CHAPTER-4
FLOW OF CONTROL

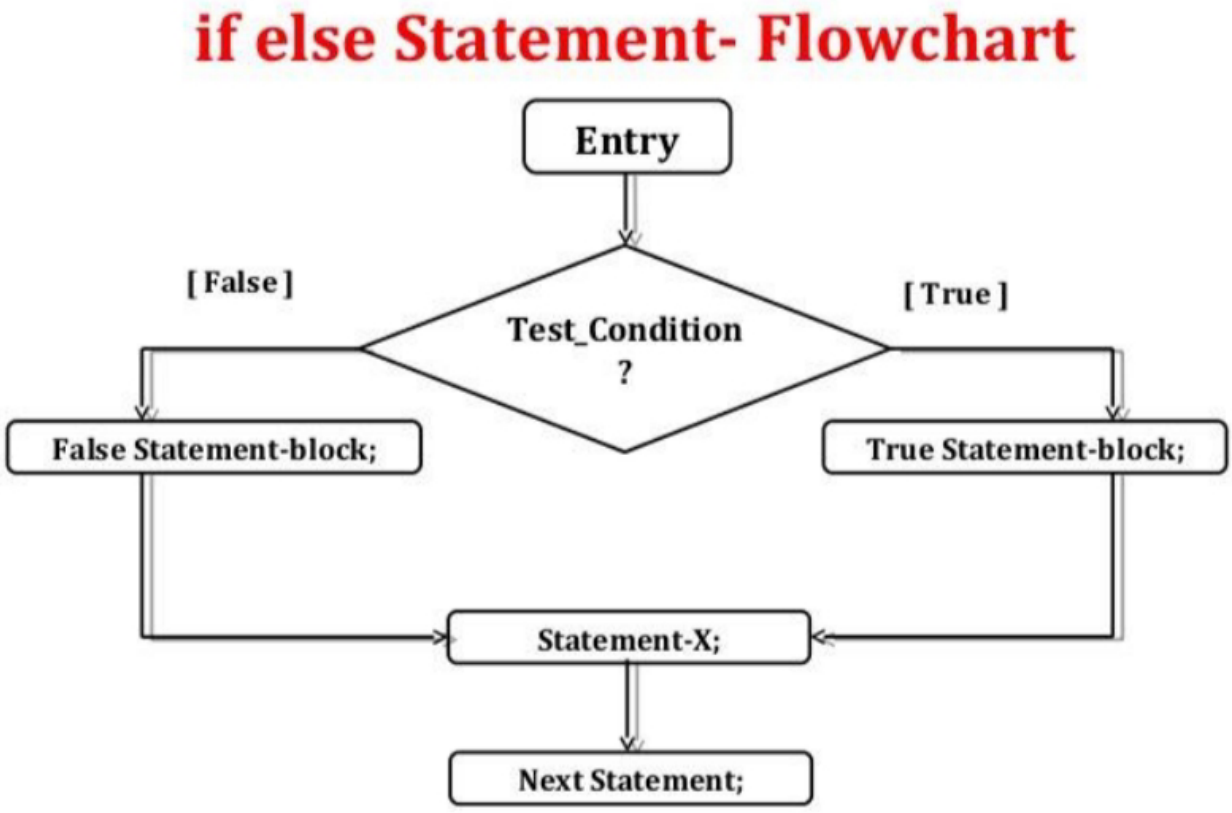
1. Decision Making and branching (Conditional Statement)

2. Looping or Iteration

3. Jumping statements

4.1 DECISION MAKING & BRANCHING

Decision making is about deciding the order of execution of statements based on certain conditions. Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome.



There are three types of conditions in python:

1. if statement

2. if-else statement

3. elif statement

1. **if statement:** It is a simple if statement. When condition is true, then code which is associated with if statement will execute.

Example:

```
a=40
b=20
if  a>b:
    print(“a is greater than b”)
```

2. if-else statement: When the condition is true, then code associated with if statement will execute, otherwise code associated with else statement will execute.

Example:

```
a=10
b=20
if a>b:
    print(“a is greater”)
else:
    print(“b is greater”)
```

3. elif statement: It is short form of else-if statement. If the previous conditions were not true, then do this condition". It is also known as nested if statement.

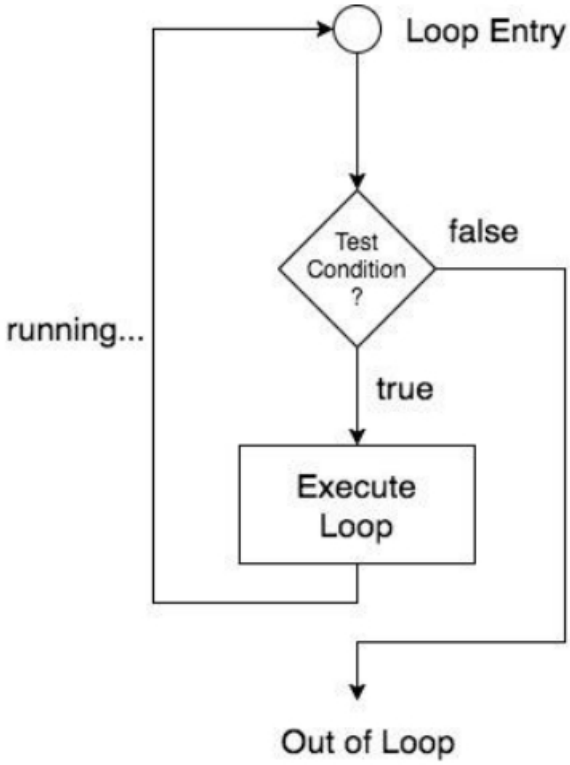
Example:

```
a=input(“Enter first number”)
b=input("Enter Second Number:")
if a>b:
```

```
    print("a is greater")
elif a==b:
    print("both numbers are equal")
else:
    print("b is greater")
```

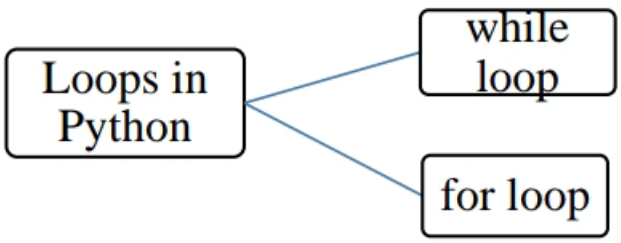
4.2 LOOPS in PYTHON

Loop: Execute a set of statements repeatedly until a particular condition is satisfied.



There are two types of loops in python:

1. while loop
2. for loop



1. **while loop:** With the while loop we can execute a set of statements as long as a condition is true. It requires to define an indexing variable.

Example: To print table of number 2

```
i=2
while i<=20:
    print(i)
    i+=2
```

2. **for loop** : The for loop iterate over a given sequence (it may be list, tuple or string).

Note: The for loop does not require an indexing variable to set beforehand, as the for command itself allows for this.

```
primes = [2, 3, 5, 7]
for x in primes:
  print(x)
```

The range( ) function:

it generates a list of numbers, which is generally used to iterate over with for loop. range( ) function uses three types of parameters, which are:

- start: Starting number of the sequence.
- stop: Generate numbers up to, but not including last number.
- step: Difference between each number in the sequence.

Python use range( ) function in three ways:

a. range(stop)

b. range(start, stop)

c. range(start, stop, step)

Note:

- All parameters must be integers.
- All parameters can be positive or negative.

1. **while loop:** With the while loop we can execute a set of statements as long as a condition is true. It requires to define an indexing variable.

Example: To print table of number 2

```
i=2
while i<=20:
    print(i)
    i+=2
```

2. **for loop** : The for loop iterate over a given sequence (it may be list, tuple or string).

Note: The for loop does not require an indexing variable to set beforehand, as the for command itself allows for this.

```
primes = [2, 3, 5, 7]
for x in primes:
  print(x)
```

The range( ) function:

it generates a list of numbers, which is generally used to iterate over with for loop. range( ) function uses three types of parameters, which are:

- start: Starting number of the sequence.
- stop: Generate numbers up to, but not including last number.
- step: Difference between each number in the sequence.

Python use range( ) function in three ways:

a. range(stop)

b. range(start, stop)

c. range(start, stop, step)

Note:

- All parameters must be integers.
- All parameters can be positive or negative.

a. range(stop): By default, It starts from 0 and increments by 1 and ends up to stop, but not including **stop** value.

Example:

```
for x in range(4):
  print(x)
```

Output:

```
0
1
2
3
```

b. range(start, stop): It starts from the **start** value and up to stop, but not including stop value.

Example:

```
for x in range(2, 6):
  print(x)
```

Output:

```
2
3
4
5
```

c. range(start, stop, step): Third parameter specifies to increment or decrement the value by adding or subtracting the value.

Example:

```
for x in range(3, 8, 2):

    print(x)
```

Output:

```
3
5
7
```

Explanation of output: 3 is starting value, 8 is stop value and 2 is step value. First print 3 and increase it by 2, that is 5, again increase is by 2, that is 7. The output can't exceed stop-1 value that is 8 here. So, the output is 3, 5, 8.

Difference between range( ) and xrange( ):

S. No.	range( )	xrange( )
1	returns the list of numbers	returns the generator object that can be used to display numbers only by looping
2	The variable storing the range takes more memory	variable storing the range takes less memory
3	all the operations that can be applied on the list can be used on it	operations associated to list cannot be applied on it
4	slow implementation	faster implementation

4.3 JUMP STATEMENTS:

There are two jump statements in python:

1. break
2. continue

1. **break statement** : With the break statement we can stop the loop even if it is true.

Example:

in while loop	in for loop

`i = 1`
`while i < 6:`
`  print(i)`
`  if i == 3:`
`  break`
`  i += 1`
	`languages = ["java", "python", "c++"]`
	`for x in languages:`
	`  if x == "python":`
	`  break`
	`  print(x)`

Output: 1 2 3	**Output:** java

Note: If the **break** statement appears in a nested loop, then it will terminate the very loop it is in i.e. if the **break** statement is inside the inner loop then it will terminate the inner loop only and the outer loop will continue as it is.

2. continue statement : With the continue statement we can stop the current iteration, and continue with the next iteration.

Example:

in while loop	in for loop

`i = 0`
`while i < 6:`
`  i += 1`
`  if i == 3:`
`  continue`
`  print(i)`
	`languages = ["java", "python", "c++"]`
	`for x in languages:`
	`  if x == "python":`
	`  continue`
	`  print(x)`

Output: 1 2 4 5 6	**Output:** java c++

4.4 Loop else statement:

The **else** statement of a python loop executes when the loop terminates normally. The else statement of the loop will not execute when the **break** statement terminates the loop.

The else clause of a loop appears at the same indentation as that of the loop keyword **while** or **for**.

Syntax:

for loop

```
for <variable> in <sequence>:
    statement-1
    statement-2
    .
    .
else:
    statement(s)
```

while loop

```
while <test condition>:
    statement-1
    statement-2
    .
    .
else:
    statement(s)
```

4.5 Nested Loop :

A loop inside another loop is known as nested loop.

Syntax:

```
for <variable-name> in <sequence>:
    for <variable-name> in <sequence>:
        statement(s)
    statement(s)
```

Example:

```
for i in range(1,4):
    for j in range(1,i):
        print("*", end=" ")
    print(" ")
```

Programs related to Conditional, looping and jumping statements

1. Write a program to check a number whether it is even or odd.

```
num=int(input("Enter the number: "))
if num%2==0:
   print(num, " is even number")
else:
   print(num, " is odd number")
```

2. Write a program in python to check a number whether it is prime or not.

```
num=int(input("Enter the number: "))
for i in range(2,num):
   if num%i==0:
      print(num, "is not prime number")
      break;
else:
   print(num,"is prime number")
```

3. Write a program to check a year whether it is leap year or not.

```
year=int(input("Enter the year: "))
if year%100==0 and year%400==0:
   print("It is a leap year")
elif year%4==0:
   print("It is a leap year")
else:
   print("It is not leap year")
```

4. Write a program in python to convert °C to °F and vice versa.

```
a=int(input("Press 1 for C to F \n Press 2 for F to C \n"))
if a==1:
        c=float(input("Enter the temperature in degree celcius:  "))
        f= (9/5)*c+32
        print(c, "Celcius = ",f," Fahrenheit")
elif a==2:
        f=float(input("Enter the temperature in Fahrenheit:  "))
        c= (f-32)*5/9
        print(f, "Fahrenheit = ",c," Celcius")
else:
        print("You entered wrong choice")
```

5. Write a program to check a number whether it is palindrome or not.

```
num=int(input("Enter a number : "))
n=num
res=0
while num>0:
   rem=num%10
```

```
    res=rem+res*10
    num=num//10
if res==n:
    print("Number is Palindrome")
else:
    print("Number is not Palindrome")
```

6. A number is Armstrong number or not.

```
num=input("Enter a number : ")
length=len(num)
n=int(num)
num=n
sum=0
while n>0:
    rem=n%10
    sum=sum+rem**length
    n=n//10
if num==sum:
    print(num, "is armstrong number")
else:
    print(num, "is not armstrong number")
```

7. To check whether the number is perfect number or not

```
num=int(input("Enter a number : "))
sum=0
for i in range(1,num):
    if(num%i==0):
        sum=sum+i
```

```
if num==sum:
    print(num, "is perfect number")
else:
    print(num, "is not perfect number")
```

8. Write a program to print Fibonacci series.

```
n=int(input("How many numbers : "))
first=0
second=1
i=3
print(first, second, end=" ")
while i<=n:
    third=first+second
    print(third, end=" ")
    first=second
    second=third
    i=i+1
```

9. To print a pattern using nested loops

```
for i in range(1,5):
    for j in range(1,i+1):
        print(j," ", end=" ")
    print('\n')
```

```
1
1 2
1 2 3
1 2 3 4
```