mathematician?' It can also be reliable even when the judges are peers at a similar level of proficiency. This opens up some exciting new ground for the assessment of skills that involves approaching problems in an open-ended way, and assessing complex skills without resorting to trying to predefine what successful students must do. Assessment organisation No More Marking is exploring this approach for English and maths in partnership with schools.

Assessment is all about students getting better at something, but it seems there are also some promising avenues for educators to get better at assessment. ^(HW)

WHAT'S IN THE BOX?

METAPHORS AND MISCONCEPTIONS

Teaching new concepts is more difficult if the related vocabulary isn't familiar to learners, but explaining variables using the box metaphor runs the risk of misconceptions

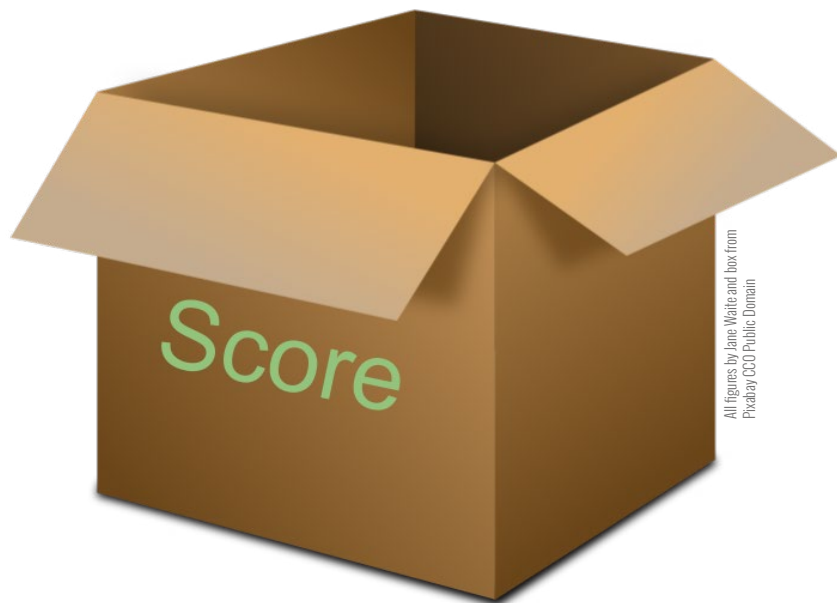
STORY BY Jane Waite, Feliene Hermans, and Efthimia Aivaloglou

The English Key Stage 2 (ages seven to eleven) Computing curriculum requires pupils to “work with variables”. Variables are the first step of students learning about data structures. Students will encounter many data structures, such as arrays, lists, and binary trees, as they progress into Key Stage 3 and beyond. But in primary, we just have to help them “work with variables”. Simply put, a variable is a reference — pointing to somewhere in memory — where a value is stored.

An example of this would be to store a date, say today's date, in memory location 1, and our birth date in memory location 2. To remember which date is where, we use variable names, such as Today's Date and Jane's Birthday.

The first mental model that pupils develop about a concept is important, as it can be very hard to change and, if it's not right, can lead to barriers to learning and loss of confidence. Variables are a fundamental idea for programming, so it's worth thinking carefully about what mental models pupils will develop for this concept.

'Variable' is a term that students might come across in a range of contexts (this is called topical word learning), such as its colloquial use referring to something that varies, or in other school subjects such as science or maths. However, each of these contexts has subtly different meanings than the computer programming concept of a variable.



All figures by Jane Waite and box from Pixabay (CC0 Public Domain)

■ Figure 1 Create a variable: make and label a box

One way to explain computer programming variables is to use a metaphor. Metaphors often have a limited shelf life — they work for so long and then they need to be replaced, as there will be differences between the working of the metaphor and the concept we are linking it to. This is true of the metaphors and methods used to explain programming variables. A common metaphor used to explain variables is the box metaphor. This seems like an attractive way to make a new word more familiar.

How does the metaphor work?

The box metaphor goes as follows:

1. As shown in **Figure 1**, the box is made and a label is placed or written on it. The box is the variable, and has a variable name (Score) — we've created a variable!
2. As shown in **Figure 2**, values can be placed in the box — simply put, these might be numbers, text, or yes/no type values, and can be the initial value or updated values
3. As shown in **Figure 3**, values can be retrieved from the box by looking inside



■ Figure 2 Set, store, change, or update values- put them in the box

So why might things go wrong?

Firstly, pupils might think that more than one value can be in the box at any point in time. For example, if 1 is added to a score, they might think that the original value and the 1 are now in the box. They might also think that to find out the current value of the variable, they need to perform some kind of action to add up all the numbers in the box. Similarly, if a text value is put in a box, pupils might think that the value that was there before is still in the box, and that there are lots of text values. This might lead them to create a mental model where a variable has not just one value, but lots of values, and to believe that they can also access the old values.

A variable, however, doesn't hold multiple values; a variable references one place in memory and holds just one single value at any point in time. The old values that were there before can't be retrieved. Simply put, as a variable is written to, the old value is overwritten.

What did the research find?

At a large museum in Amsterdam, the research team set up an experiment. They taught nearly 500 participants, two-thirds of whom were children and one-third parents, about variables using Scratch. Half of the participants were taught using the box metaphor and half using a label metaphor. All the participants were then asked some questions. One question specifically targeted whether

“ IF A TEXT VALUE IS PUT IN A BOX, PUPILS MIGHT THINK THAT THE VALUE THAT WAS THERE BEFORE IS STILL IN THE BOX

participants thought a variable could hold more than one value. Those who were taught using the box metaphor were far more likely to have the misconception that variables hold multiple values for



■ Figure 3 Select, retrieve, or get values- look in the box

COMMON MISCONCEPTIONS

The misconception of thinking that more than one thing can be in a variable can be reinforced if an unplugged activity is used to explain the metaphor. For example, if beanbags are thrown into the box to represent adding individual points to a score, or if pieces of paper are thrown into a box to represent the next answer to a question, learners can see and may even have physically enacted this idea of having multiple values in a variable.

A second misconception is that when a value is retrieved from a variable, the value is no longer in the variable, or no longer in the box - it has been

taken out. As shown in Figure 3, you can attempt to reduce this misconception by showing a telescope to represent retrieving the value, and using the phrase 'look in the box'. But it's easy for children to think they can actually take the value out of the box, particularly if an unplugged activity involves taking beanbags or pieces of paper out of a box to see what the value is.

A variable doesn't lose its value when the value is retrieved, but continues to hold the value, whether the value is retrieved or not. The value persists until the value is overwritten by a new value, or until the value is deleted.

▶ text values. However, for a simpler question on what value had been first stored in the variable, the box metaphor participants performed better than the label metaphor group.

It seems that the shelf life of the box metaphor was reached as soon as more than one value had been saved to the variable, but starting with the box metaphor was a better way to get the basic concept of saving the first value into a variable. If you want to find out more about the research, led by Felienne Hermans and Efthimia Aivaloglou at Delft University of Technology and the Open University in the Netherlands, read about it in Felienne's blog (helloworld.cc/boxmetaphor), which includes a link to the research paper.

What other metaphors or methods are there to explain variables?

Labels: A variable is a label. Here, the variable is explained as being a label, like a temperature or the name of a person. In the Dutch research on variables, they explained this by always showing 'x is 5', compared to 'x contains 5' in the box metaphor.

Hoops: This is similar to the box, but rather than using a box, a large hoop is used. This means you can more easily see the values in the variable, and you can throw things into it, such as beanbags representing points for a score. Can you see what misconception this might lead to?

Thin tubes: Rather than a big box, a thin tube is used, with cards used to represent the variable. The cards fit snugly in the tube, so only the last variable to be put in the box can be seen. This still implies that there are lots of values in the variable, and perhaps worse still, that there's an ordered stack of previous values in the tube.

Paper, pegs, and a washing line: Write the value on a piece of paper, and write or fix the variable name on the peg. Peg the

value to the washing line when a value is set, unpeg when a value changes, and peg up the new value.

Mini whiteboard: Write the variable name in small letters at the top of the card. Write the value in big letters on the board. When a new value is set, wipe out the old value and write the new value to replace it.

There are other metaphors and methods — which do you use? You can share your ideas on Felienne's blog — she would love to hear about your approaches.

Which is the best?


As far as we know, there has been no research to compare all the metaphors and methods available to teach variables, although Felienne's team are looking at this now. Nor has there been research to look at the long-term impact of the different approaches for different programming languages, nor of the long-term impact on pupil confidence of using metaphors with a short shelf life.

Jane personally likes the whiteboard method, as primary pupils are familiar with using whiteboards. They are often available in primary classes, and using them to keep a record of scores is common in games and quizzes. Using quizzes to introduce variables is a common context, so it's easy to link this method from an unplugged method to a programming activity.

Using metaphors and unplugged methods for helping children learn concepts seems like a good idea, but there can be misconceptions lurking. Misconceptions may be acceptable for a while — we may decide that being able to start the learning process off from an easy starting point is worth the problem of having to address the misconception later. However, as teachers, we need to be aware of this, so that we plan for the unlearning and reteaching, and are aware of potential

“ USING METAPHORS AND UNPLUGGED METHODS SEEMS LIKE A GOOD IDEA, BUT THERE CAN BE MISCONCEPTIONS LURKING

barriers to learning that we may have put in place because of the mental models we have helped learners to develop.

If you would like to find out more about metaphors and misconceptions, you can try out the Barefoot Computing Variables Unplugged activity with your class, which uses the whiteboard method to explain variables (helloworld.cc/variablesunplugged). Or, if you're based in the UK, sign up for one of the primary professional development sessions with the National Centre for Computing Education, or talk to your local CAS Master Teachers. 

MULTIPLE CHOICE

Can you really use multiple-choice questions to assess programming and computational thinking?

Do teachers ask questions just for fun? Or do they ask questions to find out what their students know and do not know? Multiple-choice questions (MCQs) offer the advantage of being quick to mark, and have the potential for gamification. But how can MCQs be used to assess programming concepts and computational thinking?

Question design

All MCQs should be based on sensible design criteria. You can find lots of advice, based on very good research, about designing MCQs. They have been used for summative

TOP TIPS FOR MCQ DESIGN

- Assess one, and only one, objective
- Provide four response options
- Provide one, and only one, clearly correct response
- Do not use “none of the above” or “all of the above”
- Use distractors based on misconceptions and common errors
- Make all distractors plausible
- Give necessary and clear context first, if required; then, separately, ask the question
- Use images as appropriate to support context, or as alternative responses
- Keep sentences short so they are easier to understand
- Make all response options grammatically parallel
- Make all response options similar lengths

assessment in the context of medicine for many years. However, teachers tend to use them more formatively: to identify what needs to be taught or reviewed, and to help students improve. The box at the bottom left of this page lists some top tips for writing MCQs.

Distractors and challenges

The importance of good distractors (incorrect answers) should not be underestimated. Trying to predict what a student does not know can be challenging. When choosing the alternative incorrect responses to an MCQ, consider the types of misconception you have identified among your students. For example, if a student uses a box analogy for a variable, they may believe it can hold two values at the same time. A distractor that expresses that understanding would help identify students who need additional help before moving on. You should also consider common errors in processes. A student with a misconception about indexing may always be out by one, so a response incorporating this misunderstanding would also make a good distractor.

It's not difficult to write an MCQ that assesses students' ability to recall a fact. It's much more challenging, but not impossible, to write MCQs that assess application, analysis, or creation. With application, a question could be confined to a single step in a multistep processes. Analysis can be assessed by asking 'what if' questions, in which students predict what will happen next. An even more challenging assessment of analytical skills is to define a requirement, such as an output, and ask the student to select which program solution meets that requirement. Certainly, MCQs don't lend themselves to the creation of program solutions. However, they can be used in ordering exercises, where students are

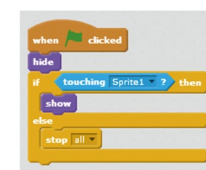


CYNTHIA SELBY

Cynthia is a senior lecturer and PGCE computer science tutor at the University of Southampton and a member of the CAS Project Quantum Content Group.

COMPUTING AT SCHOOL

EDUCATE • ENGAGE • ENCOURAGE



Identify what happens when the green flag is pressed and the sprite is not touching Sprite 1.

- A Shows
 B Shows and stops all scripts
 C Hides and stops all scripts
 D Hides

■ This question demonstrates assessment of tracing and predicting

Here are four instructions.

- 1 line 0 0 400 400 ;
- 2 background yellow ;
- 3 stroke green ;
- 4 line 200 0 200 400 ;

Here is an output.



Identify the ordering which will produce the output.

- A 3, 1, 2, 4
 B 2, 3, 4, 1
 C 4, 2, 3, 1
 D 1, 2, 3, 4

Based on original material from CS Discoveries by Coding.org (http://coding.org.uk/news/3)

■ This question shows how ordering can be used to assess the skill of sequencing

asked to choose the correct sequence of instructions to meet a requirement. Slightly simpler would be asking a question that just required the identification of a single instruction to fill a gap.

Getting started

Writing questions with a colleague is an excellent way to get started with writing MCQs. Receiving feedback on questions is the best way to improve them. As time goes on, remember to go back and review your questions. If one distractor is never being selected, you're missing an opportunity to test a true misconception.

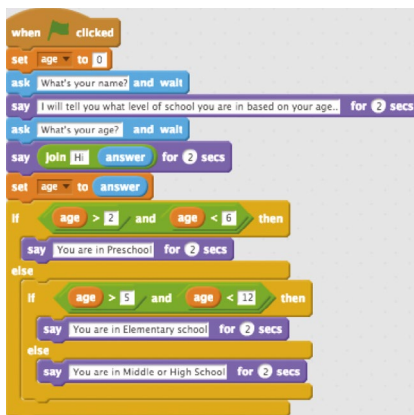
One of the simplest ways to get started with MCQs is to join a group and share your work. Project Quantum is a joint project to crowdsource computing MCQs. Find out more at helloworld.cc/quantum. (LHW)

TACKLING NOVICE LEARNERS' NAIVE CONCEPTIONS IN INTRODUCTORY PROGRAMMING

Shuchi Grover discusses common patterns of problems exhibited by students during their early programming activities

The naive conceptions of novice learners result in frequently observed sources of confusion across age levels and regions. While some aspects of novice difficulties studied over the decades are specific to text-based and object-oriented languages and programming environments (such as the confusion related to the = operator, which stands for assignment in programming and equality in mathematics), several others persist, even in the context of commonly used block-based programming environments.

Recent investigations suggest that learner misconceptions endure even when students are using child-friendly block-based interfaces. These graphical programming environments, inspired by the low floors of Seymour Papert's philosophy, are engaging. They afford a much more motivating programming experience than



■ **Figure 1** What does the programmer believe will be stored in Scratch's answer variable?

traditional text-based languages do, by allowing early programming experiences to focus on designing and creating, and avoiding issues of programming syntax.

This article describes several of the naive conceptions and difficulties that novice programmers (regardless of their age or grade level) exhibit in introductory programming settings.

The superbug

First, we hark back to the 1980s and the work of Roy Pea (doctoral advisor at Stanford University, and my mentor) on language-independent conceptual bugs. Roy and colleagues found that many such bugs in students' code in environments such as Logo could be traced to what he called a superbug: the belief that a computer can somehow interpret a student's intention with some sort of intelligent interpretive process.

Another key source of difficulties lies in a weak understanding of the temporal logic of programs and program states — the mental model of a program as a sequence of instructions, where actions at any point are the result of what has gone before (including values of variables, and other attributes of objects), and determine what happens next, unless code is placed in event handlers which are triggered by an action at any time.

These two shortcomings impact students' fundamental understanding of data and control flow in general, and in particular of concepts such as initialisation, loops, and how state changes with each iteration of the loop (including values of



■ **Figure 2** What are the roles of Counter and NumberOfTimes in this Scratch code?

variables), as well as how to create logical expressions in order to manage repetition and the conditional logic of 'if' commands.

Variables

Learners need to understand that variables are named values or quantities, and that their values can change over the course of a program. Some difficulties with variables in block-based programming environments are:

- A hangover from maths that leads learners to think variables stand for a mystery, unknown value and have names such as 'x' or 'y'. Research encourages teachers to speak of a variable as having a 'role' in a program, and having a meaningful name. Learners should be guided to make that leap.
- The metaphor of a variable as a box or placeholder leads learners to assume

that a variable can hold multiple values at a time. A teacher might target this misconception with a debugging exercise using buggy code, as in **Figure 1** (that's not the only problem with this code!).

- Despite syntactic simplifications and scaffolds such as colour and shape in environments like Scratch, learners need help understanding the semantics of variable use. What does it mean to control a loop with a variable, or an expression that contains a variable (**Figure 2**)?
- When does the value of a variable change? Many students will incorrectly answer the question in **Figure 3**, even after learning about **set** and **change** blocks in Scratch.
- Assigning a variable to another somehow links the two, causing a change in the value of one to result in a change in value of the other (even after the assignment).

Loops

In my work, I have found that students often struggle to distinguish between what goes inside a loop and what precedes or follows a loop. This is often the case when a first or last iteration is different from the 'inner' part that repeats. In addition:

- When there are multiple blocks inside a loop, instead of thinking of the loop as executing the entire sequence of actions, some students believe that each action is repeated separately before subsequent action(s) are similarly separately repeated.
- Some students struggle to understand that an expression involving the control variable of a loop can have different values in each cycle of the loop. They believe that a loop repeats the same set of actions, and expect loops to produce exactly the same output in every iteration.

■ **Figure 3** What is the value of the variable 'steps' after the following code sequence is executed?

```

set steps to 10
move steps + 10 steps
    
```

- Off-by-one, or fence post errors, in loops have been well documented in several studies involving various age groups. Most middle school students in Israel, and those in two classrooms in California, incorrectly answered a question from the 2012 Israel National Exam shown in **Figure 4**.

- Many students, in my experience, also struggle to formulate a loop-terminating condition involving a Boolean expression.

Boolean logic

Students have been found to perform less well on logic-based questions on the Advanced Placement Computer Science A Exam in the USA than on any other type of question. The Boolean AND/OR operators are often mistakenly interpreted as they are in the English language. Specifically, students tend to misinterpret the OR operator as true when one of the operands is true, but not both. We found this to be the case with middle school students learning Scratch. In order to address this, we should encourage students to create Boolean expressions to model real-world phenomena.

For example, write an expression to model the situation in which a car gives a warning beep if:

- The driver's seat belt is not locked
- The passenger's seat belt is not locked
- The car is being driven

How can teachers help?

The first part of the solution is awareness. It is a rare teacher who will not attempt to address these student difficulties if they are aware of them. Secondly, most misconceptions can be addressed if students are confronted with them explicitly. I have long argued for a balanced pedagogy of programming that combines active exploration approaches with methods of scaffolding that enable knowledge construction and deep understanding of targeted concepts.

Besides the ideas and examples shared, other useful strategies that have been known to work include demonstrating flow of control through visualisations and code tracing; having students trace code, including but not limited to 'unfurling' loops that involve

```

set number to 0
repeat until number > 6
  say number
  wait 1 secs
  change number by 1
    
```

■ **Figure 4** What will the sprite say when the above script is executed?

variables whose value changes in each iteration and involve a termination condition based on that variable; and targeting known difficulties and misconceptions head-on through code comprehension, worked examples, and debugging exercises.

Lastly, conversational metaphors, though strong pedagogical techniques to introduce concepts in new learning contexts, should be avoided. Mapping conventions for natural human language instructions onto programming can exacerbate the problem. (HW)

FURTHER READING

- Grover, S., & Basu, S. (2017). Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 267-272. helloworld.cc/grover2017
- Pea, R. D. (1986). Language-independent conceptual 'bugs' in novice programming. *Journal of Educational Computing Research*, 2(1), 25-36. helloworld.cc/pea1986



SHUCHI GROVER

Shuchi is a learning scientist and computer science and STEM education researcher (@shuchig).

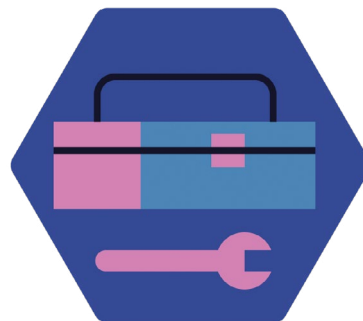


CREATE PROJECTS

- 116** PROJECT-BASED LEARNING
 - 118** HOW CHILDREN MAKE
DIGITAL PROJECTS: RESEARCH
FROM COOLEST PROJECTS 2018
 - 122** PROJECT-BASED LEARNING:
A PATH TO AGENCY
 - 124** A YEAR WITH DESIGN JOURNALS
- 

Pupils need opportunities to apply the skills, knowledge, and understanding they have developed, and project-based activities can be a great way to facilitate this. Projects give pupils a goal, an audience, and a brief to fulfil, for which they need to make autonomous decisions about the skills, knowledge, and tools they will need. Projects are a valuable context in which pupils can develop their design, analysis, and evaluation skills, as well as providing chances for them to collaborate. Projects that are rooted in learners' experience and environment are particularly powerful, as they allow them to solve the problems that matter to them, increasing their intrinsic motivation to learn.

Projects also help learners develop their skills and understanding beyond computing, as they involve them imagining, making, and sharing their ideas. Over the course of a project, learners will have to practise planning, organise their tasks to fit the time available, and communicate their idea and progress with stakeholders.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:**
project-based learning
- **What the research says:**
how children make digital projects
- Projects as a path to agency
- Using computing design journals



PROJECT-BASED LEARNING

A project-based learning approach helps learners to apply their programming knowledge to real-world scenarios

Project-based learning (PBL) is an approach to teaching computing in which the learning activities are organised around the design, creation, and evaluation of a digital artefact. Working as individuals or in small groups, learners deepen and consolidate their knowledge through hands-on, tangible experiences that allow them to reflect on their learning.¹ While a PBL approach can be applied across computing to develop a range of digital artefacts, here, we explore the specific benefits relating to programming.

Creating a strong project concept

Projects are typically split into three different stages: Imagine, Make, and Connect or Share. During the Imagine phase of PBL, the key to successful learning is choosing an appropriate project. Research shows that a successful digital project requires a well-researched idea, access to

available technology, and an appropriate level of skills.³

A strong project idea often has a personal dimension that aligns with the learner's own interests. The project aims to create something of value to them, or to solve a real-world problem that they deem important. In this phase, creating a storyboard, sketch, or design for the project helps the learner shape a realistic, visible project concept aimed at a particular set of users or the performance of a specific function.

Educators have an important role to play by designing thoughtful prompts to encourage project ideas. A good project prompt is brief and solvable, yet contains enough ambiguity so that the learner can "satisfy the prompt in their own voice".⁴

Supporting project development

Some aspects of the next stage of PBL, the Make phase, will be more challenging

for learners than others. Project-based learning is cognitively rich⁵ and requires both technical thinking about the code (and perhaps the hardware), and organisational thinking about the development of the project. One challenge for learners in PBL is the transfer of conceptual programming knowledge into the skills required to write their own programs. It can be equally challenging to manage time spent on the project, to ensure that there is progression from start to finish.

“**HANDS-ON EXPERIENCES CONSOLIDATE KNOWLEDGE**”

Educators can refer to previous activities, such as worked examples (page 90) and Parson's Problems (page 80), to support conceptual transfer. Multiple representations, such as side-by-side algorithmic and coded solutions, can help students identify patterns in the structure and generalise these to use in their own projects.

Tools such as individual or class checklists, checkpoints where teacher approval is required, and design notebooks for planning and reflection, can scaffold the development of learners' project management skills, as well as helping educators to keep an overall sense of progress.⁶

Sharing projects with others

Projects are made for people to use, and the final part of each iterative cycle is to Share. Testing, presenting, and eliciting

SUMMARY

Through project-based learning, learners can:

- Apply their existing computing knowledge to new situations
- Deepen their understanding of programming concepts
- Develop skills valued by employers in the workplace, such as planning, organisation, and communication²

Projects usually take place over a number of sessions and will typically be split into several stages:

- **Imagine:** Developing an idea of something to make, and planning the resources needed

- **Make:** Building and testing the digital artefact, with the goal of realising the original idea
- **Connect/Share:** Sharing the project with an audience to elicit feedback, and reflecting on what has been learnt during the project

In practice, the stages of PBL may not be implemented linearly. More often, they are part of an iterative process in which some stages are repeated one or more times.

PBL often involves using a mixture of software, hardware, and other physical material. The hands-on element of the project can help learners to more easily connect their digital artefact with their learning, and to better track their progress.

REFERENCES

¹ Papert, S. (1980). *Mindstorms: Computers, children, and powerful ideas*. New York, Basic Books. helloworld.cc/pbl1

² Menzies, V., Hewitt, C., Kokotsaki, D., Collyer, C., & Wiggins, A. (2016). *Project Based Learning: evaluation report and executive summary*. Education Endowment Foundation. helloworld.cc/pbl2

³ Quinlan, O., & Sentance, S. (2020). Ideas, Technology and Skills: A taxonomy for digital projects. In: Tangney, B., Byrne, J. R. & Girvan, C. (eds.) *Constructionism 2020, The University of Dublin. Trinity College Dublin, Ireland, May 26-29, TARA, 2020*. 357-365. helloworld.cc/pbl3

⁴ Martinez, S. L., & Stager, G. (2013). *Invent to Learn: Making, Tinkering, and Engineering in the Classroom*. Torrance, California, Constructing Modern Knowledge Press. helloworld.cc/pbl4

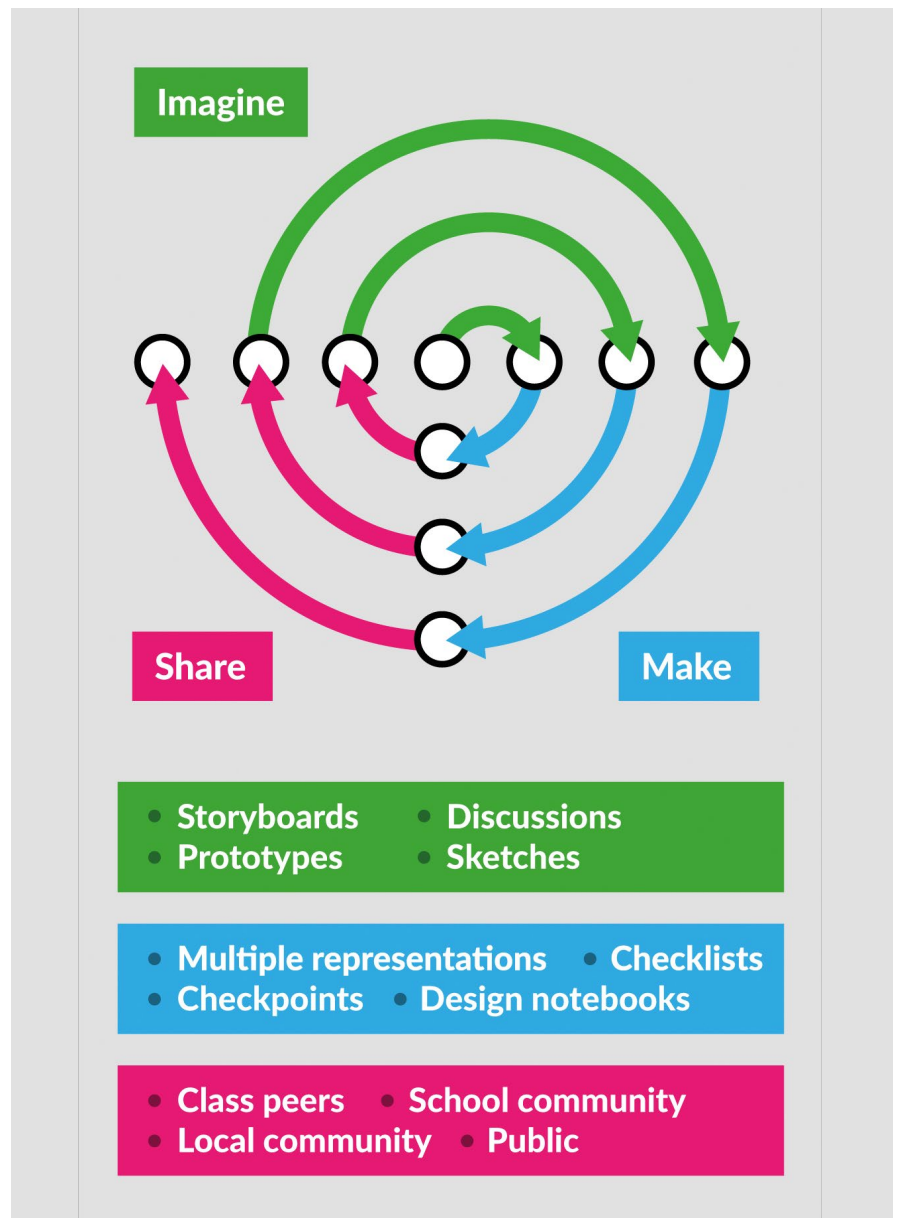
⁵ Vossoughi, S., & Bevan, B. (2014). *Making and Tinkering: A Review of the Literature*. Successful Out-of-School Learning: A Consensus Study, National Research Council Board on Science Education (1-55). helloworld.cc/pbl5

⁶ Fields, D. A., & Kafai, Y. B. (2020). Hard Fun With Hands-On Constructionist Project Based-Learning. In: Grover, S. (ed.) *Computer Science in K-12: An A to Z handbook on teaching programming* (75-82). Palo Alto, Edfinity. helloworld.cc/pbl6

⁷ Vossoughi, S., Escudé, M., Kong, F., & Hooper, P. (2013). Tinkering, Learning & Equity in the After-School Setting. Paper presented at FabLearn 2013, 27-28 October 2013, Stanford, CA. helloworld.cc/pbl7

user feedback helps learners to shape their project and identify potential improvements. The project audience can be:

- **In the classroom** — obtaining peer feedback from other learners
- **In the school community** — presenting at an assembly, or to older or younger learners
- **In the wider community** — obtaining feedback from parents or community groups
- **Public** — via the appropriate use of social media or the school website, or by entering projects into exhibitions, showcases, or competitions



Assessing PBL

Artefact-based questions (ABQs) support assessment for learning, because as learners reflect on their project, they also reflect on what they have learnt. ABQs can be about:

- **The project** — why was this project chosen? What was the original concept?
- **The code** — what does this section of the code do? Why have you taken this approach?
- **The process** — which part of the project was most successful? What changed during the project development?

- **The outcomes** — did you make what you planned to? How did user feedback shape your artefact?

At the end of a project, learners may still have areas of uncertainty and questions of their own. Connecting their learning back to the wider knowledge domain of the scheme of work or curriculum can help to situate their learning in context. Note that failure to make these links can reproduce inequities in access and opportunity.⁷ Why not try this approach out in your classroom, to help learners apply their programming knowledge to real-world scenarios? [\(HW\)](#)

HOW CHILDREN MAKE DIGITAL PROJECTS: RESEARCH FROM COOLEST PROJECTS 2018

STORY BY Lucia Flóriánová



■ Children often make projects that use technology to solve a real-world problem they care about

Coollest Projects is a science fair-style exhibition that takes place all over the world. Young digital makers bring their projects to share with others. Judges award some projects in each category, but the main emphasis is on sharing and learning from others.

In 2018, the Raspberry Pi Foundation (RPF) Research team conducted research at the international and UK events. We explored how children create with technology and what makes the events successful and special. Most importantly,

we found what makes children excited while learning, and what motivates them to take projects and learning to the next level.

Making digital projects

Problems and ideas, technology and skills

Projects had different stories. Some children started with identifying a problem that they wanted to solve and then thought of ways to achieve it. Others started with looking at their skills, or the technology they had available, and then found ways to use those in a project.

All children had to balance the relationship between three areas:

- 1 Compelling ideas or social problems they wanted to solve
- 2 Technology they had available
- 3 Skills they had or needed to develop to complete a project

Adult mentors helped them negotiate the balance and refine the ideas so that they were realistic and achievable.

Teams and roles

Some projects were created by individuals, and some by teams. In teams, participants either collaborated flexibly or took on specific roles. Sometimes the roles related to different technical aspects, such as a Python Programmer or a Hardware Engineer. In other teams, some children wrote the code and others took on non-technical roles, such as writing a story or music for a game, or developing artwork.

While some children were deeply immersed in the project creation, others took roles such as Games Tester, or just joined in to see what it was all about. Through attending alongside their friends and undertaking peripheral tasks, newcomers became familiar with practices and language within the community. It can be a beginning of their own journey towards becoming practitioners themselves. Lave and Wenger call this 'legitimate peripheral participation' (helloworld.cc/lpp).

Why Coolest Projects is so great

Agenda, drive, and learning

Children were very invested in the projects they brought to the event, and what they learnt. Many said they made more ambitious projects than they would normally do,

or took their existing ones to the next level. This was because they had set their sights on the event and wanted to create something special. Most children spent a long time on their projects and were very passionate about them. They often had a desired result in mind and worked hard to achieve it. This often required learning something new, such as programming concepts or languages. Children often mentioned that they didn't know how to do something, or how to fix a problem, but intentionally went on to find out how. Learning became a part of the fun and agenda; children learnt with enthusiasm because they decided they needed to in order to achieve the result they wanted. It was also made more effective because it happened in a context, and because children got to apply immediately what they had learnt in practice.

However, it's important for children to have an adult or materials that guide them and help them troubleshoot if they get stuck. It's also important that children don't set unachievable goals that could cause frustration.

Combining programming with other areas

Many projects were interdisciplinary and combined digital making with knowledge

IN THE CLASSROOM

Programming in schools turned out to be an important influence on many children at the events. A number of children said their first steps in programming were taken at primary school. It was through these early positive experiences that they became interested in digital making.

There are simple ways you can use what we found worked in the classroom. For example, sharing projects with each other at the end of a lesson or looking for interesting examples online is one way of getting inspiration. Motivating children to think about goals or benefits their project will have on a community can create a similar investment in learning. Presenting projects at a small exhibition for parents, school open days, or an assembly can help children feel important and build confidence.

in other areas, such as health, security, environment, or robotics. Children often created projects that used technology to solve a real-world problem that they noticed and cared about. Such an approach to computing can trigger



■ Through speaking about something they care about and know in detail, children gain confidence

▶ interest in technology and motivate children with different interests to get involved with digital making. It can help them see that technology can be used as a tool for achieving a particular goal, and isn't necessarily the goal itself. Projects also became a medium for interdisciplinary learning. Digital making can support other subjects beyond computing, bringing powerful tools to these areas.

Sharing and confidence

Even children who initially seemed shy presented their projects with confidence. Children understood the projects deeply, and focused on explaining how their projects worked and how they made them. This took away the awkwardness of talking to unfamiliar and diverse audiences. They benefited from talking about things that they were passionate about and seeing other people's interest. Through speaking about something that they know intimately and feel proud of, children experience success. They can gain confidence and become more comfortable with presenting in general.

Inspiration and the power of seeing other projects

Children often look for inspiration in projects done by others and examples of what they could do with technology and skills they have. Coolest Projects enables children to do just that; to look at projects at their own level and see what they could do. Seeing other participants' projects was the children's favourite part of both events. They spent a lot of time walking around the exhibition and asking other participants about their work. Some children said they felt more confident after seeing that their projects were similarly advanced or more advanced than others people's. Others found inspiration for improvements they could make to their own projects, as they were impressed by the potential that an idea similar to theirs could have.

You can read the full report at helloworld.cc/coolestresearch, and more research from the Raspberry Pi Foundation on page 357 of helloworld.cc/taxonomy.

You can find out more about Coolest Projects 2021, which was fully virtual, at coolestprojects.org. (HAW)

SCRATCH AT ALL LEVELS

Many of the projects we've seen were programmed in Scratch. They varied from full beginners' projects to very elaborate ones, and we were delighted to see that children with all levels of experience benefitted from showcasing their work.

In the **Pollution (or Shadow Neighbours)** game, two boys aged seven and eight created a simple Scratch game that served to raise awareness about pollution and its negative effects. It had a crab character catching plastic waste and was controlled by a keyboard.

The Rebel Quiz was a project programmed by two girls aged eight. They wanted to provide inspiring role models for girls by introducing strong female characters through the quiz. After a player answered a series of questions about their

personality and interests, they were told which Rebel Girl they were.

Dance Magic was a Scratch game with an element of physical computing. The simple dance game told a player where they should move. The player then had to press the right pressure pad with a right, left, up, or down arrow. The creators were inspired by a famous dance game.

The Random Games was an elaborate project created by a 16-year-old. He combined six different games of diverse genres into one computer program. Some of the games included a player playing a piano, or a character controlled on a 3D interface. This refutes the assumption that Scratch is only for beginners and younger children!



■ Seeing other participants' projects was children's favourite part of the event

(HW)

SUBSCRIBE TODAY

**FREE
IN PRINT**
for UK-based
educators



**FREE
PDF**

for anyone,
anywhere

Why subscribe?

- Teaching resources and ideas used by over 90 percent of our readers
- Exclusive news, research findings, and in-depth features
- Delivered to your door three times a year

Not a UK-based educator?

- Subscribe to receive the free PDF on the day it is released
- Read features and news at helloworld.cc

TO SUBSCRIBE VISIT:
helloworld.cc/subscribe



■ Learners with more agency will be more engaged in their learning

PROJECT-BASED LEARNING: A PATH TO AGENCY

Mark Calleja investigates project-based learning, a teaching method that gives students the tools to take charge of their own learning

Project-based learning is about asking your students to solve a real-world problem by designing and creating a project over a specific period of time. Getting started with project-based learning in your classroom requires a bit of a shift in thinking. Instead of providing an activity that demonstrates prior knowledge or acts as a unit assessment, students' learning is directly embedded into, and emerges from, the investigative and design processes they engage in, while solving the problem you've posed. That is to say, their project is the unit: it's the vehicle for teaching the knowledge and skills that students need to learn, as well as the assessment process that demonstrates their learning. I'm a huge proponent of teaching this way, for a whole host of reasons.

Creating solutions from the ground up

Agency: There's a massive difference for your learners between being asked whether they know the answer to a problem, and being asked to find a solution to one. The first question assumes knowledge (and thereby frames a lack of knowledge as failure) and has a narrow focus, while the second gives learners the room to be wrong or not know yet, and to develop real understanding and practical skills in a self-directed way.

Engagement: Learners with more agency are likely to be more engaged in their learning, too. A huge benefit of project-based learning is the scope it gives you to set relevant, real-world problems for your students. Being able to relate their learning to their own lives motivates students: when they see that they can apply new skills and

knowledge to other situations in their lives, they understand the true purpose of the work they're doing.

Universal skills: Project-based learning doesn't give students ready answers to a specific problem; it asks them to build a mental toolkit for understanding any problem, so that they can create solutions from the ground up. By enabling this in-depth learning, you equip your students for real life, letting them practise skills required in most industries today: taking initiative, working responsibly, decomposing and solving problems, collaborating in teams, and communicating their ideas clearly.

Giving the power back to your learners

When your learners are interested and engaged with their own learning, your job changes from passing on knowledge and



■ Learners need only teach themselves using the sheets and videos provided to make their ideas real

managing motivation to facilitating and inspiring. In project-based learning, you direct learners towards information instead of handing them answers, and you support them in creating something they didn't know they were capable of.

One way to get started with project-based learning is to use a bank of Python resources, developed with the National Citizen Service, that embody this idea (originally developed for a two-day hackathon - see resources box). The resources include 14 help sheets (with accompanying YouTube animations) that provide images of each basic electronic component, a simple wiring diagram with numbered GPIO pins, and gpiozero code snippets to execute its basic functions, all on one handy page.

The help sheets cover the most common simple components used in digital making, from LEDs to infrared motion sensors, cameras, Bluetooth remote controls, and beyond. There are also sheets that explain Raspberry Pi's Sense HAT's on-board sensors, joystick, and LED array, with accompanying code examples. We also made sheets that cover commonly used processes in digital making, such as playing sounds with Python, making a remote control with the Blue Dot Bluetooth app, and setting up a Raspberry Pi-based gadget to function automatically as your students intend it to, as soon as they power it up.

The intention behind the sheets is that you will first support your students through the design discussions they'll need to have before they start making things, and

show them the library of components they have available. When they know what functionality they want from their invention (and what's possible given their time and hardware constraints), learners need only teach themselves using the sheets and videos to make their ideas real. All of the code is broken into three sections, to make each Python script modular; students can simply combine the code snippets on the sheets to make larger scripts that create more complex functionality.

If you'd like to replicate our hackathon model, we have released the facilitator's guide to running the full two-day experience, complete with session timings, delivery notes, workshop slides, and a student support document called the Developer Manual in which participants can make notes and get discussion prompts and tips throughout their build process. (HW)



■ Why not run a hackathon with your own students?



MARK CALLEJA

Mark works for the Raspberry Pi Foundation, where he creates educational projects. He is a teacher, maker, hacker, and a Raspberry Pi Certified Educator (@M1st3r_G).

“ PROJECT-BASED LEARNING ASKS STUDENTS TO BUILD A TOOLKIT OF SOLUTIONS

RESOURCES TO SUPPORT PROJECT-BASED LEARNING

Guides to running hackathons are available to download at helloworld.cc/hackathon and helloworld.cc/componentsheets.

A YEAR WITH DESIGN JOURNALS

Introducing design journals to primary pupils, and how it went

The national curriculum for computing in England states that computing has deep links with science, and with design and technology. In my school, in both these subjects, pupils have exercise books in which they make plans and do designs. The national curriculum also states that pupils in Key Stage 2 (aged seven to eleven) should be able to design and write programs.

With this in mind, I decided to make sure that pupils spent time in the design stage of their computing projects and that I would facilitate this with the introduction of specific design journals.

So, I introduced a computing design journal which was simply an A4 book with plain pages. I decided on plain pages as I didn't want the pupils to feel constrained. I wanted them to have the freedom to develop their designs in any way they saw fit.

This article explains how we have been using design journals, and presents responses and views from the pupils' perspective. It covers three main aspects. Firstly, how the journals have helped children to better understand their computing projects. Secondly, how the journal has helped them to self-assess their work and indicate their levels of confidence. Finally, to help pupils improve their computational thinking and problem-solving skills.

Recent research papers have suggested that design can help novice and struggling programmers develop an awareness of what is doable, and also that design helps with self-regulation, much in the same way

as planning does in writing (helloworld.cc/designjournals).

The initial reaction

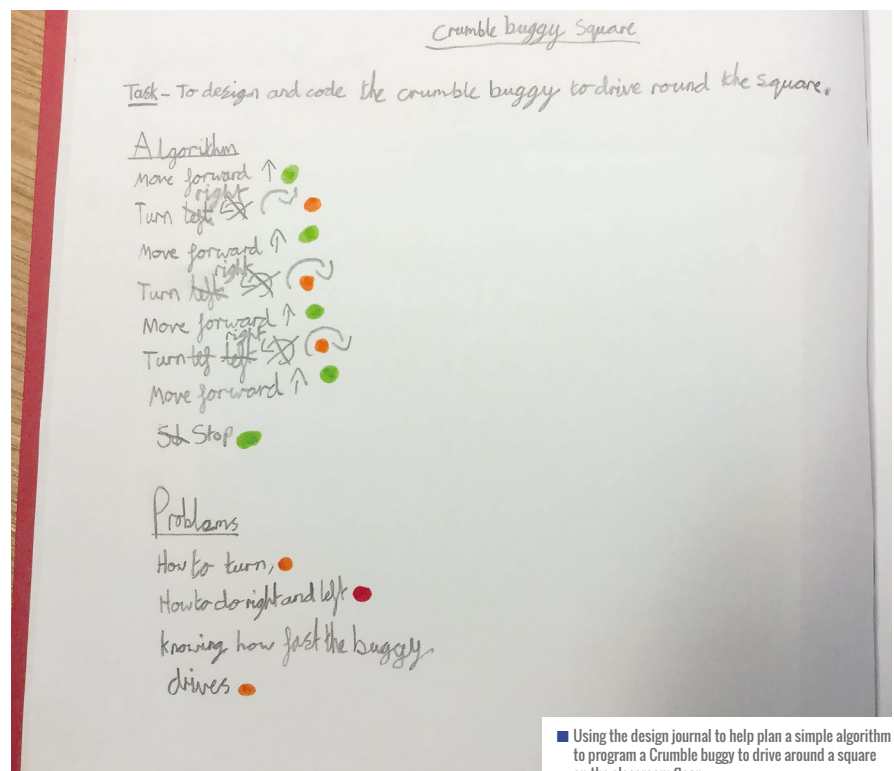
I decided to give design journals to pupils in Years 4 to 6 in my school (pupils aged eight–eleven). The initial reaction from the pupils was one of shock and delight. Many opened their books and questioned why they had blank pages. My reply was, “These are not exercise books like you get in English; they are design journals — they are a place where you can formulate your ideas and solve problems.”

The children took my word for it and were excited by the idea and intrigued by how they could use them throughout the year.

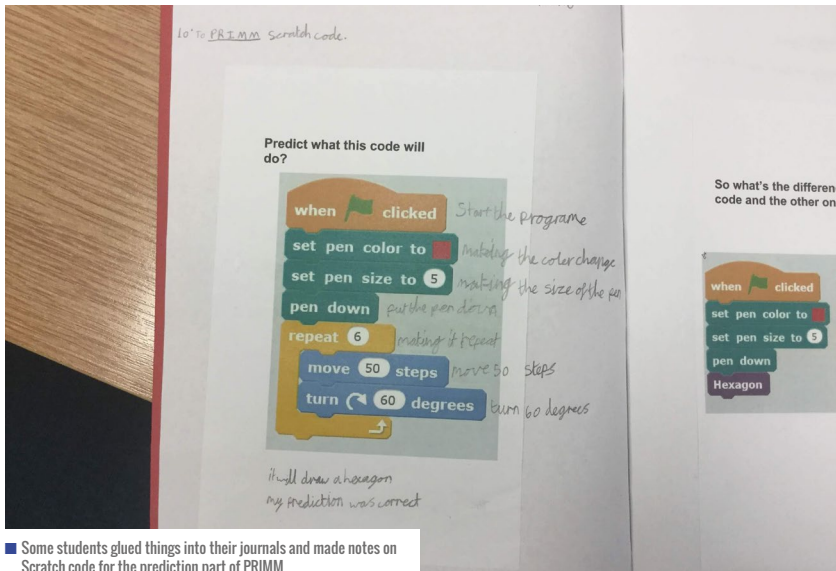
Design in computing lessons

I sold the journals to pupils as being different to exercise books. I explained that they were more like the sketchbooks they used in art or design. I didn't constrain the use of the journals, but wanted the children to engage with them and get to grips with the design process in computing.

So, whenever we did a computing project throughout the year, I would ask



■ Using the design journal to help plan a simple algorithm to program a Crumble buggy to drive around a square on the classroom floor



Some students glued things into their journals and made notes on Scratch code for the prediction part of PRIMM

the children to use their journal to plan their ideas, storyboards, and algorithms, encouraging them to make informal notes and sketches with anything that they thought might help them.

The journal was also a useful place for them to stick sheets in, for example, during the prediction part of PRIMM where they could write notes all around the sample code and self-assess their responses using

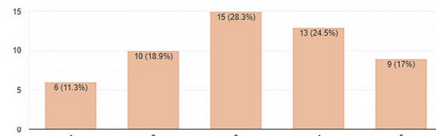
traffic light colours. They could use it if they had to figure out a problem, or design an algorithm, or look at what the levels would look like in a Scratch game.

It's fair to say that I'm still experimenting with what works and how these books fit into the workflow of the lessons. The images alongside this article highlight the versatility of the design journal and the reason I went for an A4 book with blank pages.

Year 6

On a scale 1 to 5 how useful has it been having a design book for Computing?

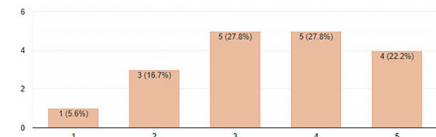
53 responses



Year 5

On a scale 1 to 5 how useful has it been having a design book for Computing?

18 responses

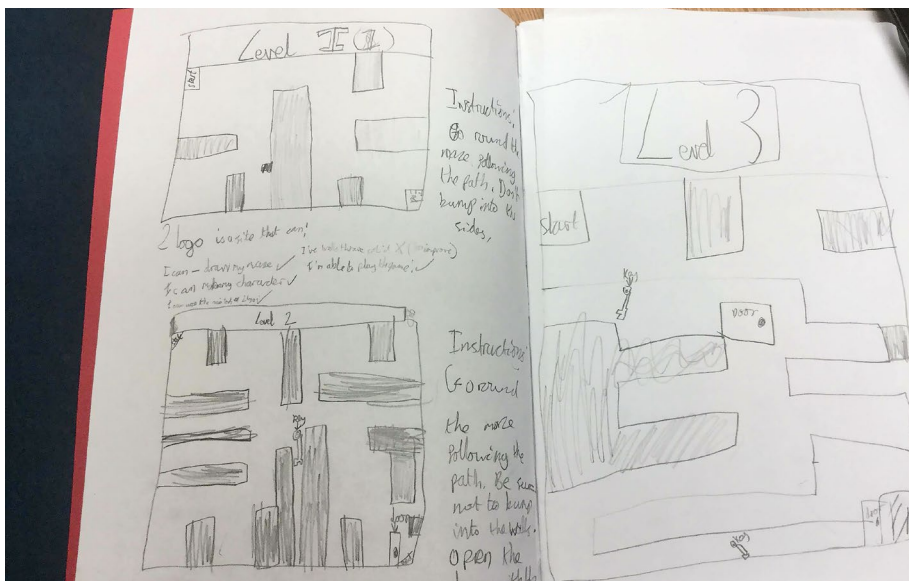


Bar charts showing responses to having a design journal in computing. The scale is 1 to 5, with 1 being 'not useful' and 5 being 'extremely useful'

Pupil voice

After a year of using the journals, I wanted to assess their impact on pupils' learning of computing. I conducted a pupil voice survey with year 5 and 6 pupils. I asked them a range of questions, and I found some interesting results. I should note that I got more responses from year 6 (53) than

I DIDN'T WANT TO CONSTRAIN THE USE OF THE JOURNALS, BUT WANTED THE CHILDREN TO ENGAGE WITH THEM



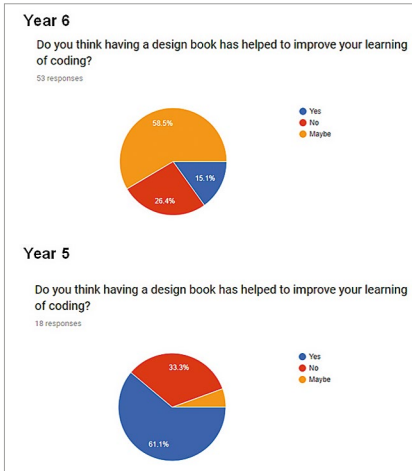
MATTHEW WIMPENNY-SMITH

Matthew is leader of digital strategy and a computing subject leader. He has worked for Headington School Oxford, UK, for the last seven years in the Prep School, teaching EVFS, Key Stage 1, and Key Stage 2. He is a CAS Master Teacher and the Oxfordshire Primary Community Leader. He is also a BCS Certified computer science teacher, Raspberry Pi Certified Educator, Google L1, and NCCE facilitator (@MWimpennyS).

JANE WAITE

Jane is a research scientist at the Raspberry Pi Foundation. Her interests include using design in primary programming, semantic waves, PRIMM, and migrating to online teaching using ABC (@janewaite).

A pupil using their journal to plan what the levels will look like for their Scratch maze game coding project



■ Pie charts showing impact of the journals on improving student learning of coding

► I did from year 5 (18), due to other events taking place in school at the same time as the survey.

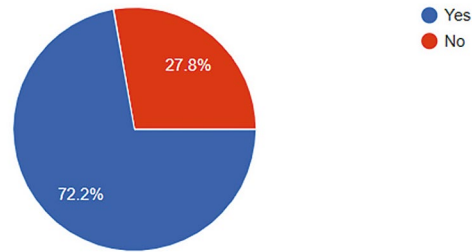
Still, the children found having a design book more useful than not for computing, and this was reflected both within the written responses of the pupils, which generally showed a positive impact on learning. It is notable that nearly half of both year five and six found it useful or extremely useful. One year six pupil commented that “it is much easier to then actually do the task because you have already planned it out! You’re more sure of what to actually do. I would definitely say to use the design books again!” You can find all the written responses from the survey at helloworld.cc/designjournalsurvey.

Despite nearly half the year five and six pupils thinking that the design journals were either very or extremely useful, they weren’t so sure it helped with their actual coding. In fact, over half of the year six respondents said ‘maybe’ when asked whether they felt having a design book improved their learning of coding. I wonder, though, if the question was a little muddled.

Although the sample size was a lot smaller, the year five response to the same question showed that well over half of the pupils were more positive when asked about improved coding from a design-first approach. Still, just shy of a third of pupils in years five and six answered no, that it didn’t help their coding. I would say that at this age group, there are always a number of pupils who don’t see the benefit of doing

Did working on a design before you started coding improve your coding or the outcome of the finished project?

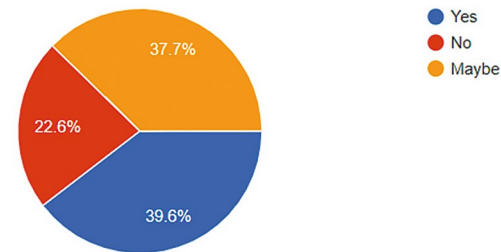
18 responses



■ Pie chart showing year 5 responses to a question on the impact of a design-first approach to coding

Did working on a design before you started coding improve your coding or the outcome of the finished project?

53 responses



■ Pie chart showing year 6 responses to a question on the impact of a design-first approach to coding

designs or planning before getting started, independent of subject.

On reflection, I should not have included ‘maybe’ as an option, or perhaps I could have included a scale similar to the usefulness of journal question. Also, it was not clear as to whether all the students associated design with improvements in coding. In the future, I will think about

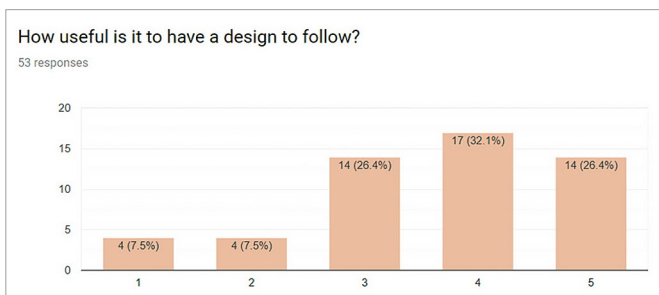
experiment and to see how the journal might fit into the design activity, rather than have the journal seen as a product in itself.

In another question to pupils. I asked about the impact of design, and whether they felt it helped improve the outcome of their coding project. I was keen to investigate students’ views on the impact of designing before coding, and whether

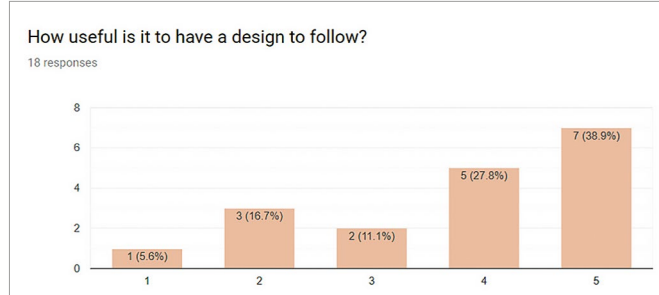
“ THIS IS EARLY DAYS, AND I DID INTENTIONALLY KEEP THE ROLE OF THE JOURNAL LOOSE TO ENABLE CREATIVITY

how I define the role of the design journal better so it has more purpose in the lessons. However, this is early days and I did intentionally keep the role of the journal loose to enable creativity. I wanted to facilitate the process of design and encourage learners (and myself) to

they felt this approach improved the outcomes of the finished project. Both year groups were positive on this, i.e. that having a design before starting the coding improved the outcomes of the project. Again, perhaps I shouldn’t have offered ‘maybe’ as an option here. I now need to



■ Bar chart showing year 6 responses to a design-first approach. The scale is 1 to 5, with 1 being 'not useful' and 5 being 'extremely useful'



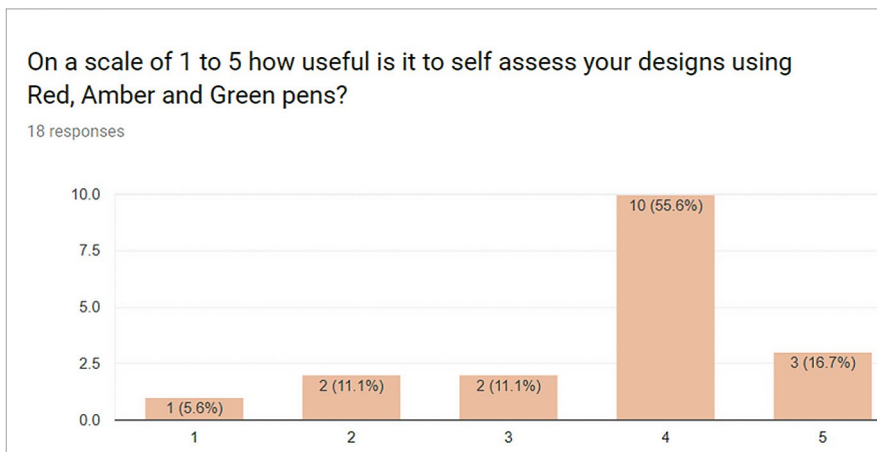
■ Bar chart showing year 5 responses to a design-first approach. The scale is 1 to 5, with 1 being 'not useful' and 5 being 'extremely useful'

develop a way to measure the impact of a design journal perhaps with a group who don't do design versus a group who do.

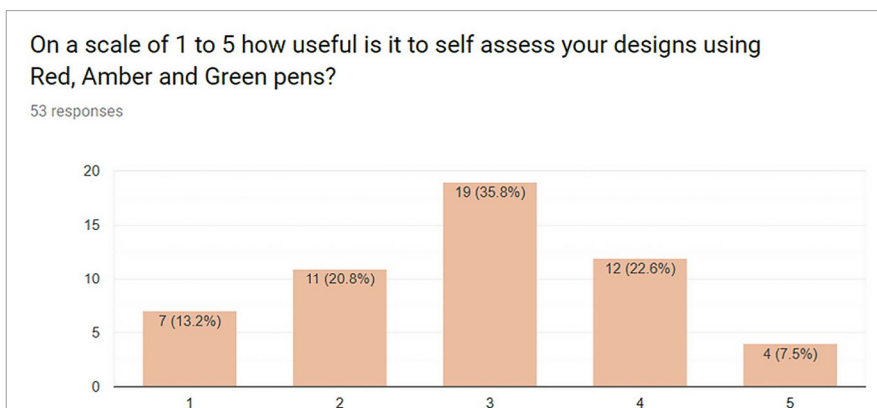
Positive responses to a design-first approach were repeated in a second question on how useful it was to have a design to follow in the first place. Interestingly though, the year five pupils found it more useful than year six. This may have been due to the unmatched sample size between the two cohorts, or it may be because of different material being covered. Or maybe something else! I will have to compare next time round and perhaps ask some open questions, or ask pupils to discuss it all between them and make notes on their discussion.

During the design stage of projects, I would often ask pupils to assess their designs for confidence using the red, amber, and green traffic light colours, along with 'purple polishing pen' annotations and edits similar to the 'purple editing pen' that they were used to using in their literacy lessons. Using this method helped the children gauge an understanding of what they considered to be doable. This continual annotation not only helped me to identify gaps in their understanding, but also helped to make their computational thinking and problem-solving more visible. It also acted as an indicator to me (and to them) of how confident they were feeling about their programming projects as they were being developed.

The responses to the survey found that over 60 percent of pupils surveyed found this method of self-assessment useful or very useful. This has highlighted to me that this is an effective way for pupils to gauge their own confidence and understanding of a project and its design, and is an effective tool for assessment for learning.



■ Bar chart showing year 5 responses to the impact of traffic light self-assessment. The scale is 1 to 5 - 1 is 'not useful' and 5 is 'extremely useful'



■ Bar chart showing year 6 responses to the impact of traffic light self-assessment. The scale is 1 to 5 - 1 is 'not useful' and 5 is 'extremely useful'


Conclusion: Stick or ditch design journals?

Based on the responses to the pupil voice survey and my own experience, I definitely intend to use design journals for this age group in the new academic year. On reflection, I need to develop my practice to incorporate them more in my teaching, and to develop their purpose and role as a tool to build their design-first approach. Also, to improve the visibility of computational thinking and problem-solving.

It's clear from the experiment that the children benefitted from having a journal in the subject at this level. I also need to be consistent with what I call them – either a design journal or a design book. I think defining them as a journal makes them more of a 'creative' and 'process' space, whereas a book is seen more as an end product; an outcome space for polished work, requiring marking and scrutiny. I might just go with journal... (HAW)



GET HANDS-ON

- 130** PHYSICAL COMPUTING
 - 132** BRINGING PHYSICAL COMPUTING TO THE CLASSROOM
 - 134** A JOURNEY INTO PHYSICAL COMPUTING
 - 137** PRIMARILY PI: TEACHING PHYSICAL COMPUTING AT PRIMARY
- 

Research shows that physical computing and making activities are highly engaging approaches for learners, giving them a sensory, tactile, and creative experience in which they can combine computing with art, craft, and design. Physical computing is both a tool to engage learners and a strategy to help them develop their understanding in more creative ways. This approach also has the benefit of supporting and engaging a diverse range of learners in tangible and challenging tasks. There is some evidence, for example, that girls engage more with physical computing because a physical project may have more immediate real-world applications.

Educators can use physical computing to teach many different areas of computing curricula across all year groups. While primarily supporting the development of programming skills, it can also support more conceptual curriculum areas.

Through physical computing, learners can encounter, develop, and practise the entire range of programming skills and concepts, including sequences, loops, conditions, functions, and data structures. Alongside applying these concepts, learners will also encounter other languages, models of programming, and novel computer systems.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:**
physical computing
- Bringing physical computing to the classroom
- Overcoming barriers to physical computing
- Physical computing for primary students

PHYSICAL COMPUTING

By embedding physical computing into their practice, educators can provide engaging, relevant, and inclusive learning experiences

Physical computing is a broad term to describe activities in which learners write programs to interact with the real world using specialist hardware. While there are many examples of physical computing devices (as well as many ways to program them), they can typically do a combination of some or all of the following:

- Control a simple output component, such as lights and buzzers
- Measure or record the environment in some way, including through sensors, buttons, and switches
- Drive and control motors to create movement

Benefits to learners

Beyond the engaging nature of physical computing, there is emerging evidence of its learning benefits in computing and beyond. Learners typically learn to program using high-level languages and produce screen-based applications, independent of the hardware on which they run.

Physical computing can promote a broader perspective, bridging learners' theoretical knowledge of how the hardware works and their program writing skills.

There is some evidence that physical computing activities can support a learner's program comprehension,¹ particularly in relation to the purpose and function of a program. The physicality of the project provides clues to the intended purpose of a program, as well as to how it is likely to work.

Depending on the context and the approach of a project, learners are also able to develop broader, more holistic skills, including collaboration, communication, and design and prototyping skills. Physical computing projects

are typically situated within meaningful contexts such as plant monitoring, social enterprise, or even performance, allowing students to develop their understanding of subjects beyond computing.

Relevance and inclusion

The last decade has seen a growth in computing across a range of education settings. Whether through formal education and curricula or the many clubs, competitions, and other non-formal settings, there are many more opportunities for young people to learn about and develop the knowledge and skills associated with computing. Despite this growth, there are significant challenges ahead for educators in how

students, with traditionally underserved minorities engaging with physical computing. Girls, in particular, have reported higher levels of confidence in computing following physical computing activities.³

Physical computing provides a hands-on experience for learners, with real and immediate feedback. In line with constructivist learning theory, there is a tangible artefact for them to touch, manipulate, observe, and build, which helps build confidence. Physical computing can also promote greater intrinsic motivation in learners through more practical and relevant examples of computing, allowing them to solve problems that matter to them. Additionally, physical computing can be applied to a

“ EVIDENCE IS EMERGING OF THE LEARNING BENEFITS OF PHYSICAL COMPUTING BOTH IN COMPUTING AND BEYOND

they address the equity, inclusivity, and relevance of computing. While educators are applying plenty of well-established good practices, there are still groups within our increasingly diverse learners that are underserved, including girls and ethnic minorities. As well as providing access and opportunities to learn about computing, educators need to consider new approaches that support a broad range of learner backgrounds.

As an approach, physical computing is established as a highly motivating experience for learners,² particularly due to its tangible and interactive outcomes. This positive response can be observed in

wide range of scenarios with some very accessible starting points and plenty of advanced concepts to explore. This breadth and depth facilitate choice, progression, and creativity for learners.

Spoilt for choice

There is no doubt that physical computing activities add some additional challenges for educators, including cost, training, logistics, and troubleshooting. However, many of these difficulties can be addressed through planning and experience. The last few years has seen an explosion in the number of educational physical computing devices. This expansion of the tools available means



■ Figure 1 This physical computing device categorisation taxonomy was adapted from Hodges et al.⁴

that there is now a diverse ecosystem of devices suitable for learners. Hardware developments mean that these devices have become more capable and cheaper. Many now incorporate simple design features such as colour-coded connectors and have improved software that make them generally more accessible and easier to use with a wide range of learners.

To better help educators understand the different features of devices across this ecosystem, Hodges et al.⁴ present a categorisation taxonomy that provides a broad distinction between devices. When considering a device(s) to work with, educators should review the features, connection method, and means of programming, as well as the flexibility that each device provides. An adaptation of this taxonomy can be found in Figure 1.

Getting started: Hardware, content, and training

Getting started with physical computing can be a rewarding experience for learners, but

it is not without its challenges. Educators should consider the following:

- Start small. Focus on a small cohort, an individual concept, or a single activity or lesson.
- Is there any suitable content available already that you could use and adapt?
- Is there any training available to support you?
- Are there other educators locally that you could collaborate with or observe?
- With the above in mind, which devices would best suit your immediate needs and allow for maximum future flexibility? Can you borrow the equipment before you buy it?

As an educator, adopting physical computing can be an engaging and highly beneficial experience for your learners. While this journey is not without its challenges, there have never been as many device options or as much support or content available as there is right now. (HW)

SUMMARY

Physical computing:

- Uses specialist hardware to interact with the real world
- Involves writing programs to control output devices (such as lights, buzzers, or motors)
- Allows learners to record and measure the environment through buttons and sensors

Benefits:

- It provides a holistic experience of computing, combining hardware and software
- It may support program comprehension by providing physical clues to a program's purpose
- It develops broader skills, including collaboration and design and prototyping
- It connects to subjects beyond computing

Relevance and inclusion:

- Physical computing can provide opportunities for a broad range of learners
- Learners (particularly girls) find physical computing engaging
- There are opportunities to vary context and level of challenge (high ceiling, low floor, wide walls)
- It provides space for learners to be expressive and creative

REFERENCES

- ¹ Jayathirtha, G., & Kafai, Y. B. (2021). Program Comprehension with Physical Computing: A Structure, Function, and Behavior Analysis of Think-Alouds with High School Students. *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, 143-149. helloworld.cc/physical1
- ² Przybylla, M., & Romeike, R. (2018). Impact of Physical Computing on Learner Motivation. *Proceedings of the 18th Koli Calling International Conference on Computing Education Research, Article 9*, 1-10. helloworld.cc/physical2
- ³ Sentance, S., & Schwiderski-Grosche, S. (2012). Challenge and Creativity: Using .NET Gadgeteer in Schools. *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, (pp. 90-100). helloworld.cc/physical3
- ⁴ Hodges, S., Sentance, S., Finney, J., & Ball, T. (2018). Physical computing: A key element of modern computer science education. *Computer*, 53(4), 20-30. helloworld.cc/physical4



■ Pupils creatively design, craft, program, and build interactive objects using Snap4Arduino and Arduino with pre-assembled sensors and actuators

BRINGING PHYSICAL COMPUTING TO THE CLASSROOM

Why and how should we teach physical computing in computer science education? Here are some useful guidelines for lesson planning and how to apply it in class

Physical computing is an interdisciplinary field involving the creative arts and design processes. It brings together hardware and software components, and joins the virtual world of computers to the physical world of humans by using concepts from embedded systems design and its neighbouring disciplines.

Products of physical computing make use of sensors and actuators to interact with their environment. Tools include microcontrollers and mini computers, often with extensions to facilitate component handling. Projects are of an iterative nature, quickly leading to working prototypes, such as the interactive garden at the end of this article. When planning and creating interactive objects, the focus is on ideas and intended interactions with the audience or environment. Purposeful tinkering is encouraged, to develop ideas and figure out how things work.

Benefits of physical computing

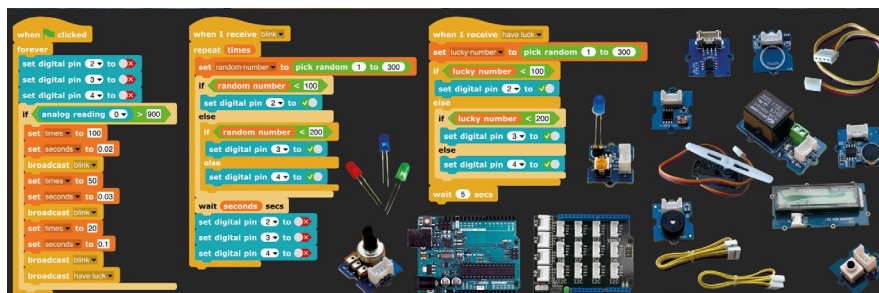
It all makes physical computing a promising approach for introducing embedded systems

and underlying concepts in computer science (CS) teaching at secondary level. Many skills and competencies are gained. Programming concepts and control structures such as decisions, loops, variables, comparisons, or arithmetic operations are used to make objects that can flash lights, move, or make sounds reacting to their environment. Content related to embedded systems design, such as sensors, measurement, control, or common practices when working in larger projects, is also relevant.

In physical computing, pupils learn with (and about) interactive objects and systems, by creating tangible real-world products from their imagination. This can be used to promote creative learning in CS education, and boost students' motivation. It fits the learning theory of constructionism. In this theory, learning is most effective in contexts where learners construct knowledge and develop competencies from their initiative and for a personally relevant purpose, while engaged in creating visible artefacts. With physical

EXEMPLARY LESSON STRUCTURE

- Introduction and motivation: provide examples in a short tutorial session
- Tinkering: let learners explore the tools - provide manuals and cheat sheets
- Brainstorming: find ideas for projects
- Project planning: give guidance with worksheets
- Presentation and discussion: reflect on ideas and plans
- Creation: let learners work, in groups of two to four
- Exhibition and reflection: present the project to an audience (open-door day, school party, etc.)



computing, these are not only visible, but also tangible — similar to artistic sculptures.

Guidelines for teaching

To make it easier to start teaching physical computing, researchers developed guidelines and design principles for teaching and learning scenarios, as well as classroom-ready materials (helloworld.cc/myig).

Using a design-based research approach, theoretical concepts were tested in classrooms and evaluated with pupils and teachers. Promoting constructionist and creative activities, such as tinkering and prototyping, has proven to be a key part of successful learning scenarios. Contrary to expectations, it is not necessary to focus strictly on project planning before introducing tools for positive impact in the evaluated domains, but it helps stimulate ideas and creativity. A commonality in less successful courses was missing structure in project planning and implementations. Providing scaffolds is particularly helpful in groups that are not used to project work.

This results in the following design principles for physical computing teaching:

- 1 Integrate tinkering activities in dedicated learning phases in which content knowledge and skills are acquired
- 2 Let learners create their own interactive objects
- 3 Let learners develop working prototypes
- 4 Provide interesting themes and open topics to trigger imagination and creativity

- 5 Integrate creative methods
- 6 Integrate technical aspects with art/crafting
- 7 Provide scaffolds to structure the process of project work, including planning from a user perspective and planning from a developer perspective (non-technical and technical viewpoints)
- 8 Choose suitable construction kits and programming environments for the target group (low floors, wide walls, high ceilings)
- 9 Provide suitable crafting material and tools for the intended projects
- 10 Prepare a joint exhibition of all objects
- 11 Present the results to an audience

MAREEN PRZYBYLLA

Mareen is head of the endowed professorship for computer science didactics at the College of Education Schwyz, Switzerland. Her research focuses on physical computing in computer science education (@MPrzybylla).

and skills. For instance, they are introduced to sensors and actuators as means of analogue and digital inputs and outputs; to ideas of events that can take place in parallel or serial; and to continuous-time and discrete systems.

In project work, learners develop prototypes in short iterations, which are discussed with their classmates and teacher throughout the process. They also detect problems and possible misunderstandings early, and find solutions to occurring problems. Finally, the students present their projects in an exhibition and discuss their experience with

“ PROVIDING SCAFFOLDS IS HELPFUL IN GROUPS NOT USED TO PROJECT WORK ”

My Interactive Garden

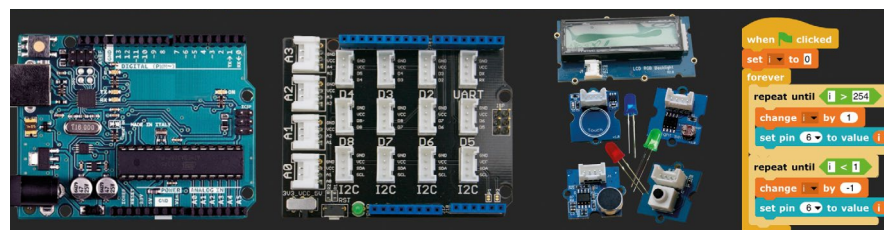
My Interactive Garden (MyIG) is a lesson series that uses these ideas. A theme of creating an interactive garden exhibition was used to call for ideas and collaboration and encourage a variety of projects. Learners design, craft, program, and build objects using a construction kit based on the microcontroller platform Arduino, with pre-assembled sensors and actuators, and using the block-based programming environment Snap4Arduino.

They are provided with material and worksheets that structure the project work. In tinkering activities and learning phases, pupils learn and acquire specific concepts

the audience. They explain the purpose and functionality of their interactive objects and reflect on their progress.

MyIG and other physical computing settings were looked at in a cross-sectional pre- and post-intervention study (helloworld.cc/myig). Courses adhering to the design principles were more successful than most others. Physical computing can motivate learners more than many other activities in CS classrooms. MyIG implementations on average showed motivation values over twice as high as general physical computing activities, and appealed better to female students. In 71 percent of the analysed MyIG courses, the existing gap between boys' and girls' learner motivation in CS classes narrowed.

The consistent implementation of design principles for physical computing teaching generally showed positive results. In addition to higher learner motivation, most pupils liked the projects, had more fun, and felt more competent than in their prior lessons. (HW)



A JOURNEY INTO PHYSICAL COMPUTING

Rebecca Franks shares some of the barriers to introducing physical computing into the classroom, and how you can overcome them

During my 15 years as a computing teacher, I always wanted to learn more about physical computing and develop activities for the classroom, but I had many barriers in my way. Now that I'm out of the classroom and developing resources for the computing curriculum in the UK, I have learnt how to remove those barriers. Physical computing is really inspiring, but doesn't always appear on a typical computing curriculum.

By exploring physical computing, I have found out why it is so important that young people experience it. I have spoken to educators around the globe who are using it successfully, and discovered some of the challenges that teachers face when trying to implement it in their classrooms. I have found fantastic resources and learnt new skills, which I will share with you in this article.

Why physical computing?

From making an LED blink to programming a robotic turtle such as a Sphero, or creating a weather station, physical computing takes programming away from being solely on the screen and moves it into the real world. When a student's project works, they can see their creation move, or light up, or record some data from the environment in real time — and these can be brilliant 'Aha!' moments.

Recent research has highlighted that physical computing can play a huge role in increasing the motivation of learners, and in particular, the motivation of girls. Learners

are creating real, tangible devices that give them instant feedback.

It's been found that girls tend to like creating things that make a difference to people's lives. If you set a class a project in which they need to make a device that can help with wider issues, such as global warming, this can motivate them to learn more about how to program and build their own devices. Physical computing also lends itself to an interactive and collaborative approach in the classroom, which is an important factor in girls' attitudes towards the subject, too.

A lot of the research on physical computing education has resonated with my own experiences. As a child in IT lessons, I loved drawing patterns with Logo on the BBC Micro, and became quite proficient at using the applications on the Acorn Archimedes. I also really enjoyed my DT lessons, in which we would use the machinery to make clocks out of acrylic, and solder wires and batteries to make pinball machines. At that age, I would never have thought that computing and soldering were linked in any way. It never came up. And I had no idea that you could

“ PHYSICAL COMPUTING INCREASES MOTIVATION AS LEARNERS CREATE TANGIBLE DEVICES THAT GIVE INSTANT FEEDBACK **”**





■ Ali Alzubaidy leads a lesson using micro-bits with his Code Club in Iraq



■ One of Rebecca's recent makes is a postcard with decorative LEDs attached

“ THEY LEARN WITHOUT REALISING IT, AND I’VE FOUND THAT THIS IS THE KEY TO CREATING POWERFUL, ENGAGING LESSONS

have a career that used those skills. It was a classic situation of I couldn't see it, so I couldn't be it.

This is why I believe it is so important that we introduce physical computing into our schools. It opens doors to all sorts of opportunities that learners might not otherwise be aware of.

Physical computing in clubs

The cross-curricular and project-based nature of physical computing, in which children can work on designing, programming, and making a product with an end goal, lends itself well to being a part of informal learning settings such as coding clubs.

Gary Quinn, who runs a Code Club at Lostock High School in Greater Manchester, uses a range of physical computing devices with his students. He uses the Makey Makey — an integrated input/output device that allows students to create devices such as keyboards made of fruit — and the micro:bit, a microcontroller board that he uses with electronics kits. He told me: “Nothing beats that feeling of achievement students get from seeing their code run before their eyes in the real world. They

love to show off their creations to peers and staff alike! The students always comment on the fun factor and want to see every project through to a conclusion. While they are learning about code, circuits, accelerometers, and everything in between, it doesn't feel like an effort to them.”

Quinn values the engaging impact that physical computing has on his learners: “The excitement and anticipation are unmatched by paper or screen-based activities, and gone are the days when the Scratch Cat moving across the screen would suffice.”

Ali Alzubaidy, a Code Club volunteer in Iraq, uses physical computing to demonstrate to his students the links between technology and other subjects. For example, to teach his group about the geography of Iraq, he loads a map of the country into Scratch, and learners create the code to allow a Makey Makey device to interact with it.

He describes the delight in his class when working with physical computing: “When the kids made anything with tech, they were so happy.” Ali told me he believes that physical computing is important because it builds students' confidence with technology,

improves their logical thinking, builds team-working skills, and prepares students for their future careers.

Challenges of getting physical computing in the classroom

So if physical computing is so popular with learners, and it has positive effects on their learning, why isn't there more of it in computing classrooms?

For me, the limiting factor was time. When I was a teacher, I purchased an Arduino starter kit — a small microcontroller board with modular sensors and electronics — to see if I could start building my own projects. I had success following the set tutorials, but really struggled if I wanted to make something independently. I kept trying to get into physical computing, but I found it tricky to put time aside to dedicate to really understanding how it all worked.

Research has shown I'm not alone. In 2018, the Raspberry Pi Foundation published a report that found Raspberry Pi computers — single-board computers that lend themselves well to physical computing projects — are quite popular in schools, but that teachers often stick to simple projects that rely on step-by-step instructions, rather than giving learners more open-ended tasks. This is potentially due to time constraints, but also the knowledge and experience of the teacher. An open-ended idea can lead to the teacher needing a wide range of physical computing knowledge.

For teachers to start using physical computing in their computing lessons, they need access to high-quality lesson plans. Good resources can help to build teachers' ▶

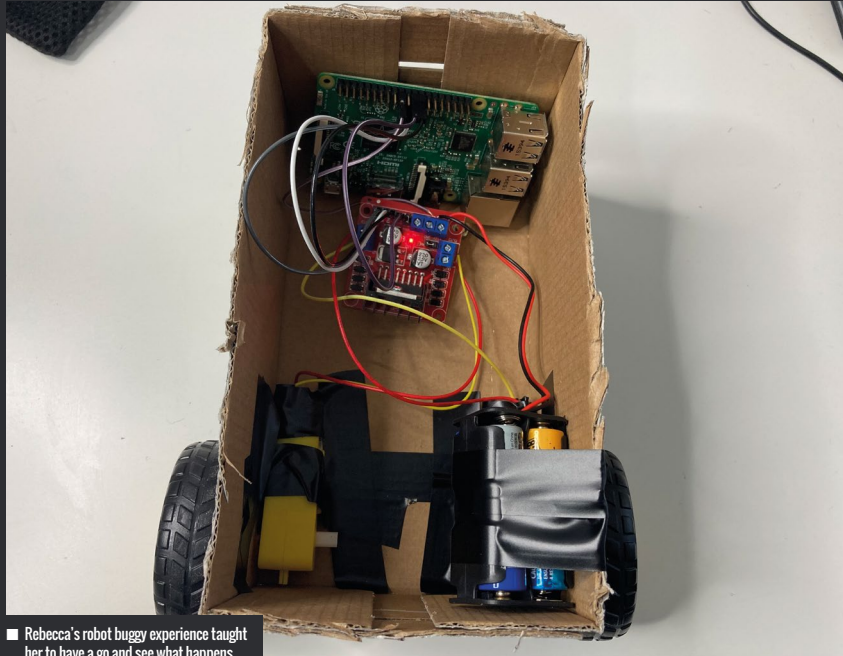
MY ADVENTURES IN PHYSICAL COMPUTING

“If I cut this wire and solder it to another wire, will it still work?”

This is a question that I found myself asking on my work Slack channel a few months ago. At the time, I was working through the Raspberry Pi Foundation’s Build a Robot Buggy course (helloworld.cc/buggy). Instead of just soldering some wires together to see what happened, I went straight to my colleagues and asked the question. I wondered, “Why am I asking this question, when I could just solder the wires together and see what happens? Why don’t I have the confidence to just do it?” I was so apprehensive about simply jumping in and trying something.

It can be really intimidating to be around people who know so much more about a subject than you. Low self-efficacy is quite a common thing for women to experience, and I definitely fell into this category when it came to physical computing. I had a little chat with myself and decided that, from then on, I was just going to give it a go and see what happens. This change in attitude has been liberating for me.

I made a commitment to just have a go at things and stop questioning myself. I started shoving loads of bits of card into my buggy to make the wheels work a bit better. I unscrewed and swapped out the wheel bearing, as it had gone sticky, and I replaced a broken sensor,



■ Rebecca’s robot buggy experience taught her to have a go and see what happens

and tried different ways to stick parts into the box. After lots of trial and error, I had a fully functional robot buggy, and it felt great!

Since building my buggy and learning how to solder properly, I have become much more confident at trying new things. And yes, I did solder those jumper wires together, and yes, they

did work! I think as adults, we can sometimes feel as if we are wasting time if we just sit and have a go at making something. We feel like it has to have a purpose or an end goal. It really doesn’t: you can get so much satisfaction from making a random thing. One of my recent makes is a postcard with decorative LEDs!

➤ confidence, making them more likely to use physical computing in their lessons.

The Teach Computing Curriculum (helloworld.cc/tcc) provides comprehensive physical computing units. They demonstrate clearly how to progress learners’ understanding of physical computing devices such as Crumbles and micro:bits. The lessons include guidance for the teacher and helpful guides that learners can use to help them debug any issues they may encounter. The resources are published under an Open Government Licence and are free for anyone to download from anywhere in the world.

Getting started

Physical computing is so important for our young people. It helps them build

connections between the real world and programming, while giving them something exciting to focus on. They learn without realising it, and I’ve found that this is the key to creating powerful and engaging lessons.

If you are in the same position that I was in with physical computing — wanting to start, but feeling nervous or uncertain of how to do so — my advice is to just relax and enjoy making something. Buy some really cheap components and have a go. See it as a journey, not a race. And try not to compare yourself too much to some of the awesome makers out there. They were all beginners in the same place you were once, and will be very willing to help you out if you need some advice. You won’t regret it — and you never know where your adventures could take you! [\(H/W\)](#)



REBECCA FRANKS

Rebecca is a learning manager at the Raspberry Pi Foundation. She writes resources for the Teach Computing Curriculum and taught computing for over 15 years ([@FranksberryPi](https://twitter.com/FranksberryPi)).



PRIMARILY PI: TEACHING PHYSICAL COMPUTING AT PRIMARY

With a little support, even our littlest learners can learn physical computing skills

Educators are often hesitant to use physical computers with primary-aged students. But with a little scaffolding, even our youngest learners can learn programming and physical computing with something like Raspberry Pi. Here are some tips for supporting primary-aged students through physical computing projects in the classroom.

Getting started

I teach in a school that does not currently have specific computer science classes, so often my students come to me with different levels of experience in coding, or with no experience at all. For that reason, I usually start our physical computing work with a couple of offline lessons about computers and algorithms. This year, my favourite resource has been the *Hello Ruby* books. These stories about a little girl named Ruby, who gets into all sorts of adventures inside her computer, are a great introduction for young students to hardware, software, coding, and the internet. The *Hello Ruby* website (helloruby.com) also has all

sorts of offline activities geared towards young students, for practising computing terminology, thinking logically, writing algorithms, and more (see page 156).

Once students have been introduced offline to vocabulary and concepts they'll need in our digital making project, we use Code.org's self-paced lessons to introduce block-based coding languages and how to write a sequence of code with blocks. Students work through a handful of lessons to get a grasp of the basics of block-based coding, including:

- Dragging and dropping blocks into a workspace
- Connecting blocks
- Writing a sequence of directions from top to bottom
- How to trash blocks when you make a change
- How to run your program
- Analysing for errors and debugging

The Code.org intro lessons also help students practise basic computing skills that many

don't have when they start school. Most of our primary students are more familiar with touchscreens and tablets than they are with computers, so we also need to teach them:

- How to use a mouse or trackpad
- How to click and drag with a mouse or on a trackpad
- How to right-click on a mouse or trackpad
- How to navigate software menus and toolbars

Learning by making

After a few lessons in Code.org, we dive into our work in Scratch and physical computing. One of the things that I loved about Picademy (the course for becoming a Raspberry Pi Certified Educator) was the project-based approach, so I decided to adapt that format for my own students. Learning within a relevant context in any subject area tends to be more motivating for my students than trying to learn skills in a siloed approach, so we start by introducing what we'll be making. Then we teach the individual coding and electronics skills that we need to get there. ▶

► I tend to integrate our physical computing projects into units currently being taught at my school, and I design the lessons with a cross-curricular approach. Some of my favourite projects have included:

- Learning about traffic lights with our Kindergarten traffic engineers (integrated with our 'community jobs' unit)
- Writing gold-finding programs in Minecraft with our fourth-graders, aged nine to ten (integrated with our 'California Gold Rush' unit)
- Designing urban wildlife cameras with my second-graders, aged seven to eight (while learning about ecosystems in their habitat-themed literacy unit)
- Digital voting booths with both second- and third-graders, aged eight to nine (integrated with our government units)

I like to embed opportunities for students to think like designers within the project. Even if I already have a plan for how we're going to make something, we launch with a student-led brainstorm, asking students what elements our digital making projects should include, and why. What might we need to create an effective wildlife camera? How should we design our digital voting booth to make voting an engaging process? If needed, I ask guiding questions to help them come up with any missing pieces in our plan: "So far you've all decided that our wildlife cameras need the camera lens, and no flash, because we don't want to scare

away the animals, but I'm still not sure how our camera will know that it's time to take a picture ... What else will it need? ... Oh, Henry, great idea ... cameras usually have a button. OK, let's add that to our plan."

Scaffolding electronics

When working with young students, hooking up the electronics (LEDs, jumper cables, picameras, motors, etc.) can sometimes become frustrating and slow down their work, especially during their first lessons on a Pi. For learners aged four to seven, I do all the electronics set-up in advance.

For my second- to fourth-grade students, I usually set up the electronics for them prior to the first couple of lessons, but as we progress through the project I release a little more of the electronics set-up each time we move to a new lesson. Diagrams and prebreadboarded models allow students to learn some basic breadboarding skills by copying examples.

One of my favourite add-ons for digital making with young students is the Pibrella HAT by Cyntech and Pimoroni. After several days of modelling, our young students quickly learn how to put HATs on Raspberry Pis properly. And with LEDs, buzzers, and buttons already built into the board, and the ability to program the HAT in Scratch, it's a great alternative to breadboarding. (Plus the inputs and outputs on the Pibrella HATs make it easy to also program motors, sensors, and more, so this versatile HAT has become one of the most used tools in our electronics kits!)

The Sense HAT add-on is also a favourite among my primary programmers and is easy for young students to code using Scratch. They love lighting up LEDs and drawing pictures on the Sense HAT with just a few blocks.

Another way that I scaffold the electronics work for my young students is by setting up breadboards for them ahead of time. I organise the LEDs, buttons, resistors, and jumper cables on the breadboard, and then walk students through how to connect the jumper cables to the GPIO pins. As they become more comfortable with using and programming on their Raspberry Pis, we move onto circuitry and get students setting up their own breadboards, or plugging in their own HATs and picameras.

Getting coding

One of the (many) reasons I'm keen on Raspberry Pi computers is the ability for students to program the physical world using Scratch. The ease with which students learn Scratch makes the introduction to physical computing that much more approachable.

Using Scratch does require some amount of reading skill, however. The colour-coded blocks help, but depending on the grade level and reading skills of our students, I sometimes start with an explicit vocabulary lesson, to help students with the language they'll need while coding. Just as we would in other subject areas, we might display the blocks we'll be using on a chart and practise reading them together, using physical cues and images or sketches to help students learn the words.

Sometimes we print out the Scratch blocks on paper (available on ScratchEd at helloworld.cc/scratched) and practise reading them and putting them into the sequence we want ahead of time before we work on the computers. We sometimes have students act out the code, as well, so that they can test whether the program they've planned will move their sprite the way that they want it to. Students can then use the plan they made on paper as a resource for finding the blocks they need once they get started in Scratch.

Other times, I've loaded a prewritten Scratch program onto the students' computers, with the blocks they need



■ During a mini-lesson, students learn next steps in their program and then return to their stations with their teams to try what they've just learned



■ Students work in teams on their physical computing projects, helping each other to learn and troubleshoot

already in the scripting area, but out of order. Students decide what order the blocks need to go in and then snap them together to create their program.

I'm also a fan of the 'learning by copying' method of learning to code. For all the projects I've done with my classes, I've created activity cards that walk students through each step of the project so they can copy (and eventually, customise, when they're ready) the code that they need, line by line. The cards include large visuals and diagrams, along with code students can copy to create their own projects. I will often include pop-outs in the diagrams that explain in a couple of words what certain blocks or lines of code mean or do, so students can learn programming and computer science concepts while making.

The activity cards allow students and teams to move at their own pace on a project, and to check and debug their work against an already-working program. The task cards have also become a great way for me to teach students how to read technical texts and follow a set of given directions in order to put something together, and a way for students to work on projects in their own time, not just at times that I'm leading the lesson for them.

Student becomes the teacher

I like to group my primary-aged students into teams of three when working on a physical computing project. We try to balance each team with students who

feel that they are most skilled or most comfortable in each of the following areas:

- Coder and typer
- Debugger and editor
- Engineer and electrician

Sometimes I create the teams ahead of time, but sometimes I ask students to think about the skill they feel most comfortable in and then get them to build and balance their own teams.

While everyone on the team will get a chance to participate in all parts of the project, we tell students that if they feel they are the strongest coder on the team, they can also be the coding coach on their team. The strongest editors will be the coach in that area, and while we expect everyone on the team to take time to check spelling, capitalisation, spacing, and so on in the code, the editor will lead the process. The engineers and electricians are the students who feel most comfortable with setting up the hardware and sometimes, setting up circuits.

This team format is a great way for us to explicitly teach students collaboration skills. Primary-aged students aren't always naturally skilled at making sure everyone gets a turn, so before students are allowed to get to work on projects, we discuss as a class what strategies we'll use to make sure that each person on the team is able to participate. I often set up the expectation that each person will take a turn dragging a



AMANDA HAUGHS

Amanda is a second-grade educator and learning designer at the Campbell School of Innovation in San Jose, California, USA. She is a Raspberry Pi Certified Educator, Apple teacher, Google Certified Educator, and Leading Edge Certified Professional Learning Leader.

block into the code and then pass the mouse to the next person, continuing the rotation until the entire program is written.

As for direct instruction skills lessons, I also try to keep my students moving frequently. I usually break my physical computing lessons into sets of mini lessons. Students come to the carpet for a mini lesson, then go back to stations to work for a bit, then I call them back for next steps, and then send them back to their stations to work. Chunking the lessons gives students smaller benchmarks to work towards, and more manageable amounts of information to try and remember at one time. And bringing them to the carpet ensures they're focused on the new lesson and less tempted to continue working while I'm giving new directions.

As they learn new skills, student experts start to emerge — one of the best benefits of the team approach to physical computing in our classroom. I don't have to be the only helper in the room. As some students start to become more and more confident in their programming skills, they become the helpers in the room, not only for their own team, but also for others in our class and, sometimes, for me as well. **(HAW)**



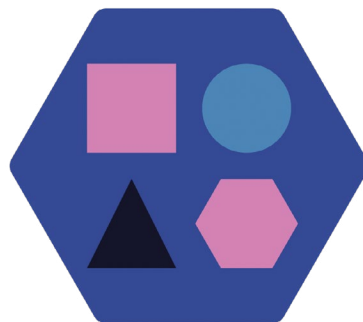
ADD VARIETY

- 143** VARIETY IN TEACHING AND ASSESSMENT OF PROGRAMMING ACTIVITIES
 - 146** STORYTELLING IN COMPUTING EDUCATION
 - 148** RETRIEVAL PRACTICE
 - 152** THE INCLUSIVE COMPUTING CLASSROOM
 - 154** ART AND ALGORITHMS
 - 156** PROGRAMMING AND PLAY
- 

Computing is a broad discipline with connections to many subject areas, especially engineering, maths, and science. Each subject area prioritises different aspects of computing and requires different pedagogical approaches. Educators will each have their own perspective on computing, which in turn may influence their approach to teaching and learning. In order to provide the greatest number of entry points to the greatest number of students, educators need to present a holistic experience of computing.

From lesson to lesson, the focus of teaching and assessment is likely to change. You may be focused on your learners acquiring new understanding, perhaps by teaching abstract concepts through representation. You might need them to explore new knowledge through explanation, demonstration, or practical activities, or even to apply past learning to create an artefact.

Expanding the approaches in your personal teaching toolbox allows you to adapt your instruction to suit different objectives, keep pupils engaged, and encourage and foster greater independence.



IN THIS SECTION, YOU WILL FIND:

- **What the research says:**
adding variety in programming
- **What the research says:**
storytelling in computing education
- Retrieval practice
- Approaches for an inclusive
computing classroom
- Algorithmic art for primary students
- Programming and play



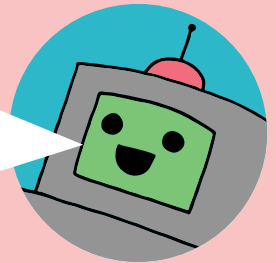
{ code club }

START A CODE CLUB IN YOUR SCHOOL!

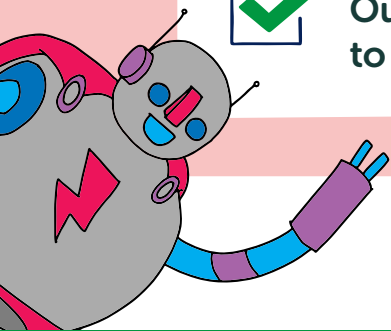
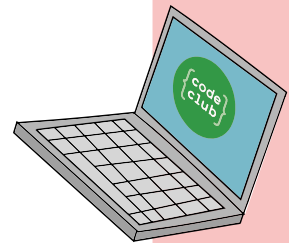
It's easy to get your school coding! Code Club provides everything you need to run coding clubs for 9- to 13-year-olds with free, step-by-step project guides for learning Scratch, Python, and HTML/CSS.

"I run a Code Club in our school for children in Year 5 and 6, which is always popular. Code Club activities encourage learning independence, and the children love to share the games they've coded with each other."

Jill, Teacher



- Code Club is free
- Code Club offers engaging, hands-on activities that have been designed to spark curiosity and inspire creative thinking
- Our free online training is perfect for teachers who are new to coding or are starting a Code Club for the first time



Join Code Clubs across the world giving young people opportunities to have fun while learning to code!

Get involved at codeclub.org



VARIETY IN TEACHING AND ASSESSMENT OF PROGRAMMING ACTIVITIES

Computing is a broad discipline. Educators should vary their perspectives, teaching approaches, and assessment strategies to suit the subject matter in question and learners' experiences and needs

A ccording to the work of Tedre,¹ computing is a broad discipline built principally on three traditions, each bringing its own perspectives. Those involved in the field of computing tend to see it as either one concerned with engineering and design, as a branch of mathematics and logic, or as a science. Each tradition has a different focus, prioritises different knowledge and skills, and invites different teaching approaches. However, all form part of computing as a whole.

- **Computing is engineering** It concerns the design and development of artefacts, including software and systems. It incorporates user research, prototyping, testing, and evaluation.
- **Computing is maths** Logic and mathematics are present throughout computing; our software and systems are built on mathematical principles, and we use mathematical techniques to describe and reason about programs.

- **Computing is science** Computing is pervasive across almost every field of science. We use computers to explore and model the physical world, and to make predictions and discoveries.

Beyond these three traditions, computing is connected to other areas such as the arts, where computing is applied as a medium, or philosophy and ethics, where the application of computing provides rich material for discussion.

Depending on our experience, we're each likely to favour one or more of these perspectives, and that may impact how we present computing to our learners. In understanding these traditions and the broader connections, educators can provide their learners with a complete and holistic computing experience. This enables them to provide a variety of meaningful entry points to the discipline supported by appropriate pedagogy.

Variety within teaching approaches

Whether during a single lesson or an entire course, computing educators need to be able to apply a variety of pedagogical strategies. These will vary depending on the subject matter, the learners, and the aims of each learning experience.

Engineering is concerned with making an artefact that solves a problem or addresses a user need, and can link computing to other areas of the curriculum. In computing, ▶

SUMMARY

Perspectives on computing

Computing is a broad discipline rooted in three main traditions:

- Computing as engineering is concerned with design and development
- Maths is integral to computing systems, software, and how we describe them
- Most fields of science apply computing to model and explore the physical world

Classroom strategies

A holistic approach to teaching computing reflects each of these perspectives, their priorities and practices:

- An engineering perspective leads to more project-based learning with scaffolding to

support individual learners. Assessment approaches might include classroom talk and ongoing feedback through code reviews, showcases, and portfolios.

- A maths perspective focuses on acquisition and construction of knowledge, using representations to explore abstract concepts, and regular recall and practice of facts and processes. Classroom talk and multiple-choice questions are used to surface learners' understanding.
- A science perspective leads to a more enquiry-based approach in which explanations, demonstrations, and practical activities develop understanding and learners predict and experiment. Practice questions and classroom talk are used to check understanding and critical thinking.



Credit: twistphoto/stock.adobe.com

■ A science-based view of computing involves prediction, exploration and observation

► an artefact could be a program, system, or digital media. Physical computing, in particular, is an obvious way of learning about computing through an engineering lens (see page 130). For example:

- Project-based learning is closely associated with this perspective. Learners apply their prior knowledge to a problem focusing on one or more aspects of the design process.
- Specifically when programming, you can adjust the scaffolding and support you provide to learners, depending on their needs and your focus.²

Maths is concerned with understanding, applying, and connecting abstract concepts. The same is true for areas of computing with links to maths, where learners need to understand abstract concepts, recall facts, and practise calculations and processes. In these situations, educators adopt approaches that focus on the acquisition and construction of understanding. For example:

- Representation is a key part of a mastery approach to maths, using different modes of representation, including physical objects, pictorial representations, and eventually, symbols and language. This approach may be a successful way to teach learners about binary number systems, for example.
- We can use varied and regular recall of concepts and processes to secure existing understanding, challenge

misconceptions, form connections, and develop a coherent understanding.

A science-based view of computing involves more enquiry-based practices in which understanding is constructed through prediction, exploration, and observation. We can also use simulations, practical demonstrations, and experiments to develop students' skills and understanding.³ For example:

“ A HOLISTIC APPROACH TO TEACHING COMPUTING REFLECTS EACH OF THESE THREE DIFFERENT TRADITIONS

- Learners develop their inquiry skills when programming with the PRIMM methodology; they predict and validate their predictions, as well as investigating and asking questions of the code (see page 22).
- Topics such as computer systems or networks⁴ contain plenty of substantive ideas or facts that can be explored through a combination of explanation, demonstration, or experimentation. Educators have to select the best balance of approaches to suit each new concept and their learners' needs.

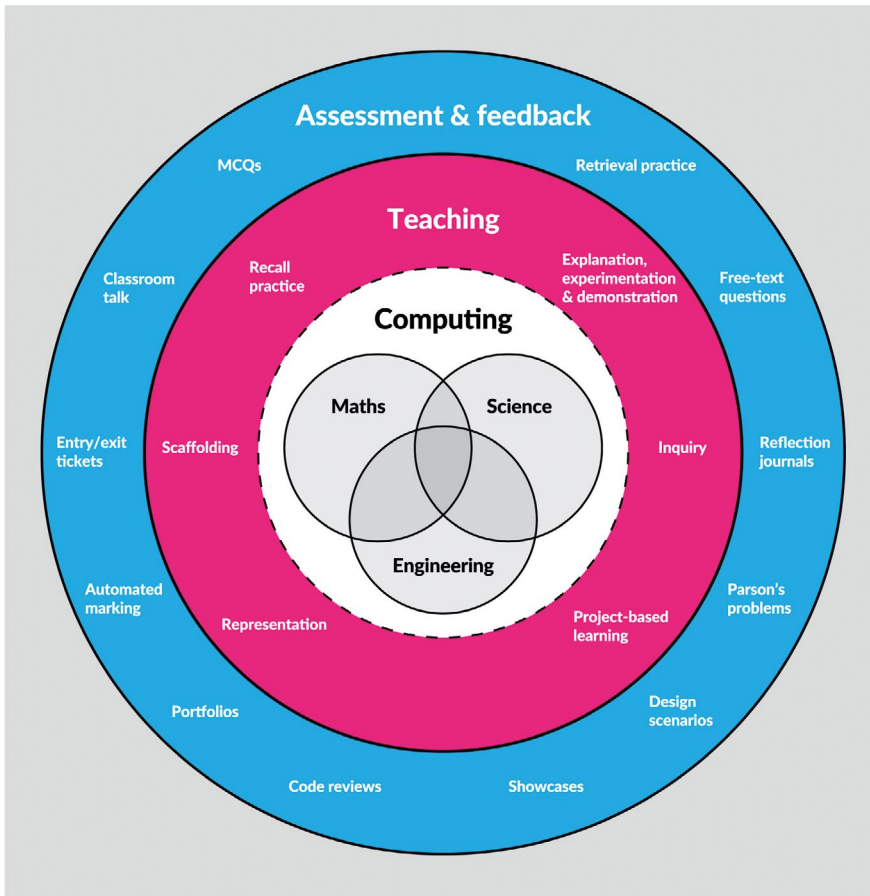
Another lens through which to understand computing is its role in society and the ethical and personal implications of

using technology. Offering learners the opportunity to discuss ideas and engage in meaningful classroom talk, whether with the teacher or their peers, can support a rich understanding of concepts.⁵ Some examples of possible approaches include collaborative methods such as pair programming and peer instruction, which can help challenge misconceptions and build confidence. Discussion and debate are particularly relevant to computing,

as the reach and impact of technology is fertile ground for legal, moral, and ethical discussions. The social and cultural connections educators draw upon have an impact on how learners engage with a topic. Rooting your practice in your students' lived experiences, cultural knowledge, and background makes their learning more relevant and accessible (see page 34).

Variety within assessment

Recent reviews of research on the assessment of computing have concluded that much work is still needed to create reliable assessment approaches suitable for all teachers and for every student. However, there is general agreement that using a



■ Introduce variety in computing lessons through different subject perspectives, teaching approaches, and assessment strategies

variety of assessment approaches helps give teachers a much better picture — a holistic view — of student progress.⁶

Classroom talk is an important assessment tool and provides teachers with an opportunity to assess student understanding in depth and provide feedback. Using design scenarios where students can discuss and adapt example programs highlight their skills as well as knowledge. Code reviews and showcases where students talk about their work can provide peer and teacher assessment opportunities.⁷

Incorporating assessment activities into lessons embeds assessment. For example, portfolio creation and analysis, or reflection journals requiring students to answer key questions during project development, encourage continuous self-assessment. Using entry and exit tickets, where students quickly record knowledge or confidence about current learning topics before and

after lessons, can be another quick and regular assessment approach, too.

More traditional assessment tasks, such as multiple-choice questions (MCQs) and free-text questions, provide formative and summative assessment opportunities. Although effective MCQs can be challenging to create and offer limited feedback, they can be a quick and low effort way to discover student understanding.

Some work can also be marked automatically, and other handy online tools are available, such as software to create Parson's Problems, where students reorder jumbled lines of code or pseudocode (helloworld.cc/parsonspseudo), or software that detects programming constructs in students' Scratch programs (drscratch.org).

What is also important to remember is that formative assessment includes a feedback loop. Students need to understand what they have or have not understood

REFERENCES

- ¹ Tedre, M. (2014). *The Science of Computing: Shaping a Discipline*. CRC Press. helloworld.cc/variety1
- ² Waite, J., & Liebe, C. (2021). Computer Science Student-Centered Instructional Continuum. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (1246-1246). helloworld.cc/variety2
- ³ GOV.UK. (2021). *Research review series: science*. helloworld.cc/variety3
- ⁴ National Centre for Computing Education. (2021, July 15). *Computer Systems and Networking Within the Computing Curriculum*. Teaching and Learning Reports. helloworld.cc/variety4
- ⁵ Sentance, S., & Waite, J. (2021). Teachers' Perspectives on Talk in the Programming Classroom: Language as a Mediator. *Proceedings of the 17th ACM Conference on International Computing Education Research* (266-280). helloworld.cc/variety5
- ⁶ Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, 148, 103798. helloworld.cc/variety6
- ⁷ Grover, S., Sedgwick, V., & Powers, K. (2020). Feedback through formative check-ins. In: S. Grover (ed.), *Computer Science in K-12: An A to Z Handbook on Teaching Programming*. Edfinity. helloworld.cc/variety7

and what is next. In choosing formative assessment approaches, teachers must consider which assessment approach gives the best feedback for their pupils for the subject material you're covering and for the context of the learning at that point in time.

To expand the range of strategies you can use in the classroom, reflect on your perspective of computing. Does your perspective impact the approaches you favour? What new practices could you try that could increase entry points for your students and enhance their experience? How else could you capture and assess your learners' understanding? [\[HW\]](https://helloworld.cc/variety7)

STORYTELLING IN COMPUTING EDUCATION

STORY BY Hayley Leonard

Children's traditional stories, nursery rhymes, and literature provide a rich source of sequences and repetition. At the 14th Workshop in Primary and Secondary Computing Education in Glasgow in 2019 (WiPSCE'19), Sarah Twigg and colleagues presented an approach that uses children's literature to teach computing to primary school pupils. The team from Lancaster University hopes the familiar contexts of children's stories will engage pupils and raise the confidence of non-specialist teachers in delivering the curriculum.

Identifying computing concepts in children's literature

The team reviewed 50 popular children's picture books to identify key computing constructs, namely sequencing, repetition, and selection. The books included programming constructs to varying degrees, but 16 books included all three constructs, and all 50 could be used to demonstrate sequencing. Several books were then used as the basis for sample teaching resources, which were trialled in some classrooms and code clubs.

The teaching approach: Read, Act, Model, and Program (RAMP)

An approach named RAMP builds up subject knowledge and appropriate vocabulary in a storytelling context. The format begins by reading through the story; the teacher asks questions about what is happening, and introduces computing terminology. Children then act out the story and are asked about repeating patterns

of behaviour and what triggers them. The model element of the approach then involves constructing the sequence of events in the story, using either images from the book, or printouts of lines of code or blocks from Scratch. Children are asked to identify repeating patterns and choice points in the story, making links to the computing terminology throughout. The program step is supported by the sample lesson resources developed for specific books. It involves using the computer to produce the program that has been designed through the previous unplugged activities.

Teachers' evaluation of the resources

Responses from teachers who were asked about their experiences were highly positive. In particular, teachers said that the first three stages (Read, Act, and Model) were very engaging for the pupils. They provided multiple opportunities for differentiation and working together at different levels of ability.

Some teachers suggested, however, that some non-specialist computing teachers might find the step up to the Program stage daunting. The authors are using this feedback to help them develop the teaching resources. They are continuing to work with teachers

LESSON EXAMPLE

Computing concepts represented in *We're Going on a Bear Hunt* by Michael Rosen:

Sequencing: A list of events to be followed in order.

Example: The characters on the bear hunt go through six different environments in

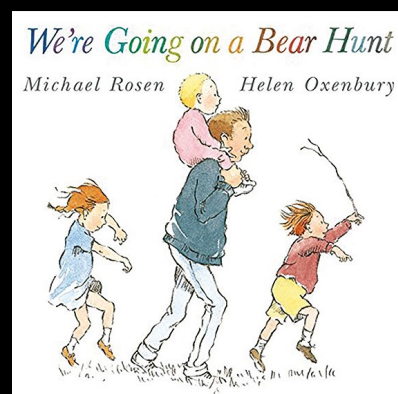
order. On their way back home, they go through the environments in reverse order.

Repetition: At least one example of a pattern of repeated dialogue, actions, or environment.

Example: Repetition of particular phrases in each environment, for example, "We're going on a bear hunt."

Selection: At least one example of a choice of dialogue, actions, or environment.

Example: The end of the repeated dialogue varies depending on the environment: for example, in the river they say, "Splash splosh!"



Text © 1989 Michael Rosen, Illustrations © 1989 Helen Oxenbury, From *WE'RE GOING ON A BEAR HUNT* by Michael Rosen. Reproduced by permission of Walker Books Ltd, London SE11 5JU, www.walker.co.uk

“ THE APPROACH IS LOW-COST AND USES FAMILIAR CONTEXTS

to investigate whether more support is required for the transition to the final stage, or whether this should be separated from the other elements of the approach.

An inclusive approach

Teaching computing principles to young children can be a challenge. A creative storytelling approach is low-cost and uses familiar contexts that are intuitive to teachers and parents. Twigg and her colleagues suggest that this approach has benefits for diversity in computing, and they are particularly interested in its use with disabled students. The collaborative activities and active discussion involved could also be beneficial in engaging more girls in computing. A pilot study testing this approach more widely started in England in September 2020, as part of the Gender Balance in Computing programme of research (helloworld.cc/genderbalance). (HW)

FURTHER READING

- ✓ Twigg, S., Blair, L., & Winter, E (2019). Using children’s literature to introduce computing principles and concepts in primary schools: work in progress. *Proceedings of the 14th Workshop in Primary and Secondary Computing Education (WiPSCÉ’19)*. Association for Computing Machinery, New York, NY, USA, Article 23, 14. helloworld.cc/literature



Credit: stock.adobe.com/Rawpixel.com

Credit: stock.adobe.com/Bnrichaz Benoit



■ Get students to talk through what they learnt last lesson to a rubber duck!

RETRIEVAL PRACTICE

Gemma Moine explains how she uses retrieval practice, an effective learning strategy and an evidence-based teaching technique, in her computer science classroom

In recent years, educational research has praised the findings of cognitive psychology research on retrieval practice: the idea that bringing information to mind can boost learning. It's been revealed that the mechanics of the memory have a large impact on learning. Understanding research and translating this into the classroom is key, but it can be very difficult for teachers to put into practice when workloads are high and time is precious.

The majority of retrieval tasks undertaken in my classroom have been taken (maggied) from amazing teachers who have posted examples on social media, and adapted for computer science. I highly recommend you research the topic further using the original source links provided for further examples and ideas. This article aims to give teachers a toolbox of simple retrieval tasks that can

easily be embedded into the classroom. Tasks are ordered in preparation time, to help teachers decide which ones they may like to trial in the classroom.

Brain dumps

How: Brain dumps are among the easiest retrieval practice tasks you can incorporate into the classroom with minimal preparation. Students have a maximum of five minutes to write down as much as they can recall on a specific topic given by the teacher. Answers can be written on paper or — my favourite — straight onto the desk with board markers. Students love a bit of desk graffiti.

Extend: Another useful task is for students to identify areas to revisit by highlighting missed areas on the brain dumps on their knowledge organisers or mind maps. This could be extended further to a homework activity in which students are required to

make flashcards on the identified missing topics and then test one another at the beginning of the next lesson.

Original idea: Maggied and adapted from @RetrieveLearn.

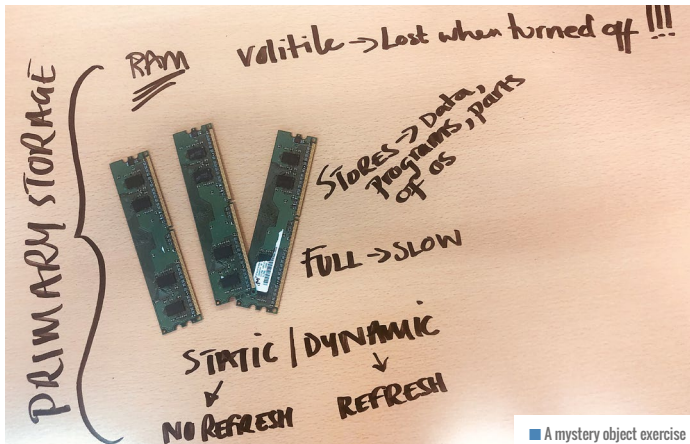
Take three

How: This is a super-quick and easy-to-embed retrieval task. Ask students to write down on paper three things they learnt last lesson, last week, or last term. Give students the opportunity to think-pair-share what they have written down.

Extend: Ask students to find peers with written comments from the same topic (or where there are linkages between topics) and discuss.

Mystery object

How: Place an object on the desk, then ask students to recall as much as they can about



■ A mystery object exercise



the item. It could be a stick of RAM, an old network switch, two different types of wires, input/output devices, or an old floppy disk. Students write down what they can recall and compare that to mind maps, knowledge organisers, or notes.

Workload: Contact your IT network manager for old computers or spare parts.

Fill in the blanks

How: Another quick retrieval task is to take a knowledge organiser or mind map, cover some of the words, and ask students to try to recall what is missing. This is a slight cheat, as students can see visual prompts, so it's more of a guided retrieval practice. I tend to use it as a follow-on from a previous lesson's retrieval practice task in which gaps in knowledge or understanding were identified.

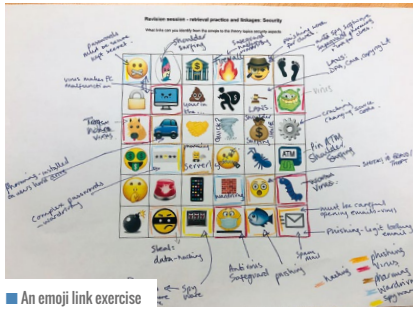
Workload: Laminate knowledge organisers and mind maps in advance to make them reusable, then use stickies or page flags to cover the words.

Talk to the duck!

How: Many computing classrooms have adopted debugging code with the aid of the faithful rubber duck (helloworld.cc/rubberduck). Make the duck part of an easy retrieval task by asking students

to talk through with the duck what they learnt last lesson, week, or term. Bringing that information to mind changes the way information is stored and makes it easier for students to recall later.

Workload: Use a screwdriver to remove the squeak and save teacher headaches!



■ An emoji link exercise

Emoji links

How: Students link emojis to sections from a theory topic; this should take no more than five minutes to complete. Emojis should be purposefully selected, with a few random emojis, to see what fun ideas the students can come up with. This retrieval task is also a nice example of incorporating dual coding in the classroom.

Extend: This task could be extended as homework, with students using a different-coloured pen to find further links using their knowledge organisers, mind maps, notes, or flashcards.

Workload: Reduce the teacher workload by setting as homework the challenge to design the emoji grids based on a given topic.

Flash cards

How: Following the Leitner flash card method is an effective technique for retrieval

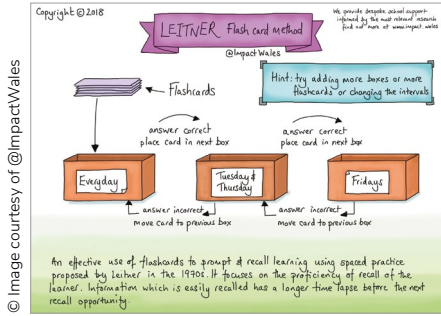
TOP TIPS

- Make the tasks universal, so the activity can be easily modified across topics.
- Make the tasks quick to complete, so they do not dominate the lesson.
- Tasks should involve everyone. Each activity outlined here can be completed individually, in pairs, or small groups.
- Each task described here takes from three to ten minutes. Students may take longer at first, but pace should quicken once the task has been used in lessons a couple of times.
- Tasks need to be low-stake and should not require recording of results. It's important to circulate, to observe any common misconceptions.
- Feedback is essential, so that students know what they got right! This does not mean more work for the teacher - students should self-mark by comparing their answers to mark schemes, knowledge organisers (single A4 documents that contain key basic facts and knowledge on a given topic), mind maps, flash cards, or notes.

GEMMA MOINE

Passionate about computer science, Gemma is a secondary computer science teacher at the British School Al Khubairat in Abu Dhabi, United Arab Emirates (@BSAKComputing, @BSAKAbuDhabi).

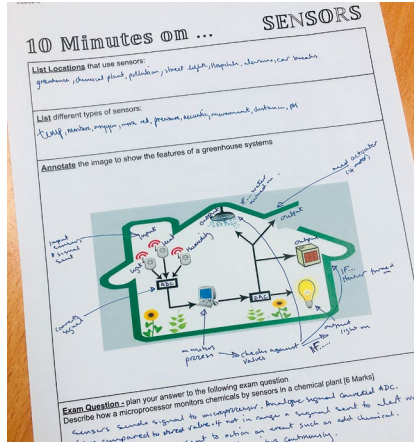




practice. This involves using spaced practice and recall, by writing answers down before turning flash cards over to check the answers. Impact Wales has an amazing poster to help guide students through this process (helloworld.cc/leitner), but students can use envelopes to store different piles. This task should take no more than three to five minutes of lesson time and is an excellent method to help prepare students for revision. **Workload:** Students should create their own flash cards, as the process of writing them helps reinforce learning.

Great packet race

How: Students race across a map of the USA, claiming states (renamed with a tech twist) as they answer questions correctly. The map is generic and can be played on any topic. If played on a teacher's board, students can work in teams to answer questions. The game can also be printed and played in small groups. Questions and answers can be prepared, or students can use their knowledge organisers, mind maps, or flash cards to generate questions. **Workload:** Use the retrieval grids mentioned



■ One of the 'Ten minutes on' sheets

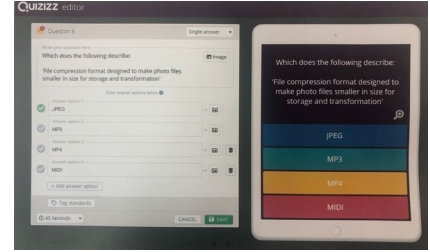
later in the article for the teacher question bank, to reduce workload. **Extend:** Set the task of annotating the map after the game as homework, linking the name of tech states to theory topics and keywords. **Original idea:** Magpied and adapted from @SPBeale (helloworld.cc/maggame).

Ten minutes on

How: Students have ten minutes to list, explain, or compare one specific topic. They also need to annotate an image, which nicely incorporates dual coding, and plan how they would answer an exam-style question. **Extend:** Students can complete their final exam-style question and answer as homework.

Quizizz self-marking quiz

How: Create a free teacher account on quizizz.com and make a bank of quizzes on each topic needed. The questions are



■ Quizizz is a useful tool that allows for self-marking, while also letting teachers track progress

multiple choice and allow for pictures and up to five possible answers. The quiz can be set as a live game, homework, or solo game, and can be played by students using a pin code or shared via Google Classroom or Remind. Quizzes should take three to five minutes for students to complete, and the system self-marks. Quizizz also has a feature that allows teachers to track students' progress. **Workload:** You can search for existing quizzes on Quizizz to save and edit if needed. Once quizzes are saved, they can be reused with very little preparation.

Retrieval grids

How: This task involves a little more planning, but retrieval grids are a flexible tool for low-stakes retrieval and feedback. Each box is colour-coded to represent whether the knowledge was learnt last lesson, last week, last month, or way back! Simply add questions on one slide and answers on the following slide. Display the grid on the teacher's board or on students' screens, then ask students to fill in their answers individually on a printed blank grid or directly into their books. The answer grid can be displayed and students self-mark their work. **Workload:** It can be time-consuming to prepare this retrieval task, but you could set up a template at the beginning of the academic year with 20 blank question-and-answer slides. As you work through topics, you can add questions to different slides.

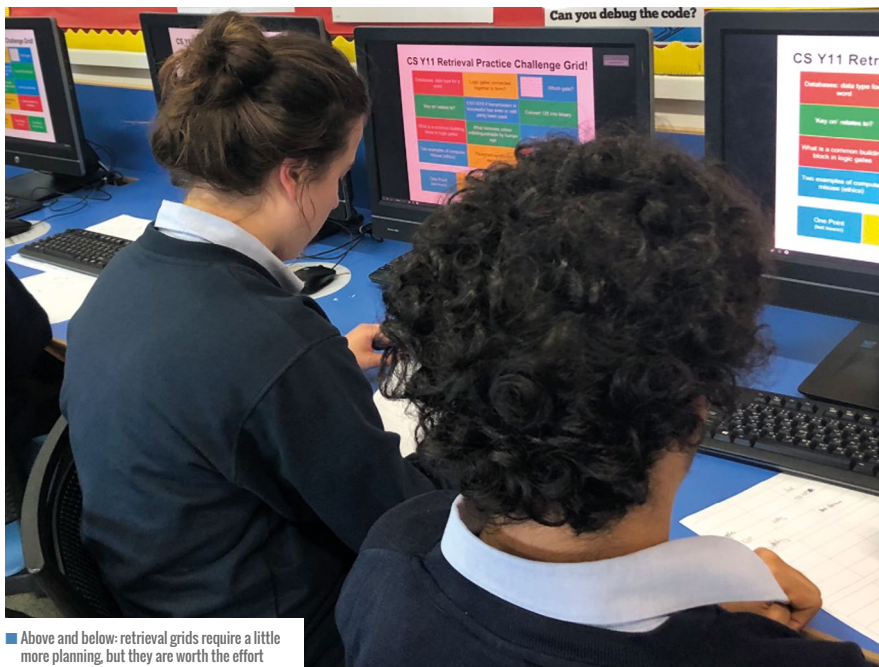
Extend: Ownership can also be taken by students themselves. My sixth-form students were each given a blank template at the beginning of the year and asked to periodically update their own versions. You can also print and laminate the questions and answer grids back to back and use them periodically with individuals. **Original idea:** Magpied and adapted from @KateJones_teach.

IT'S BEEN REVEALED THAT THE MECHANICS OF THE MEMORY HAVE A SIGNIFICANT IMPACT ON LEARNING



■ The great packet race exercise

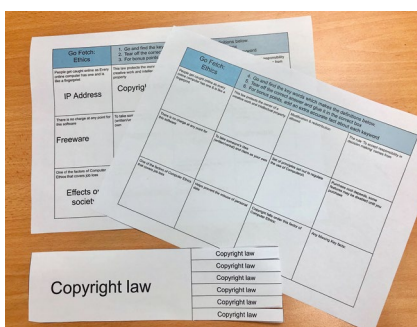




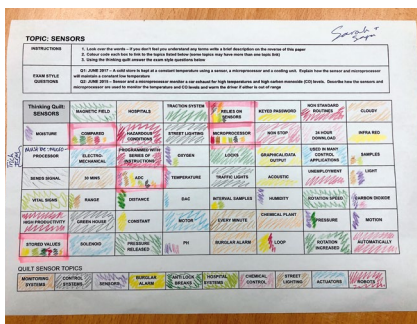
■ Above and below: retrieval grids require a little more planning, but they are worth the effort

CS Y11 Retrieval Practice Challenge Grid!

Convert F3 to denary	Where are binary numbers commonly used in registers	Example of interrupt	
7 contains data about one type of item, person, event	Example use of a buffer	Draw a NOT gate	
Where is hexadecimal used	Odd or even parity? 01101001	Where is the OS loaded to	
State one function of an operating system	Software that allows you to modify the source code	Convert 11100011 into denary	
One Point (last lesson)	Two Points (last week)	Three Points (a bit ago)	Four Points (wow ages ago)



■ The 'Go fetch!' activity involves placing answers around the classroom, and gets people on their feet!



■ An example of a thinking quilt

Go fetch!

How: This retrieval task is a high-energy one that gets students moving around the classroom. Complete the template squares with a description or a key fact on part of a topic. Students race to find the corresponding answers on snippable sheets placed around the classroom. The answers are glued or written into the corresponding box, and students can add extra details for bonus points.

Extend: Make it harder by not including all the answers on the snippable sheets.

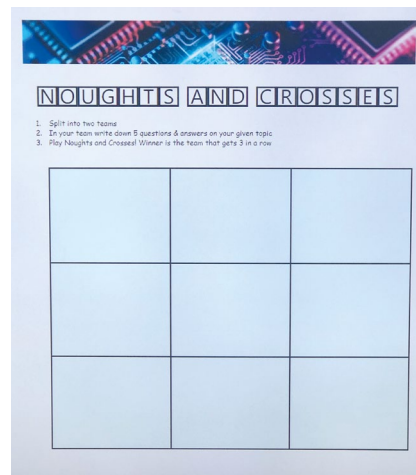
Original idea: Magpied and adapted from @SPBeale.

Thinking quilts

How: Students use the topics along the bottom of the grid to identify related keywords on the main grid. Links between the grid and the topics should be the same colour, with some keywords potentially sharing several topics. Students then use the grid to answer exam-style questions. The task has visual prompts, so is not a complete recall task, but it's a great revision resource.

Extend: Blank spots can be left for students to fill in additional keywords from their own recall, or a blank topic can be left along the bottom for students to identify.

Original idea: Magpied and adapted from @KKNTEachLearn.



Noughts and crosses

How: Display a simple noughts and crosses (tic-tac-toe) grid on the teacher's interactive board. Split the class into two teams and get each one to write down five questions.

Each team takes turns to answer questions, and if a team gets an answer right, places the team's symbol (nought or cross), in a chosen space on the grid. The winning team is the one with three symbols in a row. Alternatively, students work in pairs with their own printed grid and play between two.

Speed it up: Previous homework could be for students to bring three questions and answers to the lesson, with the team then selecting five questions to use, or students could use their flash cards as question banks.

Original idea: Magpied and adapted from @BsaktL. (HW)

FURTHER READING

Retrieval practice is such a powerful technique! If you want to learn more about the topic, I would highly recommend the following books: *Make It Stick* by Peter C. Brown, Henry L. Roediger, and Mark A. McDaniel, and *Powerful Teaching* by Pooja K. Agarwal and Patrice M. Bain.

Finding time to read can be difficult, so I also recommend searching online for chapter summaries on the books. The retrievalpractice.org website is brimming with techniques and further information on the topic.

THE INCLUSIVE COMPUTING CLASSROOM

Make your computing lessons more accessible and inclusive for learners with special educational needs and disabilities, with effective approaches that are beneficial for all students

Consider the students in your computing classes. What are their strengths and weaknesses; their passions and hates; the barriers they face in learning? In an ideal world we would be able to meet the precise needs of every individual learner, but this is simply not practicable.

However, there are some ways to make your computing lessons more inclusive and accessible for the greatest number of learners, in particular those with special educational needs and disabilities (SEND). The good news is that these approaches harness effective pedagogy that can benefit all students in your computing classroom.

In this article, I will outline a few ways for you to improve your practice, in terms of how information is presented to students and how students interact with the learning material. And a word of warning before we get started: these approaches won't work for 100 percent of students, 100 percent of the time! Some students may require specific adaptations. Talk to your individual students and find out what works for them.

Reduce cognitive load

One of the most effective changes we can make as teachers is to reduce the cognitive load of learning new information and concepts. There has been an increased focus on understanding the role of cognitive load in the classroom in recent years and, for more detail, you can read Phil Bagge's article in Hello World issue 8, and the article on page 20 of this special edition.

In essence, the amount of new information being introduced to learners can lead to cognitive overload, as the capacity of their

working memory is finite. The complexity of the information and how it is presented can also increase cognitive load. Many students with SEND have poor working memory and so can reach overload sooner than their peers.

Here are some ways to reduce cognitive load for learners with SEND:

- Teach key vocabulary in advance of a topic. Provide word lists that can be sent home for students to learn, ideally with image support. When the word is then met in the classroom, the learner doesn't have to use up working memory decoding the word, or remembering how it is spelt, and can concentrate on understanding it in context.
- Teach basic skills explicitly and routinely. Once knowledge is transferred to long-term memory, it doesn't use space in working memory. As such, if students have a routine of logging on and accessing work from the same folder at the same stage in each lesson, it's more likely to become habit.
- Introduce content in smaller chunks, and practise what is learnt before moving on.
- Use familiar contexts to introduce new programming and computer science concepts, such as repetition and selection. This can be done effectively in unplugged tasks, to reduce the amount of new information being introduced at one time.

Accessible teaching materials

Provide information in a range of formats — such as text, images, video, and audio — so that students with sensory disabilities can access content, and to support weaker readers. In addition, presenting information both verbally (narration or text) and non-


verbally (for example as an image) allows the learner to access more working memory capacity, and can enhance recall. This is the central idea of Paivio's dual coding theory (helloworld.cc/paivio1971).

When it comes to font, colour, layout, and content, ensure materials are accessible by the greatest number of students. For example:


- Use a sans-serif font of at least 12pt in documents or 24pt in presentations
- Avoid italics and underlining, which make text harder to read
- Include lots of white space and break up text with titles, paragraphs, and bullet points to help readers make sense of content
- Make sure that the text colour contrasts well with the background colour, and don't use colour as a sole indicator of meaning
- Use simple language where possible, and keep your sentences short

Finally, you can make your teaching materials more accessible by making the most of in-


Part of a computer – key words



screen

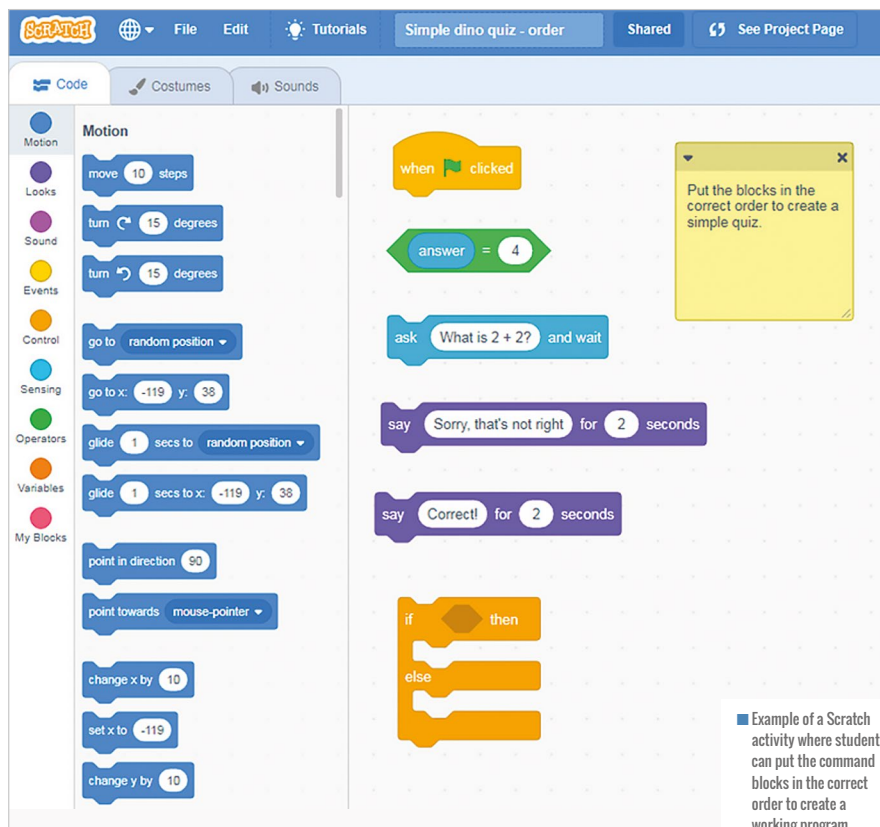


mouse



keyboard

■ Provide word lists with image support so learners can prelearn key words



built assistive technology in the everyday tools you use. Immersive Reader in Microsoft OneNote and Office 365 will read text aloud and highlight key parts of speech, and includes a picture dictionary (see onenote.com/learningtools). Pupils with poor or slow typing can use Voice Typing in Google Docs

through physical objects that can be touched and described. “This can make it much easier to explore the concepts involved and makes it easier to ask questions about things that aren’t understood ... By providing a physical representation, the learner can point to and ask the question at the level of the analogy

incredibly useful for scaffolding learning across the computing curriculum (also the more detailed PRIMM framework, see page 22). The cognitive load associated with creating digital content or writing a program from first principles is much greater than when adapting a working model. Students can begin by running working programs or playing a good-quality animation. They can learn about the key concepts and features without having to worry about making mistakes or writing a large amount of text.

Students can then move on to modifying a working program or a template of digital content to create a more personalised version. This provides a level of guaranteed success in the activity, which will help to boost the confidence of learners with SEND and increase engagement with the learning. This model also helps teachers to include every student in the lesson, with a number of different entry levels. You can find a selection of Scratch activities at sheffielddcl.net/scratch that can be used to scaffold learning for all students, with options to debug, order, explore, and modify code.

For more information on these strategies, and more specific advice on programming environments and activities, try the Raspberry Pi Foundation’s free online course *Creating an Inclusive Classroom: Approaches to Supporting Learners with SEND in Computing* (helloworld.cc/sendcourse). (HW)

“ ONE OF THE MOST EFFECTIVE CHANGES WE CAN MAKE IS TO REDUCE COGNITIVE LOAD

to add content. Android tablets and iPads also include a number of accessibility options, including magnifier, zoom, screen readers, and colour options.

Provide variety in activities

Provide a variety of ways for students to interact with learning material, for example by using a mix of unplugged activities, physical computing devices, and screen-based tasks.

Unplugged activities tackled away from technology are an effective way of introducing computer science and programming concepts. Curzon et al. detail how they can help students make sense of abstract concepts

rather than having to fully verbalize it at the technical level.” (helloworld.cc/curzon2018).

Allow students to present their learning through a range of media, for example animations, videos, comic strips, and graphic organisers. A student with poor spelling and slow writing may be able to express themselves far more effectively, and with a wider vocabulary, by recording a video or audio clip or drawing a diagram than via a written answer to a question.

Scaffolding learning activities

The Use–Modify–Create model posited by Lee et al (helloworld.cc/lee2011) is



CATHERINE ELLIOTT

Catherine is the SEND lead for the Sheffield eLearning Service (sheffielddcl.net), and she works on ways to make the subject accessible to all learners. She is a member of the CAS Include working group, and leads the SEND Virtual and the Sheffield and South Yorkshire Secondary CAS Communities (@[catherinelliott](https://twitter.com/catherinelliott)).

ART AND ALGORITHMS

Katharine Childs takes inspiration from the artist Sol LeWitt to create algorithmic art in primary computing lessons



■ This wall drawing by Sol LeWitt, in the Spoleto Carandente Museum, is an inspirational starting point for children to write algorithms for their own artwork

Credit: Wikimedia Commons

At first sight, computing and art are an unlikely combination: computing seems precise, prepared, and predictable; art seems creative, expressive, and ambiguous. Sol LeWitt (1928–2007) was an American artist who conceptualised that the instructions for creating art were as important as the finished piece. He gave his instructions and diagrams to other people so that they could follow them and create wall paintings.

I have developed primary computing activities based on a particular type of LeWitt's art. These comprise coloured stripes, and involve children writing their own algorithm to draw a stripy picture. An algorithm is a sequence of instructions or a set of rules to achieve something — in this case, some artwork. Once the algorithms have been written, children will work in pairs. One child can take the role of the artist who reads out their instructions, and the other will be the creator who carries out the directions exactly as they are instructed. It's worth noting that LeWitt made some of his instructions deliberately ambiguous, whereas in the classroom, children will need to be precise with their algorithms so that they can be interpreted by a computer, or by a human artist working in a robotlike way.

Adapting across learning stages

This idea works well as an unplugged activity; choose art materials that can quickly create blocks of colour, for example painting on large sheets of paper taped to the floor, or using chalks on the playground tarmac. Just like LeWitt's art, the creative space needs to have some boundaries marked, so that children can use positional language to describe where to add the stripes. Horizontal or

vertical stripes are the easiest to work with, and the activity can be made simpler by using only two colours, or made more complex by using stripes in a repeating pattern. Children will need to think carefully about when to put the pen (or paint or chalk) down onto the paper, and when to pick it up again.

At upper primary, the activity could also be used to write an algorithm to create art in Scratch, a free online programming environment (scratch.mit.edu). It is useful for children to do a small-scale, unplugged version of their design to help them plan their work and give them the structure of their algorithm. Although Scratch contains a paint editor for freehand drawing, the activity works best by programming a sprite to draw the stripes one by one on the stage. This will involve some careful calculations to determine the starting and finishing position of the sprite on each line, and pupils could use variables to store and update the values of these positions. As the sequence of instructions grows, children may notice that they repeatedly go to the starting position,

these omissions, and can help children to debug their algorithm by finding and fixing errors in it.

- 2 Missing out detail:** Children often make the steps in their algorithms too ambiguous. Adding in detail such as positional language can help to make algorithms more precise. This might include using vocabulary such as 'left', 'right', 'under', or 'touching' at lower-primary level, while upper-primary pupils may be able to incorporate precise measurements. Watching someone else carry out their instructions is a useful way to get children to think about how to improve their algorithm by adding precision.

Effective scaffolding

You can support pupils to write precise algorithms by scaffolding this activity with examples. A useful technique is to model how to decompose a task into small steps, including deciding what to do first. Pupils can write the first part of the algorithm, test it on a small-scale, check it for errors, amend what

need support to understand that their role is to help the artist by working in a robotlike way, rather than adding in steps or detail.

At the end of this activity, the artwork and their algorithms make a great display. Match algorithm and art together, or jumble them up and see if others can do the matching!

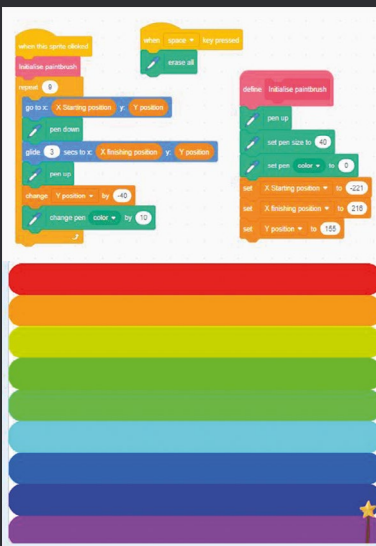
For further ideas for algorithmic art, check out Hello World issue 9, focused on computing and the arts, particularly pages 74–75 and 87–89. [\(HW\)](#)



KATHARINE CHILDS

Katharine is a programme coordinator at the Raspberry Pi Foundation and a former primary school teacher (@IAmKatharineC).

FIND OUT MORE



More information about Sol LeWitt:
helloworld.cc/Tate.LeWitt

An unfinished basic version of a Scratch project referencing LeWitt's work:
helloworld.cc/LeWitt1

A completed Scratch project referencing LeWitt's work:
helloworld.cc/LeWitt2

MODELLING HOW TO LEARN FROM FAILURE CREATES A CLASSROOM CULTURE IN WHICH MISTAKES AREN'T A PROBLEM

put the pen down, draw the stripe, pick the pen up, then change the coordinates and the pen colour ready for the next line. Spotting this pattern helps to determine where to use a repeat loop in the algorithm.

Common misconceptions

When writing algorithms, pupils often underestimate how precise the instructions need to be so that a computer can understand them. This precision can be misunderstood in two ways:

- 1 Missing out steps:** It's easy to make assumptions about what needs to happen to create a piece of art. In these activities, a common misconception is that children forget to explicitly say when to put the pen down on the paper and when to pick it up. Having someone else carry out their instructions is a good way of spotting

they have done so far, and then write some more. This run–test–fix–add cycle is a good habit to develop, and can be used as part of a toolbox of techniques to be displayed on tables or on the whiteboard.

There are many opportunities for gotcha moments when writing an algorithm for someone else, so expect to hear comments such as, "I didn't mean that!" Modelling how to learn from failure at the start of the activity creates a classroom culture in which mistakes aren't a problem in themselves, as long as you persevere and put them right. Classroom displays often include the final version of a piece of work, but displaying an algorithm including errors side by side with the debugged version sends a powerful message that it's OK to fail and learn.

Learning from failure is also important when pupils take the role of the creator and follow the algorithm. Some children may

PROGRAMMING AND PLAY

How embracing play made me a better educator

We can find inspiration to become better technology educators in unlikely places. Follow me on my journey to rediscover playfulness in programming, which took me, among other places, to a small town in Italy.

When I first started writing storybooks about programming, I knew almost nothing about pedagogy. I enjoyed programming, but I mixed Piaget with Papert and didn't differentiate between computational thinking and constructivism. I just had a strong sense of the kind of world I'd like to create.

CREATIVITY IN PROGRAMMING

■ Decomposition and logical thinking

As Ruby says, "Even the biggest problems in the world are just tiny problems stuck together." Every programmer starts by breaking down the problem at hand.

■ Creativity and collaboration

Even though the instructions a programmer gives to a computer need to be exact, in the right sequence, and carefully named, programming is also highly creative. Encourage students to instruct each other on how to brush their teeth, and see how many different ways there are to give the necessary commands!

■ Debugging and persistence

Learning to program is all about learning to overcome mistakes. Even the best programmers forget a semicolon from time to time, and need to go back and find the mistake.

For me, computing was magical, charming, and imaginative — but the materials teaching it were often dull and uninspiring.

Programming as make-believe

Most of my childhood was spent in a very immersive world of make-believe. My siblings and I built small villages in the woods for Moomins and created galactic maps around Star Wars heroes. On the asphalt in our front yard, we sailed in a self-made raft and imagined a mysterious sea around us.

This is the way I relate to programming, even today. Being able to build ever more complicated worlds and structures without the need for physical components like LEGO bricks is fascinating, especially for a child. Most children, at least once in their lives, feel very powerless. Someone else comes up with the rules — but not in programming.

How is programming related to play?

When I decided to learn programming back in 2009, using narrative as a learning tool was a natural fit. I was learning a programming language called Ruby, and every time I ran into a word or concept I didn't understand (such as 'object-oriented programming' or 'garbage collection'), I would try to explain the concept as a six-year-old girl called Ruby would explain it. This project eventually turned into a series of books explaining and celebrating computing, from the tiniest Booleans to the most immense algorithms.

Luckily, on my journey to writing about computing in early childhood, I stumbled upon the work of Seymour Papert and Alan Kay, which made me realise that

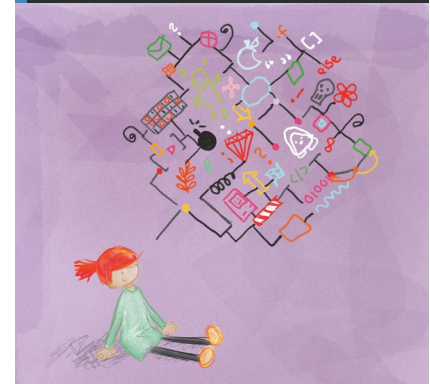
creativity was something that had been built into computing education, but was something we'd somehow lost. Even Alan Turing wrote a whimsical note in an early artificial intelligence paper on how we should teach computers like children, teaching them to learn to learn.

Whenever I asked educators about play and programming, they would direct me to apps that gamified learning. Programming education was experienced through completing challenges, collecting points, or winning competitions. But the type of programming I really enjoyed was full of other types of play: finding and giving support, exploration, and the joy of finding a new way to solve a problem. To rediscover playfulness in programming, I needed to visit a small town in Italy.

"The computer is like a foreigner, and if you want to talk to it, you have to speak its language."

"Yes, but the computer has to understand how we talk, too, and it has to do what we want it to do."

[The Hundred Languages of Children: the Reggio Emilia experience in transformation](#)





■ Download this free exercise to build a computer out of paper and get to know the components inside it at helloruby.com/play

The hundred languages of children

In an Italian city called Reggio Emilia, I finally found the framework of thinking I needed to create Ruby's world. Reggio Emilia is an educational approach for preschool and primary education, named after the city where it began. From the outside, it has very little to do with computing: the approach highlights respect, responsibility, and community through artistic exploration and discovery.

The first thing I learned to love from Reggio Emilia was the idea of a hundred languages. Its core idea is that a child has hundreds of ways of expressing themselves: with clay, gestures, paint, and rubber stamps. However, in schools we often limit children to writing and reading. Reggio Emilia educators treat the computer as just one more material to learn, alongside paper, ruler, pens, and movement — one of the hundred languages.

The second thing I fell in love with in Reggio Emilia was the open-ended nature of projects that can take all sorts of twists and turns. Many of my own favourite exercises start with kids posing questions that interest them, like “What kind of computer would a dolphin doctor need?”, “What is the world’s most dangerous animal?”, or “What if my paper computer could print candy?” Throughout the process of exploring and experimenting, they learn about abstraction, collaboration, and media literacy, and develop a plethora of powerful ideas I would never have anticipated. That’s why most of the exercises I create for kids include discussion points, and very few of them have right or wrong answers. I think it is important to give kids permission to trust themselves, and to allow for many right answers to a question.

The third thing that resonated with me is the idea of observing children at work and responding to their unique needs. I’ve learnt to simplify my writing, creating exercises and materials that have only a single concept to teach. Reggio Emilia also suggests that we shouldn’t use words as shortcuts to knowledge. Computer science is riddled with abstract words such as functions, Booleans, and decomposition. But what does a loop feel like? And can we find conditionals in the everyday surroundings of kids, such as the way they choose clothes for a rainy or sunny day? Computational thinking concepts are more fascinating when we notice their presence all around us. Inspired by Reggio Emilia, I’ve practised making computer science concrete, specific, and understandable to the child. A computer can take a thousand forms.

I wish to see programming become one tool in a big box of self-expression — along with crayons, blocks of wood, prisms, and pipettes. This can help us to present a more colourful, exciting computing culture. Why does Reggio Emilia keep inspiring me after 70 years of existence? I think the answer lies in wonder. These pedagogical movements have helped me to rediscover my own wonder around technology. It is this wonder that allows me to invent new teaching practices that offer unusual and beautiful pathways to computing. (HAW)

FURTHER READING

- Hello Ruby: helloruby.com
- The Hundred Languages of Children: helloworld.cc/100lang
- Understanding the Reggio Approach: Early Years Education in Practice: helloworld.cc/reggio



LINDA LIUKAS

Linda is a programmer, storyteller, and illustrator from Helsinki, Finland. She is author of the ‘Hello Ruby’ series.

“HELLO, WORLD!”

Everything you need to know about our computing and digital making magazine for educators

Q WHAT IS HELLO WORLD?

A Hello World is a magazine for computing and digital making educators. Written by educators, for educators, the magazine is designed as a platform to help you find inspiration, share experiences, and learn from each other.

Q WHO MAKES HELLO WORLD?

A The magazine is a joint collaboration between its publisher, Raspberry Pi, and Computing at School (part of BCS, the Chartered Institute for IT).

Q WHY DID WE MAKE IT?

A There's growing momentum behind the idea of putting computing and digital making at the heart of modern education, and we feel there's a need to do more to connect with and support educators, both inside and outside the classroom.

Q WHEN IS IT AVAILABLE?

A Your 100-page magazine is available three times per year. Check out our new podcast too, to get more great Hello World content between issues.



IT'S FREE!

Hello World is free now and forever as a Creative Commons PDF download. You can download every issue from helloworld.cc. Visit the site to see if you're entitled to a free print edition, too.

WANT TO GET INVOLVED?

There are numerous ways for you to get involved with the magazine.
Here are just a handful of ideas to get you started.

- **Give us feedback**

Help us make your magazine better – your feedback is greatly appreciated.

- **Ask us a question**

Do you have a question you'd like to ask? We'll feature your thoughts and ideas.

- **Tell us your story**

Have you had a success (or failure) you think the community would benefit from hearing about?

- **Write for the magazine**

Do you have an interesting article idea? Visit helloworld.cc/writeforus to submit your idea.

GET IN TOUCH

Want to talk? You can reach us at:
contact@helloworld.cc

FIND US ONLINE

www.helloworld.cc

 [@HelloWorld_Edu](https://twitter.com>HelloWorld_Edu)

 fb.com/HelloWorldEduMag

**NEXT
ISSUE
OUT**
OCTOBER
2021

**(Hello
World)**

helloworld.cc

