Lab 04: Loops

Goals for this lab:

1. Use `if`, `elif`, `else` statements to allow a Python program to take different actions depending on input
2. Use `while` loops to allow the program to repeatedly execute the same instructions
3. Use `for` loops to allow the program to repeatedly execute the same instructions
4. Use sentinel values to detect when the user has completed entering data
5. Use the random module and function to generate random numbers in a range
6. Convert text data input into numeric data using appropriate functions

General Guidelines:

- Use meaningful variable names
- Use appropriate indentation
- Use comments, especially for the header, variables, and blocks of code.

Example Header:

```
# Author: Your name
# Date: Today's date
# Description: A small description in your own words that describes what
the program does. Any additional information required for the program to
execute should also be provided.
```

1. Preparation

You will be using one premade Python file for this lab, so be sure to download **uncertainty.zip** from the Canvas Lab04 link before you begin the lab, as it contains the **uncertainty.py** file you need.

2. Uncertainty Redux

In the previous lab, you created a particle guessing game that allowed the user only a single opportunity to guess the right position. However, that may have been a bit too challenging. For this part of the lab, you will need to provide the user three chances to guess the position, and some hints on if their guesses are above or below the actual position. The file **uncertainty.py** has been provided as a starting point, so be sure to download and use that file for this portion of the lab, though you should rename it as **uncertaintyRedux.py** .

Your program should now perform the following steps:

- Initialize an appropriately named variable to 3, representing the user's guesses
- Use a `while` loop to control the game and continue looping as long as the user has guesses remaining
    - You may need to indent portions of the provided code to make sure the appropriate lines are in the loop
- Inside the loop:
    - Update the first provided `input()` message to also tell the user how many guesses they currently have remaining
    - Update the `if/elif/else` block such that if the user guesses the correct number, the user is informed they correctly guessed the position and the loop is immediately exited
    - Update the `if/elif/else` block such that if the user guesses the wrong position within the range
        - If the guessed x position is greater than the correct x position, the user is informed of this with an appropriate message
        - Else if the guessed x position is less than the correct x position, the user is informed of this with an appropriate message
        - If the guessed y position is less than the correct y position, the user is informed of this with an appropriate message
        - Else if the guessed y position is less than the correct y position, the user is informed of this with an appropriate message
    - Decremented the number of chances they have by 1
    - Determine if the user still has any chances remaining and if they do not, inform them they have run out of chances and the value of the correct number, with an appropriate message
- Update the description of the program in the header comments now that you've made updates to the program's functionality.

Make sure you save your **uncertaintyRedux.py** as you will be submitting this file for Lab 04.

3. Going the Distance

It is well known that a trebuchet can launch a 90-kilogram projectile over a distance of 300 meters. However, there is always a bit of variance to the final distance based on wind speed, exact projectile weight, and other factors. For this portion of the lab, your program, **trebuchet.py**, will be recording the distances the projectile

flies for a number of trials, and keeping track of the three trials with the furthest distance. Note that you do not know how many trials will be conducted, so your program will need to keep prompting for additional trials until the user is done.

Your program should perform the following steps:

- Initialize three appropriately named variables to 0 to store the top three distances
- Initialize three appropriately named variables to 0 to store the trial numbers of the top three trials
- Initialize an appropriately named variable to `"Y"` to store whether the user wishes to input additional trials
- Initialize an appropriately named variable to 1, to keep track of the current trial number
- Create a `while` loop that tests if the user wishes to input in additional trials, assuming that if the user inputs a `"Y"` they wish to enter more trials
- Inside the loop:
    - Prompt the user for an integer distance and store it in a new variable (this will store data temporarily)
    - Using `if/elif/else` statements determine:
        - If the new distance is greater than the farther distance, move the current farthest distance and trial number to the second farthest place, the second farthest distance and trial number to third the third farthest place, and the new distance and trial number to the first farthest place
        - If the new distance is greater than the second farther distance but less or equal to the current farthest distance, move the current second farthest distance and trial number to the third farthest place, and the new distance and trial number to the second farthest place
        - If the new distance is greater than the third farthest distance but less than or equal to the second farthest distance, move the new distance and initials to third farthest place
        - Otherwise, make no changes
    - Prompt the user, with an appropriate message, if they wish to enter another distance and trial number
        - If they type “Y” execute the loop again
        - If they type anything other than “Y”, exit the loop
        - Be sure the user knows what their options are in the appropriate message!
    - Increment the trial number
- After the loop, with an appropriate message, tell the user you are going to list the top three farthest distances and their trial numbers
    - Each line should be in the format `Trial Distance` (e.g. `5 325`)
    - The very first line should have the column headers: `Trial Distance`
    - The next three lines should have the top three trials in descending order of distance
    - The two columns should be neat and aligned, so be sure to set an appropriate width for each column to achieve this.

Make sure you save your **trebuchet.py** as you will be submitting this file for Lab 04.

Odd Sums

Just like the `while` loop, a `for` loop allows you to repeat your codes for the desired number of times. In this exercise, we will write a program, **oddSums.py**, to find the sum of all odd numbers between any two randomly generated numbers, the boundaries are included if they are odd. For example, if the randomly generated integers are 4 and 13, you need to sum from 5 to 13. Another example: if the randomly generated integers are 7 and 17, you need to sum from 7 to 17.

Your program should perform the following steps:

- Generate the first random integer between 1 and 10, inclusively, and store it in an appropriate variable
- Generate the second random integer between 11 and 20, inclusively, and store it in an appropriate variable
- Initialize a variable to store the sum
- Inside a `for` loop:
    - Compute the sum of all odd integers between the two random integers. The random integers should be included in sum if they are odd
- Display the random integers as well as the sum on the screen with suitable messages

Make sure you save your **oddSums.py** as you will be submitting this file for Lab 04.

Submission

Once you have completed this lab it is time to turn in your work. Please submit the following files to the Lab 04 assignment in Canvas:

- **uncertaintyRedux.py**
- **trebuchet.py**
- **oddSums.py**

Sample Output

uncertaintyRedux.py

```
The particle
The particle is somewhere in this space! You have 3 chances to guess it.
What do you think its x coordinate is (1-10)? 3
What do you think its y coordinate is (1-10)? 5
Bad luck! The particle's x position is greater than your x position!
Bad luck! The particle's y position is greater than your y position!
The particle is somewhere in this space! You have 2 chances to guess it.
What do you think its x coordinate is (1-10)? 5
What do you think its y coordinate is (1-10)? 7
Bad luck! The particle's x position is less than your x position!
Bad luck! The particle's y position is less than your y position!
The particle is somewhere in this space! You have 1 chances to guess it.
What do you think its x coordinate is (1-10)? 4
What do you think its y coordinate is (1-10)? 8
Bad luck! The particle's y position is less than your y position!
Oh no! You ran out of chances. (4,6) was the particle's position!
```

```
The particle is somewhere in this space! You have 3 chances to guess it.
What do you think its x coordinate is (1-10)? 4
What do you think its y coordinate is (1-10)? 6
Good guess! (4,6) was the position!
```

trebuchet.py

```
Please enter your distance for trial 1: 298
Would you like to input another trial? (Y/N):Y
Please enter your distance for trial 2: 305
Would you like to input another trial? (Y/N):Y
Please enter your distance for trial 3: 301
Would you like to input another trial? (Y/N):Y
Please enter your distance for trial 4: 312
Would you like to input another trial? (Y/N):Y
Please enter your distance for trial 5: 307
Would you like to input another trial? (Y/N):N
The top three distances for the trebuchet are:
Trial Distance
    4      312
    5      307
    2      305
```

oddSums.py

```
The first random number was 2, the second random number was 16, and the
sum is 63.
```

Rubric

For each program in this lab, you are expected to make a good faith effort on all requested functionality. Each submitted program will receive a score of either 100, 75, 50, or 0 out of 100 based upon how much of the program you attempted, regardless of whether the final output is correct (See table below). The scores for all of your submitted programs for this lab will be averaged together to calculate your grade for this lab. Keep in mind that labs are practice both for the exams in this course and for coding professionally, so make sure you take the assignments seriously.

Score	Attempted Functionality
100	100% of the functionality was attempted
75	<100% and >=75% of the functionality was attempted
50	<75% and >=50% of the functionality was attempted
0	<50% of the functionality was attempted