1. A Python program utilizes a single core unless the programmer explicitly uses multithreading. Fortunately, the threading module has Thread class which we can inherit as the starting point for our threads. Each Thread object might be scheduled by the PVM to run on separate processor cores. If we run the following program OUTSIDE of IDLE by double-click on the file incrementingThreads.py, it increments a single variable a specified number of times using a specified number of threads. Thread methods are:

<table>
<thead>
<tr>
<th>Thread Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong>(name = None)</td>
<td>Constructs a thread object with an optional name</td>
</tr>
<tr>
<td>getName()</td>
<td>Returns the thread's name</td>
</tr>
<tr>
<td>setName(newName)</td>
<td>Sets the thread's name to newName</td>
</tr>
<tr>
<td>run()</td>
<td>Executed when the thread acquires the CPU.</td>
</tr>
<tr>
<td>start()</td>
<td>Makes the new thread ready. Raises an exception if run more than once.</td>
</tr>
<tr>
<td>isAlive()</td>
<td>Returns True if the thread is alive or False otherwise</td>
</tr>
</tbody>
</table>

```python
# File: incrementingThreads.py  Illustrates concurrency with multiple threads.
import random
from threading import Thread
from time import clock

class IncrementingThread(Thread):
    # Represents an incrementing thread
    
    def __init__(self, number, counterListRef, increment):
        # Create a thread with the given name and
        # increments the counter by incrementAmount
        Thread.__init__(self, name = "Thread " + str(number))
        self._counterList = counterListRef
        self._incrementAmount = increment

    def run(self):
        # Increments the counter by the incrementAmount by a loop.
        for count in xrange(self._incrementAmount):
            self._counterList[0] += 1
        print self.getName(), "is done."

def main():
    # Create the user's number of threads. Then start the threads.
    maxThreads = input("Enter maximum number of threads: ")
    targetAmount = input("Enter number of times all threads should increment: ")
    for numThreads in xrange(1,maxThreads+1):
        start = clock()
        incrementAmount = targetAmount/numThreads
        incrementAmountLastThread = targetAmount - (numThreads-1)*(incrementAmount)
        threadList = []
        counter=[0]
        for count in xrange(numThreads-1):
            threadList.append(IncrementingThread(count+1, counter, incrementAmount))
        threadList.append(IncrementingThread(numThreads,counter,incrementAmountLastThread))
        for thread in threadList:
            thread.start()
        allThreadsDone = False
        while not allThreadsDone:
            for thread in threadList:
                if thread.isAlive():
                    allThreadsDone = False
            end = clock()
            runTime = end - start
            print "Counter should be", targetAmount,"but the counter =",counter, "with a",
            print "time of %.6f sec." % runTime

main()
raw_input("Hit <Enter> to quit")
```
Consider the following screen capture with a maximum of 5 threads and an increment amount of 10,000,000.

<table>
<thead>
<tr>
<th>Thread</th>
<th>Counter Value</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[10000000]</td>
<td>18.204649</td>
</tr>
<tr>
<td>2</td>
<td>[7248169]</td>
<td>14.204239</td>
</tr>
<tr>
<td>3</td>
<td>[5442444]</td>
<td>12.019376</td>
</tr>
<tr>
<td>4</td>
<td>[4281350]</td>
<td>11.449488</td>
</tr>
<tr>
<td>5</td>
<td>[4032293]</td>
<td>10.473196</td>
</tr>
</tbody>
</table>

Enter maximum number of threads: 5
Enter number of times all threads should increment: 10000000
Thread 1 is done.
Counter should be 10000000 but the counter = [10000000] with a time of 18.204649 sec.
Thread 1 is done.
Thread 2 is done.
Counter should be 10000000 but the counter = [7248169] with a time of 14.204239 sec.
Thread 1 is done.
Thread 3 is done.
Thread 2 is done.
Counter should be 10000000 but the counter = [5442444] with a time of 12.019376 sec.
Thread 4 is done.
Thread 2 is done.
Thread 1 is done.
Thread 3 is done.
Counter should be 10000000 but the counter = [4281350] with a time of 11.449488 sec.
Thread 1 is done.
Thread 4 is done.
Thread 2 is done.
Thread 3 is done.
Thread 5 is done.
Counter should be 10000000 but the counter = [4032293] with a time of 10.473196 sec.

Hit <Enter> to quit

a) Why do the threads sometimes complete out-of-order?

b) Why is the counter only correct with a single thread?