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:

```
for element in sequence :

    Body of for
```

Here, element 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.

Flow Chart

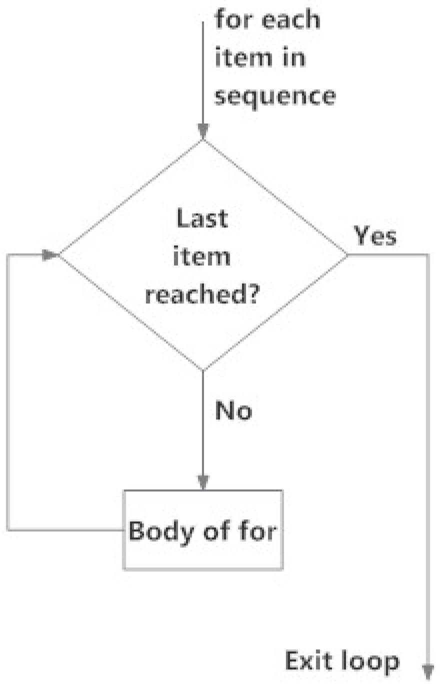

Fig: operation of for loop

Example

In [1]:

```
#Find product of all numbers present in a list

lst = [10, 20, 30, 40, 50]

product = 1
#iterating over the list
for ele in lst:
    print(type(ele))
    product *= ele

print("Product is: {}".format(product))
```

```
<class 'int'>
<class 'int'>
<class 'int'>
<class 'int'>
<class 'int'>
Product is: 12000000
```

In [2]:

```
ele
```

Out[2]:

```
50
```

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.

In [3]:

```
#print range of 10
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [4]:

```
#print range of numbers from 1 to 20 with step size of 2
for i in range(0, 20, 5):
    print(i)
```

```
0
5
10
15
```

In [6]:

```
lst = ["apple", "orange", "litchi", "mango", "watermelon"]

#iterate over the list using index
#for index in range(len(lst)):
#    print(lst[index])
for ele in lst:
    print(ele)
```

```
apple
orange
litchi
mango
watermelon
```

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.

In [7]:

```
numbers = [1, 2, 3]

#iterating over the list
for item in numbers:
    print(item)
else:
    print("no item left in the list")
```

```
1
2
3
no item left in the list
```

In [8]:

```
for item in numbers:
    print(item)
    if item % 2 == 0:
        break
else:
    print("no item left in the list")
```

```
1
2
```

Python Program to display all prime numbers within an interval

In [9]:

```
index1 = 20
index2 = 50

print("Prime numbers between {0} and {1} are :".format(index1, index2))

for num in range(index1, index2+1):      #default step size is 1
    if num > 1:
        isDivisible = False;
        for index in range(2, num):
            if num % index == 0:
                isDivisible = True;
        if not isDivisible:
            print(num);
```

```
Prime numbers between 20 and 50 are :
23
29
31
37
41
43
47
```

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.

Syntax:

```
while test_expression:

    Body of while
```

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.

Flow Chart

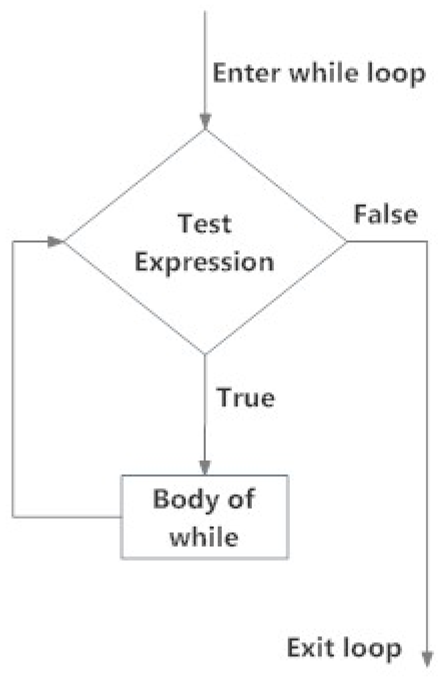

Fig: operation of while loop

Example

In [10]:

```
#Find product of all numbers present in a list

lst = [10, 20, 30, 40, 60]

product = 1
index = 0

while index < len(lst):
    product *= lst[index]
    index += 1

print("Product is: {}".format(product))
```

```
Product is: 14400000
```

while Loop with else

Same as that of for loop, we can have an optional else block with while loop as well.

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.

In [11]:

```
numbers = [1, 2, 3,4,5]

#iterating over the list
index = 0
while index < len(numbers):
    print(numbers[index])
    index += 1

else:
    print("no item left in the list")
```

```
1
2
3
4
5
no item left in the list
```

Python Program to check given number is Prime number or not

In [13]:

```
num = int(input("Enter a number: "))        #convert string to int


isDivisible = False;

i=2;
while i < num:
    if num % i == 0:
        isDivisible = True;
        print ("{} is divisible by {}".format(num,i) )
    i += 1;

if isDivisible:
    print("{} is NOT a Prime number".format(num))
else:
    print("{} is a Prime number".format(num))
```

```
Enter a number: 2
2 is a Prime number
```

```
In [ ]:
```