3.Logical operators :-

i/p 1	i/p 2	o/p
1	1	1
1	0	0
0	1	0
1	o	0

1) Logical ‘and’:-

- It is an operator which will return the result as True if both the operands are true,else it will return false.
- Syntax operand and operand2
- If op1 == True
- o/p ==op2
- if op1 == False
- o/p == op1
- all default values intranally False and all nondefalt value intranally True
- examples:-

```
>>> True and 8
>>> 8
>>>2 and ''
>>>''
>>>set() and {8}
>>>set()
>>> 0 and 0.0
>>> 0
>>> 'hai' and 'hello'
>>> 'hello'
>>> 1 and 3 and 5
>>> 5
```

2) Logical or :-

i/p 1	i/p 2	o/p
1	1	1
1	0	1
0	1	1
1	o	0

- Or is on operator which will return the result as true if any one of input is True,else it will return false.
- Syntax op1 or op2

If op1 == True

o/p = op1

if op1 == False

o/p = op2

- Example:-
  - 1 or ' '
  - 1
  - 6 or 7
  - 6
  - [ ] or [1,2,3]
  - [1,2,3]
  - 5.6 or 7.8
  - 5.6
  - " " or " "
  - " "
  - True or True
  - True

3) Logical not :-

i/p	o/p
1	0
0	1

- not is an operator which will return the result as true if given i/p is false,if the given i/p is true it will return false.
- Syntax not(operand)

Ex :-

- not(56)
- False
- not(False)
- True

4.BITWISE OPERATORS :-

- Those are the operator which will convert the given operand into binary value and it will perform bit by bit operation.

1.Bitwise and (&) operation :-

Syntax op1 & op2

- bin() :- is built in function it will return the binary value of argumemnt
- 7 & 4

i/p 1	i/p 2	o/p
1	1	1
1	0	0
0	1	0
1	o	0

- 4
- 13 & 3
- 1
- -5 &2
- 2
- 13 &3
- 1

1	1	0	1 ==13
0	0	1	1==3
0	0	0	1==1

2.Bitwise or (|) operation :-

Syntax `op1 | op2`

- 7 | 4
- 7
- 13 | 3
- 15
- -5 |2
- -5
- 13 |17
- 29

i/p 1	i/p 2	o/p
1	1	1
1	0	1
0	1	1
1	o	0

1	1	0	1 ==13
0	0	1	1==3
1	1	1	1== 15

3.Bitwise (xor) (^):-

- It is an operator which will return as true if any one of i/p true or if both the i/p not equal else if will return false.
- 4^2
- 6
- 8^3
- 11

i/p 1	i/p 2	o/p
1	1	0
1	0	1
0	1	1
1	0	0

4.Bitwise not (~) :-

- It is an operator which will return the positive o/p if thegiven i/p is negative and it will return -ve o/p if the givin operand is +ve.
- Syntax :- `~(op)`

- To perform bitwise not operation internally control will make use of formula
- -(op+1)
- ~(5)
- -6 #-(5+1)
- ~(-8)
- 7
- ~(7)
- -8

5.Bitwise leftshift:- (<<) :-

- It is an operator which is used to add the number of zeroes to rigtht side of a given binary value based on the specified value for n.
- **Syntax :- operand<<n**
- 1<<2
- 4
- 7<<7
- 896
- 5<<1
- 10

6.**Bitwise rightshift :- (>>)**

- It is an operator which will remove the number of bits from the right side of binary values based an the value for n specified by the user.
- **Syntax operand>>n**
- **Ex**
- **7>>2**
- **1**
-

- **3>>7**
- **0**
- **15>>2**
- **3**
- **22>>3**
- **2**

5.Assignment operator :- (=)

- It is an operator which is used to assign a value to a variable.
- By using assignment operator we can increase the efficiency of program
- **Ex**
- A = 10
- B = [1,2,3] # normal assignment
- A = A+2 # compound assignment
- We can used Compound assignment for all arthmatic operation and logical operation
- a = a+b or ➔ a+=b
- a = a-b ➔ a-=b
- a = a*b ➔ a*=b
- a = a/b ➔ a/=b
- a = a//b ➔ a//=b
- a = a%b ➔ a%=b
- a = a&b ➔ a&=b
- a = a|b ➔ a|=b
- a = a^b ➔ a^=b
- a = a<<b ➔ a<<=b
- a = a>>b ➔ a>>=b
- a = [1,2,3]

- a +=[4]
- a
- [1,2,3,4]

6.Membership operator :-

- It is an operator which is used to check whether the given the value is present inside the collection or not.
- Membership operator got classified into 2 types.

1) in operator :-

- This will return true if the value is present inside the collection.
- Syntax value in collection

Ex

- 'h' in 'hello'
- True
- 30 in (1,2,30)
- True
- 4 in 7
- Error
- 'ho' in 'hello'
- False
- (1,2) in (13,5,(1,2))
- True
- 'a' in {'a':1,'b':2}
- True

2) not in :-

- This operator will return the result as true if the value is not present inside the collection.
- Syntax Value not in collection

>>> Ex.

>>> 'a' not in 'abcd'

>>> False

>>> [1,2,3] not in [1,2,3]

>>> True # [1,2,3] this is considerd as single value hence it is True

>>> [1,2,3] in not (1,2,3,[1,2,3])

>>> False

7) IDENTITY OPERATOR :-

- It is an operator which is used to check whether both the variable are pointing to the same memory location or not.
- Identity operator got classified into two types.

1) is :-

- This operator will return the result as true if both the variable are pointing to the same memory location
- Syntax var1 is var2

Ex

>>> a,b,c = 10,20,10

>>> a is c

>>> True

>>> b is c

>>> False

>>> x = ‘hai’

>>> y = ‘hai’

>>> x is y

>>> True

2) Is not :-

>> This operator will return the result as true if both the variable are not pointing same memory location.

- >>> Syntax var1 is not var2

>>> c = [3,4,5]

>>> d = c

>>> c is not d

>>> False

>>> a = 10

>>> b = 20

>>> a is not b

>>> True