Looping Statements in Python

In [ ]:
```
Looping Statements in Python

When working with loops in Python, a programmer must remember three important parts

1. Initialization Part
        •       This tells us where to start.
Example: i = 1

2. Conditional Part
        •       This tells us where to stop.
Example: i <= 10

3. Updation Part
        •       This handles increment or decrement (forward or backward movement).
Example: i = i + 1

_____

Example: Generate numbers from 1 to 10
1
2
3
4
5
6
7
8
9
10
```

In [3]:
```
i = 1  # Initialization
while i <= 10: # Condition
    print(i)
    i = i + 1  # Updation
```

```
1
2
3
4
5
6
7
8
9
10
```

In [17]:
```
i=10
while i<=20:
    print(i)
    i=i+1
```

```
10
11
12
13
14
15
16
17
18
19
20
```

In [15]:
```
i=20
while i<=30:
    print(i)
    i=i+1
```

```
20
21
22
23
24
25
26
27
28
29
30
```

I want to generate 5 to 1

In [20]:
```
i=5
while i>=1:
    print(i)
    i=i-1
```

```
5
4
3
2
1
```

In [26]:
```
i=10 # initialization
while i>=5: # Conditon
    print(i)
    i=i-1 # Updation
```

```
10
9
8
7
6
5
```

2. Looping (Iterative or Repetitive) Statements

The purpose of Looping or Iterative or Repetitive Statements is to perform a certain operation or task repeatedly for a finite number of times until the test condition becomes false.

In Python programming, we have two types of looping statements:

1. while loop or while…else loop
2. for loop or for…else loop

1. While loop or While...else loop

while(Test_Cond):

→ Indentation

→ Statement-1

→ Statement-2

→ Statement-n Block of Statements

else: → Else Block of Statements

Other statements in Program

Explanation • while and else are Python keywords used in looping.

- When the Test Condition is True, the Python Virtual Machine (PVM) executes the indented block of statements.
- After completing the block, the PVM checks the Test Condition again.
- This cycle continues (repeats) a finite number of times until the condition becomes False.
- When the Test Condition becomes False, the PVM:

1. Executes the else block (if it is written), and
2. Then executes the remaining statements in the program.

Program for generating 1 to N number where n is +ve:

In [36]:
```
n=int(input("Enter How numbers you want generate:"))
if(n<=0):
    print("{} is Invalid input".format(n))
else:
```

```
    i=1 # Initialization part
    while(i<=n): # Conditional part
        print(i)
        i=i+1
```

```
-20 is Invalid input
```

In [40]:
```
n=int(input("Enter How numbers you want generate:"))
if(n<=0):
    print("{} is Invalid input".format(n))
else:
    i=1 # Initialization part
    while(i<=n): # Conditional part
        print(i)
        i=i+1
```

```
1
2
3
4
5
```

In [44]:
```
n=int(input("Enter How numbers you want generate:"))
if(n<=0):
    print("{} is Invalid input".format(n))
else:
    i=1 # Initialization part
    while(i<=n): # Conditional part
        print(i)
        i=i+1
```

```
0 is Invalid input
```

Program for generating N to 1 number where N is +ve

In [4]:
```
n=int(input("Enter How numbers you want to generate :"))
if(n<=0):
    print("{} Invalid input:".format(n))
else:
    i=n
    while(i>=1):
        print(i)
        i=i+1
```

```
-5 Invalid input:
```

In [6]:
```
n=int(input("Enter How numbers you want to generate :"))
if(n<=0):
    print("{} Invalid input:".format(n))
else:
    i=n
    while(i>=1):
        print(i)
        i=i+1
```

```
0 Invalid input:
```

In [2]:
```
n=int(input("Enter How numbers you want to generate :"))
if(n<=0):
    print("{} Invalid input:".format(n))
else:
    i=n
    while(i>=1):
        print(i)
        i=i-1
```

```
5
4
3
2
1
```

In [4]:
```
n=int(input("Enter How numbers you want to generate :"))
if(n<=0):
    print("{} Invalid input:".format(n))
else:
    i=n
    while(n>=1):
        print(n)
        n=n-1
```

```
7
6
5
4
3
2
1
```

Program for generating Mul table for a given +ve Number

In [25]:
```
n=int(input("Enter a number for generating Mul table :"))
if (n<=0):
    print("{} is Invalid input :".format(n))
else:
    print("-" * 50)
    print("Mul table for {}".format(n))
    print("-" * 50)
    i=1
    while(i<=10):
        print("\t{} x {}={}".format(n,i, n*i))
        i=i+1
    else:
        print("-" * 50)
```

```
-5 is Invalid input :
```

In [23]:
```
n=int(input("Enter a number for generating Mul table :"))
if (n<=0):
    print("{} is Invalid input :".format(n))
else:
    print("-" * 50)
    print("Mul table for {}".format(n))
```

```
    print("-" * 50)
    i=1
    while(i<=10):
        print("\t{} x {}={}".format(n,i, n*i))
        i=i+1
    else:
        print("-" * 50)
```

```
0 is Invalid input :
```

In [21]:
```
n=int(input("Enter a number for generating Mul table :"))
if (n<=0):
    print("{} is Invalid input :".format(n))
else:
    print("-" * 50)
    print("Mul table for {}".format(n))
    print("-" * 50)
    i=1
    while(i<=10):
        print("\t{} x {}={}".format(n,i, n*i))
        i=i+1
    else:
        print("-" * 50)
```

```
--------------------------------------------------
Mul table for 12
--------------------------------------------------
        12 x 1=12
        12 x 2=24
        12 x 3=36
        12 x 4=48
        12 x 5=60
        12 x 6=72
        12 x 7=84
        12 x 8=96
        12 x 9=108
        12 x 10=120
--------------------------------------------------
```

In [27]:
```
n=int(input("Enter a number for generating Mul table :"))
if (n<=0):
    print("{} is Invalid input :".format(n))
else:
    print("-" * 50)
    print("Mul table for {}".format(n))
    print("-" * 50)
    i=1
    while(i<=10):
        print("\t{} x {}={}".format(n,i, n*i))
        i=i+1
    else:
        print("-" * 50)
```

```
--------------------------------------------------
Mul table for 10
--------------------------------------------------
	10 x 1=10
	10 x 2=20
	10 x 3=30
	10 x 4=40
	10 x 5=50
	10 x 6=60
	10 x 7=70
	10 x 8=80
	10 x 9=90
	10 x 10=100
--------------------------------------------------
```

In [29]:
```
n=int(input("Enter a number for generating Mul table :"))
if (n<=0):
    print("{} is Invalid input :".format(n))
else:
    print("-" * 50)
    print("Mul table for {}".format(n))
    print("-" * 50)
    i=1
    while(i<=10):
        print("\t{} x {}={}".format(n,i, n*i))
        i=i+1
    else:
        print("-" * 50)
```

```
--------------------------------------------------
Mul table for 11
--------------------------------------------------
	11 x 1=11
	11 x 2=22
	11 x 3=33
	11 x 4=44
	11 x 5=55
	11 x 6=66
	11 x 7=77
	11 x 8=88
	11 x 9=99
	11 x 10=110
--------------------------------------------------
```

In [ ]:
```
'''THE SESSION IS COMPLETED WILL COTINEU IN NEXT SESSION

Regards,
MAHABOOB KHAN'''
```