Operators in Python

Operators are used to perform operations on variables and values.

Python Operator falls into below categories:

- Python Arithmetic Operator
- Python Relational Operator
- Python Assignment Operator
- Python Logical Operator
- Python Bitwise Operator

Python Arithmetic Operator

These Python arithmetic operators include Python operators for basic mathematical operations.

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

a. Addition(+)
Adds the values on either side of the operator.
Expression : 3+4
Output: 7

b. Subtraction(-)
Subtracts the value on the right from the one on the left.
Expression : 3-4
Output: -1

c. Multiplication(*)
Multiplies the values on either side of the operator.
Expression : 3*4
Output: 12

d. Division(/)
Divides the value on the left by the one on the right. Notice that division results in a floating-point value.
Expression : 3/4
Output: 0.75

e. Exponentiation(**)
Raises the first number to the power of the second.
Expression : 3**4
Output: 81

f. Floor Division(//)
Divides and returns the integer value of the quotient. It dumps the digits after the decimal.
Expression : 3//4
Expression : 4//3
Output: 1

Expression : 10//3
Output: 3

g. Modulus(%)
Divides and returns the value of the remainder.
Expression : 3%4
Output: 3

Expression : 4%3
Output: 1

Expression : 10%3
Output: 1

Expression : 10.5%3
Output: 1.5

Python Relational Operator

Relational Python Operator carries out the comparison between operands.
They tell us whether an operand is greater than the other, lesser, equal, or a combination of those.

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

a. Less than(<)
This operator checks if the value on the left of the operator is lesser than the one on the right.
Expression : 3<4
Output: True

b. Greater than(>)
It checks if the value on the left of the operator is greater than the one on the right.
Expression : 3>4
Output: False

c. Less than or equal to(<=)
It checks if the value on the left of the operator is lesser than or equal to the one on the right.
Expression : 7<=7
Output: True

d. Greater than or equal to(>=)
It checks if the value on the left of the operator is greater than or equal to the one on the right.
Expression : 0>=0
Output: True

e. Equal to(= =)

This operator checks if the value on the left of the operator is equal to the one on the right. 1 is equal to the Boolean value True, but 2 isn't. Also, 0 is equal to False.
Expression : 3==3.0
Output: True

Expression : 1==True
Output: True

Expression : 7==True
Output: False

Expression : 0==False
Output: True

Expression : 0.5==True
Output: False

f. Not equal to(!=)
It checks if the value on the left of the operator is not equal to the one on the right. The Python operator <> does the same job, but has been abandoned in Python 3.

When the condition for a relative operator is fulfilled, it returns True. Otherwise, it returns False. We can use this return value in a further statement or expression.

Expression : 1!=-1.0
Output: False

Python Assignment Operator

An assignment operator assigns a value to a variable. It may manipulate the value by a factor before assigning it.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

a. Assign(=)
Assigns a value to the expression on the left. Notice that = = is used for comparing, but = is used for assigning.
Expression : a=7
Expression : print(a)
Output: 7

b. Add and Assign(+=)
Adds the values on either side and assigns it to the expression on the left. a+=10 is the same as a=a+10.
The same goes for all the next assignment operators.
Expression : a+=2
Expression : print(a)
Output: 9

c. Subtract and Assign(-=)
Subtracts the value on the right from the value on the left. Then it assigns it to the expression on the left.
Expression : a-=2
Expression : print(a)
Output: 7

d. Divide and Assign(/=)
Divides the value on the left by the one on the right. Then it assigns it to the expression on the left.
Expression : a/=7
Expression : print(a)
Output: 1.0

e. Multiply and Assign(*=)
Multiplies the values on either sides. Then it assigns it to the expression on the left.
Expression : a*=8
Expression : print(a)
Output: 8.0

f. Modulus and Assign(%=)
Performs modulus on the values on either side. Then it assigns it to the expression on the left.
Expression : a%=3
Expression : print(a)
Output: 2.0

g. Exponent and Assign(**=)
Performs exponentiation on the values on either side. Then assigns it to the expression on the left.
Expression : a**=5
Expression : print(a)
Output: 32.0

h. Floor-Divide and Assign(//=)
Performs floor-division on the values on either side. Then assigns it to the expression on the left.
Expression : a//=3
Expression : print(a)
Output: 10.0

Python Logical Operator

These are conjunctions that we can use to combine more than one condition. We have three Python logical operator – and, or, and not that come under python operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

a. and
If the conditions on both the sides of the operator are true, then the expression as a whole is true.
Expression : a=7>7 and 2>-1
Expression : print(a)
Output: False

b. or
The expression is false only if both the statements around the operator are false. Otherwise, it is true.
Expression : a=7>7 or 2>-1
Expression : print(a)
Output: True

'and' returns the first False value or the last value; 'or' returns the first True value or the last value

Expression : 7 and 0 or 5
Output: 5

c. not
This inverts the Boolean value of an expression. It converts True to False, and False to True. As we can see below, the Boolean value for 0 is False. So, not inverts it to True.

Expression : a=not(0)
Expression : print(a)
Output: True

Python Bitwise Operator

Let us now look at Bitwise Python Operator.

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

On the operands, these operate bit by bit.

a. Binary AND(&)
It performs bit by bit AND operation on the two values. Here, binary for 2 is 10, and that for 3 is 11. &-ing them results in 10, which is binary for 2. Similarly, &-ing 011(3) and 100(4) results in 000(0).
Expression : 2&3
Output: 2

Expression : 3&4
Output: 0

b. Binary OR(|)
It performs bit by bit OR on the two values. Here, OR-ing 10(2) and 11(3) results in 11(3).
Expression : 2|3
Output: 3

c. Binary XOR(^)
It performs bit by bit XOR(exclusive-OR) on the two values. Here, XOR-ing 10(2) and 11(3) results in 01(1).
Expression : 2^3
Output: 1

d. Binary One's Complement(~)
It returns the one's complement of a number's binary. It flips the bits. Binary for 2 is 00000010. Its one's complement is 11111101. This is binary for -3. So, this results in -3. Similarly, ~1 results in -2.
Expression : ~-3
Output: 2

Again, one's complement of -3 is 2.

e. Binary Left-Shift(<<)

It shifts the value of the left operand the number of places to the left that the right operand specifies. Here, binary of 2 is 10. 2<<2 shifts it two places to the left. This results in 1000, which is binary for 8.

Expression : 2<<2

Output: 8

f. Binary Right-Shift(>>)

It shifts the value of the left operand the number of places to the right that the right operand specifies. Here, binary of 3 is 11. 3>>2 shifts it two places to the right. This results in 00, which is binary for 0. Similarly, 3>>1 shifts it one place to the right. This results in 01, which is binary for 1.

Expression : 3>>2

Expression : 3>>1

Output: 1