LOOPS

1. Why is looping used ?

Looping is used to repeatedly perform a block of statements over and over again.

2. What are the different types of loops in Python ?

1. For Loop:

For loop is used to iterate over a sequence, starting from the first value to the last. The number of iterations to be performed depends upon the length of the list.

Syntax:

```
for iteratingVariable in sequence:
    statement 1
    statement 2
    - - -
    - - -
    statement n
```

Example:

```
numbers = [1, 2, 3, 4, 5]
for number in numbers:
   print(number, end=‘ ‘)
```

Output:

```
   1 2 3 4 5
```

Note: end = ‘ ’ is used to end the print statement with a white space instead of a new line

In the example mentioned above, the variable “number” is used to iterate over the list “numbers”. Here, the loop is executed 5 times since the length of the list is 5. At first iteration, the variable “number” holds the value “1”, at the second iteration value it holds the value “2” and so on.

2. While Loop:

"While loop" is used to repeatedly execute a block of statements as long as the condition mentioned in the "while loop" holds true.

Syntax:

```
while condition:
    statement 1
    statement 2
    - - -
    - - -
    statement 3
```

Example:

```
length = 1
while length <= 3:
   print("Value of length = ", length)
   length = length + 1
```

Output:

```
   Value of length = 1
   Value of length = 2
   Value of length = 3
```

In this example, the execution of while loop depends on the value stored in the variable "length". Each time the block of code in while loop gets executed, we increment the value of "length" by 1. When the value stored in the variable length is "4", the condition "length <= 3" turns false and the loop breaks.

3. Nested Loop:

A loop within another loop is called a nested loop. The concept of nested loop could be a little bit of trouble understanding at first, but can be simplified with the help of an example.

```
   outerLoopValue = 1
   innerLoopValue = 1

   while outerLoopValue <= 2:
      # This inner loop runs three times for every
iteration of the outer loop
      while innerLoopValue <=3:
         print("Outer loop value = ",
outerLoopValue)
         print("Inner loop value = ",
innerLoopValue)
         # Increment the inner loop iteration
value
         innerLoopValue = innerLoopValue + 1
         # Increment the outer loop iteration
value
         outerLoopValue = outerLoopValue + 1
         # Reset the inner loop value to 1
         innerLoopValue = 1
```

In a nested loop, only after the completion of all iterations of the innermost loop does the outer loop proceed to its next iteration. Please refer the table below with reference to the above example.

Iteration	Outer Loop value	Inner Loop value	Comments
1	1	1	On successful execution of outer loop, inner loop starts its execution
2	1	2	Inner loop continues its iteration in the first iteration of outer loop and increments its value to 2
3	1	3	Inner loop continues its iteration in the first iteration of outer loop and increments its value to 3
4	2	1	Since the while condition in inner loop broke when the innerLoopValue became “4”, the execution once again started from outer loop after incrementing its value to 2, and iteration of inner loop once again starts from 1
5	2	2	Inner loop continues its iteration in the second iteration of outer loop and increments its value to 2
6	2	3	After this, outerLoopValue becomes 3 and the loop breaks

3. What are break, continue and else statements ?

Break:

A break statement is used to stop a loop from further execution.

Example:

```
length = 1
while length > 0:
   if length == 3:
      break
   print(“Length = “, length)
   length = length + 1
```

Output:

```
Length = 1
Length = 2
Length = 3
```

In the above example, when length = 3, the break statement gets executed and the while loop breaks.

Continue:

Continue statement is used to skip a particular iteration of the loop.

```
Example:
   length = 1
   while length <= 4:
      if length == 2:
         length = length + 1
         continue
      print(“Length = “, length)
      length = length + 1
```

Output:

```
Length = 1
Length = 3
Length = 4
```

In the above example, when length = 2, continue statement stops further execution of that iteration and moves on to the next iteration

Note: While break statement stops the whole loop from execution, continue stops just an iteration of that loop.

Else:

The block of statements in the else block gets executed if the break statement in the looping condition was executed

Example:

```
length = 1
while length <= 3:
   if length == 5:
      break
   print(“Length = “, length)
   length = length + 1
else:
   print(“Break statement was not 
executed”)
```

Output:

```
Length = 1
Length = 2
Length = 3
Break statement was not executed
```

In the above example, the break statement does not get executed and the statement in else block gets executed after the loop.