CSCI 1100 — Computer Science 1 Homework 1
Calculations and Functions

Overview

This homework is worth **75 points** toward your overall homework grade, and is due Thursday, September 11, 2014 at 11:59:59 pm. There are three parts to the homework, each to be submitted separately. If any part is late you will be charged late days according to the latest submitted part. Remember, you have three late days for the whole semester. Save them for the later homeworks.

As you learned in Lab 1, all homework will be submitted electronically through the Department of Computer Science server. Please use the link

```
https://www.cs.rpi.edu/submit/submit.php?course=csci1100
```

we used in Lab 1 for all assignments. This link is also available on the Piazza site and will not be repeated in each homework handout. The homework submission server for HW#1 will be up and running by Monday, September 8, 2014, perhaps sooner.

Make a habit of testing your program by running it on Wing. First, make sure that it can run. Then, go through the logic to make sure that it is correct. Submission server can get slow at busy times. Use it for submissions, not for testing your code. Learning to test your code is a big part of programming. However, you will not be penalized for submitting a program multiple times.

General Comments

In each part of this homework, you are given some data and are asked to provide an output. There are a few things that you will be graded on:

- First and foremost program correctness will determine your grade. But you will also get or lose points based on other criteria discussed below.
- No hard-coding of output values. Even though you can easily compute the output and write it directly into your code, you will loose points if you do that. We want to you to store values in variables, use these variables to compute the results. You can store the output of operations into different variables or directly output.

  How do you know if you are hard-coding? Simple. If one of the input values we provided you needs changing, this should be doable with a single change in your program. Otherwise, you are hard-coding.
- Your program should follow the program structure we discuss in Lecture 4 and Lab 2.

  First, a small comment section explaining the purpose of your code and the author, i.e. your name and username.

  Second, you should list all the function definitions with no other code.

  Finally, you should have the main program code.
- Your output must match the solution on submission server. This is part of your grade. Formatting output is a part of the program.

- In part 3, you must use functions. If you do not, you will lose points.
- For each part, you must submit a program with the correct name. Otherwise, the submission server will not be able to execute your program and we will not be able to give you a grade.

  Your programs for each part will be named:

  ```
  hw1_part1.py
  hw1_part2.py
  hw1_part3.py
  ```

  respectively. Each should be submitted separately.

Part 1: Ice Bucket Challenge

How quickly did the Ice Bucket Challenge catch on? Ok, here is some hard data. The following are the number of posts with hash tag #icebucketchallenge and #alsicebucketchallenge on Twitter each day:

	8/10	8/12	8/14	8/16	8/18	8/20	8/22
#icebucketchallenge	200	500	2,000	12,000	24,000	65,000	70,000
#alsicebucketchallenge	100	300	1,500	13,000	25,000	105,000	85,000

Write a program that shows the percentage increase on each day for both hashtags side by side. See the expected output at the end of the homework.

How to solve this? We will help you get started with your first program.

First, try producing the first line with a print statement. It should be very easy.

Second, try to compute and print the first line. `150` is the percentage increase from `200` to `500`, and `200` is the percentage increase from `100` to `300`. For example, from `200` to `500` is an increase of `300` which is `300/200=1.5` or a `150` percent increase.

Once you are done with this first part, you have a choice. You can copy and paste like we did in Lab 1. But, you can also reduce the repetitive code by using functions like we did in Lab 2. Best is to try both and make sure you understand how it works.

When you have tested your code and sure that it works, please submit it as `hw1_part1.py`.

Part 2: Olympic Figure Skating

You are going to help out with Olympic Figure Skating scoring. We will simplify it quite a bit for this homework. You are given the scores of a competitor in two different components: short program and the free skate. For each component, you have scores from 5 judges. The final score for a component is computed by dropping the highest and the lowest score for that component, adding up the remaining scores. The total score is then given as the sum of the scores for the two components. Below is an example:

Component	Judge 1	Judge 2	Judge 3	Judge 4	Judge 5
Short Program	21	32	28	24	29
Free Skating	24	28	19	23	24

Short program: drop `21` and `32`. Free skating: drop `19` and `28`. The final score then should be: `(28+24+29) + (24+23+24) = 152`.

The spread of scores for a component is given by: `(max - min)/(average of remaining scores)`.

For example, for the short program, the average score is: `(28+24+29)/3=27`. The spread is then given by `(32-21)/27= 0.40`.

Write Python code that uses the scores given above, and

- computes the spread of the scores for each component, and prints it,
- the total score for each component, and prints it,
- the total score for the competitor, and prints it,

You will need to think a bit about how to drop the max and min value. Hint: you have some built-in python functions that can be very useful here. You do not need to use anything fancy for this. In fact, we want to you to learn to think differently about these simple calculations.

You can use 10 variables for the above scores or use a function to simplify things. Both are fine. Always do the thing that appears simplest to you first, make sure it is correct. Then, if you have time, try to simplify your program by rewriting parts of it and by introducing functions.

When you have tested your code, please submit it as `hw1_part2.py`.

Part 3: The Panama Canal and so much rain!

In this part of the homework, you must write functions.

Did you know that Panama Canal runs on fresh water? Basically, between the two locks on the two ends of the canal is a man made lake called the Gatun Lake. One lock on the east side connects the lake to Atlantic Ocean, and the lock on the west side connects to the lake to the Pacific Ocean.

Every time a ship passes through the canal, the following happens. First one of the locks is filled with water from the Gatun lake which allows the ship to move from one ocean to the lake. Then, the second lock is again filled with water from the lake which allows the ship to move from the lake to the ocean on the other side. The water used to fill both locks is forever lost to the ocean.

Did I say that the lake is filled with rain water? How much does it rain in Panama exactly? It rains 9 months a year heavily in Panama. Let's try to compute how much rain it would take to power this canal and that would give you an idea of a lower bound on the amount rain.

In this part of the homework, you are going to write a number of functions to help with the computation. Each function is described below. Write a short description after the `def` statement for each function to explain its purpose.

1. `volume_solid(width, length, depth)` Returns the volume of water needed to fill a lock of these given dimensions. It is simply a box of these given dimensions.
2. `water_needed_perlock(volume)` Returns the amount of water needed to fill a lock with the given volume for a full year. Your function should have a local variable called `ships_per_day`, which is currently set to 35. Basically, you will need to compute and return the total volume of water you need to fill a single lock 35 times a day for a full year (365 days).

To fill `600,000 cubic meters` of water in the Gatun lake, you must get 1 millimeter of rain fall. The dimensions of the lock: `width = 32 meters, length = 320 meters, depth = 10 meters`.

Use of all this information to compute the daily rain in Panama during the rainy season. Here are the steps you need to go through to compute this:

- Find and output the total volume of water used in the canal (two locks) yearly in cubic meters. You must call both of the functions defined above for this.
- Find and output the total amount of rain in cubic meters that must fall on a given day in the rainy season (any day in the 9 months of the year that is rainy). Assume a month is 30 days.
- Now, convert and print this amount to the millimeters of rain (generally rain is measured in terms of millimeters or inches).

Do you want to understand how much this is? Compare this to New York state where average rain (or snow) per day is about 2.5 millimeters.

When you have tested your code, please submit it as `hw1_part3.py`.

Expected Output

Here we provide the expected output for each part for the given input values for testing your code. Remember: no hard-coding of values.

Part 1.

```
#icebucketchallenge vs #alsicebucketchallenge, percentage change
150 vs 200
300 vs 400
500 vs 766
100 vs 92
170 vs 320
7 vs -19
```

Part 2.

```
Short program scores 21 32 28 24 29
Free skating scores 24 28 19 23 24
Spread of the short program is 0.407407407407
Spread of the free skating is 0.380281690141
Total score for the short program is 81
Total score for the free skating is 71
The total score for the competitor is 152
```

Part 3.

```
Panama canal statistics:
The total volume of water needed in Gatun lake: 2616320000 m^3
In rainy season, daily rain should be at least: 9690074 m^3
This means, it rains about 16.1501234568 millimeters every day during the rainy season
```