Overview

This homework is due Friday, February 13, 2015 at 11:59:59PM. There are three parts to the homework, each to be submitted separately.

Late Policy: If you submit this homework after the deadline, there will be a 10% penalty per day late. If any of the three parts are late, you will be charged late days according to the timestamp of the last submitted part. Further, no submissions will be accepted 5 days after the deadline. Note that this will be the late policy for all homeworks.

Submitting Your Work: As you will learn in Lab 2, all homework will be submitted electronically through the Department of Computer Science homework server. The link will be made available on the course website and will be up and running by Monday, February 9, 2015, perhaps sooner.

Strategy: Make a habit of testing your program by running it in the Wing IDE. First, make sure that it does indeed run without syntax errors. Then, go through the logic to make sure that it is correct. Note that the submission server might be slow during busy times. Use it for submissions, not for testing your code. Learning to test your code is a crucial part of programming.

General Comments

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

- First and foremost, program correctness will determine your grade. 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 lose points if you do that. We want to you to store values in variables, then use these variables to compute the results. You can store the output of operations into different variables or output results directly.

  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 described below.

  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 of your function definitions.

  Finally, you should have the "main" program code.

- Your output must match the solution on the submission server. This is part of your grade. Formatting output is an important part of the program.

- In part 3, you must use functions. If you do not, you will lose points.
Part 1: Ice Bucket Challenge

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

<table>
<thead>
<tr>
<th>Date</th>
<th>#icebucketchallenge</th>
<th>#alsicebucketchallenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/10</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>8/12</td>
<td>500</td>
<td>300</td>
</tr>
<tr>
<td>8/14</td>
<td>2,000</td>
<td>1,500</td>
</tr>
<tr>
<td>8/16</td>
<td>12,000</td>
<td>13,000</td>
</tr>
<tr>
<td>8/18</td>
<td>24,000</td>
<td>25,000</td>
</tr>
<tr>
<td>8/20</td>
<td>65,000</td>
<td>105,000</td>
</tr>
<tr>
<td>8/22</td>
<td>70,000</td>
<td>85,000</td>
</tr>
</tbody>
</table>

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 do you solve this problem?

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 or you can reduce the repetitive code by using functions. Best is to try both and make sure you understand how and why both approaches work.

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

Part 2: Programming Competition

For this part, you are going to help score a programming competition. You are given the scores of a competitor in two different components of the competition: computational programming and robotics freestyle. 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, then 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:

<table>
<thead>
<tr>
<th>Component</th>
<th>Judge 1</th>
<th>Judge 2</th>
<th>Judge 3</th>
<th>Judge 4</th>
<th>Judge 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational</td>
<td>21</td>
<td>32</td>
<td>28</td>
<td>24</td>
<td>29</td>
</tr>
<tr>
<td>Robotics Freestyle</td>
<td>24</td>
<td>28</td>
<td>19</td>
<td>23</td>
<td>24</td>
</tr>
</tbody>
</table>

Computational programming: drop 21 and 32. Robotics freestyle: 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: \( \frac{\text{max} - \text{min}}{\text{average of remaining scores}} \).
For example, for the computational programming component, the average score is: \( \frac{28+24+29}{3}=27 \).
The spread is then given by \( \frac{32-21}{27}=0.40 \).
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 manmade lake called the Gatun Lake. The lock on the east side connects the lake to Atlantic Ocean, while the lock on the west side connects 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).

To understand how much this is, compare this to New York State where the 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.**

Computational Programming scores 21 32 28 24 29
Robotics Freestyle scores 24 28 19 23 24
Spread of the Computational Programming is 0.407407407407
Spread of the Robotics Freestyle is 0.380281690141
Total score for the Computational Programming is 81
Total score for the Robotics Freestyle 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