University Of Engineering and Technology, Lahore
Computer Engineering Department

Course Name: Fundamentals of Programming & Data Science	**Course Code:** CMPE-112L
Assignment Type: Lab	**Dated:** 22nd January 2024
Semester: 2nd	**Session:** 2023
Lab/Project/Assignment #: 2	**CLOs to be covered: CLO 1**
Lab Title: Loops in Python	**Teacher Name: Engr.** Afeef Obaid

Lab Evaluation:

CLO 1	Apply appropriate programming techniques to create executable programs to solve well defined problems					
Levels (Marks)	**Level1**	**Level2**	**Level3**	**Level4**	**Level5**	**Level6**
(10)						
					Total	**/10**

Rubrics for Current Lab Evaluation

Scale	**Marks**	**Level**	**Rubric**
Excellent	**10**	L1	Submitted all lab tasks, BONUS task, have good understanding.
Very Good	**8**	L2	Submitted the lab tasks but have good understanding
Good	**6**	L3	Submitted the lab tasks but have weak understanding.
Basic	**4**	L4	Submitted the lab tasks but have no understanding.
Barely Acceptable	**2**	L5	Submitted only one lab task.
Not Acceptable	**0**	L6	Did not attempt

LAB No. 2

Lab Goals/Objectives:

By reading this manual, students will be able to:

- To understand the concept of loops in Python
- To learn how to iterate the while-loop in Python
- To know about how to use nested-loop in python

Equipment Required: Computer system with Pycharm IDE and python package installed on it

Loops In Python

In Python, Loops are used to execute a group of instructions or a block of code multiple times, without writing it repeatedly. The block of code will be executed as many times as the control statement will hold true and the Loop will be terminated when the conditions in the control statement become false.

The structure of a Loop can be virtually divided into two parts

- Control Statement
- Body

Control Statement:

The control statement of a Loop comprises the conditions that have to be met for the execution of the body of the Loop. For every iteration of the Loop, the conditions in the control statement have to be true

Body:

The body of a Loop comprises the block of code or the sequence of logical statements that are to be executed multiple times

Types of Loops in Python

- While Loop
- For Loop
- Nested Loop
- Infinite Loop

While Loop

A while loop is a control flow statement in Python that allows code to be repeatedly executed as long as a certain condition is true. It has the following syntax:

```
while condition:

   statement(s)
```

The condition is evaluated before each iteration of the loop, and if it is true, the statement(s) inside the loop are executed. Once the statement(s) have been executed, the condition is evaluated again, and the loop continues as long as the condition remains true.

The statement(s) inside the loop can include any valid Python code, including other loops, conditional statements, and function calls. It is important to note that if the condition is initially false, the loop will not be executed at all.

It is important to be careful when using while loops, as it is possible to create an infinite loop if the condition is never false. In this case, the loop will continue to execute indefinitely, which can cause the program to crash or become unresponsive. To avoid this, make sure that the condition will eventually become false.

Example

```
password = "secret"
max_attempts = 3
attempts = 0
while attempts < max_attempts:
   user_password = input("Enter the password: ")
   attempts += 1
   if user_password == password:
      print("Access granted!")
      break
   else:
      print("Incorrect password. Please try again.")
if attempts == max_attempts:
   print("Maximum number of attempts reached. Access denied.")
```

For Loop

A for loop in Python is a loop that iterates through code in its body for a set number of times until a condition is met. This is helpful in repetitive instances where a user needs to perform the same task a large number of times; or when a user needs to iterate through a sequence. The syntax for a for loop in Python is as follows:

```
for i in range(n):
 # Loop body
```

The letter n is a placeholder for the number of times one wants to iterate through the for loop. The loop body is where the executable code is placed.

The program will set i equal to 0 and run through the loop body n number of times, adding 1 to i after each iteration. In other words, range(n) tells the program, "for i = 0, every time i < n, run through the loop and add 1 to i." i is an integer that has been initialized to 0 for the purpose of the for loop.

Example 1:

```
num=int(input("Enter Limit "))
sum=0
for i in range(num+1):
   sum=sum+i
print(f"The sum is {sum}")
```

Example 2:

```
word=input("Enter any word ")
for i in word:
   print(f"{i}--->{ord(i)}")
```

Nested Loop

A nested loop is a loop inside the body of the outer loop. The inner or outer loop can be any type, such as a while loop or for loop. For example, the outer for loop can contain a while loop and vice versa.

The outer loop can contain more than one inner loop. There is no limitation on the chaining of loops.

In the nested loop, the number of iterations will be equal to the number of iterations in the outer loop multiplied by the iterations in the inner loop.

Nested loops are typically used for working with multidimensional data structures, such as printing two-dimensional arrays, iterating a list that contains a nested list.

```
for element in range(n):

    body of outer for loop

    for element in range(n):

        body of inner for loop
```

Example 1:

```
n = 5
for i in range(1, n+1):
   for j in range(1, i+1):
      print(j, end='')
   print()
```

Example 2:

```
word = "python"
for i in range(len(word)):
   for j in range(i+1):
      print(word[j], end='')
   print()
```

Infinite Loop

A loop becomes infinite loop if a condition never becomes FALSE. You must use caution when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.

Example # 1

```
x=int(input("Enter any Positive number"))
i=0
while i<=x:
   print(i)
```

As we are not increasing the value of i, that's why the condition remains true and this while loop becomes infinite loop

```
x=int(input("Enter any Positive number"))
i=0
while i<=x:
   print(i)
   i=i+1
```

Now the while loop becomes finite loop

Example # 2 (Intentional Infinite Loop)

```
x=int(input("Enter Range "))
sum=0
i=0
while True:
   print(i)
   i=i+1;
   if(i>=x):
      break;
```

Lab Tasks

- Write a program that convert a binary number into its decimal equivalent.
- Write a program that takes a sentence from the user and calculate the number of vowels and consonants in the sentence.
- Write a program that takes the range from the user and display all the prime numbers in that range and also print their sum.
- Write a program that generates a random password for the user. Program must ask the valid characters for the password from the user like length of password, is password contain uppercase and lowercase alphabets, is password contain digits, is passwords has special characters and then generates password according to it.
- Write a program that draw the below pattern by using nested loop

```
*

*  *

*  *  *

*  *  *  *

*  *  *

*  *

*
```