

&	<p>Bitwise and operator, this operator is used to apply and gate</p> <p>Truth table</p> <table><tr><th>Opr1</th><th>Opr2</th><th>Opr1 & Opr2</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table> <pre>>>> a=10 >>> b=5 >>> c=a&b >>> print(a,b,c) 10 5 0 >>> print(bin(a),bin(b),bin(c)) 0b1010 0b101 0b0 >>></pre>	Opr1	Opr2	Opr1 & Opr2	1	1	1	1	0	0	0	1	0	0	0	0
Opr1	Opr2	Opr1 & Opr2														
1	1	1														
1	0	0														
0	1	0														
0	0	0														
	<p>Bitwise or operator, this operator is used to apply or gate</p> <p>Truth table of or gate</p> <table><tr><th>Opr1</th><th>Opr2</th><th>Opr1 Opr2</th></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table> <pre>>>> a=10 >>> b=5 >>> c=a b >>> print(a,b,c) 10 5 15 >>> print(bin(a),bin(b),bin(c)) 0b1010 0b101 0b1111 >>></pre>	Opr1	Opr2	Opr1 Opr2	1	0	1	0	1	1	1	1	1	0	0	0
Opr1	Opr2	Opr1 Opr2														
1	0	1														
0	1	1														
1	1	1														
0	0	0														
~	<p>Bitwise not operator</p> <p>This is complement operator</p> <p>Formula➡ -(num+1)</p>															

	<pre>>>> a=5 >>> b=~a >>> print(a,b) 5 -6 >>> b=-6 >>> c=~b >>> print(b,c) -6 5 >>></pre>															
^	<p>Bitwise XOR operator, this is used to apply XOR gate</p> <p>Truth table</p> <table><tr><th>Opr1</th><th>Opr2</th><th>Opr1 ^ Opr2</th></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table> <pre>>>> a=10 >>> b=10 >>> c=a^b >>> print(a,b,c) 10 10 0 >>> print(bin(a),bin(b),bin(c)) 0b1010 0b1010 0b0 >>> a=10 >>> b=5 >>> c=a^b >>> print(a,b,c) 10 5 15 >>></pre>	Opr1	Opr2	Opr1 ^ Opr2	1	0	1	0	1	1	0	0	0	1	1	0
Opr1	Opr2	Opr1 ^ Opr2														
1	0	1														
0	1	1														
0	0	0														
1	1	0														

Assignment Operators or update operators

Augmented assignment statements

Augmented assignment is the combination, in a single statement, of a binary operation and an assignment statement

Operators	Description and Example
-----------	-------------------------

+=	a=5 a=a+2 → a+=2 b=5 b+=4*2
-=	A=5 A=A-1 → A-=1
*=	A=5 A=A*2 → A*=2 B=3 A=A*B → A*=B
/=	A=5 A=A/2 (OR) A/=2
//=	A=5 A=A//2 (OR) A//=2 A=5 A+=5-=2 → Syntax Error
%=	A=5 A=A%2 → A%=2
=	N1=4 N1=N12 → N1**=2
>>=	A=5 A=A>>2 → A>>=2
<<=	A=5 A=A<<2 → A<<=2
&=	A=5 B=2 A&=B → A=A&B
 =	A=8 B=5 A =B → A=A B
^=	A=6 B=5 A^=B → A=A^B

Salary=5000

Salary=salary+200 → Salary+=200

Balance=90000

Balance=balance+10000 → balance+=10000

Balance-=5000 → balance=balance-5000

Identity Operator

Every object in python is having identity, which is called address

In order to compare identity or address of objects we use identity operators.

1. **is** → return True, if two variable hold address of same object
2. **is not** → return True, if two variable hold different object addresses

```
>>> a=10
>>> b=10
>>> id(a)
92582472272
>>> id(b)
92582472272
>>> a==b
True
>>> a is b
True
>>> list1=[10,20,30,40,50]
>>> list2=[10,20,30,40,50]
>>> list1==list2
True
>>> list3=[30,20,10,40,50]
>>> list1==list3
False
>>> list1 is list2
False
>>> id(list1)
92596326208
>>> id(list2)
92626761152
>>> list4=list1
>>> list1 is list4
True
>>>
```

What difference is between == and is operator?

`==` operator is used to compare state of the object or values of the object
`is` operator is used to compare id of objects (OR) address of objects

```
>>> f1=1.5
>>> f2=1.5
>>> f1==f2
True
>>> id(f1)
92626355184
>>> id(f2)
92626354544
>>> f1 is f2
False
>>> f1 is not f2
True
>>>
```

Membership operator

Membership operators are used to search a given value exists within group values or collection of values

1. `in`
2. `not in`

Membership operator return Boolean value (True/False)

It is a binary operator, it required 2 operands

1. value
2. collection

```
>>> 10 in [10,20,30,40,50]
True
>>> 60 in [10,20,30,40,50]
False
>>> "a" in "java"
True
>>> "python" in "python is language"
True
>>>
```

Example:

```
# write a program to find input character is vowel or not
ch=input("Enter any character")
print("vowel") if ch in "aeiouAEIOU" else print("not vowel")
```

Walrus operator

Walrus assignment operator :=

This operator is introduced in python 3.8 version

Example:

```
a=2
b=3
c=(x:=a**2)+(y:=b**2)+(z:=2*a*b)
print(a,b,c)
print(x,y,z)
```

Output:

```
2 3 25
4 9 12
>>>
```

Example:

write a program to add two numbers

```
n3=(n1:=int(input("Enter first number")))+(n2:=int(input("Enter second
number")))
print(n1,n2,n3)
```

Output:

```
Enter first number10
Enter second number20
10 20 30
>>>
```