CMSC 201 Fall 2016
Homework 5 – For Loops

Assignment: Homework 5 – For Loops
Due Date: Wednesday, October 12th, 2016 by 8:59:59 PM
Value: 40 points

Collaboration: For Homework 5, collaboration is allowed. Make sure to consult the syllabus about the details of what is and is not allowed when collaborating. You may not work with any students who are not taking CMSC 201 this semester. If you work with someone, remember to note their name, email address, and how you collaborated at the top of your file.

If you did not work with anyone else on this assignment, your collaboration statement should state that
`I did not collaborate with anyone on this assignment.`

If you did work with someone else (or multiple other people), your collaboration statement should state something *similar* to the following:
`I collaborated with Fox Mulder (fmulder1@umbc.edu); I helped him understand the loop.`

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
# Collaboration:
#    COLLABORATION STATEMENT GOES HERE
```

Homework 5 is designed to help you practice using all of the structures we have learned so far including **`for`** loops, lists, iterating over lists, and branching selection structures. More importantly, you will be solving problems using algorithms you create and code yourself.

You cannot use `while` loops for this homework!

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Instructions

In this homework, you will be doing a series of exercises designed to help you practice using **`for`** loops, control statements like **`if/else`**, **`print()`** statements, and algorithmic thinking. Each one of these exercises should be in a **separate python file**. For this assignment, you may assume that all the input you get will be of the correct type (*e.g.,* if you ask the user for a whole number, they will give you an integer).

For this assignment, you'll need to follow the class coding standards, a set of rules designed to make your code clear and readable. The class coding standards are on the website, linked at the top of the "Assignments" page. You can also access them directly via this URL (http://goo.gl/yEoGfC).

NOTE: ***You must use*** **`main()`** *as seen in your* **`lab2.py`** *file, and as discussed in class.*

At the end, your Homework 5 files must run without any errors.

Details

Homework 5 is broken up into four separate parts. **Make sure to complete all 4 parts.**

NOTE: Your filenames for this homework must match the given ones exactly.

And remember, filenames are case sensitive!

Questions

Each question is worth the indicated number of points. Following the coding standards, having complete file headers, and having correctly named files is worth 4 points.

hw5_part1.py **(Worth 6 points)**

For this part of the homework you will create a "modulus" table.

Your program should prompt the user for two integers. The first integer is the number for which to make a modulus table; the second is how high you want the table to go. You can assume the user will enter integers greater than 0.

The output of the program should be a number, the modulus sign, the number you are modding it by, the equal sign, and the final answer.

Here is some sample output, with the user input in blue.
(Yours does not have to match this exactly, but it should be similar.)

```
bash-4.1$ python hw5_part1.py
Please enter the number to mod by: 4
Please enter how high you'd like to go: 13
0 % 4 = 0
1 % 4 = 1
2 % 4 = 2
3 % 4 = 3
4 % 4 = 0
5 % 4 = 1
6 % 4 = 2
7 % 4 = 3
8 % 4 = 0
9 % 4 = 1
10 % 4 = 2
11 % 4 = 3
12 % 4 = 0
13 % 4 = 1
```

hw5_part2.py **(Worth 10 points)**

For this part of the homework you will write code to draw a box.

Your program should prompt the user for these inputs, **in exactly this order:**

1. The width of their box
2. The height of their box
3. The symbol the box will be *outlined* in
4. The symbol the box will be *filled* with

For these inputs, you can assume the following:

- The height and width will be positive integers
- The symbols will be a single character each

Use the *first symbol* to draw a box of the height and width chosen by the user. The box should then be filled in with the *second symbol*. Please note that on your terminal a square box may not look exactly square.

Here is some sample output, with the user input in blue.
(Yours does not have to match this exactly, but it should be similar.)

```
bash-4.1$ python hw5_part2.py
Please enter the width of the box: 8
Please enter the height of the box: 4
Please enter a symbol for the box outline: $
Please enter a symbol for the box fill: -
$$$$$$$$
$------$
$------$
$$$$$$$$

bash-4.1$ python hw5_part2.py
Please enter the width of the box: 1
Please enter the height of the box: 1
Please enter a symbol for the box outline: X
Please enter a symbol for the box fill: O
X
```

(HINT: Use the concatenate and repetition operators we learned in class to print the right number of characters for each line.)

hw5_part3.py **(Worth 8 points)**

Write a program that prints the numbers from 101 down to 1 (inclusive), one per line. However, there are three special cases where instead of printing the number, you print a message instead:

1. If the number you would print is **divisible by 5**, print the message:
   Where do you see yourself in five years?
2. If the number you would print is **divisible by 6**, print the message:
   I'll believe six impossible things before breakfast.
3. If the number you would print is **divisible by 5 and 6**, instead print out:
   Thirty days hath September.

Print the exact strings given above! Failing to do so will lose you points.

Here is some partial sample output, showing from 101 down to 79.

```
bash-4.1$ python hw5_part3.py
101
Where do you see yourself in five years?
99
98
97
I'll believe six impossible things before breakfast.
Where do you see yourself in five years?
94
93
92
91
Thirty days hath September.
89
88
87
86
Where do you see yourself in five years?
I'll believe six impossible things before breakfast.
83
82
81
Where do you see yourself in five years?
79
```

(HINT: Look at the course notes on `range()` *for details on how to use it to count down, rather than up.)*

hw5_part4.py **(Worth 12 points)**

*(**WARNING**: This part of the homework is the most challenging, so budget plenty of time and brain power. And read the instructions carefully!)*

Finally, you are going to write code that checks whether a string contains another, smaller string. Your program should be case insensitive.

Your program will ask the user for two strings; it should ask for the longer string first. You then need to check the entire longer string for instances of the shorter string. If you find one, print out the index the match started at. Do this until you've checked the entire longer string.

(HINT: String slicing will probably be a big part of your solution. Pay careful attention to indexes, and make sure you check the entire length of the string.)

Here is some sample output, with the user input in blue.
(Yours does not have to match this exactly, but it should be similar.)

```
bash-4.1$ python hw5_part4.py
First  (longer)  string: The dog is a good dog, doggonit.
Second (shorter) string: dog
At index 4 found a slice of dog
At index 18 found a slice of dog
At index 23 found a slice of dog

bash-4.1$ python hw5_part4.py
First  (longer)  string: Pied pipers of PIE, on the pier!
Second (shorter) string: PIE
At index 0 found a slice of PIE
At index 15 found a slice of PIE
At index 27 found a slice of PIE

bash-4.1$ python hw5_part4.py
First  (longer)  string: Hello all, toll the bell, l l
Second (shorter) string: ll
At index 2 found a slice of ll
At index 7 found a slice of ll
At index 13 found a slice of ll
At index 22 found a slice of ll
```

(HINT: Working through this yourself with one of the examples in the sample output should help you better understand how to tackle the problem.)

Submitting

Once your **hw5_part1.py**, **hw5_part2.py**, **hw5_part3.py**, and **hw5_part4.py** files are complete, it is time to turn them in with the **submit** command. (You may turn in individual files as you complete them. To do so, only **submit** those files that are complete.)

You must be logged into your GL account, and you must be in the same directory as your Homework 4 python files. To double-check this, you can type **ls**.

```
linux1[3]% ls
hw5_part1.py  hw5_part2.py  hw5_part3.py  hw5_part4.py
linux1[4]%
```

To submit your Homework 4 python files, we use the **submit** command, where the class is **cs201**, and the assignment is **HW5**. Type in (all on one line) **submit cs201 HW5 hw5_part1.py hw5_part2.py hw5_part3.py hw5_part4.py** and press enter.

```
linux1[4]% submit cs201 HW5 hw5_part1.py hw5_part2.py
hw5_part3.py hw5_part4.py
Submitting hw5_part1.py...OK
Submitting hw5_part2.py...OK
Submitting hw5_part3.py...OK
Submitting hw5_part4.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**