Looping or Iterative or Repetitive Statements :

- The purpose of Looping or Iterative or Repetitive Statements is that "Perform Certain Operations Repeatedly for finite number of times until Test Condition Becomes False".
- There may be a situation when you need to execute a block of code several number of times.
- To iterate over times or elements in a given siquence like str, list, tuple, 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.

In Python Programming, we have 2 types of looping statements. They are

```
1. for loop or for-else loop
2. while loop or while-else loop
```

Iterations or loops - repeate actions :

- **for loop --> Iterates for all times present in iterating_sequence.**
- **while loop --> untill the condition satisfies.**

To deal with any any looping statements, we have to follow 3 steps. They are

```
1. Initlization Part
2. Conditional Part
3. Updation Part (Incrementation or Decrementation)
```

1. for loop or for-else loop

```
Syntax1: -
-----------
                for varname  in  Iterable_object:
                    ----------------------------------------
                    Indentation block of stmts
                    ----------------------------------------
                ------------------------------------------
                Other statements in Program
                ------------------------------------------

---------------
Syntax2:
---------------
                for   varname   in Iterable_object:
                    ----------------------------------------
                    Indentation block of stmts
                    ----------------------------------------
                else:
                    ----------------------------------------
                    else block of statements
                    ----------------------------------------
                ------------------------------------------
                Other statements in Program
                ------------------------------------------
```

Explanation :

- Here 'for' and 'else' are keywords.
- Here Iterable_object can be Sequence(bytes, bytearray, range, str), list(list, tuple), set(set, frozenset) and dict.
- The execution process of for loop is that " Each of Element of Iterable_object selected, placed in varname and executes Indentation block of statements". This Process will be repeated until all elements of Iterable_object completed.
- After execution of Indentation block of statements, PVM executes else block of statements which are written under else block and later PVM executes Other statements in Program.
- Writing else block is optional.
- for loops else part runs if no break occurs.

In [1]:
```
# Without using loops Example
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 [1]:
```
# With loops Example
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 [2]:
```
# With loops Example
for i in range(1,4):
    j=i+5
    k=j+10
    print(i,j,k)
print("srk")
```

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

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

```
3 8 18
```

In [6]:
```
# print all the numbers between 10 to 50 (Both inclusive)
for i in range(10,51):
    print(i,end=" ")
```

```
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 3
6 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

In [7]:
```
# print Even numbers from descending order (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 [72]:
```
''' Write a python program to get all the even numbers for
a given range both the Inclusive.'''

# Ask the user to enter the start and stop value.
lb=int(input("Enter the lower bond value:"))
ub=int(input("Enter the upper bond value:"))

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

```
Enter the lower bond value:20
Enter the upper bond value:50
20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
```

In [73]:
```
''' WAPP that gives numbers between 1 to 100 which are
divisible by 5 and divisible by 7.'''
for i in range(1,101):
    if (i%5==0 and i%7==0):
        print(i,end=" ")
```

```
35 70
```

In [12]:
```
seq = [1,2,3,4,5]

# Iterating over the list
for i in seq:
    print(i)
```

```
1
2
3
4
5
```

In [17]:
```
seq = [1,2,3,4,5]

# Iterating over the list
for i in seq:
    print("srk")
```

```
srk
srk
srk
srk
srk
```

In [18]:
```
items = ["item1", "item2", "item3", "item4"]

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

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

In [21]:
```
# Sum of items in a given list.
lst = [1,3,5,7,9]

# Initialization
sum=0

for i in lst:
    sum=sum+i

print(sum)
```

```
25
```

In [24]:
```
# Sum of even numbers in between given range (Both inclusive).

#Ask the user to enter the start(lower bond) and end(upper bond) values.
lb=int(input("Enter the lower bond value:"))
ub=int(input("Enter the upper bond value:"))

# Initialization
sum=0

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

print("\nSum is",sum)
```

```
Enter the lower bond value:30
Enter the upper bond value:60
30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60
Sum is 720
```

In [25]:
```
# WAPP which will print the user difined multiplication table.

#Ask the user to enter the which multiplication table they want.
a = int(input("Enter multiplication table:"))

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

```
Enter multiplication table:8
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
```

In [31]:
```
# WAPP which will compute the factorial of a given numbers.

# Ask the user to enter the number.
a = int(input("Enter the number:"))

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

print("The factorial of",a,"is:",fact)
```

```
Enter the number:4
The factorial of 4 is: 24
```

In [32]:
```
# WAPP which will compute the factorial of a given numbers.

# Ask the user to enter the number.
a = int(input("Enter the number:"))

# If user enters -ve number.
if (a<0):
    print("{} is invalid input".format(a))

# If user enters +ve number.
else:
    fact=1
    for i in range(a,0,-1):
        fact=fact*i
    print("The factorial of",a,"is:",fact)
```

```
Enter the number:5
The factorial of 5 is: 120
```

In [1]:
```
# Program for finding Factors of a given number.

# Ask the user to Enter a Number to find its Factor.
n = int(input("Enter a Number to find its Factorial:"))

# If user enters -ve number.
if (n <= 0):
    print("{} is invalid input:".format(n))

# If user enters +ve number.
else:
    print("Factors of a given Number:{}".format(n))
    for i in range(1, (n // 2) + 1):
        if (n % i == 0):
            print("\t{}".format(i))
```

```
Enter a Number to find its Factorial:40
Factors of a given Number:40
        1
        2
        4
        5
        8
        10
        20
```

In [24]:
```
# WAPP which will add 10 to the given list of numbers and print the added list.

# User Given list.
l = [4,9,14,20,1]

# Creating Empty list.
e=[]
for i in l:
    # Adding 10 with each iterated element and assign in object "a".
    a=i+10
    # Adding each modified element to the  new list.
    e.append(a)

# print the modified list with increment of 10 with each elements in a list.
print(e)
```

```
[14, 19, 24, 30, 11]
```

In [16]:
```
# Program for finding First N Natural Nums sum, Squares Sum and Cubes sum.

# Ask the user to enter How Many Natural Nums Sum u want to find.
n=int(input("Enter How Many Natural Nums Sum u want to find:"))

# If user enters -ve number.
if(n<=0):
    print("{} is invalid input:".format(n))

# If user enters +ve number.
else:
    # Initialization
    s,ss,cs=0,0,0
    # print "Nat Nums  Squares  Cubes" display purpose.
    print("-"*50)
    print("\tNat Nums\tSquares\t\tCubes")
    print("-"*50)
    for i in range(1,n+1):
        s=s+i
        ss=ss+i**2
        cs=cs+i**3
        print("\t{}\t\t{}\t\t{}".format(i,i**2,i**3))
    else:
        print("-"*50)
        print("\t{}\t\t{}\t\t{}".format(s,ss,cs))
        print("-"*50)
```

```
Enter How Many Natural Nums Sum u want to find:5
--------------------------------------------------
        Nat Nums        Squares         Cubes
--------------------------------------------------
        1               1               1
        2               4               8
        3               9               27
        4               16              64
        5               25              125
--------------------------------------------------
        15              55              225
--------------------------------------------------
```

Inner or Nested for loop :

- The process of writing one loop in inside of another loop is called Inner or Nested Loop.
- The execution process of Inner loops is that "For Every Value of Outer Loop, inner loop executes repeatedly for finite number of times".
- Inner or Nested loop can be used with the following Syntaxes.

```
-------------------------------------------------------------
Syntax-1   for loop in for loop
-------------------------------------------------------------
      for Varname1 in Iterable_Object1:              # Outer Loop
            -------------------------------
            -------------------------------
            for Varname2 in Iterable_Object2:    # Inner Loop
                  ---------------------------------------
                  ---------------------------------------
            else:
                  ---------------------------------------
       else:
          -------------------------------------------------
```

In [22]:
```
for i in range(3):
    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
```

In [23]:
```
for i in range(2):
    for j in range(2):
        print("ds")
```

```
ds
ds
ds
ds
```

In [26]:
```
for i in [1,5]:
    a=14
    b=a+14
    for j in range(1,4):
        c=j+b
        print(c)
```

```
29
30
31
29
30
31
```

In [27]:
```
'''WAPP which will takes two list l1=[11,14,12,11] & l2=[10,10,12,9]
 and gives the output as l=[21,24,24,20] by using for loop.'''

l1=[11,14,12,11]
l2=[10,10,12,9]

# Taking empty list
l=[]
for i in range(0,4):
    # Adding l1 & l2 with each iterated element and assign in object "a".
    a=l1[i]+l2[i]

    # Adding each modified element to the new list.
    l.append(a)

# print the new list.
print(l)
```

```
[21, 24, 24, 20]
```

In [28]:
```
'''WAPP which will takes two list l1=[11,14,12,11] & l2=[10,10,12,9]
 and gives the output as l=[21,24,24,20] by using Nested for loop.'''

l1=[11,14,12,11]
l2=[10,10,12,9]

# Taking empty list
l=[]
for i in range(0,len(l1)):
    for j in range(0,len(l2)):

        # If "i" value is equal to "j" then add the elements.
        if (i==j):
            # Adding l1 & l2 with each iterated element and assign in object "a".
            a=l1[i]+l2[i]

            # Adding each modified element to the new list.
            l.append(a)

# print the new list.
print(l)
```

```
[21, 24, 24, 20]
```

Loops Termination

Break

- break is a key word.

- The purpose of break statement is that "To terminate the execution of loop logically when certain condition is satisfied and PVM control comes of corresponding loop and executes other statements in the program".
- when break statement takes place inside for loop or while loop then PVM will not execute corresponding else block (bcoz loop is not becoming False) but it executes other statements in the program.
- **If for loop gets executed without break then else will execite.**
- **If for loop gets executed with break then else will not execute.**

```
--------------------
=>Syntax1:
--------------------
                    for varname in Iterable_object:
                            ------------------------------
                            if (test cond-1):
                                    break
                            ------------------------------
                            ------------------------------
--------------------
=>Syntax2:
--------------------
                    while(Test Cond-1):
                            ------------------------------
                            if (test cond-2):
                                    break
                            ------------------------------
                            ------------------------------
```

In [41]:
```
numbers = [1,2,3,4,5]

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

```
1
2
3
```

In [43]:
```
numbers = [1,2,3,4,5]

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

```
3
```

In [44]:
```
numbers = [1,2,3,4,5]

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

```
3
```

In [46]:
```
numbers = [1,2,3,4,5]

for i in numbers:
    if i==3:
        break
        print(i)  # No Output because printing after break statement.
```

In [56]:
```
# Program for demonstrating break statement--for loop

s="PYTHON"
for v in s:
    print(v)
print("*"*30)
for v in s:
    if (v=="H"):
        break
    print(v)
```

```
P
Y
T
H
O
N
******************************
P
Y
T
```

In [57]:
```
# Program for demonstrating break statement--for loop

s="PYTHON"
for v in s:
    print(v)
print("*"*30)
for v in s:
    if (v=="H"):
        break

    # If for loop gets executed without break then else will execite.
    else:
        print(v)
```

```
P
Y
T
H
O
N
******************************
P
Y
T
```

In [75]:
```
# Program for demonstrating break statement--for loop

s="PYTHON"
for v in s:
    print(v)
print("*"*30)
for v in s:
    if (v=="H"):
        break

    # If for loop gets executed without break then else will execite.
    else:
        print("-"*30)

# If for loop gets executed with break then else will not execite.
else:
    print(v)
```

```
P
Y
T
H
O
N
******************************
------------------------------
------------------------------
------------------------------
```

In [59]:
```
#Program for demonstrating break statement--for loop

# User given list.
lst=[10,"Rossum",34.56,True,2+3j,False,"Python"]

for val in lst:
    # printing the each elements within the list.
    print("\t{}".format(val))

# Here else block will executes, because there is no any break statement.
else:
    print("-----------------------------------------------")

for val in lst:
    # If "val==2+3j" then break.
    if(val==2+3j):
        break
    # If for loop gets executed without break then else will execite.
    else:
        print("\t{}".format(val))

# If for loop gets executed with break then else will not execite.
else:
    print("=========================")
```

```
        10
        Rossum
        34.56
        True
        (2+3j)
        False
        Python
-----------------------------------------------
        10
        Rossum
        34.56
        True
```

In [66]:
```
# Program for deciding whether the given number is prime or not.

# To take input from the user.
n=int(input("Enter a number:"))

# If user enters less than or equal to 1, then print the invalid input.
if(n<=1):
    print("{} is invalid input:".format(n))

# If user enters greater than 1, then take the input.
else:
    # define a dec variable.
    dec="PRIME"

    # check for factors.
    for i in range(2,n):
        # If n is divided by i, then break out the loop with "NOTPRIME".
        if(n%i==0):
            dec="NOTPRIME"
            break

    # check if dec is True, then print "n is prime number".
    if(dec=="PRIME"):
        print("{} is PRIME NUMBER:".format(n))

    # check if dec is False, then print "n is Not Prime number".
    else:
        print("{} is NOT PRIME NUMBER:".format(n))
```

```
Enter a number:19
19 is PRIME NUMBER:
```

Continue

- continue is a keyword.
- continue statement is used for making the PVM to go to the top of the loop without executing the following statements which are written after continue statement for that current Iteration only.
- continue statement to be used always inside of loops.
- when we use continue statement inside of loop then else part of corresponding loop also executes provided loop condition becomes false.

```
-----------------
=>Syntax: -
----------------
      for varname   in Iterable-object:
            ----------------------------------------
            if (Test Cond):
                  continue
            statement-1  # written after continue statement
            statement-2
            statement-n
            ----------------------------------------
            ----------------------------------------
```

In [70]:
```
numbers = [1,2,3,4,5,6,7]

for num in numbers:
    if (num==4):
        continue
    print(num)
```

```
1
2
3
5
6
7
```

In [71]:
```
#Program for demonstrating continue statement

# To take input from user.
s="PYTHON"

# for loop without continue statement.
for v in s:
    print("\t{}".format(v))
else:
    print("---------------------------------------")

# for loop with continue statement.
for v in s:
    if(v=="H"):
        continue
    else:
        print("\t{}".format(v))  # else block will be executes.
```

```
        P
        Y
        T
        H
        O
        N
---------------------------------------
        P
        Y
        T
        O
        N
```