Assignment 2

Course: CS65 - Introduction to Computer Science (Spring 2022)
Instructor: Md Alimoor Reza, Assistant Professor of Computer Science, Drake University
Due: Tuesday March 22, 11:50 PM

Introduction

Loops (for and while) allow us to solve a repetitive task. This assignment will allow you to explore this specific feature. Your programs should be written with multiple user-defined functions in the same file. This helps break down the task into smaller and simpler pieces. Functions also make it easier to test individual components, as parts of the program can be tested separately.

Logistics

In this assignment, You will be creating a python file for each task separately (with a **.py* extension), where you will save your python instructions. By default, Thonny opens an unnamed file in the a *text editor* (top text pane). It is an excellent practice to write a formal header and suppress it as comments. A line of code with a preceding # is considered as a comment in Python. You should type in the necessary parts as shown below, e.g., *your name, your contact email, description, etc.*. Save the file using the format as follows *firstname_lastname_a2_task*.py* (all in lowercase letters). For example, I saved my file as *md_reza_a2_task1.py*. Those who prefer to work alone, please email me. Otherwise, work with your existing group. You should only work together but can submit a separate copy of the assignment.

Task 1 (20 points): while loops

There are several small sub-tasks, many of which were solved individually/collectively during class. You should solve each subtask using a while loop.

Subtask 1: summation using while loop

Prompt the user to enter one integer number. Then your program should find the summation of all the numbers from 1 up to that integer number. For example, if the user enters **3**. Your program should print **6** as the summation of 1+2+3 is equal to 6.

Subtask 2: summation of odd numbers using while loop

Prompt the user to enter one integer number. Then your program should find the summation of all the **odd numbers** from 1 up to that integer number. For example, if the user enters **5**. Your program should print **6** as the summation of 1+3+5 is equal to 9.

Subtask 3: finding regional state using while loop

Prompt the user to enter the name of a state from the following options:

- "DE", "NC", "NJ", "VA"
- "IA", "IN", "KS", "WI"
- "TX", "LA", "AL", "AK"
- "CA", "OR", "WA", "NV"

Your program should print the state's geographic location from one of the categories: "Eastern", "Midwestern", "Southern", or "Western". Your program will terminate only when the user enters "END".

Task 2 (30 points): for loops

This task comprises of solving several small sub-tasks. Many of the given tasks were solved individually/collectively during class time. You should solve each task by using either **value for loop** or **index for loop** (as appropriate).

Subtask 1: printing a special character N times using for loop

Prompt the user to enter one of the special characters from the following options: @, #, $, *. Also, prompt the user to enter another integer number **N**. Your program should print the special character **N** times. For example, if the user enters a special character choice of @ and enters a value of 5 for **N**, then your program should print @@@@@.

Subtask 2: summation of multiple of 5s using for loop

Prompt the user to enter one integer number. Then your program should compute the summation of all the multiple of **5**s from 1 up to the given integer number. For example, if the user enters **15**. Your program should print **30** as the summation of 5+10+15 is equal to **30**.

Subtask 3: finding a number in a list using for loop

Initialize a list variable *my_list*=[2, 4, 6, 8, 10, 11, 13, 15, 17, 19, 22, 24, 26, 26, 28]. Now, prompt the user to enter one integer number. If the number entered by the user exists in the list above, your program should print FOUND; otherwise, it should print NOT FOUND.

Subtask 4: counting the frequency of a number using for loop

Initialize a list variable *new_list*=[1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, , 9, 9, 9, 9, 9, 9, 9, 9, 9]. Now, prompt the user to enter one integer number. If the number entered by the user exists in the list above, your program should count how many times that number appears in the given list. After finding the count, your program should print the count. For example, if the user enters **1**, your program should print the following message 1 appears 3 times in the list. Alternatively, if the user enters 15, your program should print 15 appears 0 times in the list.

Subtask 5: finding location of a number in a list using for loop

Initialize a list variable *my_list*=[2, 4, 6, 8, 10, 11, 13, 15, 17, 19, 22, 24, 26, 28]. Now, prompt the user to enter one integer number. If the number entered by the user exists in the list above, your program should print its location. Location starts at 0 and ends at 13 for the above list (its length is 14). For example, if the user enters **2**, your program should print the following message 2 appears at location 0. Alternatively, if the user enters 15, your program should print 15 appears at location 7.

Subtask 6: finding the maximum and minimum numbers in a list using for loop

Initialize a list variable *my_list*=[10, 3, 15, -7, 90, 11]. Your program should print the following messages Maximum number is 90 and Minimum number is -7 in two separate lines. Hint: you should use two separate for loops to separately find the maximum and minimum numbers. Although it is possible to find it using one for loop, for simplicity, I encourage you to do it using two separate for loops. **You should not use**

Python's max() or min() functions. You will not receive any credit if you use those two built-in functions to solve this problem.

Task 3 (40 points): nested for loops

You should use nested for loop in this task. Your program should build and display three different shapes with increasing complexities. It is recommended that you solve subtask 1 before you attempt the other two subtasks.

Subtask 1: building a rectangle

You should prompt the user to enter the lengths of the two sides of a rectangle. First, prompt the user to enter the **height** of the rectangle. Then, prompt the user to enter the **width** of the rectangle. Based on the provided **height** and **width**, you should display a rectangular shape which is filled in with the character * as follows:

```
enter the height of the rectangle: 5
enter width of the rectangle:  10


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

As shown above, the rectangle is represented by 5 rows and 10 columns in a grid-like structure, where each grid point is represented by a *.

Subtask 2: building a right-facing right-triangle

You should prompt the user to enter the length of a right triangle's **adjacent** and **opposite**. You can also assume that these two sides are of equal length. Based on the provided length of its **adjacent** and **opposite**, you should display a right-facing triangular shape which is filled in with the character * as shown below:

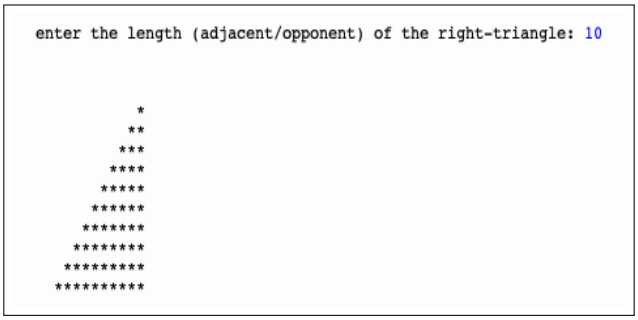
```
enter the length (adjacent/opponent) of the triangle: 10


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

As shown above, the triangle is represented by 10 rows and 10 columns in a grid-like structure, where each grid point is represented by a *. It is a **right triangle** (90° angle at the bottom-left corner of the triangle).

Subtask 3: building a left-facing right-triangle

You should prompt the user to enter the length of a right triangle's **adjacent** and **opposite**. You can also assume that these two sides are of equal length. Based on the provided length of its **adjacent** and **opposite**, you should display a left-facing triangular shape which is filled in with the character * as shown below:



As shown above, the triangle is represented by 10 rows and 10 columns in a grid-like structure, where each grid point is represented by a *****. It is a **right triangle** (90° angle at the bottom-right corner of the triangle).

What to turn in

You should submit your source code (three python files). You **do not need** to submit screenshots of your outputs.

Grading Rubric:

Submitted correctly	05
Code is well commented	05
loop usage	45
Code produces correct outputs	45
Total	100