Assignment 1

Course: CS65 - Introduction to Computer Science (Fall 2025)
Instructor: Md Alimoor Reza, Assistant Professor of Computer Science, Drake University
Due: Friday, October 31, 2025, 11:55 PM

Introduction

Loops (for and while) allow us to solve a repetitive task. This assignment will allow you to explore this specific feature. 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. 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_a1_task*.py* (all in lowercase letters). For example, I saved my three files as *md_reza_a1_task1.py*, *md_reza_a1_task2.py*, and *md_reza_a1_task3.py*.

Task 1 (15 points): while loops

You should solve each subtask using a **while loop**.

Subtask 1 (5 points): 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 (5 points): 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 (5 points): 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. You should solve each task by using either **value for loop** or **index for loop** (as appropriate).

Subtask 1 (5 points): 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 (5 points): 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 (5 points): 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 (5 points): 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, 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 (5 points): 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 (5 points): 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 (30 points): nested for loops

You should use nested for loop in this task. Your program should build and display different shapes with increasing complexities.

Subtask 1 (5 points): making a square with a diagonal pattern

You should prompt the user to enter the length of a square shape. Either the **height** or the **width** of the square should be prompted as they are both equal for a square shape. Based on the provided **side length**, you should display square shape which should be filled in with the characters **X** and **.** as follows:

```
enter the side length: 5
X....
.X...
..X..
...X.
....X
```

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

Subtask 2 (5 points): making a square with a cross shape

Modify the previous solution so that it displays a square shape filled with the characters **X** and **.** as shown below:

```
enter the side length: 5

X...X
.X.X.
..X..
.X.X.
X...X
```

Subtask 3 (5 points): making a square with an upside-down 'L' shape

Modify the previous solution so that it displays a square shape filled with the characters **X** and **.** as shown below:

```
enter the side length: 6

XXXXXX
.....X
.....X
.....X
.....X
.....X
```

Subtask 4 (5 points): making 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 **.** (dot symbol) as shown below:

```
enter the length (adjacent/opponent) of the right-triangle: 7

.
..
...
....
.....
......
.......
```

As shown above, the triangle is represented by 7 rows and 7 columns in a grid-like structure, where each grid point is represented by a **.** (dot symbol). It is a **right triangle** (90° angle at the bottom-left corner of the triangle).

Subtask 5 (5 points): making a left-facing right-triangle

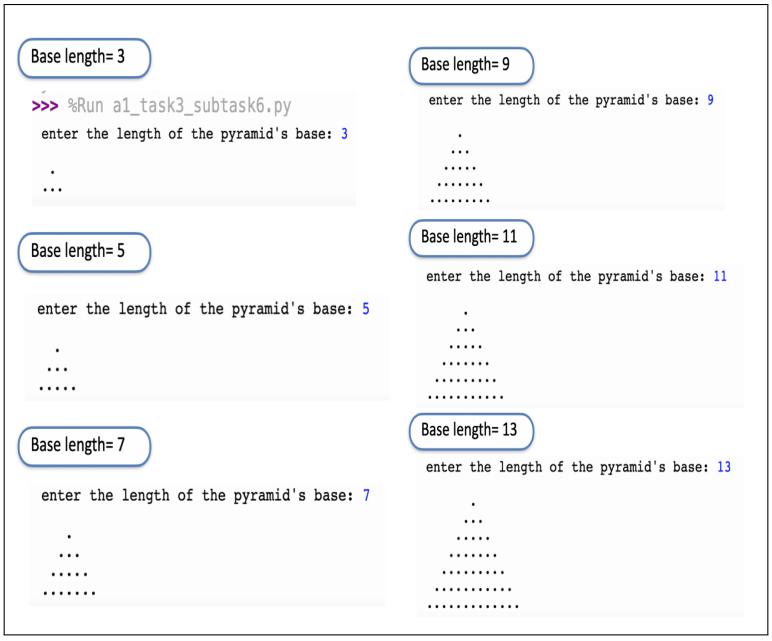
Modify your previous solution so that based on the provided length of its **adjacent** and **opposite**, you should display a left-facing triangular shape which is filled in with the character **.** (dot symbol) as shown below:

```
enter the length (adjacent/opponent) of the right-triangle: 7

      .
     ..
    ...
   ....
  .....
 ......
.......
```

Subtask 6 (5 points): making a pyramid shape

You should prompt the user to enter the length of the base of a pyramid. Based on the entered base length, display a pyramid shape filled with the character **.** as shown below:



What to turn in

You should submit your source code (three python files).