CSE 1321L: Programming and Problem Solving I Lab

Lab 5

Flow Control (Part 2)

What students will learn:

- Using **WHILE** loops.
- Using **FOR** loops.

Content

If there is one thing computers are good at, it is repeating something over and over. The concept of repetition (which some call "iteration" and other "looping") is not terribly difficult since we humans repeat things in our daily lives. Any decent programming language is going to support iteration and usually allows for two different kinds of "looping templates". These templates are exactly what this lab is going to cover.

The two kings of loops we will cover are the **FOR** and **WHILE** loop. You want to memorize the template for these:

```
while <condition>:
      <body>

for <element> in <iterable>:
      <body>
```

In a **WHILE** loop, the <condition> can be anything (variable or expression) that resolves to a **Boolean** value (**True** or **False**). This could mean **Boolean** variables, as well as **comparison** or **logical expressions**.

In a **FOR** loop, the <iterable> can be anything that can be iterated over (we will see more about these at a later module. Right now, the only things that you can iterate over are **strings** and **ranges**). At each iteration of the loop, <element> will hold one element from the <iterable>. These elements are retrieved one by one, in the order that they appear in the <iterable>.

It is important to know **when** to use them. Here is an overall guideline to help you out:

- Use a **FOR** loop when you want to repeat something **a certain number of times**. For example:
  - If you want to repeat something 100 times.
  - If you want to count from 50 to 3000 in increments of 10.
- Use a **WHILE** loop when you **do not know how many times** something will repeat. For example:

- If you ask the user to enter a number between 1 to 10 and they consistently enter a higher number such as 45, then this loop could go on forever. Eventually, the user would enter a valid number.

As with previous weeks, all labs should have the appropriate file names:

- Lab5A.py
- Lab5B.py
- Lab5C.py

Lastly, make sure you review the sample output and make sure the output of your program follows the exact same format including the input statements, print statement, etc. As always, user input is shown in **red** and **bold**.

Lab5A: Largest of 10

For this lab, you will write a program that will ask the user to input **10 positive integer** numbers, **one at a time**. While it does this, the program should also keep track of the largest number the user has entered so far. Once the user enters 10 positive integers, the program should display the largest number entered by the user.

Requirements:

- Your solution must use a **FOR** loop exclusively for this lab.
- The user input must be handled inside the loop, and it must be handled dynamically so **do not** manually write 10 input statements.
- The program assumes the user will only enter valid positive integers, so no input validation is required.
- The program must keep track of the highest number entered **so far** by the user.
- We have not covered collections yet, so your solution **should not** use lists or other data structures.
- Make sure to review the Sample Output as guidelines for the prompts and print statements. Your output **must** match the Sample Output provided.

Hints:

- The user will input **positive integer** numbers, so what do you think is the lowest value the user may input?
- Your solution should not contain a collection such as a list, therefore the program cannot remember all the user input.

Sample Output #1

```
Please enter 10 numbers and this program will display the largest.
Please enter number 1: 50
Please enter number 2: 51
Please enter number 3: 10
Please enter number 4: 1
Please enter number 5: 99
Please enter number 6: 1000
Please enter number 7: 1010
Please enter number 8: 42
Please enter number 9: 86
Please enter number 10: 1000

The largest number was 1010
```

Lab5B: Factorial Calculator

For this lab we will write a program that will calculate the factorial of a given 0 or positive integer input.

Remember that the factorial of a number *"n"* is the product of all positive integers from 1 up to *"n"*.

For example:

- Factorial of 3 or 3! = 3 * 2 * 1 = 6
- Factorial of 4 or 4! = 4 * 3 * 2 * 1 = 24
- Factorial of 5 or 5! = 5 * 4 * 3 * 2 * 1 = 120
- And so on…

In this lab your solution should prompt the user for a number and calculate the factorial of that number using a **FOR** loop.

One important consideration regarding factorials, factorials of 0 and 1 will always be 1:

- 0! = 1
- 1! = 1

Requirements:

- Your solution **must** implement a **FOR** loop to **calculate the factorial**.
  - Any other approaches will be counted as **0**.
- You may use **any type of loop** for anything else if required.
- The user should be able to input an integer number.
- The program should validate whether the user input is positive or zero.
- Input values less than 0 should output an error (review sample output for format) and re-prompt.
- The program should **keep prompting** for a number until the user enters a valid number.
- Given a valid input number, the program should proceed by calculating the factorial of said number.
- After the factorial calculation is performed, the program should output the calculated value (review sample output for format).
- Review the sample output format provided, your solution should follow the specified output format.

Hint:

- In the case of factorial of 3 or **3!** We perform **2** calculations: **3 * 2 = 6**, and **6 * 1 = 6**
- In the case of factorial of 4 or **4!** We perform **3** calculations: **4 * 3 = 12**, **12 * 2 = 24**, and **24 * 1 = 24**

Sample Output #1

```
[Factorial Calculator]
Enter a number: -1
Error: Number must be 0 or positive, try again.

Enter a number: -23
Error: Number must be 0 or positive, try again.

Enter a number: 3
```

```
Calculating 3! ...
3! is 6

Program Terminated
```

Sample Output #2

```
[Factorial Calculator]
Enter a number: 4

Calculating 4! ...
4! is 24

Program Terminated
```

Sample Output #3

```
[Factorial Calculator]
Enter a number: 8

Calculating 8! ...
8! is 40320

Program Terminated
```

Lab5C: Say “please”

For this lab, we will create a simple program that keeps looping until the user inputs “please”.

Requirements:

- Since we do not know how many times this loop will keep iterating, the most fitting type of loop to be used here is a **WHILE** loop.
- Your solution must implement a **WHILE** loop exclusively.
- Keep asking ‘*If you would like to stop this program, say “please”: ’*
- Keep looping until the user enters “please”.
- The program should terminate if the user enters “please”
    - The user input should be case-sensitive, meaning the input **must** be exactly “please” in all lowercase letters.

Sample Output #1

```
If you would like to stop this program, say "please": please
Program complete
```

Sample Output #2

```
If you would like to stop this program, say "please": pLeAsE
If you would like to stop this program, say "please": Please
If you would like to stop this program, say "please": please
Program complete
```

Sample Output #3

```
If you would like to stop this program, say "please": no
If you would like to stop this program, say "please": nah
If you would like to stop this program, say "please": nuh-uh
If you would like to stop this program, say "please": ok fine
If you would like to stop this program, say "please": please
Program complete
```

Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
    - Lab5A.py
    - Lab5B.py
    - Lab5C.py
- Programs (source code files) must be uploaded to Gradescope by the due date.