CS 121 – Lab #4 – Loop practice

On your USB drive or computer account, please create a new directory called lab04. Put all of today's work into this folder. Today, you will write some short programs to practice with loops. Open Anaconda Spyder as usual.

Program #1: `summation.py`

A common application of loops is to find a sum of several numbers. This program will find the sum of perfect squares. The program will ask the user to enter a positive integer, which we might call n. Then, the program will calculate the sum of $1^2 + 2^2 + 3^2 + 4^2 + \ldots + n^2$.

1. Since we know that the loop will have exactly n iterations, which type of loop makes more sense to use: a while loop, or a for loop?

2. When you run your program, the I/O should look something like this:

```
Please enter a positive integer:  10
The sum of the first 10 perfect squares is 385.
```

3. Once you have verified that your program calculates the total correctly, let's add error checking. Notice that we prompt the user for a positive integer. It doesn't make sense to run this program with zero or negative input. So, modify the beginning of your program to do the error checking. As explained in the Loop handout in class, you will need a boolean variable needInput. And you should enclose the input statement inside a while loop, so that if the user makes a mistake, the program can continue asking for proper input.

Program #2: `summation2.py`

Create a new source file called summation2.py. Copy all of the code from summation.py into summation2.py. This time, we will compute a different summation. Rather than finding the sum of i^2, this program should find the sum of $3i^2 – 2i + 1$. To make this change, it is only necessary to change the calculation of the total inside your loop. You should also modify the comment(s) accordingly, since they had been copied from a different program. If you run with 10 as input, the output should be 1055.

Program #3: `box.py`

Now, let's practice with nested for-loops to create rectangular designs on the screen. Create a new source file called box.py. This program should first ask the user to enter two integers. The first number will represent the number of (horizontal) rows of the rectangle. The second number will be the number of (vertical) columns. Both numbers need to be at least 3 for the program to show meaningful output, but you do not need to perform error checking in the code.

After the program has obtained the number of rows and columns from the user, it can draw rectangles. The first rectangle will be a *solid* rectangle of stars (asterisks). In pseudocode, the algorithm could go like this:

```
for i = 1 to the number of rows
    for j = 1 to the number of columns
        print a star
    print a newline character to finish the line
```

The second rectangle will be a *hollow* rectangle. Here is its pseudocode:

```
for i = 1 to the number of rows
    for j = 1 to the number of columns
        if i = the first or last row, or j is the first or last column
            print a star
        else
            print a space
    print a newline character to finish the line
```

Write Python code that implements the above pseudocode loops. Be sure to label each rectangle in your output and print a blank line between them, so that we know where one rectangle ends and the next one begins. Here is what an example run should look like:

```
How many rows?  5
How many columns?  8
Solid rectangle:
********
********
********
********
********

Hollow rectangle:
********
*      *
*      *
*      *
********
```

Program #4: `diagonal.py`

Let's draw something more interesting than a rectangle. If we can draw diagonals, then we can print a giant letter "X".

Create a new source file called diagonal.py. This program will ask the user to enter a positive number n. This will be the size of the X. Ideally, this number of should be an *odd* number at least 3.

Once the user provides the program with the value of n, we are ready to draw the big letter X. Consider the following pseudocode algorithm:

```
for i = 1 to n:
    for j = 1 to n:
        if i equals j,
            print a star
        else
            print a space
    print a newline character to end the line
```

If you implement the above algorithm, you should see one of the diagonals of the X. You need to add some code to the if-statement in order to accommodate the other diagonal. To visualize what your program should do, here is a diagram that may help. It illustrates the X for n = 5. Look at the / diagonal. What mathematical property do all these squares have?

	j = 1	j = 2	j = 3	j = 4	j = 5
i = 1	Print star				Print star
i = 2		Print star		Print star	
i = 3			Print star		
i = 4		Print star		Print star	
i = 5	Print star				Print star

Program #5: `triangle.py`

This program will be similar in structure to the previous program. Create a new source file called triangle.py. The purpose of this program is to display a right triangle on the screen.

The program triangle.py should first ask the user to enter a positive integer n, which will indicate the size of the triangle. Any positive integer will do. For example, if n is 7, the program should print:

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

The structure of the nested for-loop will be the same as for the diagonal program. In other words, the pseudocode would have this structure:

```
for i = 1 to n
    for j = 1 to n
        if _______________
            print a star
        else
            print a space
    print a newline to finish the line
```

What condition should go in the blank above? To motivate your solution, consider the example of n = 7. The following grid may be of assistance. What mathematical property is shared by all the squares where we must print a star?

	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6	j = 7
i = 1	Print star						
i = 2	Print star	Print star					
i = 3	Print star	Print star	Print star				
i = 4	Print star	Print star	Print star	Print star			
i = 5	Print star	Print star	Print star	Print star	Print star		
i = 6	Print star	Print star	Print star	Print star	Print star	Print star	
i = 7	Print star	Print star	Print star	Print star	Print star	Print star	Print star

Please have the instructor or lab aide check your work. You're done!