❖ Python Operators:

- Operators are used to perform operations on variables and values.
- Operators are the special symbols which indicate operation to be perform.
- Following are the types of operators present in Python.
  1. Arithmetic Operators
  2. Comparison Operators
  3. Logical Operators
  4. Bitwise Operators
  5. Assignment Operators
  6. Membership Operators
  7. Identity Operators

1. Arithmetic Operators:

- Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.
- Following are the list of Arithmetic operators present in Python

Operator	Meaning
+	Addition
-	Substraction
*	Multiplication
/	Division
//	Floor Division
%	Modulus – remainder calculator
**	Exponent

- Example:

```
x = 15
y = 4

print("x + y =",x+y)

print("x - y =",x-y)

print("x * y =",x*y)

print("x / y =",x/y)

print("x // y =",x//y)

print("x ** y =",x**y)
```

When you run the above program, the output will be:

```
x + y = 19
x - y = 11
x * y = 60
x / y = 3.75
x // y = 3
x ** y = 50625
```

2. Comparision Operators:

- Comparison operators are used to compare values.
- It give result either True or False according to the condition.
- Following are the list of comparision operators present in Python

Operator	Meaning
<	Less Than
>	Greater Than
<=	Less Than or Equal To
>=	Greater Than or Equal To
==	Equal To
!=	Not Equal To

- Example:

```
x = 10
y = 12

print("x > y  is", x>y)

print("x < y  is", x<y)

print("x == y is", x==y)

print("x != y is", x!=y)

print("x >= y is", x>=y)

print("x <= y is", x<=y)
```

When you run the above program, the output will be:

```
x > y  is False
x < y  is True
x == y is False
```

```
x != y is True
x >= y is False
x <= y is True
```

3. Logical Operators:

- Logical operators are the **and, or, not** operators.
- It give result either True or False according to the condition.
- Following are the list of Logical operators present in Python

Operator	Meaning
and	It returns True if its both sides are True otherwise False
or	It returns False if its both sides are False otherwise True
not	Reverse the Result, returns False if the result is True.

- Example:

```
print("10>5 and 10==10 Result is ",10>5 and 10==10)

print("150<50 or 15!=15 Result is",150<50 or 15!=15)

print("not(12>5) Result is",not(12>5))
```

When you run the above program, the output will be:

```
10>5 and 10==10 Result is  True
150<50 or 15!=15 Result is False
not(12>5) Result is False
```

4. Bitwise Operators:

- Bitwise operators are used to work with binary numbers
- Bitwise operator works on bits and performs bit by bit operation.
- Following are the list of Bitwise operators present in Python

Operator	Meaning	Description
&	Bitwise AND	If both bits are 1 then output is 1 otherwise 0.
|	Bitwise OR	If both bits are 0 then output is 0 otherwise 1.
^	Bitwise XOR	If both bits are different then output is 1 otherwise 0
~	Bitwise NOT	Inverts all the bits
<<	Bitwise left Shift	Shift left by pushing zeros in from the right side and let the leftmost bits fall off
>>	Bitwise right shift	Shift right by pushing zeros in from the left side and let the rightmost bits fall off

- Example:

```
x=10
y=4
print("x&y Result is", x&y);
print("x|y Result is", x|y);
print("x^y Result is", x^y);
print("~x Result is", ~x);
print("x<<2 Result is", x<<2);
print("x>>2 Result is", x>>2);
```

When you run the above program, the output will be:

```
x&y Result is 0
x|y Result is 14
x^y Result is 14
~x Result is -11
x<<2 Result is 40
x>>2 Result is 2
```

5. Assignment Operators:

- Assignment operator is used to assign right side expression result or value to the left side variable.
- Assignment operators are used in Python to assign values to variables.
- Following are the Assignment operators in Python:

Operator	Example	Same as
=	X=5	X=5
+=	X+=5	X=X+5
-=	X-=5	X=X-5
*=	X*=5	X=X*5
/=	X/=5	X=X/5
//=	X//=5	X=X//5
**=	X**=5	X=X**5
&=	X&=5	X=X&5
|=	X|=5	X=X|5
^=	X^=5	X=X^5
<<=	X<<=5	X=X<<5
>>=	X>>=5	X=X>>5

6. Membership operators:

- in and not in are the membership operators in Python.
- They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).
- In a dictionary we can only test for presence of key, not the value.
- Following are the Membership operators in Python:

Operator	Meaning
in	True if value/variable is found in the sequence
not in	True if value/variable is not found in the sequence

- Example:

```
x=[10,20,30,40,50]

print("10 in x =>", 10 in x);
print("10 not in x =>", 10 not in x);
print("120 in x =>", 120 in x);
print("120 not in x =>", 120 not in x);
```

When you run the above program, the output will be:

```
10 in x => True
10 not in x => False
120 in x => False
120 not in x => True
```

7. Identity operators:

- **is** and **is not** are the identity operators in Python.
- Identity operators are used to compare the objects/variables.
- They are used to check if two values (or variables) are located on the same part of the memory.
- Following are the identity operators in Python:

Operator	Meaning
is	True if operands are indentical
is not	True if operands are not identical

- Example:

```
x=10;
y=10;
```

```
z=20;

print("x is y =>", x is y);
print("x is not y =>", x is not y);
print("x is not z =>", x is not z);
print("x is z =>", x is z);
```

When you run the above program, the output will be:

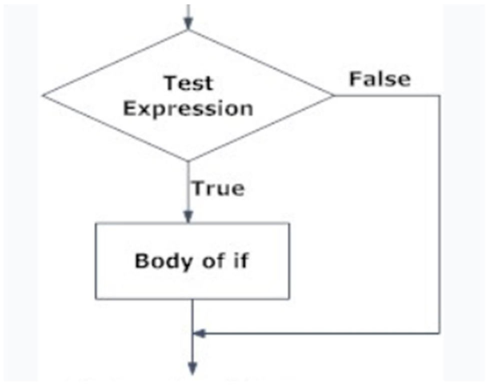
```
x is y => True
x is not y => False
x is not z => True
x is z => False
```

❖ if Statement:

- if statement is one of the types of Decision making statement.
- Decision making statement is required when we want to execute a code only if a certain condition is satisfied.
- **if** is a predefined keyword which should be written in small letters.
- Syntx:

```
if condition:
   statement-1
   statement-2
   ---
   ---
   statement-n
```

- Program controller first test the condition, if condition is True then program controller executes the body of if statement.
- If condition is False then program controller will not executes the body of if statements.
- In Python programming language, any non-zero and non-null values are assumed as TRUE, and if it is either zero or null, then it is assumed as FALSE value.
- if statement flowchart:



- Example:

```
a=int(input("Enter first Number:"))
b=int(input("Enter second Number:"))
if a==b:
   print("Both Numbers are Equals");
```

When you run the above program, the output will be:

```
Enter first Number:15
Enter second Number:15
Both Numbers are Equals
```

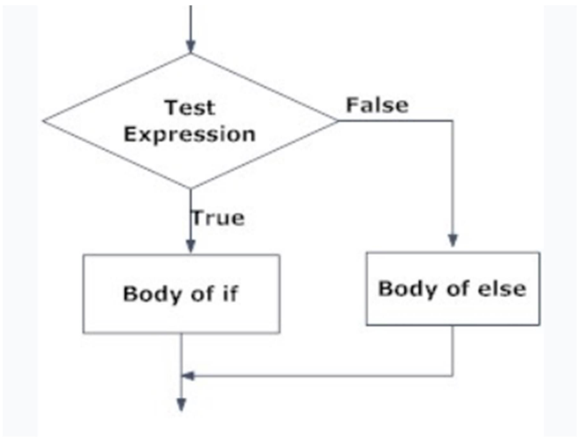
- In the above example, a==b is the condition. The body of if is executed only if condition a==b is True otherwise it will not executes the body of if statement.

❖ if-else Statement:

- if-else statement is one of the types of Decision making statement.
- **if** and **else** both are predefined keywords which should be written in small letters.
- Syntx:

```
if condition:
   statement-1
   ---
   ---
else:
   statement-1
   ---
   ---
```

- Program controller first test the condition, if condition is True then program controller executes the body of if statement.
- If condition is False then program controller executes the body of else statements.
- In Python programming language, any non-zero and non-null values are assumed as TRUE, and if it is either zero or null, then it is assumed as FALSE value.
- if-else statement flowchart:



- Example:

```
a=int(input("Enter first Number:"))
b=int(input("Enter second Number:"))
if a==b:
   print("Both Numbers are Equals");
else:
   print("Both Numbers are not Equals");
```

When you run the above program, the output will be:

```
Enter first Number:15
Enter second Number:15
Both Numbers are Equals
```

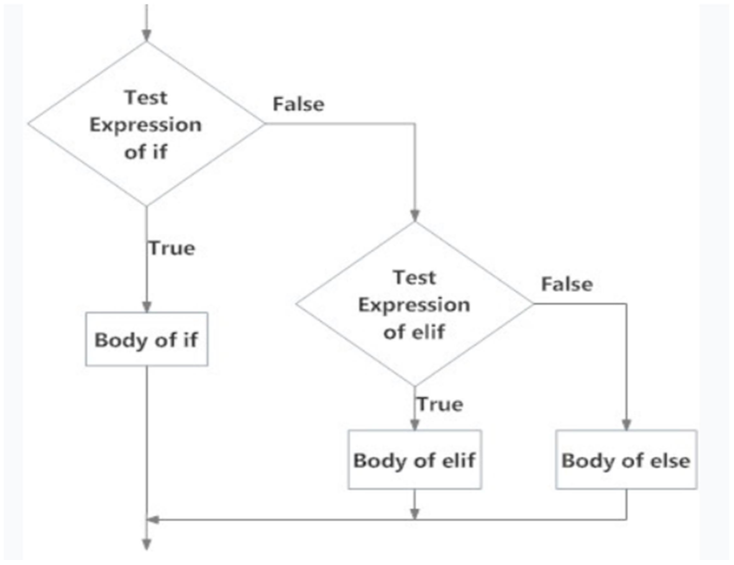
- In the above example, a==b is the condition. The body of if is executed only if condition a==b is True otherwise it will executes the body of else statement.

❖ if-elif-else Statement:

- if-elif-else statement is one of the types of Decision making statement.
- **if, elif** and **else** all are predefined keywords which should be written in small letters.
- Syntx:

```
if condition-1:
   statements
   ---
   ---
elif condition-2:
   statements
   ---
   ---
elif condition-3:
   statements
   ---
   ---
else:
   statements
   ---
```

- Program controller first test the condition-1, if condition-1 is True then program controller executes the body of if statement.
- If condition-1 is false then program controller test the condition-2, if condition-2 is True then it will executes the body of if statement.
- If all conditions are False then program controller executes else block statements.
- if-elif-else statement flowchart:



- Example:

```
marks=int(input("Enter your Marks:"))
if marks>=75:
   print("You got Distinction");
elif marks>=60:
   print("You got First Class");
elif marks>=40:
   print("You are Pass only");
else:
   print("You are Fail");
```

When you run the above program, the output will be:

```
Enter your Marks:89
You got Distinction
```

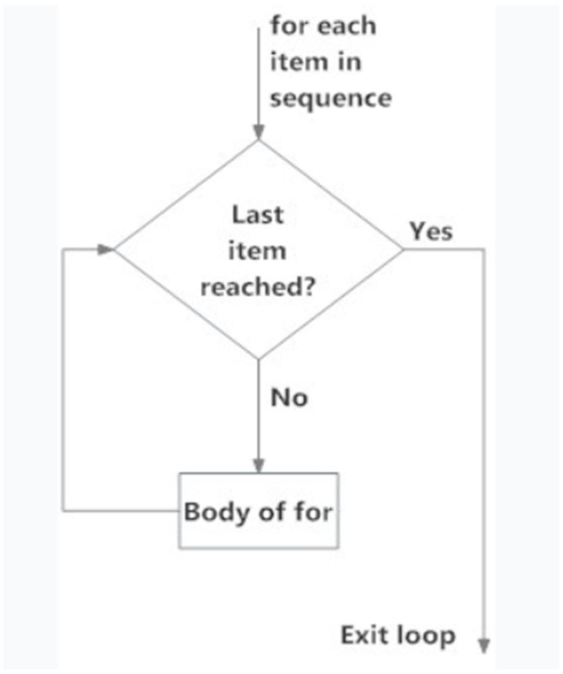
- In the above example, When variable marks value is 89, then ‘`You got Distinction`’ is printed. When variable marks value is 66, then ‘`You got First Class`’ is printed. When variable marks value is 45, then ‘`You are Pass only`’ is printed. When variable marks value is 35, then ‘`You are Fail`’ is printed.

❖ For Loop Statement:

- for loop statement is one of the types of looping statement in Python.
- **for** is a predefined keyword which should be written in small letters.
- If we want to run the block of code N no of times then we can use the looping statements.
- 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.
- Syntx:

```
for val in sequence:
    Body of for loop
    -------
    -------
    -------
```

- In above syntax, val is the variable that takes the value of the item inside the sequence on each iteration.
- Program controller executes the for loop until it reach the last item in the sequence.
- for loop statement flowchart:



- Example:

```
# Program to find the sum of all numbers in a list

numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

sum = 0
for val in numbers:
  sum = sum+val
```

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

When you run the above program, the output will be:

```
The sum is 48
```

- In the above example, When variable marks value is 89, then 'You got Distinction' is printed. When variable marks value is 66, then 'You got First Class' is printed. When variable marks value is 45, then 'You are Pass only' is printed. When variable marks value is 35, then 'You are Fail' is printed.

The range() function:

- We can generate a sequence of numbers using range() function. For example, range(10) will generate numbers from 0 to 9.
- We can also define the start, stop and step size as range(start,stop,Increment size). Increment size defaults to 1 if not provided.
- This function does not store all the values in computer memory. So it remembers the start, stop, step size and generates the next number.
- We can use the range() function in for loop to iterate through a sequence of numbers.
- Example:

```
# Program to display 'VJTech Academy'message on output screen 5 times

for i in range(1,6):
    print("VJTech Academy")
```

When you run the above program, the output will be:

```
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
```

for loop with else keyword:

- A for loop can have an optional else block as well.
- The else part is executed if all items in the sequence are finished.
- When we use break statement to stop a for loop then program controller ignore the else part execution.
- In for loop, else part runs if break statement is not executed inside the body.
- Example:

```
# Program to display ‘VJTech Academy’message on output screen 5 times

for i in range(1,6):
    print("VJTech Academy")
else:
    print("End of the program")
```

When you run the above program, the output will be:

```
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
End of the program
```

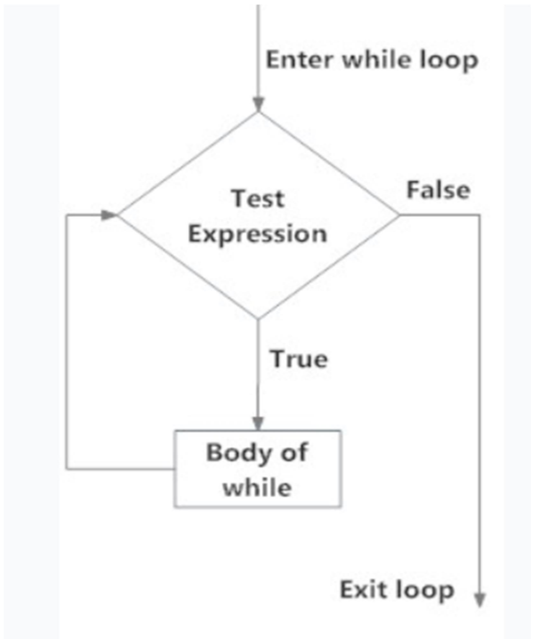
- In above example, the for loop prints message on output screen until the loop ends. When the for loop ends, it executes the block of code in the else statement.

❖ **While Loop Statement:**

- while loop statement is one of the types of looping statement in Python.
- **while** is a predefined keyword which should be written in small letters.
- If we want to run the block of code N no of times then we can use the looping statements.
- In while loop statement, program controller first test the condition if condition if True then it will execute the body of while loop.
- Again, it will goes to test the condition if condition is again True then it will execute the body of while loop.
- This process continue until while loop condition become False.
- Syntx:

```
while condition:
    Body of while loop
    -------
    -------
    -------
```

- In above syntax, program controller executes the while loop body until condition is True.
- While loop statement flowchart:



- Example:

```
# Program to display ‘VJTech Academy’message on output screen 5 times
i=1
while i<=5:
    print("VJTech Academy")
    i=1+1
```

When you run the above program, the output will be:

```
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
```

- In the above example, program controller print ‘VJTech Academy’ message on output screen 5 times. It will stop the execution of while loop when i value become 6.

while loop with else keyword:

- A while loop can have an optional else block as well.
- The else part is executed when while loop condition is False.
- When we use break statement to stop a while loop then program controller ignore the else part execution.
- In while loop, else part runs if break statement is not executed inside the body.
- Example:

```
# Program to display ‘VJTech Academy’message on output screen 5 times
i=1
while i<=5:
    print("VJTech Academy")
else:
    print("End of the program")
```

When you run the above program, the output will be:

```
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
VJTech Academy
End of the program
```

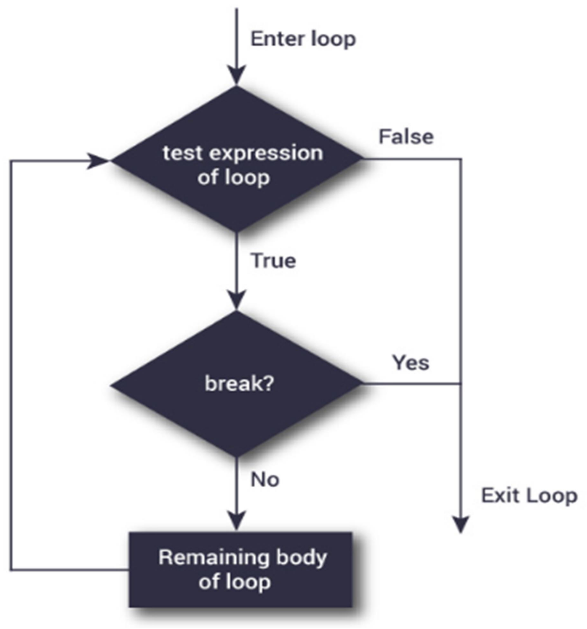
- In above example, the while loop print message on output screen until the loop ends. When the while loop condition is false then it executes the block of code in the else statement.

❖ Break keyword:

- **break** is a predefined keyword in Python which should be written in small letters.
- We can use break keyword inside the body of looping statements.
- Normally, program controller executes the body of loop unil condition becomes false. But sometimes their could be situation where we want to terminate the execution of the loop then we can use the break keyword inside the body of loop.
- When break statement is executed inside the body of loop then program controller terminates the execution of loop immediately and it will comes out of loop.
- Syntax:

```
break
```

- Flowchart of break:



- Example:

```
# Program to show the use of break keyword
for i in range(1,6):
    if i==3:
        break
    print("Value of i=",i)
```

When you run the above program, the output will be:

```
Value of i= 1
Value of i= 2
```

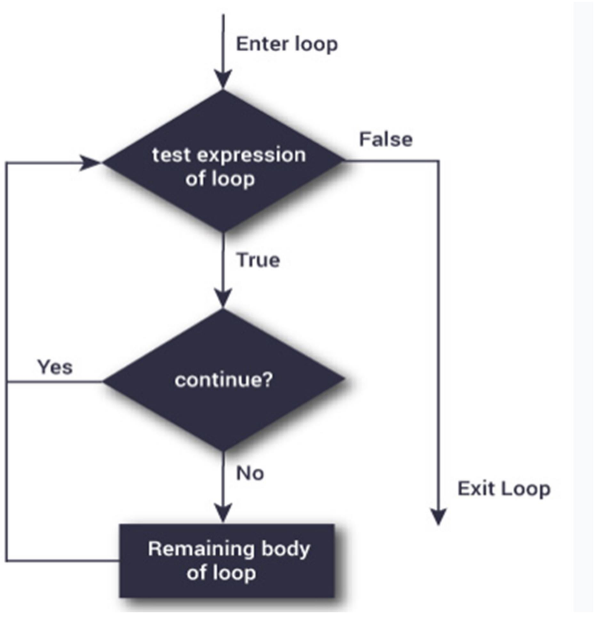
- In above program, we can see for loop will iterate through the range(1,6). We check that value of i is 3 if it is 3 then break will executed. Hence, we see in our output that value of i is printed up to 2. After that, the loop is terminated.

❖ Continue keyword:

- **continue** is a predefined keyword in Python which should be written in small letters.
- We can use continue keyword inside the body of looping statements.
- When continue statement is executed inside the body of loop then program controller goes for the next iteration and skip the execution of rest of the statements.
- Loop does not terminate but continues on with the next iteration.
- Syntax:

```
continue
```

- Flowchart of continue:



- Example:

```
# Program to show the use of continue keyword
for i in range(1,6):
    if i==3:
        continue
    print("Value of i=",i)
```

When you run the above program, the output will be:

```
Value of i= 1
Value of i= 2
Value of i= 4
Value of i= 5
```

- In above program, we can see for loop will iterate through the range(1,6). We check that value of i is 3 if it is 3 then continue is executed. W can see value of i=3 is not printed in output.

❖ Pass keyword:

- **pass** is a predefined keyword in Python which should be written in small letters.
- In Python programming, pass is a null statement.
- The difference between a comment and pass statement in Python is that, while the interpreter ignores a comment entirely, pass is not ignored.
- However, nothing happens when pass is executed. It results into no operation (NOP).
- Syntax:

```
pass
```

- Suppose we have a loop or a function that is not implemented yet, but we want to implement it in the future.
- They cannot have an empty body. The interpreter would complain. So, we use the pass statement to construct a body that does nothing.
- Example:

```
# Program to show the use of pass keyword
numbers=[10,20,30,40,50]
for val in numbers:
    pass
```

When you run the above program, the output will be blank:

```

```