CMSC 201 Fall 2015
Homework 5 – Nested and While Loops

Assignment: Homework 5 – Nested and While Loops
Due Date: Tuesday, October 13th, 2015 by 8:59:59 PM
Value: 4% of final grade

Homework 5 is designed to help you practice using **`while`** loops, lists, iterating over lists, branching selection structures, and printing. More importantly, you will be solving problems using algorithms you create and code yourself.

Remember to enable Python 3 before you run your programs:

```
/usr/bin/scl enable python33 bash
```

Instructions

In this homework, we will be doing a series of exercises designed to make you practice using **`while`** and **`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 Blackboard under "Course Documents" in a file titled "CMSC 201 - Python Coding Standards."
You will **lose major points** if you do not following the 201 coding standards.

A very important piece of following the coding standards is writing a complete **file header comment block**. Make sure that each file has a comment block at the top (see the coding standards document for an example).

NOTE: **You must use `main()` in each of your files.**

Details

Homework 5 is broken up into four 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.

hw5_part1.py

For this part of the homework you will write code to get valid input from the user. You will need to continue to re-prompt them until they provide a number between 0 and 100, inclusive. (In other words, 0, 1, 2, 3, … 97, 98, 99, 100 are all valid inputs.)

For this program, you can assume that the user will always enter an integer.

Here is some sample output, with the user input in blue.

```
bash-4.1$ python hw5_part1.py
Please enter a number (between 0 and 100, inclusive): -1
That is an invalid number, please try again.
Please enter a number (between 0 and 100, inclusive): 111
That is an invalid number, please try again.
Please enter a number (between 0 and 100, inclusive): 100
Thank you for selecting the number 100

bash-4.1$ python hw5_part1.py
Please enter a number (between 0 and 100, inclusive): 0
Thank you for selecting the number 0

bash-4.1$ python hw5_part1.py
Please enter a number (between 0 and 100, inclusive): 37
Thank you for selecting the number 37
```

hw5_part2.py

For this part of the homework you will write code to create a "counting" box. *(WARNING: This part of the homework is the most challenging, so budget plenty of time and brain power. And read the instructions carefully!)*

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

For these inputs, you can assume the following:

- The height and width will be positive integers

Using this width and height, you will create a box where there are **`width`** numbers on each line and **`height`** rows. Your numbers should count up starting from 1, and should continue counting up (do not restart the numbering).

HINT: You can keep the **`print()`** *function from printing on a new line by using putting* **`end=""`** *at the end:* **`print("Hello", end="")`**. *If you do want to print a new line, you can call print without an argument:* **`print()`**.

Here is some sample output, with the user input in blue.

```
bash-4.1$ python hw5_part2.py
Please enter a width:  4
Please enter a height: 2
1 2 3 4
5 6 7 8
bash-4.1$ python hw5_part2.py
Please enter a width:  3
Please enter a height: 5
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
bash-4.1$ python hw5_part2.py
Please enter a width:  9
Please enter a height: 2
1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18
```

hw5_part3.py

Next, you are going to write code that simulates the up and down movement of a hailstone in a storm.

You will ask the user for a positive integer, this will be the starting height of the hailstone. Based on the current value of the height, you will repeatedly do the following:

- If the current height is 1, quit the program
- If the current height is even, cut it in half (divide by 2)
- If the current height is odd, multiply it by 3, then add 1

(HINT: You will want to check if the number is 1 ***before*** *you check if it is odd. This will ensure you quit, rather than changing the 1 to (1*3)+1 = 4 instead.)*

You will keep updating the number, following the above rules, until the number is 1. At that point, your program should end.

For example, given a starting value of 6, here are the numbers we output:
12 -> 6 -> 3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1

Here is some sample output, with the user input in blue.

```
bash-4.1$ python hw5_part3.py
Please input the starting height of the hailstone: 12
Hail is currently at height 12
Hail is currently at height 6
Hail is currently at height 3
Hail is currently at height 10
Hail is currently at height 5
Hail is currently at height 16
Hail is currently at height 8
Hail is currently at height 4
Hail is currently at height 2
Hail is currently at height 1
```

(HINT: If you want to prevent your program from outputting decimal numbers like 6.0 and 3.0, you will need to use integer division.)

hw5_part4.py

Finally, you will write a program that takes in two lists of numbers from the user, and creates a pairwise sum. In other words, if you have 2 lists of 4 numbers **`[1, 2, 3, 4]`** and **`[5, 6, 7, 8]`**, you should create a new list that is **`[1+5, 2+6, 3+7, 4+8]`**, or **`[6, 8, 10, 12]`**.

The user will enter the numbers for first list, and type "end" when they are done. They will then enter the numbers for the second list, and type "end" when they are done.

For these inputs, you can assume the following:

- The user will only enter integers or the string "**end**"
- The length of the two lists will be the same

*(HINT: You will want to check if the user's input is "***end***" before you cast it to an integer.)*

Here is some sample output, with the user input in blue.

```
bash-4.1$ python hw5_part4.py
Please enter a number (or 'end' to stop): 20
Please enter a number (or 'end' to stop): 30
Please enter a number (or 'end' to stop): 50
Please enter a number (or 'end' to stop): 60
Please enter a number (or 'end' to stop): 90
Please enter a number (or 'end' to stop): end
Thank you for completing the first list.
Please enter a number (or 'end' to stop): 7
Please enter a number (or 'end' to stop): 4
Please enter a number (or 'end' to stop): 9
Please enter a number (or 'end' to stop): 1
Please enter a number (or 'end' to stop): 3
Please enter a number (or 'end' to stop): end
Thank you for completing the second list.
[27, 34, 59, 61, 93]
```

(HINT: Calling **`print()`** *with a list (for example:* **`print(myList)`** *) will print out the list contents including the square brackets and commas.)*

Submitting

Once all four parts of your Homework 5 are complete, it is time to turn them in with the **`submit`** command.

Don't forget to complete the header block comment for each file! Make sure that you updated the header block's file name and description for each file.

You must be logged into your GL account, and you must be in the same directory as the Homework 5 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 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 **double-check that all four homework files were submitted** by using the **`submitls`** command. Type in **`submitls cs201 HW5`** and hit enter.

And you're done!