CSCI 1100 — Computer Science 1
Fall 2018
Homework 1: Calculations and Strings

Overview

This homework, worth **100 points** total toward your overall homework grade, is due Thursday, September 20, 2018 at 11:59:59 pm. The three parts should each be submitted separately. All parts should be submitted by the deadline or your program will be considered late.

Please refer to the `Submission Guidelines` document before starting this assignment. It will give you details on what we expect and will answer some of the more common questions, including that you need to submit your program through `Submitty` and that `Submitty` will open by **Monday, September 17, 2018** or earlier.

Remember that your output must match EXACTLY the format we put here. The purpose of this is to make testing easier and at the same time teach you how to be precise in your programming using the tools we gave you. We love creativity, but not in HW output formatting!

Part 1: The Lifeguard Problem (25 pts)

The lifeguard must get to the swimmer as quickly as possible in order to prevent them from drowning. The lifeguard has to go through the sand and then swim. The direction which the lifeguard chooses determines the total distance that they would need to travel because their swimming speed is lower than their running speed by some constant factor. We need to find the time required to reach the swimmer. The geometry of the problem is shown in Figure 1 below:

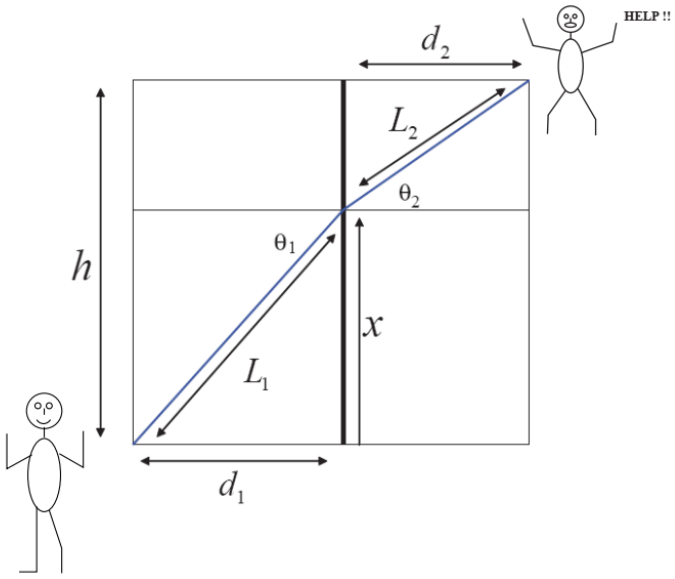

Figure 1: The lifeguard problem (picture credit: Massachusetts Institute of Technology, `https://ocw.mit.edu/courses/mechanical-engineering/2-71-optics-spring-2009/assignments/MIT2_71S09_usol1.pdf`)

Write a program called **hw1_part1.py** that asks the user for 6 values:

- the shortest distance from the lifeguard to water, d_1 (in yards),
- the shortest distance from the swimmer to the shore, d_2 (in feet),
- the lateral displacement between the lifeguard and the swimmer, h (in yards),
- the lifeguard's running speed on sand, v_{sand} (in miles per hour),
- the lifeguard's swimming slowdown factor, n, and
- the direction of lifeguard's running on sand, θ_1 (in degrees).

All values should be converted to floats for computation. You may also assume that there are 3 feet in a yard and 5280 feet in a mile. Remember that trigonometric functions from the `math` module expect the argument to be in radians. Now compute the total time required for the lifeguard to reach the swimmer. You can use the following formulas:

$$L_1 = \sqrt{x^2 + d_1^2}, L_2 = \sqrt{(h-x)^2 + d_2^2}$$

If the running speed on sand is v_{sand}, the swimming speed is $v_{swim} = \frac{v_{sand}}{n}$. The total time is then given by:

$$t = \frac{1}{v_{sand}} (L_1 + nL_2)$$

In the output, the value of the direction (angle) should be printed as an integer and the value of time as a fixed-point floating number with the precision of 1.

Here is an example run of the program (how it will look when you run it using Wing IDE):

```
Enter the shortest distance from the lifeguard to water, d1 (yards) => 8
8
Enter the shortest distance from the swimmer to the shore, d2 (feet) => 10
10
Enter the lateral displacement between the lifeguard and the swimmer, h (yards) => 50
50
Enter the lifeguard's running speed on sand, v_sand (MPH) => 5
5
Enter the lifeguard's swimming slowdown factor, n => 2
2
Enter the direction of lifeguard's running on sand, theta1 (degrees) => 39.413
39.413
If the lifeguard starts by running in the direction with theta1 of 39 degrees,
they will reach the swimmer in 39.9 seconds
```

We will test your code for the above values as well as a range of different values. Test your code well and when you are sure that it works, please submit it as a file named **hw1_part1.py** to Submitty for Part 1 of the homework.

Part 2: Reading the United States Constitution (35 pts)

As we are approaching September 17, it is a good opportunity to remember that on that day in 1787 the delegates to the Constitutional Convention met for the last time to sign the declaration that they had created. This declaration has become known as the United States Constitution. Although some states were recognizing the Constitution Day as early as 1911, it was not until 2004 that the Constitution Day became an official holiday.

As we are celebrating the Constitution Day, let's take a moment and read this outstanding document following the advice of Thomas Jefferson who said, "It is every Americans' right and obligation to read and interpret the Constitution for himself".

Figure 2: The United States Constitution

We are providing a module called `constitution` which contains full original text of the United States Constitution and implements two useful functions. A brief description of these functions along with usage examples are given below:

1. `get_all_text()` provides full text of the United States Constitution. It has no parameters, and the return value is a string containing the text of the Constitution. Usage example (shortened to preserve space; omitted parts are shown by an ellipsis inside brackets [...]):

```
>>> import constitution
>>> print(constitution.get_all_text())
We@the@People@of@the@United@States,[...]We@have@hereunto@subscribed@our@Names
```

2. `substring()` allows you to extract part of the string. This function has three parameters:

   - `string` is the string from which a substring will be extracted
   - `start` determines the index of the first character that will be extracted
   - `count` determines the number of characters that will be extracted

   Note that `string` is not changed; the extracted string is a new string which is returned to the caller.

Here's a usage example:

```
>>> import constitution
>>> print(constitution.substring(constitution.get_all_text(), 21975, 90))

 #Section.@2.#The@Citizens@of@each@State@shall@be@entitled@to@all@Privileges@and@Immunities
```

In order to use module `constitution`, download `constitution.py` file from the `Course Materials` section of Submitty and save it to the same directory where you will be creating your Python code for this part of the homework assignment.

As you can see from the examples above, something bad happened to the text of the United States Constitution, and it is no longer readable. Luckily, you are able to figure out that the original text through the following transformations applied in this order:

1. All double new lines were replaced with a `#` character
2. All single new line characters were replaced with a `^` character
3. All spaces were replaced with a `@` character

Your first task is to write Python code to recover the original text of the Constitution from the string that `get_all_text()` returns. Remember that you must not modify the `constitution.py` file. Only work within your **hw1_part2.py** file. Also keep in mind that any code that you write should perform necessary computations to determine the values that we are asking for. **Do not compute those values manually and insert them into your code as literal values. This approach (called hardcoding) is strictly forbidden and will result in a grade of 0 for the entire part.**

When writing your code, try following some naming conventions and other code style guidelines. Usually, different naming conventions are used by different companies or teams of developers. For instance, for Python there is a "PEP 8" style guide, a Google Python Style Guide, and many others. It is not as important which particular guide you are using, as it is to strictly follow all conventions outlined in the guide your company, team, or instructor have selected. For this class, we will be following the PEP 8 Style Guide for Python Code (`https://www.python.org/dev/peps/pep-0008/`) which is widely used by the Python community. You do not need to read this guide all the way through right now but you might want to take a look at what it is and how it is organized. For subsequent assignments we will be enforcing the rules for naming conventions outlined in this guide, and you will be penalized for not following them. For the first assignment, just try to follow the naming conventions for your variables.

Remember to use readable names for variables in your own programs, especially for variables that hold important values. You will thank yourself when you need to understand your own programs. Shorter variable names are better, but sometimes you need longer names too. Names which are too short are often not very descriptive and should be avoided. Generally, it is recommended to have variable names which are between 3 and 31 characters long.

Once you have recovered the text of the constitution, print the text that you recovered between two lines that act as visual separators. One separator line would be immediately before the text, the other one would be immediately after. Both separator lines are identical and consist of a number of `*` characters that is equal to the average length of a line in the (recovered) text of the Constitution

rounded to the nearest integer ('\n' is not counted as a character for the purposes of determining the line length). An empty line is considered to have a length of 0.

Next print is a message saying how many articles and sections (across all articles) there are in the Constitution; followed by two lines of output containing information about indices within the (recovered) text of the Constitution corresponding to the first and the last characters of "Article. I." The first and the last words of this Article should also be printed. For the purposes of this problem, a word is defined as a sequence of characters before (for the first word) or after (for the last word) a space character and does not include a newline character.

Finally, your program should ask the user to enter a word and then print a message saying how many times that word appears in the United States Constitution. The word in the message should be printed in all caps but when performing counting ignore the case (i.e., if the user entered "TiME" then both "time" and "Time" should be matched with this word when counting the number of occurrences).

To summarize, your program should execute these steps:

1. Obtain a copy of the Constitution text from the `constitution` module
2. Recover the original text of the Constitution by correctly replacing special characters (`#`, `^`, and `@`)
3. Output the text of the Constitution with a string of `*`'s immediately above and below where the number of asterisks in each line is the same as the average length of a line in the Constitution (rounded to the nearest integer)
4. Calculate and print the number of articles and the total number of sections in the Constitution
5. Calculate and print requested information about the beginning and end of "Article. I."
6. Read a word from the user, count the number of occurrences of this word (case insensitive), and print that results

The output of your program might look like the following (some of the text of the Constitution has been omitted for brevity; longer lines have been wrapped to the next line to fit the page width of this document but there should be no additional wrapping in the output that your code generates):

```
****************************************************************************************************
We the People of the United States, in Order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the
    common defence, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and
    establish this Constitution for the United States of America.

Article. I.

Section. 1.

All legislative Powers herein granted shall be vested in a Congress of the United States, which shall consist of a Senate and House of
    Representatives.

Section. 2.

 [...]

Attest William Jackson Secretary

done in Convention by the Unanimous Consent of the States present the Seventeenth Day of September in the Year of our Lord one thousand
    seven hundred and Eighty seven and of the Independance of the United States of America the Twelfth In witness whereof We have
    hereunto subscribed our Names
****************************************************************************************************

There are 7 articles and 21 sections in the United States Constitution

Text of Article I starts at position 342 (character 'S') with the word "Section."
Text of Article I ends at position 13589 (character '.') with the word "delay."

Enter the word to count in the Constitution => TiME
TiME
Word "TIME" appears 21 times (without regard to case) in the text of the United States Constitution
```

Test your code well and when you are sure that it works, please submit it as a file named **hw1_part2.py** to Submitty for Part 2 of the homework.

Part 3: Printing a grid (40 pts)

In this part of the homework, we will have a number of different outputs. Each output is a simple modification of the previous one. So, please complete them in sequence. Then, think about how to modify the solution from the previous step to solve the next part. Feel free to cut and paste your code from one part to the next and modify.

If you cannot complete the whole problem, you will get partial credit for all the parts you completed. The aim is to teach you how to solve problems in an iterative way, test your solution and then add more complexity.

You must not use any IF statements or loops in your program. We have not learned them yet, they are not needed, and they will not make your solution better or more elegant.

Write a simple program that reads a word and two integers. The word will be used in the optional challenge (d) grid. The integers refer to the number of columns and rows in a grid, respectively. Then your program should print increasingly complex grids as explained below. You can find a sample run of the program at the end of this homework statement.

You may assume that the values entered for column and row are odd numbers. You may also assume that the number of columns is at least 3 *and that the number of rows is at least* 5*. This will simplify the program logic.*

(a.) Print the word you just entered (`Your word is:`) and then a grid of three stars `***` given by the number of columns and rows. The columns are separated by a single space. There are no spaces at the end of the line.

(b.) Now, add to your program a second set of print statements that will print `CS1` in the middle of your grid, i.e., in the middle row and column. The remaining grid remains the same.

(c.) Now, add to your program a third set of print statements that will keep the grid from part (b) but also add `^` and `v` in the top and bottom mid points of the grid, respectively, `/` and `\` in the middle of both three character groups (the other two characters in the group should be spaces) that are immediately to the left and to the right of the middle group, respectively, of the second and penultimate rows, and, finally, `|` in the middle of both three character groups (the other two characters in the group should be spaces) that are immediately to the left and to the right of the middle group, respectively, for all other rows (i.e., all rows except the first two and the last two).

Here is a simple run of this program (how it will look when you run it using Wing IDE):

```
Word => Hello
Hello
#columns => 5
5
#rows => 7
7
Your word is: Hello

(a)
*** *** *** *** ***
*** *** *** *** ***
*** *** *** *** ***
```

```
*** *** *** *** ***
*** *** *** *** ***
*** *** *** *** ***
*** *** *** *** ***

(b)
*** *** *** *** ***
*** *** *** *** ***
*** *** *** *** ***
*** *** CS1 *** ***
*** *** *** *** ***
*** *** *** *** ***
*** *** *** *** ***

(c)
*** ***  ^  *** ***
***  /  ***  \  ***
***  |  ***  |  ***
***  |  CS1  |  ***
***  |  ***  |  ***
***  \  ***  /  ***
*** ***  v  *** ***
```

Test your code well and when you are sure that it works, please submit it as a file named **hw1_part3.py** to Submitty for Part 3 of the homework.

Optional Challenge. If you are done early and want to challenge yourself, try the following. This part is optional and will not be graded on Submitty, nor will you get additional points for doing it. It is just one more programming exercise for you to try on your own. Instead of stars, put the word that you entered in the grid. Remember to center the word `CS1` and all the `|`, `^`, `v`, `/` and `\` characters in your grid by padding them with spaces on both sides depending on the length of the input word. You may assume that the input word is at least 3 characters long. Think about how we solved Lab 2.

Please make sure that if you solved this Optional Challenge you do not submit it on Submitty because it would produce additional lines of text causing your output to be different from the results for parts (a) through (c) which are the only parts Submitty expects.

If you solved this part, you will have the following additional output (try with words of odd or even lengths):

```
(d)
Hello Hello   ^   Hello Hello
Hello   /   Hello   \   Hello
Hello   |   Hello   |   Hello
Hello   |    CS1    |   Hello
Hello   |   Hello   |   Hello
Hello   \   Hello   /   Hello
Hello Hello   v   Hello Hello
```
