Python Loops

- The following loops are available in Python to fulfil the looping needs. Python offers 3 choices for running the loops. The basic functionality of all the techniques is the same, although the syntax and the amount of time required for checking the condition differ.
- We can run a single statement or set of statements repeatedly using a loop command.

The following sorts of loops are available in the Python programming language.

Sr.No.	Name of the loop	Loop Type & Description
1	**While loop**	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2	**For loop**	This type of loop executes a code block multiple times and abbreviates the code that manages the loop variable.
3	**Nested loops**	We can iterate a loop inside another loop.

While Loop in Python

- In Python, a while loop is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.

Python While Loop Syntax:

```
while expression:
    statement(s)
```

- All the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.
- Let's learn how to use a while loop in Python with Examples:
- Example of Python While Loop

Let's see a simple example of a while loop in Python. The given Python code uses a 'while' loop to print "Hello Geek" three times by incrementing a variable called 'count' from 1 to 3.

```
count = 0
while (count < 3):
    count = count + 1
    print("Hello Geek")
```

Output

```
Hello Geek
Hello Geek
Hello Geek
```

Infinite While Loop in Python

- If we want a block of code to execute infinite number of times, we can use the while loop in Python to do so.

- The code uses a 'while' loop with the condition (count == 0). This loop will only run as long as count is equal to 0. Since count is initially set to 0, the loop will execute indefinitely because the condition is always true.

```
count = 0
while (count == 0):
    print("Hello Geek")
```

Note: It is suggested not to use this type of loop as it is a never-ending infinite loop where the condition is always true and you have to forcefully terminate the compiler.

For Loop in Python

For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is “for in” loop which is similar to foreach loop in other languages. Let us learn how to use for loops in Python for sequential traversals with examples.

For Loop Syntax:

```
for iterator_var in sequence:
    statements(s)
```

It can be used to iterate over a range and iterators.

Example:

The code uses a Python for loop that iterates over the values from 0 to 3 (not including 4), as specified by the range(0, n) construct. It will print the values of ‘i' in each iteration of the loop.

```
n = 4
for i in range(0, n):
    print(i)
```

Output

```
0
1
2
3
```

Nested Loops in Python

- Python programming language allows to use one loop inside another loop which is called nested loop. Following section shows few examples to illustrate the concept.

Nested Loops Syntax:

```
for iterator_var in sequence:
   for iterator_var in sequence:
       statements(s)
   statements(s)
```

- The syntax for a nested while loop statement in the Python programming language is as follows:

```
while expression:
   while expression:
       statement(s)
   statement(s)
```

- A final note on loop nesting is that we can put any type of loop inside of any other type of loops in Python. For example, a for loop can be inside a while loop or vice versa.

- Example: This Python code uses nested ‘for' loops to create a triangular pattern of numbers. It iterates from 1 to 4 and, in each iteration, prints the current number multiple times based on the iteration number. The result is a pyramid-like pattern of numbers.

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

Output

```
1
2 2
3 3 3
4 4 4 4
```