Loops

- There may be a situation when you need to execute a block of code several number of times.
- to iterate over items or elements in a given seq like str, list, tup, set or dict
- In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Iterations or loops - repeate actions

- for loop --> iterates for all items present in iterating_sequesnce:
- while loop --> until the condition satisfies

For loop

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

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 all items completed in the sequence.

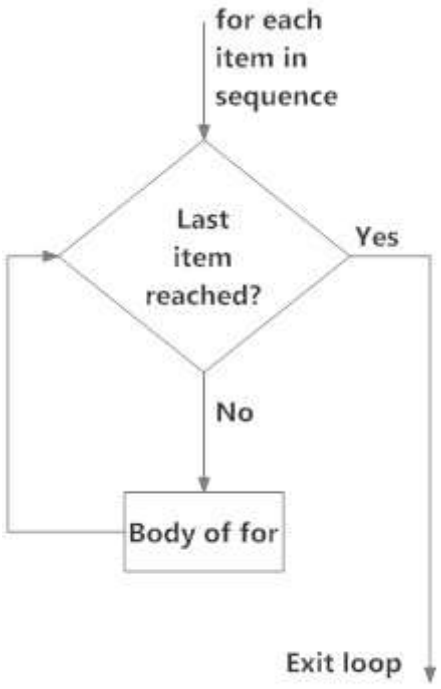


Fig: operation of for loop

In [1]:
```
i=1
j=i+5
k=j+10
print(i,j,k)

i=2
j=i+5
k=j+10
print(i,j,k)

i=3
j=i+5
k=j+10
print(i,j,k)
```

```
1 6 16
2 7 17
3 8 18
```

In [2]:
```
for i in range(1,4):
    j=i+5
    k=j+10
    print(i,j,k)
```

```
1 6 16
2 7 17
3 8 18
```

In [3]:
```
# for loop to print all the numbers between 10 and 50 (both inclusive)
for i in range(11,51):
    print(i,end=" ")
```

```
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3
7 38 39 40 41 42 43 44 45 46 47 48 49 50
```

In [4]:
```
#print even numbers in descending order from 1 to 100
for i in range(100,1,-2):
    print(i,end=' ')
```

```
100 98 96 94 92 90 88 86 84 82 80 78 76 74 72 70 68 66 64 62 60 58 56 54 52 50
48 46 44 42 40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2
```

In [5]:
```
# Write a python program to get all the even numbers for a given range both are i

lb = int(input('enter lower bound values:'))
ub = int(input('enter upper bound values:'))

for i in range(lb,ub+1):
    if i%2==0:
        print(i,end=' ')
```

```
enter lower bound values:10
enter upper bound values:22
10 12 14 16 18 20 22
```

In [6]:
```
seq = [1,2,3,4,5]

for i in seq:
    print(i)
```

```
1
2
3
4
5
```

In [7]:
```
seq = [1,2,3,4,5]

for item in seq:
    print(item)
```

```
1
2
3
4
5
```

In [8]:
```
items = ['item1', 'item2', 'item3', 'item4']

for i in range(len(items)):
    print("item at index ",i," is: ", items[i])
```

```
item at index  0  is:  item1
item at index  1  is:  item2
item at index  2  is:  item3
item at index  3  is:  item4
```

In [9]:
```
#Write program that gives numbers between 1 to 100 which are divisible by 5 and r
for num in range(1,101):
    if num%5 == 0 and num%7==0:
        print(num)
```

```
35
70
```

In [10]:
```
# to print sum of n whole numbers
a=[1,2,3,4]

sum =0 # initialization

for i in a:
    sum=sum+i
    print("after", i,"iteration the sum:",sum)
```

```
after 1 iteration the sum: 1
after 2 iteration the sum: 3
after 3 iteration the sum: 6
after 4 iteration the sum: 10
```

In [11]:
```
# Python Program sum of even numbers in between given range
lb=int(input())
ub=int(input())

s=0

for i in range(lb,ub+1):
    if (i%2==0):
        s=s+i
        print(i,end=' ')

print('\nsum is',s)
```

```
10
20
10 12 14 16 18 20
sum is 90
```

In [12]:
```
#Program to print multiple mathematical tables
a=int(input('enter multplication table:'))

for i in range(1,11):
    print(a,'*',i,'=',a*i)
```

```
enter multplication table:5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

In [13]:
```
# Write a program which can compute the factorial of a given numbers.
a=int(input())

fact=1

for i in range(a,0,-1):
    fact=fact*i

print(fact)
```

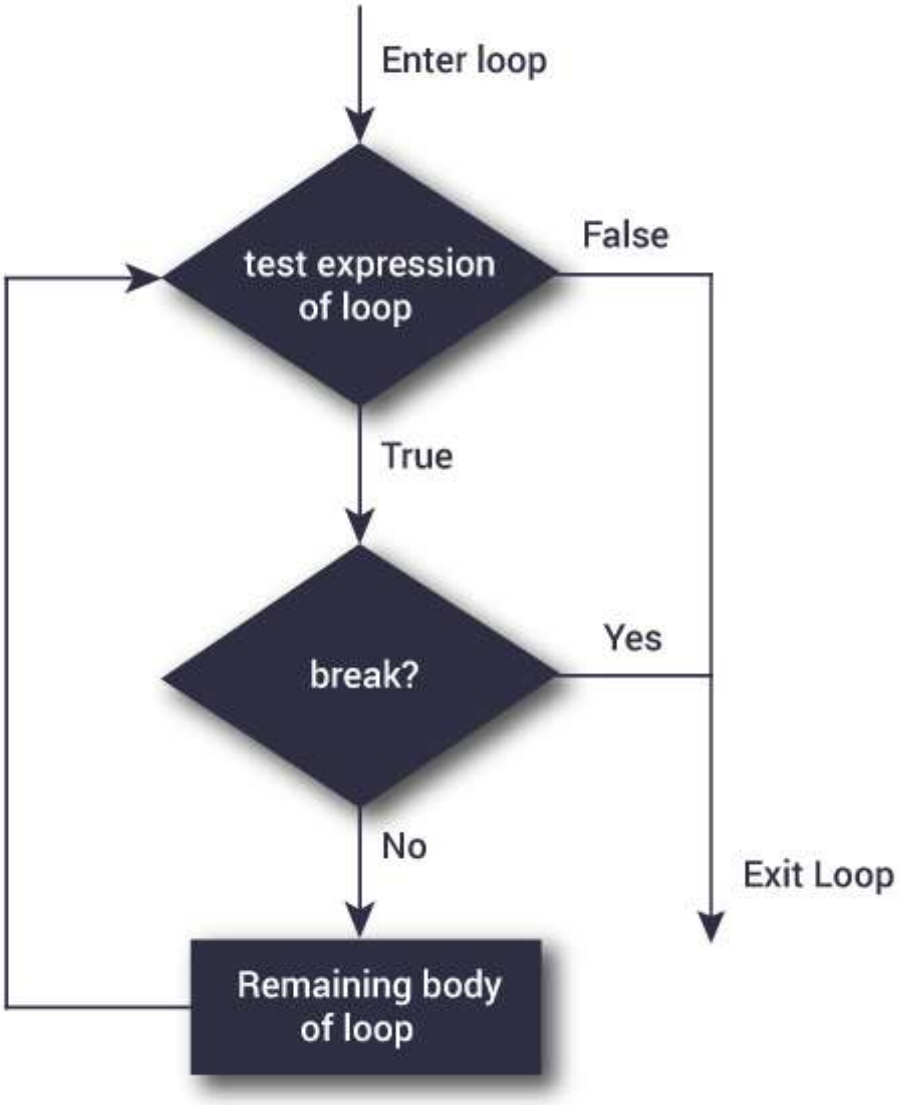
```
5
120
```

In [14]:
```
n=3

for i in range(n):
    for j in range(2):
        print("i=",i,"j=",j)
```

```
i= 0 j= 0
i= 0 j= 1
i= 1 j= 0
i= 1 j= 1
i= 2 j= 0
i= 2 j= 1
```

Python break Statement

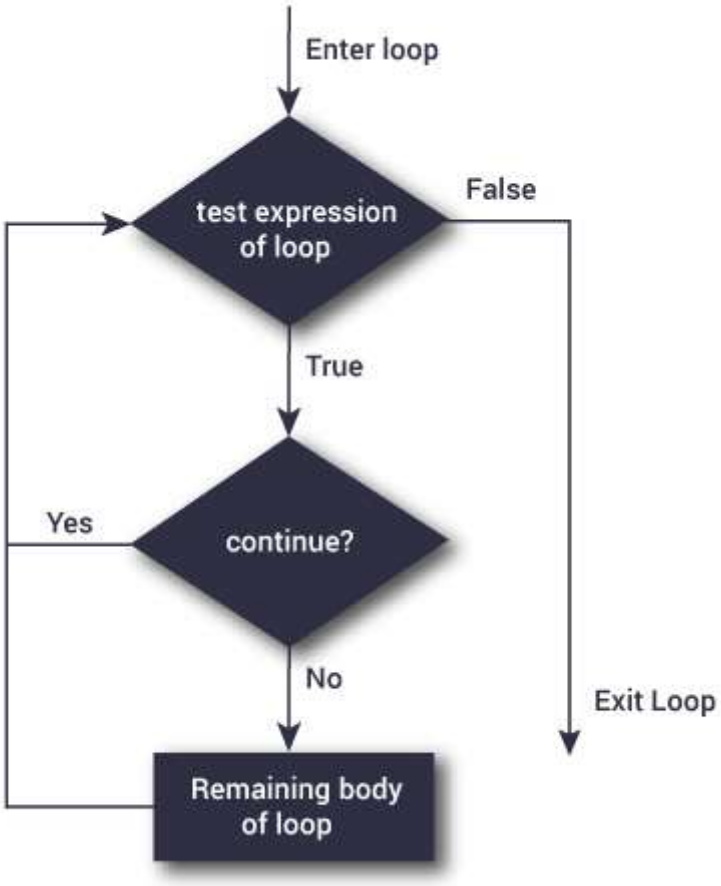


In [15]:
```
numbers = [1, 2, 3, 4]

for num in numbers:
    print(num)
    if num == 3:
        break
```

```
1
2
3
```

Continue




```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

In [16]:
```
#print odd numbers present in a list
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num % 2 == 0:
        continue
        print(num)
```

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 [17]:
```
numbers = [1, 2, 3]

#iterating over the list
for item in numbers:
    print(item)
    if item==4:
        break
else:
    print("no item left in the list")
```

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

In [18]:
```
numbers = [1, 2, 3, 4]

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

```
2
```

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.

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.

while loop 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.

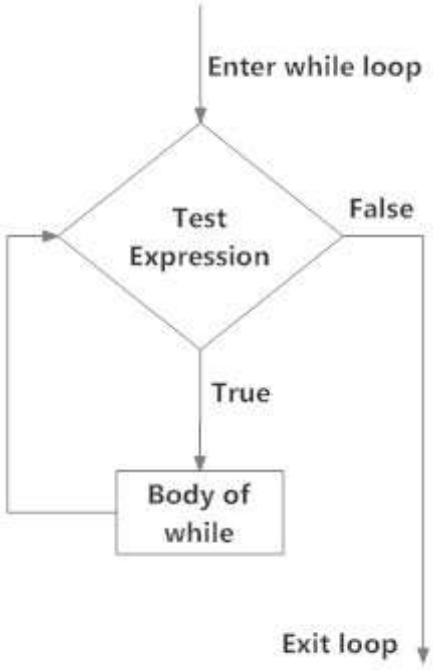


Fig: operation of while loop

In [19]:
```
x=0

while x < 6:
    print(x)
    x=x+1
```

```
0
1
2
3
4
5
```

In [22]:
```
i=0
s=0

while (i<=5):
    s=s+i
    i+=2
    print(s)
```

```
0
2
6
```

In [23]:
```
x=0

while x < 5:
    x=x+1

print(x)
```

```
5
```

In [24]:
```
# program to print 0 to 100 (inclusive) whole numbers
i=0
while(i<=100):
    print(i,end=' ')
    i=i+1
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 5
6 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82
83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

In [25]:
```
# Program to "n" add natural numbers --> sum = 1+2+3+...+n

n = int(input("Enter n: "))
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1

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

```
Enter n: 10
The sum is 55
```

In [27]:
```
#Find product of all numbers present in a list

lst = [1, 2, 3, 4, 6]

p = 1
i = 0

while i < len(lst):
    p = p*lst[i]
    i = i+1

print("Product is:",p)
```

```
Product is: 144
```

In [28]:
```
#Ask the use to enter the numbers and sum it

s=0

while True:
    a=input('enter a value:')
    if (a=='done'):
        break
    s=s+int(a)

print(s)
```

```
enter a value:5
enter a value:55
enter a value:40
enter a value:50
enter a value:50
enter a value:done
200
```

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 [29]:
```
numbers = [1, 2, 3, 4, 5]

index = 0

while index < len(numbers):
    print(numbers[index])
    index = index+1
    if index==9:
        break

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

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

In [30]:
```
x = 8
while x >= 1:
    print("x:",x)
    x = x-1
    if x==9:
        break
else:
    print("x is no longer greater than 1")
```

```
x: 8
x: 7
x: 6
x: 5
x: 4
x: 3
x: 2
x: 1
x is no longer greater than 1
```