Loops

In any programming language, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.

A sequence of statements is executed until a specified condition is true. This sequence of statements to be executed is kept inside the Loop body. After every execution of loop body, condition is verified, and if it is found to be true the loop body is executed again. When the condition check returns false, the loop body is not executed.

Types of Loops

There are two types of loops in Python that is given below.

- while
- for

While Loop

In While loop it iterates the code until condition is false. Here, condition is given before the code. So code may be executed 0 or more times.

Syntax

```
while expression:
    Statement
```

Example	Output
a=1	1
while a<5:	2
print a	3
a=a+1	4

Infinite while loop

A loop becomes infinite loop if a condition never becomes FALSE. You must use caution when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.

Example	Output
a = 1	Enter a number 10
while a == 1 :	You entered: 10
a1=input("Enter a number")	Enter a number 20
print "You entered: ", a1	You entered: 20
print "this is infinite loop""	Enter a number 30
	You entered: 30
	Enter a numberTraceback (most recent call last):
	File "main.py", line 3, in <module>
	a1=input("Enter a number")
	EOFError: EOF when reading a line

else Statement with While Loops

Python enables us to use the while loop with the while loop also. The else block is executed when the condition given in the while statement becomes false. Like for loop, if the while loop is broken using break statement, then the else block will not be executed and the statement present after else block will be executed.

Example	Output
`a=0`	0
`while a<5:`	1
`    print a`	2
`    a=a+1`	3
`else:`	4
`    print "Value of a is",a`	Value of a is 5

For Loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```
for var in sequence:
    statement
```

Example	Output
a=0	Value of a is 0
for a in range(0,5):	Value of a is 1
print "Value of a is",a	Value of a is 2
	Value of a is 3
	Value of a is 4

Example	Output
list =[10,20,30,40,50]	current list member is 10
for a in list:	current list member is 20
print 'current list member is', a	current list member is 30
	current list member is 40
	current list member is 50

Nested for loop

Python allows us to nest any number of for loops inside a for loop. The inner loop is executed n number of times for every iteration of the outer loop.

```
for var in sequence:
   for var in sequence:
      statements
   statements
```

Example	Output
list =[10,20,30,40,50]	10+10 = 20

list1 = [10]
for a in list:
    for b in list1:
        print '%d+%d = %d'%(a,b,a+b)
	20+10 = 30
	30+10 = 40
	40+10 = 50
	50+10 = 60

else Statement with for Loops

The else keyword in a for loop specifies a block of code to be executed when the loop is finished and in while loop block of code to be executed when condition of loop becomes false.

Example	Output

list =[10,20,30,40,50]
for a in list:
   print 'current list member is', a
else:
    print 'loop is finished'
	current list member is 10
	current list member is 20
	current list member is 30
	current list member is 40
	current list member is 50
	loop is finished