IS111 – Lab 6
while-Loops

Instructions

You should attempt the questions using Visual Studio Code. Create your own *.py files to complete the lab exercises. No starting code is provided.

Q1: [★] Gender

Prompt the user for his/her gender. The user has to enter one of the following options: 'M' or 'F'. The program keeps prompting the user until the user enters a valid option.

A sample run of the code can be found below:

```
What's your gender? Please enter M or F: T
Wrong input!

What's your gender? Please enter M or F: f
Wrong input!

What's your gender? Please enter M or F: F

Thanks! Your gender is female.
```

Q2: [★] Two Numbers

Write a program that prompts the user for two numbers. The program keeps prompting the user until **both numbers are positive** and **the sum of the two numbers is strictly smaller than 100**.

A sample run of the program can be found below:

```
Enter the first number: 90.5
Enter the second number: 0
Conditions not satisfied!

Enter the first number: 90.5
Enter the second number: 10.5
Conditions not satisfied!

Enter the first number: -5
Enter the second number: 50
Conditions not satisfied!

Enter the first number: 5.5
Enter the second number: 6.5

Thanks!
```

Q3: [★★] Number Guessing Game

Write a piece of code that does the following:

- The program first generates a random number between 1 and 100 (both inclusive). You can use the `randrange()` or `randint()` function from the `random` module to generate the number.
- The program then keeps prompting the user to guess the random number generated until the user's guess is correct. When the guess is wrong, the program gives a hint about whether the guess is too low or too high.

A sample run of the program can be found below. Note that in this sample run 95 is the random number generated in the beginning.

```
Enter your guess (between 1 and 100) :34
Your guess is too low!

Enter your guess (between 1 and 100) :58
Your guess is too low!

Enter your guess (between 1 and 100) :73
Your guess is too low!

Enter your guess (between 1 and 100) :86
Your guess is too low!

Enter your guess (between 1 and 100) :99
Your guess is too high!

Enter your guess (between 1 and 100) :90
Your guess is too low!

Enter your guess (between 1 and 100) :95

Bingo!
```

Another sample run is below. This time 47 is the random number generated in the beginning.

```
Enter your guess (between 1 and 100):50
Your guess is too high!

Enter your guess (between 1 and 100):25
Your guess is too low!

Enter your guess (between 1 and 100):37
Your guess is too low!
```

```
    Enter your guess (between 1 and 100):43
    Your guess is too low!

    Enter your guess (between 1 and 100):47

    Bingo!
```

Note: You cannot reproduce the same output as shown above because your program will likely generate a different random number.

Q4: [★★] Shopping Cart

Part I

Write a program that prompts the user for the items in his/her shopping cart. The program keeps prompting the user until there's no more item in the shopping cart. The program displays the total price of all the items in the shopping cart.

A sample run of the code can be found below:

```
Do you have any item left in your shopping cart? Please enter Y or N: Y
Please enter the name of the item : bread
Please enter the price of the item : $2.2
Please enter the quantity of the item : 2

Do you have any item left in your shopping cart? Please enter Y or N: Y
Please enter the name of the item : apples
Please enter the price of the item : $0.5
Please enter the quantity of the item : 6

Do you have any item left in your shopping cart? Please enter Y or N: Y
Please enter the name of the item : milk
Please enter the price of the item : $5.3
Please enter the quantity of the item : 1

Do you have any item left in your shopping cart? Please enter Y or N: N

Total price: $12.7
```

Part II

Now modify your code such that in the end the program prints out not only the total price but also all the items in the shopping cart.

A sample run of the program can be found below:

```
Do you have any item left in your shopping cart? Please enter Y or N: Y
Please enter the name of the item: bread
Please enter the price of the item: $2.2
Please enter the quantity of the item: 2

Do you have any item left in your shopping cart? Please enter Y or N: Y
Please enter the name of the item: apples
Please enter the price of the item: $0.5
Please enter the quantity of the item: 6

Do you have any item left in your shopping cart? Please enter Y or N: Y
Please enter the name of the item: milk
Please enter the price of the item: $5.3
Please enter the quantity of the item: 1

Do you have any item left in your shopping cart? Please enter Y or N: N

You've entered the following items:
    bread   $2.2    2
    apples  $0.5    6
    milk    $5.3    1

Total price: $12.7
```

Q5: [★★] Two Strings

Prompt the user for two strings. Keep prompting the user until the following conditions are satisfied:

1. Neither string contains a space.
2. The second string is longer than the first string.
3. Every character in the first string can be found in the second string.

A sample run of the program can be found below:

```
    Enter the first string: SMU SIS

    Enter the second string: SMUSIS

    Conditions not satisfied!


    Enter the first string: SMUSIS

    Enter the second string: SMUSIS
```

```
Conditions not satisfied!

Enter the first string: SMUSIS
Enter the second string: SMUSMUSMU
Conditions not satisfied!

Enter the first string: SMUSIS
Enter the second string: SMUSISS

Bingo!
```

Q6: [★★★] Multiplication Table

Write a program that helps the user to practice multiplication table. The program keeps asking whether the user wants to continue practicing until the user decides to quite. If the user wants to continue, the program randomly picks two numbers between 1 and 9 (both inclusive) and asks the user for his/her answer. If the answer is wrong, the program keeps prompting until the answer is correct.

A sample run of the program can be found below:

```
Do you want to practice multiplication table? Please enter Y or N: Y
What's the result of 4 times 9? 27
Wrong answer! Please try again: 36
You are right!

Do you want to continue your practice? Please enter Y or N: Y

What's the result of 4 times 2? 10
Wrong answer! Please try again: 9
Wrong answer! Please try again: 8
You are right!

Do you want to continue your practice? Please enter Y or N: Y

What's the result of 3 times 4? 12
You are right!

Good-bye!
```

Q7: [★★★] Line-up

A group of students need to line up for the procession of an event. We only have the information about every pair of students who are next to each other. We use a list of tuples to represent this line-up information: Each element of the list is a tuple that contains the name of a student (call him/her A) and the name of the student behind A.

For example, ("Alice", "Bob") means Bob is behind Alice. The list [("Alice", "Bob"), ("Bob", "Chris")] means Bob is behind Alice and Chris behind Bob.

For the student who's at the very end of the line-up, we use an empty string to indicate that there's nobody behind this student. E.g., ("Darren", "") means there's nobody behind Darren, i.e., Darren is at the end of the line-up.

Given a list of tuples that contains the line-up information of every pair of students who are next to each other, implement a function that returns the students in a list based on their order in the line-up.

Example 1: `get_lineup([("Chris", "Darren"), ("Alice", "Bob"), ("Darren", ""), ("Bob", "Chris")])` should return `["Alice", "Bob", "Chris", "Darren"]`.

Example 2:

Given the following code,

```
info = [("Mary", "Jason"), ("John", "Alan"), ("Jason", "George"), ("
Alan", "Christie"), ("Christie", "Mary"), ("George", "")]

print(get_lineup(info))
```

we should see the following output:

```
['John', 'Alan', 'Christie', 'Mary', 'Jason', 'George']
```

Note: If the list passed to the function is empty, the function returns an empty list.