

Python's Loop

In this lecture , you'll learn to iterate over a sequence of elements using the different variations of for loop.

Python while Loop:

The **while** loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.



We generally use this loop when we don't know beforehand, the number of times to iterate.

Syntax of **while** Loop in Python

```
While test expression:
```

```
    Body of while
```

In while loop, test expression is checked first. The body of the loop is entered only if the **test_expression** evaluates to **True**. After one iteration, the test expression is checked again.

This process continues until the **test_expression** evaluates to **False**. In Python, the body of the while loop is determined through indentation.

Body starts with indentation and the first unindented line marks the end.

Python interprets any **non-zero** value as **True**. **None** and **0** are interpreted as **False**.

Flowchart of while Loop:

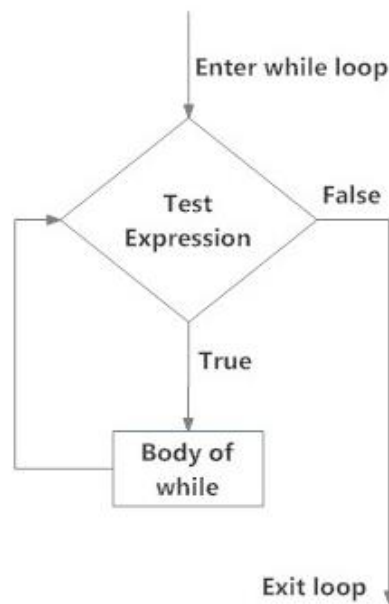


Fig: operation of while loop

Example: Python while Loop

```
script.py  IPython Shell
1  # Program to add natural
2  # numbers upto
3  # sum = 1+2+3+...+n
4
5  # To take input from the user,
6  # n = int(input("Enter n: "))
7
8  n = 10
9
10 # initialize sum and counter
11 sum = 0
12 i = 1
13
14 while i <= n:
15     sum = sum + i
16     i = i+1    # update counter
17
18 # print the sum
19 print("The sum is", sum)
```

When you run the program, the output will be:

Enter n: 10

The sum is 55

While loop with else

The **else** part is executed if the condition in the while loop evaluates to **False**. The while loop can be terminated with a **break statement**.

In such case, the **else** part is ignored. Hence, a while **loop's else** part runs if no break occurs and the condition is false.

Here is an example to explain this.

```
script.py  IPython Shell
1  # Example to illustrate
2  # the use of else statement
3  # with the while loop
4
5  counter = 0
6
7  while counter < 3:
8      print("Inside loop")
9      counter = counter + 1
10 else:
11     print("Inside else")
```

Output

```
Inside loop
```

```
Inside loop
```

```
Inside loop
```

```
Inside else
```

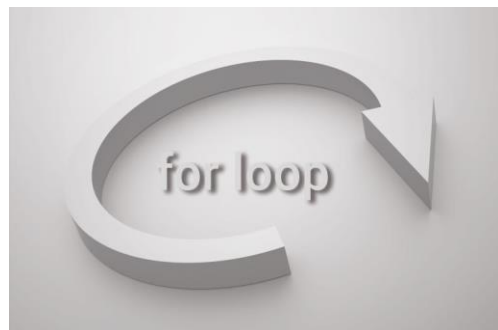
Here, we use a counter variable to print the string **Inside loop** three times.

On the forth iteration, the condition in while becomes **False**. Hence, the **else** part is executed.

```
*****
```

Python 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.

Flowchart of for Loop

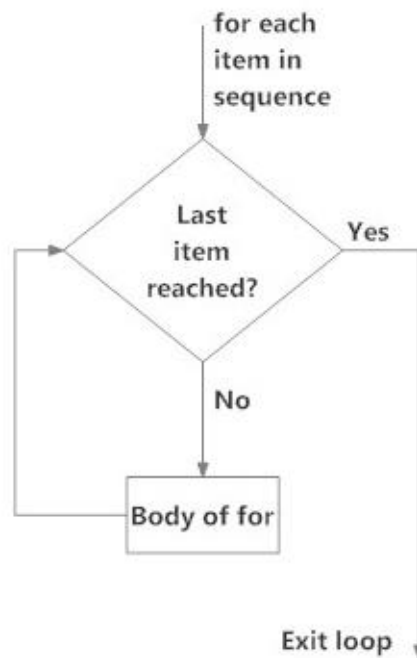


Fig: operation of for loop

Example: Python for Loop

```
script.py  IPython Shell
1  # Program to find the sum of all numbers stored in a list
2
3  # List of numbers
4  numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
5
6  # variable to store the sum
7  sum = 0
8
9  # iterate over the list
10 for val in numbers:
11     sum = sum+val
12
13 # Output: The sum is 48
14 print("The sum is", sum)
```

when you run the program, the output will be:

```
The sum is 48
```

The range() function

We can generate a sequence of numbers using `range()` function. `range(10)` will generate numbers from 0 to 9 (10 numbers).

We can also define the start, stop and step size as :

`range(start, stop, step size).`

step size defaults to 1 if not provided.

This function does not store all the values in memory, it would be inefficient. So it remembers the **start, stop, step size** and generates the next number on the go.

We can use the `range()` function in for loops to iterate through a sequence of numbers. It can be combined with the `len()` function to iterate through a sequence using indexing. Here is an example.

```
script.py  IPython Shell
1  # Program to iterate through a list using indexing
2
3  genre = ['pop', 'rock', 'jazz']
4
5  # iterate over the list using index
6  for i in range(len(genre)):
7      print("I like", genre[i])
```

When you run the program, the output will be:

```
I like pop  
  
I like rock  
  
I like jazz
```

for loop with else

A for loop can have an optional else block as well. The **else** part is executed if the items in the sequence used in for loop exhausts.

break statement can be used to stop a for loop. In such case, the else part is ignored. Hence, a **for loop's else** part runs if no break occurs.

Here is an example to explain this.

```
script.py  IPython Shell  
1  digits = [0, 1, 5]  
2  
3  for i in digits:  
4      print(i)  
5  else:  
6      print("No items left.")
```

When you run the program, the output will be:

```
0  
  
1  
  
5
```

No items left.

*****Python Program to Find the Factorial of a Number:***

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

FACTORIAL

To understand this example, you should have the knowledge of following [Python programming](#) topics:

- [Python if...else Statement](#)
- [Python for Loop](#)

The factorial of a number is the product of all the integers from 1 to that number.

For example, the factorial of 6 (denoted as 6!) is $1*2*3*4*5*6 = 720$. Factorial is not defined for negative numbers and the factorial of zero is one, $0! = 1$.

Code :


```
script.py  IPython Shell
1  # Python program to find the factorial of a number provided by the user.
2
3  # change the value for a different result
4  num = 7
5
6  # uncomment to take input from the user
7  #num = int(input("Enter a number: "))
8
9  factorial = 1
10
11 # check if the number is negative, positive or zero
12 ▾ if num < 0:
13     print("Sorry, factorial does not exist for negative numbers")
14 ▾ elif num == 0:
15     print("The factorial of 0 is 1")
16 ▾ else:
17 ▾     for i in range(1,num + 1):
18         factorial = factorial*i
19     print("The factorial of",num,"is",factorial)
```

Output

```
The factorial of 7 is 5040
```

Note: To test the program, change the value of **num**. Try negative numbers as well.

Here, the number whose factorial is to be found is stored in **num** and we check if the number is negative, zero or positive using **if...elif...else** statement.

If the number is positive, we use **for** loop and **range()** function to calculate the factorial.

Summary:

Syntax of while Loop in Python

```
while test_expression:
```

 Body of while

Syntax of for Loop

```
for val in sequence:
```

 Body of for