Eg:- >>> a = 10
>>> b = 20
>>> a == b
False.

Walrus operator (Python 3.8)

:= (colun and equal-to) is called walrus operator as assignment expression operator.

Eg:- >>> a = (b := 1+5) * (c := 5-2)
>>> print(a)
18
>>> print(b)
6
>>> print(c)
3.

08/11/23
Wed

Inside the expression we can use assign operator.

Eg:- >>> a = (b = 1+5) * (c = 5-2)
Syntax error:- invalid syntax. May be meant '==' or ':=' instead of '='?

Eg:- X := 10
Syntax error: invalid syntax.

Bitwise operators:-

→ Bitwise operators are used to perform operation on binary data.

1) Shift operator.
   a) Left shift operator.
   b) Right shift ".

2) Bitwise & (and) operator.

3> Bitwise | (or) operator.

4> Bitwise ^ (XOR) operator.

5) Bitwise ~ (NOT) operator.

1> Shift operators :- Shifts operators are used to shift or move bits towards left side or right side.

⇒ these operators are used to ↑ (increment) or ↓ (decrement) addings bits or removing bits.

Applications :-

→ Memory Management.
→ Encryption and Decryption.

i> Right shift operator >>
ii) Left " " <<

• >> right shift operator is used to shift number of bits towards right side.

Eg: 0b1100 Formula :- Value // (2) power n (no of bits)

Eg: >>> b = a >> 5

>>> a = 20

>>> b = a >> 2

>>> print(a)
20

>>> print(b)
5.

(2) → 2x → binary.

=4 → position.

• << left shift operator is used to shift numbe of bits towards left side.

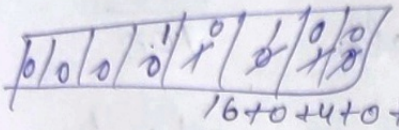
⇒ By adding (shifting) number of bits towards left side the value get increments.

a = 0b1010

b = a<<1.

Syntax:- Value<<n

formula:- Value * 2 power n.



Eg:- a = 10

>>> print(bin(a))

>>> 0b1010

>>> b = a<<1

>>> print(bin(b))

~~Bitwise & (and) operator:~~

Logical gates :- Logical gates is a device that acts as a building block for digital circuits.

→ They perform basic logical functions that are fundamental to digital circuits. Most electronic device we use today will have some form of logic gates in them. Ex:- Logic gates can be used in technologies such as smartphones, tablets or neither devices.

Bitwise & (and) operator:-

→ This operator is used to apply and gate.

Truth table

opr1	opr2	opr1 & opr2
1	0	0
0	1	0
1	1	1
0	0	0

if any input bit is 0, output is 0.

a = 0b1010 — 1010
b = 0b1110 — 1110
c = a&b — 1010

a = 5 — 101
b = 4 — 100
c = a&b — 100

2	5
2	2
2	1
	0
	1

2	4
2	2
2	1
	0
	1

*NOTE:- Bitwise operators are applied only one integer data types.

(1) Eg:- a=5
```
>>> b=4
>>> c=a&b
>>> print(a,b,c)
    5 4 4
print(bin(a), bin(b), bin(c))
    ↓        ↓        ↓
  0b101    0b100    0b100
```

(2) Eg:-
```
>>> X = 0b1010
>>> Y = 0b1110
>>> Z = X&Y
>>> print(bin(X), bin(Y), bin(Z))
        ↓         ↓         ↓
     0b1010    0b1110    0b1010.

>>> print(X,Y,Z)
    10 14 10.
```

Bitwise (|) or operator.

→ This operator is used to apply or gate.

Truth table of | operator

opr1	opr2	opr1 | opr2.
1	0	1
0	1	1
1	1	1
0	0	0

If any input bit is 1, output 1

```
>>> x=0b101
>>> y= 0b110
>>> z=x|y
>>> print(bin(x), bin(y), bin(z))
   0b101  0b110  0b111

>>> a=12
>>> b=20
>>> c=a|b
>>> print(bin(a), bin(b), bin(c))
        0b1100   0b100   0b11100

>>> print(a,b,c)
   12  20  28

>>> a and b
   20.
```

Bitwise XOR (^) operator :-

→ This operator is used to apply XOR gates.

Truth Table

opr1	opr2	opr1 ^ opr2
1	0	1
0	1	1
0	0	0
1	1	0

```
>>> a = 0b101
>>> b = 0b100
>>> c = a^b
>>> print(bin(a), bin(b), bin(c))
        0b101  0b100  0b1
```

Bitwise not (~) operator :-

Bitwise not operator is unary operator, this operator required 1 operand.

formula - (value + 1)

Truth table

opr1	~ opr2
1	0
0	1

```
>>> a = 5
>>> b = ~a
>>> print(a)
 5
>>> print(b)
 -6
>>> print(bin(a), bin(b))
        0b101 , -0b110
```

Assignment operator OR update operator:-

- Any of the operators can be combined with assignment. The means that +=, -=, *=, /=, //=, %=, **=, >>=, <<=, &=, ^=, and |=
- Augument assignment statement is a single operator perform two operations.
  1> Binary operations
  2> Assignment.

```
>>> a = 5
>>> a = a + 2
>>> print(a)
 7

>>> a += 2
>>> print(a)
 9
```

Eg:- >>> b=10
>>> b-=1
>>> print(b)
9

Eg:- >>> c=16
>>> c/=4
>>> print(c)
4.0

Eg:- >>> a=12
>>> a>>=2
>>> print(a)
3.