LOOP St:-

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

Python programming language provides following types of loops to handle looping requirements.

Sr.No.	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 Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	nested loops You can use one or more loop inside any another while, for or do..while loop.

➢ While loop:-

A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

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

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

Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

In Python, 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.

Ex.
```
count = 0
while (count < 9):
  print 'The count is:', count
  count = count + 1

print "Good bye!"
```

Using else statement with while loops: As discussed above, while loop executes the block until a condition is satisfied. When the condition becomes false, the statement immediately after the loop is executed. 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.

If else like this:

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

And while loop like this are similar

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

Ex.

```
#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:

```
Hello Geek
Hello Geek
Hello Geek
In Else Block
```

For Loop

The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects. Iterating over a sequence is called traversal.

Syntax of for Loop

```
for val in sequence:
          Body of for
```

Here, val is the variable that takes the value of the item inside the sequence on each iteration.
Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.

Ex.

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

```
# Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# Variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
          sum = sum+val

print("The sum is", sum)
```

Nested Loops: Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax:

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

Nested Loop

Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax

```
for iterating_var in sequence:
   for iterating_var in sequence:
      Statements(s)
   Statements(s)
```

The syntax for a **nested while loop** statement in Python programming language is as follows −

```
while expression:
   while expression:
      Statement(s)
   Statement
```

A final note on loop nesting is that you can put any type of loop inside of any other type of loop.

For example a for loop can be inside a while loop or vice versa.

Example

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error. It is used when a statement is required syntactically but you do not want any command or code to execute.

The pass statement is a null operation; nothing happens when it executes.

Example

```
for x in [0, 1, 2]:
  pass
```