Lab 4

Objectives

- Practice with count driven loops

Exercises

Download `lab4.py` and save it to your Lab4 folder in your H: drive and complete the function designs according the following descriptions:

1. Design a function called `print_temps` that will print a table of temperatures. The table should have a header row, followed by rows with temperature in Celsius and a temperature in Fahrenheit.

   The function should take 3 arguments: the number of rows to print, the starting temperature in Celsius, and the amount the temperature changes by from one row to the next.

   Good software development practice is to write a helper function for each separate piece of computation. In your solution to this problem, you should write and use the helper function `celcius_to_farenheit`. This function should take a floating point number that represents a temperature in Celcius and coverts it to Farenheit and returns the converted value.
   Reminder: farenheit = celcius * 9 / 5
   Your `print_temps` function must call `celcius_to_farenheit`

   An example, `print_temps(4, 33, 10)` should print something like this:

```
Celsius   Farhenheit
33         91.4
43         109.4
53         127.4
63         145.4
```

Trouble getting started? Start by getting the correct number of rows to print.

Within that loop call the `celcius_to_farenheit` function with the same starting temperature every time. So for `print_temps(4, 33, 10)` it would print:

```
Celsius   Farhenheit
33         91.4
33         91.4
33         91.4
33         91.4
```

Now, think about how to change the call to `celcius_to_farenheit` in the loop. Remember, you can have multiple instructions in the loop and you can update the value of Celsius each iteration!

CHECK POINT – get your lab TA to check off that you have completed this part. They will want to see your function run and see that you used a helper function, you have proper documentation and that you have tests.
REMEMBER: testing a function that does not return a value can be done by simply calling that function and then inspecting the output of that call to ensure it is what you expect.

2. Write a function called `print_line` that takes a string s and an integer n as parameters and prints s on one line n times with one new line at the end. You **must** use a loop.
   For example, `print_line('ab', 4)` should print:
   ```
   Abababab
   ```

3. Write a function called `print_rectangle` that takes a string s, an integer width and an integer height as parameters and prints height lines with s printed width times on each line.
   For example, `print_rectangle('*=', 4, 3)` should print:
   ```
   *=*=*=*=
   *=*=*=*=
   *=*=*=*=
   ```

   Did you use your `print_line` function as a helper function in your implementation?

CHECK POINT – get your lab TA to check off that you have completed this part. They will want to see the output of the tests for your functions. They will also want to see your documentation, your tests and your use of a helper function.

4. Write a function called `print_shape` that takes an integer representing the size of the shape and prints a shape corresponding to the following examples.

   `print_shape(1)` should print:
   ```
   \*\
   OOO
   /*/
   ```

   `print_shape(2)` should print:
   ```
   \***\
   \\*\\
   OOOOO
   //*//
   /***/
   ```

   `print_shape(4)` should print:
   ```
   \*******\
   \\*****\\
   \\\***\\\
   \\\\*\\\\
   OOOOOOOOO
   ////*////
   //***///
   //*****//
   /*******/
   ```

CHECK POINT – get your lab TA to check off that you have completed this part. They will want to see the output of the tests for your functions. They will also want to see your documentation, your tests

and your use of a helper function.
Finished Early? Get started on Assignment 4 – you can find it in conneX!