Code Style/Quality, Activity #2

Consider the following solutions for - Func G2
https://www.cs.uni.edu//~schafer/cohort23/FOP/lessons/Unit2/w9/GP/index.html

```
#=============  Solution 1
def qbCategory(passerRating):
    if passerRating < 0:
        return "ERROR"
    elif passerRating >=0 and passerRating <= 85:
        return "Bad"
    elif passerRating > 85 and passerRating <= 90:
        return "Mediocre"
    elif passerRating >90 and passerRating <= 95:
        return "Good"
    elif passerRating > 95 and passerRating <= 158.33333:
        return "Great"
    else:
        return "ERROR"

#=============  Solution 2
def qbCategory(answer):
    if answer > 158.333333 or answer < 0 :
      answer = "ERROR"
    elif answer <= 85:
            answer = "Bad"
    elif answer <= 90:
            answer = "Mediocre"
    elif answer <= 95:
            answer = "Good"
    else:
            answer = "Great"
    return answer

#=============  Solution 3
def qbCategory(rating):

    if rating <0 or rating >=158.4:
        return ("ERROR")

    elif rating <=85:
        return ("Bad")

    elif rating >85 and rating <=90:
        return ("Mediocre")

    elif rating >90 and rating <=95:
        return ("Good")

    else:
        rating >95 and rating <=158.333333
        return ("Great")
```

```
#=============  Solution 4
def qbCategory(cRating):
    if cRating <0:
        return ("ERROR")
    elif cRating >-0.0000001 and cRating <85.0000001:
        return ("Bad")
    elif cRating >85 and cRating <90.0000001:
        return ("Mediocre")
    elif cRating >90 and cRating <95.0000001:
        return ("Good")
    elif cRating >95 and cRating <158.3333334:
        return ("Great")
    else:
        return ("ERROR")
```

Consider the following solutions for - Func G4
https://www.cs.uni.edu//~schafer/cohort23/FOP/lessons/Unit2/w9/GP/index.html

```
#=============  Solution 1
def dogAge(humanyrs):

    if humanyrs < 0:
        return -1

    else:
        if humanyrs <= 2:
            age = 10.5 * humanyrs

        else:
            age =  21 + (humanyrs - 2) * 4

        cleaned = round(age, 1)
    return cleaned

#=============  Solution 2
def dogAge(years):
    yr_float= float(years)
    if yr_float>2:
        result = round(21.0 + (yr_float - 2)*4, 1)
    elif yr_float>=0 and yr_float<=2:
        result = round(yr_float * 10.5, 1)
    else:
        return -1
    return result

#=============  Solution 3
def dogAge(humanYears):
    if humanYears < 0:
        return -1
        print(-1)
    elif humanYears <= 2:
        dogYears = round(humanYears*10.5,1)
    else:
        dogYears = round(21+(humanYears-2)*4,1)
    return dogYears
    print(dogYears)

#=============  Solution 4
def dogAge(human_years):
    if human_years < 0:
        return -1

    if human_years <= 2:
        dog_years = human_years * 10.5
    else:
        dog_years = 2 * 10.5 + (human_years -2) * 4
```

```
    return round(dog_years, 1)
```

Consider the following solutions for
https://www.cs.uni.edu//~schafer/cohort23/FOP/lessons/Unit3/w13/GP/index.html

```
#=============  Solution 1
def addDensities(filename, newfile):
  fin = open(filename, "r", encoding = "utf-8")
  fout = open(newfile, "w", encoding = "utf-8")
  header = fin.readline()
  header_strip=header[0:-1]
  new_header = header_strip + ",Density\n"
  fout.write(new_header)
  for territory in fin:
    data = territory.split(",")
    pop_dens = float(data[1]) / float(data[2])
    fout.write(territory[0:-1])
    fout.write("," + str(pop_dens) + "\n")
  fout.close()
  fin.close()

#=============  Solution 2
def addDensities(filename, outfile):
    fin = open(filename,"r")
    fout = open(outfile,"w")

    header = fin.readline()

    fout.write(header[0:-1])

    fout.write(",Density\n")

    for r in fin:
        data = r.split(",")

        location = (data[0])
        pop = data[1]
        population = int(pop)

        s = (data[2])
        size = float(s)

        density = (population)/(size)

        fout.write(r[0:-1])

        fout.write("," + str(density))

        fout.write("\n")

    fout.close()

    fin.close()
```

```
#=============  Solution 3
def calculateDensity(population,area):
    density = population/area
    return density

def addDensities(if,of):
    fin = open(if,"r")
    fout = open(of,"w")

    header = fin.readline()
    fout.write(header[0:-1] + "," + "Density\n")

    for number in fin:
        fout.write(number[0:-1])
        data = number.split(",")

        population = int(data[1])
        area = float(data[2])
        density = calculateDensity(population,area)

        fout.write("," + str(density))
        fout.write("\n")
    fin.close()
    fout.close()

#=============  Solution 4
def addDensities (inFile,outFile):
    fin = open(inFile, "r")
    fout = open (outFile, "w")
    header = fin.readline()
    fout.write(header[:-1])
    fout.write(",")
    fout.write("Density")
    fout.write("\n")
    for state in fin:
        data = state.split(",")
        print(data)
        population = float(data[1])
        print(population)
        area = float(data[2])
        print(area)

        density = population/area
        density_str = str(density)
        print(density_str)

        fout.write(state[:-1])
        fout.write(",")
        fout.write(density_str)
        fout.write("\n")
```

```
    fin.close()
    fout.close()


#=============  Solution 5
def addDensities (in_file, out_file):
    fin = open(in_file, "r")
    fout = open(out_file, "w")
    fout.write(fin.readline()[0:-1]+ ",Density\n")
    for t in fin:
        td = t.split(",")
        fout.write(t[0:-1]+"," + str(int(td[1])/float(td[2]))+"\n")
    fin.close()
    fout.close()
```