Loop Statement

- A **loop** is an instruction that repeats multiple times as long as some condition is met.
- Looping simplifies complicated problems into smooth ones. It allows programmers to modify the flow of the program so that rather than writing the same code, again and again, programmers are able to repeat the code a finite number of times.
- In Python, there are three different types of loops: for loop, while loop, and nested loop.

Here, we will read about these different types of loops and how to use them.

❖ Types of Loops

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.

❖ For loop

- 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 for each loop in other languages. Let us learn how to use for in loop for sequential traversals.
- It can be used to iterate over a range and iterators.

Syntax :

```
for value in sequence:
    {loop body}
```

Example :

```
# Code to find the sum of squares of each element of the list using for loop

# creating the list of numbers
numbers = [3, 5, 23, 6, 5, 1, 2, 9, 8]
# initializing a variable that will store the sum
sum_ = 0
# using for loop to iterate over the list
for num in numbers:

sum_ = sum_ + num ** 2

print("The sum of squares is: ", sum_)
```

Output:

```
The sum of square is 774
```

❖ Iterating by the index of sequences

- We can also use the index of elements in the sequence to iterate.
- The key idea is to first calculate the length of the list and in iterate over the sequence within the range of this length.

Example:

```
# Python program to illustrate  Iterating by index
list = ["Topper", "World"]
for index in range(len(list)):
        print(list[index])
```

Output:

```
Topper
World
```

❖ While Loop

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

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.

Example:

```
# Python program to illustrate while loop
count = 0
while (count < 3):
        count = count + 1
        print("Topper World")
```

Output:

```
Topper World
Topper World
Topper World
```

❖ Using else statement with While Loop in Python

- The else clause is only executed when your while condition becomes false. If you break out of the loop, or if an exception is raised, it won't be executed.

Syntax:

```
while condition:
    # execute these statements
else:
    # execute these statements
```

Examples:

```
# Python program to illustrate combining else with while
count = 0
while (count < 3):
        count = count + 1
        print("Hello Geek")
else:
        print("In Else Block")
```

Output:

```
Topper World
Topper World
Topper World
In Else Block
```

❖ Infinite While Loop in Python

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

```
# Python program to illustrate  Single statement while block
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.

❖ Nested Loops

Python programming language allows to use one loop inside another loop.

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)
```

Example:

```
# Python program to illustrate
# nested for loops in Python
from __future__ import print_function
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
```