CS-UY 1114: Lab 4

For Loops, Nested Loops

You must get checked out by your lab CA prior to leaving early. **If you leave without being checked out, you will receive 0 credits for the lab.**

Restrictions

The Python structures that you use in this lab should be restricted to those you have **learned in lecture so far (topics learned in Rephactor may only be used if they have also been taught in lecture)**. Please check with your teaching assistants in case you are unsure whether something is or is not allowed!

Create a new python file for each of the following problems.

Your files should be named *lab[num]_q[num].py* similar to homework naming conventions.

Problem 1: For and Nested Loops Galore

Solve this problem by hand. Format the output exactly how it would be when the code snippet is run.

What would be printed for each of the following code snippets?

(a)

```
for i in range(0,10):
    print(i)
```

(b)

```
for i in range(1, 13, 2):
    print(i)
```

(c)

```
for i in range(43, 31, -3):
    print(i)
```

(d)

```
for i in range(4):
    for j in range(4):
        print(i*j, end = " ")
    print()
```

(e)

```
for i in range(5):
    for j in range(i+1, 5):
        print(i, j)
```

(f)

```
for i in range(1, 10):
    if i%2 == 0:
        for j in range(i, 10, 2):
            print(j, end = " ")
        print()
```

(g)

```
for i in range(1, 10):
    if i%2 == 0:
        for j in range(0, i, 2):
            print(j, end = " ")
        print()
```

Problem 2: Higher and Higher

Write a program that will find the largest value out of X-number of user-entered **positive** values. This will be done in three steps:

1. Asking the user to enter the number of values they want to enter.
2. Asking the user to enter the values
3. Printing the largest of these values (what happens if the user entered 0 in step 1?)

The behavior of the program should look like this:

```
Please enter how many positive values you want to consider: 5
Enter your values:
24
27.5
30.234
100
1
The largest of these values is 100.0
```

Do **NOT** use lists for this problem, if you know what they are. That's no fun!

Problem 3: Line By Line

This problem will be printing a single string in multiple lines. The program will need to do the following:

1. Prompt the user to enter a string in the format “[text]-[text]-[text]...” that is to be printed on multiple lines.
2. Print the string line by line with lines being demarcated by each hyphen in the string.

You may assume that the user will input a string in the correct format. You may also assume that there will be at least one line to print. You CANNOT use string methods or lists in this problem.

Here are some examples of this program’s execution.

```
Enter a string to be printed (in the format [text]-[text]...)
Hello, my name is Greg.-Hey Greg, nice to meet you!
Printing:
Hello, my name is Greg.
Hey Greg, nice to meet you!
```

```
Enter a string to be printed (in the format [text]-[text]...)
One-Two-Three-Four-Five
Printing:
One
Two
Three
Four
Five
```

Problem 4: Arrowhead

This question will task you with creating a geometric shape using nested loops. Take in a input that determines the length of the longest line.

For example, and input of 5 should output:

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

HINT: think of using end and think about how many nested loops you need!

Problem 5: I Think Python Really Likes Me!

Python has really gotten to know you over the last few weeks and so, **Python** wants to send you a "XOX" message! It needs to know how many characters wide its message can be.

Create a program that asks for **one** input: the character-length of the message. Then the program will display an output containing **Python**'s message along with a personal signature from **Python**. X's and O's are expected to be drawn as follows:

```
X   X
 X X
  X
 X X
```

```
X   X
```

```
 OOO
O   O
O   O
O   O
 OOO
```

Note: The "O" drawing does *not* have an "O" character in the corners of the drawing. You may also notice each drawing fits within a square.

Python wants to make sure its message looks as pretty as possible. So, we're only concerned about how the output looks in cases where the message length is at least 3 characters wide and an odd number (since the X's and O's can look a little unusual otherwise).

The following are examples of possible outputs:

```
Python needs to tell you a secret. How many characters wide can
its message be? 3
X X
 X
X X
 O
O O
 O
X X
 X
X X

- From Python
```