CSCI 1100 — Computer Science 1
Fall 2017
Homework 1: Calculations and Strings

Overview

This homework, worth **75 points** total toward your overall homework grade, is due Thursday, September 21, 2017 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 18, 2017** 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: Eclipse (20 pts)

We just had an eclipse over a significant portion of the United States. In this part of the homework, you will write a Python program to do a bit of eclipse related computation. Did you know that finding out when eclipses will occur lead to one of the earliest types of computers, one of which was called *Antikythera Mechanism* *https://en.wikipedia.org/wiki/Antikythera_mechanism*? Scientists are still studying how these machines worked. The real computation involves too many factors. We will do a simple exercise in this part instead. Remember, the next solar eclipse in April 8, 2024 will cover about 97% of the sun here in Albany!

Eclipses are only possible because while the sun is much larger than the moon, the sun is also much farther away from the Earth. So, we will compute when sun and moon appear the same size from our location.

Write a program called **hw1_part1.py** that asks the user for 5 values:

- the radius of the sun (in miles),
- the radius of the moon (in miles),
- the distance from the Sun to the Earth (in miles),
- the distance of the Moon to the Earth (in miles), and
- the rate at which the Moon is moving away from the Earth (**in inches**).

All values should be converted to floats for computation. Now compute the following.

1. Disregard the last two values you read for a minute, i.e. assume you do not know the actual distance of Moon to Earth. Calculate and print the distance between the Moon and Earth at which the Moon would have exactly the same apparent size as the Sun. Here is the formula for this you can use:

   ```
   Distance_Moon_to_Earth = Distance_Sun_to_Earth*(Radius_of_Moon/Radius_of_Sun)
   ```

2. Now, given that the moon moves a little farther away from the Earth every year, how many years will it take to reach the distance you computed above? Using the last two values you read, compute the number of years it will take the Moon to look the same size as the Sun. Print the number of years rounded to the nearest integer. Remember that the final parameter is in *inches/year* while all of your other distances are in miles. You will need to convert. You can use 5280 feet in a mile and 12 inches in a foot.

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

```
Enter the radius of the Sun (miles) -> 432288
432288
Enter the radius of the Moon (miles) -> 1079
1079
Enter the maximum distance to the Sun (miles) -> 94500000
94500000
Enter the minimum distance to the Moon (miles) -> 221500
221500
Enter the rate the Moon is moving away (in/year) -> 1.6
1.6
The Moon will have exactly the same apparent size as the Sun when it is 235874.00 miles
    away
The Moon will need to recede another 14374.00 miles
Which will occur in 569210426 years at the current rate.
```

We will test your code for the above values as well as 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: Naming conventions (20 pts)

Software development is a team effort, often requiring large teams. These teams generally need to adopt conventions for naming variables so that the code is readable to other members of the team! Of course teams also spend countless hours on arguing which convention is better, oh well!

Here are some basic conventions of a variable name involving two words: `first` and `value`:

Variable name	Convention Type	Potential use
`first_value`	lower case	for ordinary variables
`FIRST_VALUE`	upper case	for constants, values that should not be modified during program execution
`FirstValue`	upper camel case	for class names
`_first_value`	lower case and underscore	for system variables

Write a simple program that reads two strings from the user and outputs variable names in each of the following conventions. You can do this in various ways: by constructing the string first and printing, or by using the various options in the print statement. Also print one final version that replaces all the letters `a` and `e` in both the strings with underscore to show a *bad* variable name. Remember all the string functions we have available. Here is a simple run of this program (how it will look when you run it using Wing IDE):

```
First => circle
circle
Second => radius
radius
Example variable names
Lower case: circle_radius
For constants: CIRCLE_RADIUS
Camel case: CircleRadius
System variables: _circle_radius
Silly variable: circl__r_dius
```

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.

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 (35 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 FOR loops in your program, for the simple reason that we did not learn them yet. Furthermore, they will not make your solution better or more elegant.

Write a simple program that reads a word and two integers. 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 greater than 1*. 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.

(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 `|` in the top, bottom, left and right mid points of the grid.

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
#row => 7
7
Your word is: Hello

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

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

(c)
*** ***  |  *** ***
*** *** *** *** ***
*** *** *** *** ***
 |  *** CS1 ***  |
*** *** *** *** ***
*** *** *** *** ***
*** ***  |  *** ***
```

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 `|` characters in your grid by padding it with spaces on both sides depending on the length of the input word. Think about how we solved Lab 2.

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 Hello Hello Hello
  |   Hello  CS1  Hello   |
Hello Hello Hello Hello Hello
Hello Hello Hello Hello Hello
Hello Hello   |   Hello Hello
```