Bitwise Operators

Bitwise Operators are used to perform operations on bits or binary data.

Python support the following bitwise operators

Operators	**Description**
>>	Right shift operator
<<	Left shift operator
&	Bitwise and operator
|	Bitwise or operator
^	Bitwise xor operator
~	Bitwise not operator

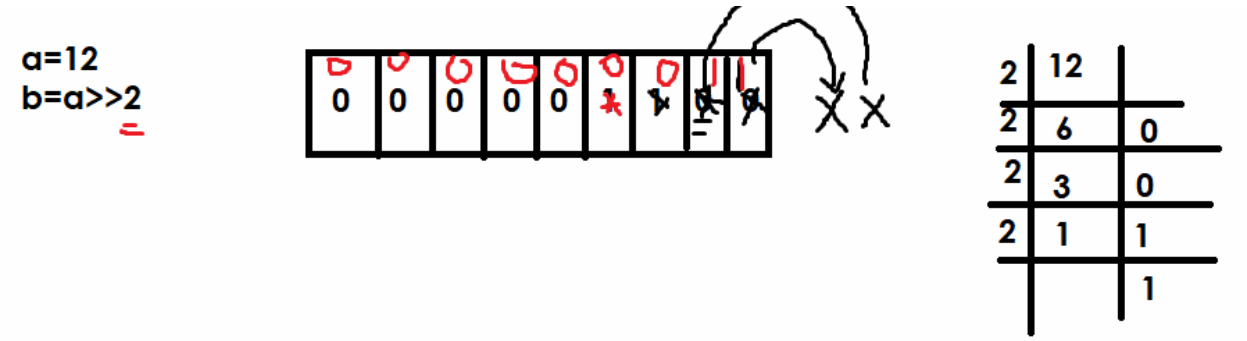
I**n application development bitwise operators are used,**

1. Encrypting and Decryption (OR) Encoding and Decoding
2. Images, Audio, Video processing
3. Operating System for representing access permission
4. Memory Management
5. Low Level Programming (Embedded Applications)
6. To perform arithmetic operations

Bitwise operators can be applied only on integer data type or integer value.
Bitwise operators always perform operations by converting integer value into 0's and 1's

>> Right shift operator
This operator is used for shifting bits towards right side
This allows shifting n bits towards right side
This allows decrement value by shifting **n** bits towards right side

Syntax: Opr>>n




```
>>> a=12
>>> b=a>>2
>>> print(a)
12
>>> print(b)
3
>>> print(bin(a),bin(b))
0b1100 0b11
>>> x=15
>>> y=x>>3
>>> print(bin(x),bin(y))
0b1111 0b1
>>> print(x,y)
15 1
```

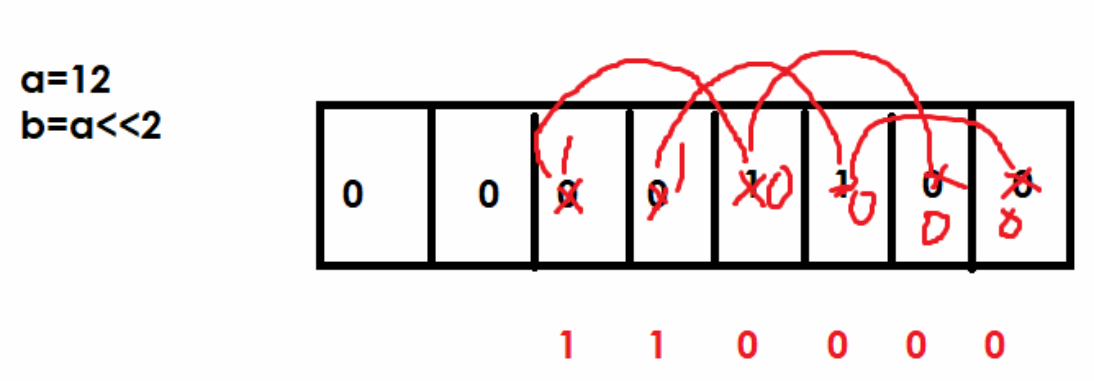
Formula : opr//2 pow n

```
>>> a=12
>>> b=a//2**2
>>> print(a)
12
>>> print(b)
3
>>> a=126
>>> b=a>>5
>>> print(a,b)
126 3
>>> print(bin(a),bin(b))
0b1111110 0b11
```

Left shift operator (<<)
This operator is used to shift n bits towards left side

It allows adding n bits at right side.
This operator increment value by adding bits at right side

Syntax: opr<<n




```
>>> a=12
>>> b=a<<2
>>> print(bin(a),bin(b))
0b1100 0b110000
>>> print(a,b)
12 48
>>> x=10
>>> y=x<<4
>>> print(bin(x),bin(y))
0b1010 0b10100000
>>> print(x,y)
10 160
```

Formula : opr *2pow n

```
>>> a=12
>>> b=a<<3
>>> print(a,b)
12 96
>>> print(bin(a),bin(b))
0b1100 0b1100000
```

What is logic gate?
Logic gates are fundamental building blocks of digital circuits, performing basic logical operations on binary inputs (0 or 1) to produce a single output

Application of logic gates

Digital Circuits: Logic gates are the foundation of all digital systems, including computers, smartphones, and other electronic devices.
Boolean Algebra: Logic gates implement Boolean operations, which are used in various fields like mathematics, computer science, and engineering.
Microprocessors: Logic gates are used in the design and functionality of microprocessors.
Error Detection: XNOR gates can be used in error-detecting circuits.
Arithmetic Circuits: Logic gates are used in arithmetic circuits, such as adders.
Encryption Circuits: Logic gates are used in encryption circuits

Python support logic gate using bitwise logical operators

1. Bitwise logical & (and) operator
2. Bitwise logical | (or) operator
3. Bitwise logical ^ (XOR) operator
4. Bitwise logical ~ (Not) operator

Bitwise & (and) operator

This operator is used to apply logical and gate
Truth table of bitwise (&) and operator

Opr1	Opr2	Opr1 & Opr2
1	0	0
0	1	0
1	1	1
0	0	0

```
>>> a=0b1010
>>> b=0b1100
>>> c=a&b
>>> print(bin(a),bin(b),bin(c))
```

Bitwise | (or) operator

Bitwise | (OR) operator is used to apply logical or gate
Truth table of bitwise | operator

Opr1	Opr2	Opr1| Opr2
1	0	1
0	1	1
1	1	1
0	0	0

```
>>> x=0b1011
>>> y=0b1100
>>> z=x|y
>>> print(bin(x),bin(y),bin(z))
0b1011 0b1100 0b1111
>>> print(x,y,z)
11 12 15
>>> a=11
>>> b=12
>>> c=a|b
>>> print(bin(a),bin(b),bin(c))
0b1011 0b1100 0b1111
>>> print(a,b,c)
11 12 15
>>> p1=1.5
>>> p2=2.0
>>> p3=p1|p2
Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    p3=p1|p2
TypeError: unsupported operand type(s) for |: 'float' and 'float'
```

Bitwise ^ (XOR) operator

This operator is used to apply XOR grate

The truth table XOR operator is

Opr1	Opr2	Opr3
1	0	1
0	1	1
0	0	0
1	1	0

```
>>> a=0b101
>>> b=0b110
>>> c=a^b
>>> print(bin(a),bin(b),bin(c))
0b101 0b110 0b11
>>> print(a,b,c)
5 6 3
```

Example:

```
# Write a program to swap two numbers without
# using third variable with help of bitwise XOR operator

x=int(input("Enter First Number "))
y=int(input("Enter Second Number "))

print(f'Before Swaping {x},{y}')
x=x^y
y=x^y
x=x^y

print(f'After Swaping {x},{y}')
```

Output

```
Enter First Number 10
Enter Second Number 20
Before Swaping 10,20
After Swaping 20,10
```

Bitwise ~ (not) operator

Truth table of bitwise (~) operator

It is a unary operator and required 1 operand to perform operation.

Opr1	**~Opr1**
1	**0**
0	**1**

Formula : -(opr+1)

```
>>> x=10
>>> y=~x
>>> print(x,y)
10 -11
>>> print(bin(x),bin(y))
0b1010 -0b1011
>>> p=-11
>>> q=~p
>>> print(p,q)
-11 10
>>> print(bin(p),bin(q))
-0b1011 0b1010
```

Assignment Operators (OR) Compound Assignment Statements (OR) Update Operators

Compound assignment operator, is a single operator which perform 2 operations

1. Binary operation
2. Assignment

Operators	Description
+=	A=10 A=A+5 (OR) A+=5 X=10 Y=5 X=X+Y (OR) X+=Y
-=	A=10 B=5 A-=B (OR) A=A-B
*=	A=5 B=2 A*=B (OR) A=A*B
/=	A=5 B=3

	A/=B (OR) A=A/B
//=	A=5 B=3 A//=B (OR) A=A//B
**=	A=5 B=2 A**=B (OR) A=A**B
%=	A=5 B=3 A%=B (OR) A=A%B
>>=	A=5 B=2 A>>=B (OR) A=A>>B
<<=	A=5 B=2 A<<=B (OR) A=A<<B
&=	A=10 B=20 A&=B (OR) A=A&B
|=	A=10 B=15 A|=B (OR) A=A|B
^=	A=10 B=12 A^=B (OR) A=A^B

Walrus Operator (OR) Assignment Expression Operator (:=)
Walrus operator is introduced in python 3.8 version