Python Operators with Examples

Python Operators perform operations by taking one or more value, to give another value. These operations are performed on variables and values. Following are the operators:

- Arithmetic Operators
- Assignment Operators
- Comparison/ Relational operators
- Identity Operators
- Membership Operators
- Bitwise Operators

The types of operators are explained here with examples, where **x = 5, y =10**. Following are the Python operators:

Arithmetic Operators

Let us see the below table describing Arithmetic operators with examples:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x – y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y

Assignment Operators

To assign values to variables, use the Assignment operators. Let us see the below table describing Assignment operators with examples:

Comparison/ Relational operators

To compare the value of two operands, Python has comparison. Relational operators. Let us see the below table describing comparison operators with examples:

Logical operators

Logical operators combine conditional statements. Let us see the below table describing logical operators with examples:

Identity Operators

To compare the objects to check they are with the same memory location, work with the Python Identity operators. Let us see the below table describing Identity Operators in Python:

Membership Operators

The Membership operator is used to check for membership in a data structure, like tuple, strings, lists, dictionary, etc. Let us see the below table describing Membership operators with examples:

Bitwise Operators

To compare Binary number, bit by bit, on operands, use the Bitwise Operators. Following are the Bitwise Operators:

- Binary AND
- Binary OR
- Binary XOR
- Binary NOT
- Binary Left Shift
- Binary Right Shift

Binary AND Bitwise operator

The Binary AND operator set each bit to 1, if both the bits are 1 i.e., if both the operands are 1, then result is 1.

Let us see an example:

```
## Bitwise AND operator

x = 18          # 18 = 0001 0010
y = 11          # 11 = 0000 1011
res = 0

res = x & y;
print ("x AND y = ", res)
```

The output is as follows:

```
x AND y = 2
```

Binary OR Bitwise operator

The Binary OR operator sets each bit to 1, if any of the two bits are 1 i.e., if any of the two operands are 1, then result is 1.

Let us see an example:

```
## Bitwise OR operator

x = 18          # 18 = 0001 0010
y = 11          # 11 = 0000 1011
res = 0

res = x | y;
print ("x OR y = ", res)
```

The output is as follows:

```
x OR y = 27
```

Binary XOR Bitwise operator

The Binary XOR operator sets each bit to 1, if only one of them is 1, i.e. if only one of the two operands are 1.

Let us see an example:

```
## Bitwise XOR operator

x = 18          # 18 = 0001 0010
y = 11          # 11 = 0000 1011
res = 0

res = x ^ y;
print ("x XOR y = ", res)
```

The output is as follows:

```
x XOR y = 25
```

Binary NOT Bitwise operator

The Binary NOT Bitwise operator negates the bits i.e., 1 for 0 and 0 for 1.

Let us see an example:

```
## Bitwise NOT operator

x = 18          # 18 = 0001 0010
res = 0
```

```
res = ~x
print (" ~x = ", res)
```

The output is as follows:

```
~x =  -19
```

Binary LEFT SHIFT Bitwise operator

The Binary LEFT SHIFT operator shits the left operand value to the left, by the number of bits in the right operand. The leftmost bits while shifting fall off.

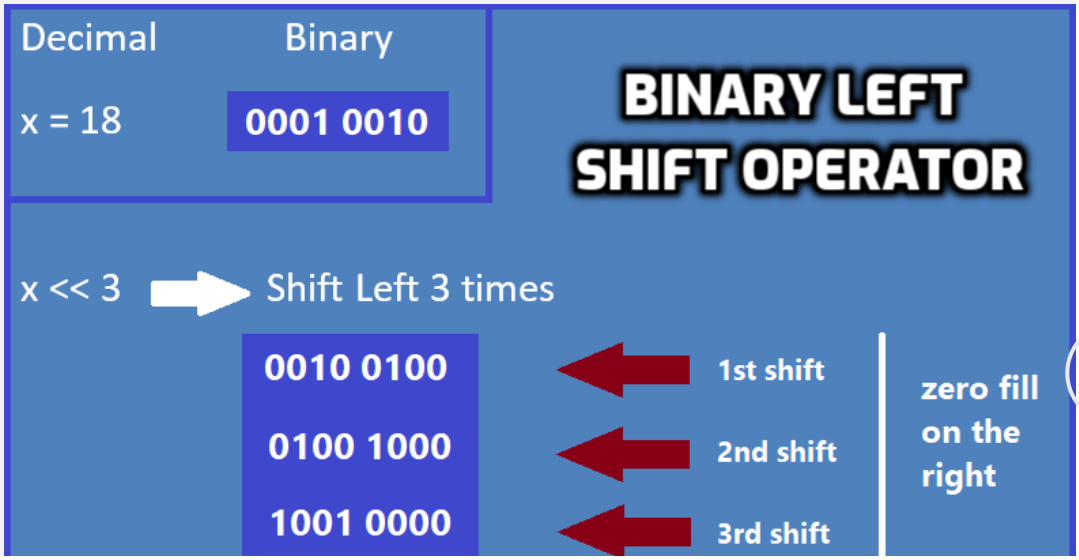
Let us see an example to implement the LEFT SHIFT Bitwise Operator and shift 3 times:

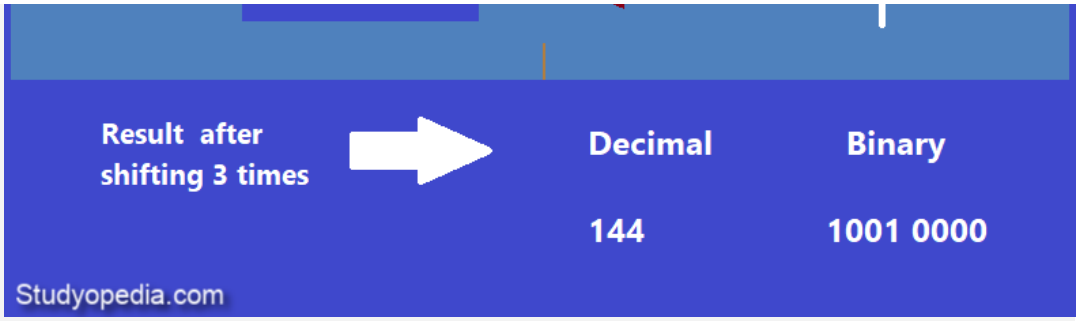
```
## Bitwise Left Shift operator

x = 18          # 18 = 0001 0010
res = 0

res = x << 3
print (" Left Shift = ", res)
```

The output is as follows:

```
Left Shift = 144
```

Now, let us see how the above result calculated. We have to shift 3 times left and fill zero on the right:





Binary RIGHT SHIFT Bitwise operator

The Binary RIGHT SHIFT operator shits the left operand value to the right, by the number of bits in the right operand. The rightmost bits while shifting fall off.

Let us see an example to implement the RIGHT SHIFT Bitwise operator and shift 2 times:

```
## Bitwise Right Shift operator

x = 18        # 18 = 0001 0010
res = 0

#shift 2 times
res = x >> 2

print (" Right Shift = ", res)
```

The output is as follows:

```
Right Shift =  4
```