Lab 03: Conditionals, Loops, & Randoms

Goals for this lab:

1. Use `if`, `elif`, `else` statements to allow the program to take different actions depending on input
2. Use Unicode values to output particular symbols
3. Use the random module and function to generate random numbers in a range
4. Convert text data input into numeric data using appropriate functions
5. Use `while` loops to allow the program to repeatedly execute the same instructions
6. Use `for` loops to allow the program to repeatedly execute the same instructions
7. Write and execute simple Python programs using the given instructions

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. Numbers to Symbols

As was discussed, your computer uses numeric values to represent different symbols that humans have given meaning to, such as the letters of the English alphabet or Hindu-Arabic numerals. Python supports Unicode encoding, and so long as you know the Unicode numeric value associated with a particular symbol, you can output any of the 65000+ symbols.

For this program, you will be creating a program named **romanNumerals.py** which converts a user specified number in the range of 1-9, inclusively, into the equivalent Roman numeral according to the following table:

Hindu-Arabic	Roman Value	Unicode
1	Ⅰ	2160
2	Ⅱ	2161
3	Ⅲ	2162
4	Ⅳ	2163
5	Ⅴ	2164
6	Ⅵ	2165
7	Ⅶ	2166
8	Ⅷ	2167
9	Ⅸ	2168

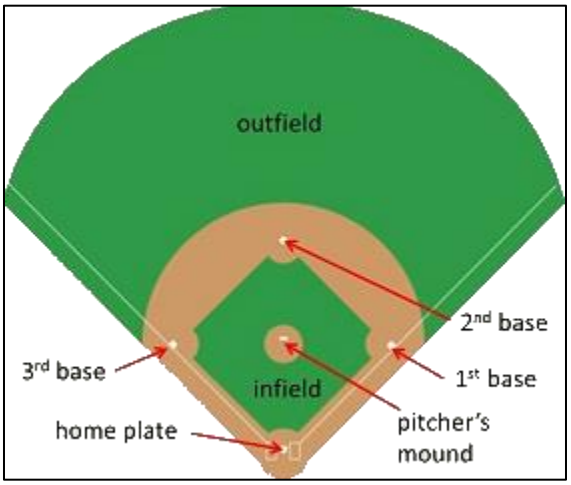
Your program should perform the following steps:

- Using an appropriate message, prompt the user for a value in the range of 1-9, inclusively, and store it in an appropriately named variable
    - Do not forget to convert the text into an integer value!
- Using appropriately connected `if`, `elif`, `else` statements, determine which roman numeral the program should output, based on the user's input, and output that symbol along with an appropriate message
    - In order to output the symbol you will need to add a \u in front of the Unicode value e.g. printing "\u2164" would output Ⅴ
    - Additionally, if the user's number is outside of the 1-9 range, output an appropriate message informing them of the mistake

Make sure you save your **romanNumerals.py** as you will be submitting this file for Lab 03.

2. Batter Up!

In baseball, the playing field into three main parts, the infield, which is roughly 135 feet from the home plate, the outfield which extends from the edge of the infield up to the fences roughly 400 feet from the home plate, and the area beyond the fences. Ideally, when the ball is pitched to the batsman, the batsman will hit the ball as far as possible to provide them more time to run around the bases.



For this lab, we consider the case of a batsman hitting the ball a random distance, and we will assume that the ball always travels in straight line from home plate across second base. To help simulate the outcome of the batter's hit, write a program named **batterUp.py** which should:

- Import the random module at the beginning of the program
- Generate a random integer in the range of 0 to 450, inclusively, to represent the distance, in feet, the ball traveled after being hit and store it in an appropriately named variable
- Using appropriately connected `if, elif, else` statements, determine the outcome of the action in the following way:
    - If the ball's distance was greater than 400, the batter hit a homerun. An appropriate message should output the distance the batter hit the ball and that the batter hit a home run and scored a run for the team
    - Else if the ball's distance was between 400 and 135, inclusively, the batter hit the ball into the outfield. An appropriate message should output the distance the batter hit the ball and that the batter hit the ball into the outfield and made it to third base.
    - Else if the ball's distance was between 134 and 10, inclusively, the batter hit the ball into the infield. An appropriate message should output the distance the batter hit the ball and that the batter hit the ball into the infield and made it to second base.
    - Else if the ball's distance was between 9 and 1, inclusively, the batter bunted the ball. An appropriate message should output the distance the batter hit the ball and that the batter bunted and made it to first base.
    - Else if the ball's distance was 0, the batter made a strike. An appropriate message should output that the batter made a strike.

Make sure you save your **batterUp.py** as you will be submitting this file for Lab 03.

3. Uncertainty

The Heisenberg uncertainty Principle states that we cannot know both the position and momentum of a particle with perfect accuracy due to the methods required to obtain either piece of information. For the first part of this lab, you will construct a program named **uncertainty.py** which will play a position guessing game with the user. The program should:

- Declare an appropriately named variable representing the user's guesses and assign it the value 3
- Declare two appropriate variables to store the x and y coordinates of the particle, and assign each a number of your choosing between 1 and 10, inclusively
- 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 `while` loop:
    - Using an appropriate message, inform the user of how many guesses they currently have remaining, and then prompt the user for two numbers in the range of 1 to 10, inclusively, representing their guess as to the particle's x and y coordinates and store their responses in appropriately named variables
        - Do not forget to convert the text into an integer value!
    - Using appropriately connected `if`, `elif`, `else` statements, test the user's coordinates and:
        - If the user's coordinates match the particles coordinates, inform the user they guessed correctly and that they win, and immediately exit the loop
        - Else if either of the user's coordinates are outside of the range of the game, inform the user that their guess was outside of the coordinate bounds
        - Otherwise, try to give the user a hint
            - If the guessed x position is greater than the correct x position, inform the user of this with an appropriate message
            - Else if the guessed x position is less than the correct x position, inform the user of this with an appropriate message
            - If the guessed y position is less than the correct y position, inform the user of this with an appropriate message
            - Else if the guessed y position is less than the correct y position, inform the user of this with an appropriate message
    - Decrement the number of chances they have by 1
    - Test if the user still has any chances remaining and if they do not, inform them they have run out of chances and lost the game, and provide them with the correct position of the particle, with an appropriate message

Make sure you save your **uncertainty.py** as you will be submitting this file for Lab 03.

4. 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 03.

Submission

Once you have completed this lab it is time to turn in your work. Please submit the following files to the Lab 03 assignment in Canvas:

- **romanNumerals.py**
- **batterUp.py**
- **uncertainty.py**
- **oddSums.py**

Sample Output

romanNumerals.py

```
Please enter a number between 1 and 9 to convert to a roman numeral: 4
Your roman numeral is: IV

Please enter a number between 1 and 9 to convert to a roman numeral: 10
10 is outside the allowed range of 1-9.
```

batterUp.py

```
The ball flew 423 feet and the batter scored a home run! That's one point
for our team!

The ball flew 215 feet and the batter made it to third base!

The ball flew 88 feet and the batter made it to second base!

The ball flew 6 feet because the batter bunted, and made it to first base!

The batter has made a strike! Oh no!
```

uncertainty.py

```
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)? 15
What do you think its y coordinate is (1-10)? 2
No good! (15,2) is outside of the range!
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!
```

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