LAB # 07

ITERATION WITH LOOPS

OBJECTIVE

To get familiar with the different types of loops - While

THEORY

A loop can be used to tell a program to execute statements repeatedly. In other word, to keep a computer doing useful work we need repetition, looping back over the same block of code again and again.

Type of Loop Statements in Python:

There are 2 types of loop statements in Python language. They are,

1. for
2. **while**

The while loop:

A **while** loop executes statements repeatedly as long as a condition remains true.

Syntax:

The syntax of a **while** loop in Python programming language is:

```
while loop-continuation-condition:
   # Loop body
     Statement(s)
```

How while loop works:

- ✓ In while loop first the condition (boolean expression) is tested.
- ✓ If it is false the loop is finished without executing the statement(s).
- ✓ If the condition is true, then the statements are executed and the loop executes again and again until the condition is false.
- ✓ Each loop contains a loop-continuation-condition, a Boolean expression that controls the body's execution

Example program for 'while loop': Finite Time

```
count = 0
while count < 3:
    print("Programming is fun!")
    count += 1
```

Output:

```
>>> %Run Lab_07.py
Programming is Fun!
Programming is Fun!
Programming is Fun!
>>>
```

Example program for ‘while loop’: Infinite

```
while(True):
    print("Hello Students!")
```

Output:

```
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!
Hello Students!Traceback (most recent call last):
  File "C:\Users\Dell\Documents\Python Labs\Lab_07.py", line
10, in <module>
    print("Hello Students!")
KeyboardInterrupt: Execution interrupted
```

The break Statement

With the break statement we can stop the loop even if the while condition is true.

Example program for ‘while loop’: Break Statement

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

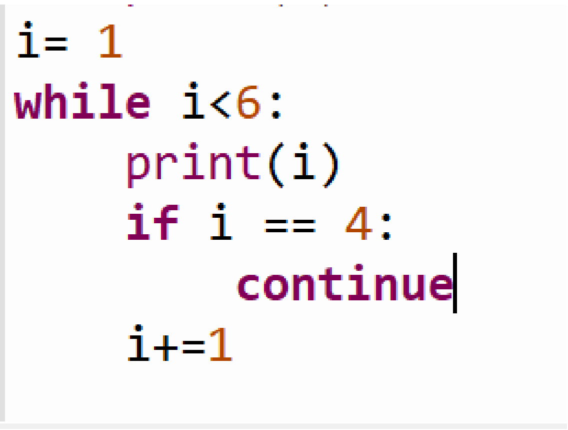
Output:

```
>>> %Run Lab_07.py
1
2
3
4
```

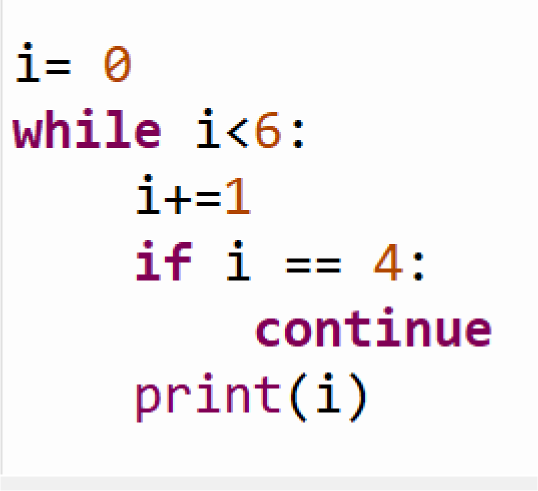
LAB # 07

```
>>>
```

Example: Differentiate them.

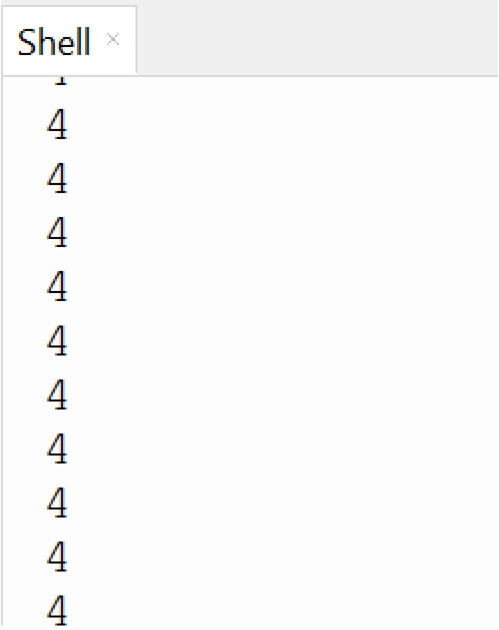


```
i= 1
while i<6:
    print(i)
    if i == 4:
        continue
    i+=1
```

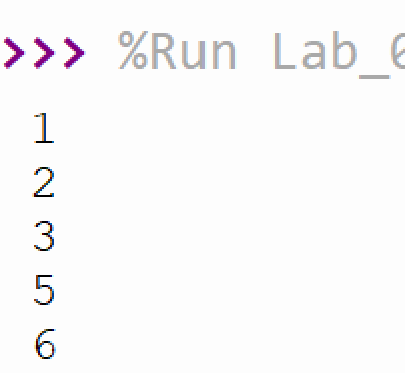


```
i= 0
while i<6:
    i+=1
    if i == 4:
        continue
    print(i)
```

Output



```
1
4
4
4
4
4
4
4
4
4
4
```



```
>>> %Run Lab_6
1
2
3
5
6
```

The else Statement

Example

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

LAB # 07

Output

```
>>> %Run Lab_07.py
1
2
3
4
5
i is no longer less than 6
>>
```

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code:

```
i = 0
while i < 5:
    print("Hello")
```

2. Code:

```
i = 1
while i <= 5:
    print(i)
    i = i – 1
```

3.Code

```
i = 10
while i > 10:
    print(i)
    i -= 1
```

LAB # 07

B. What will be the output of the following programs:

1. Code

```
i = 1
while i <= 10:
    print(i)
    i += 1
```

Output

2. Code

```
user_input = ""
while user_input != "stop":
    user_input = input("Type something (or 'stop' to end): ")
```

Output

3. Code

```
i = 1
total = 0

while i <= 10:
    total += i
    i += 1

print("Sum of first 10 natural numbers is:", total)
```

Output

LAB # 07

C. Write Python programs for the following:

1. **Take input and find the factorial of a number** using a while loop.
2. **Take two inputs (start and stop)** and print all numbers between them using a while loop.
3. **Reverse a number** entered by the user (e.g., 123 → 321) using a while loop.
4. **Display the Fibonacci series** (using a while loop) up to a given number of terms.
5. **Simulate a digital clock** that counts seconds from 0 to 59 using a while loop.