EduTECH
WARJE
Trusted by student's and professional's

Operators In Python

Get in touch with us

edutechwarje@gmail.com

9607727888

Office No-13, Opposite Warje Flyover Above Bank Of Maharashtra , Warje, Pune-58

www.edutechwarje.com

Operators are the constructs which can manipulate the value of operands.

Consider the expression 4 + 5 = 9. Here, 4 and 5 are called operands and + is called operator.

Types of Operator

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Let us have a look on all operators one by one.

Python Arithmetic Operators

Assume variable a holds 10 and variable b holds 20, then –

Operator	Description	Example
+	Addition Adds values on either side of the operator.	a + b = 30
-	Subtraction Subtracts right hand operand from left hand operand.	a – b = -10
*	Multiplies values on either side of the Multiplication operator	a * b = 200
/	Division Divides left hand operand by right hand operand	b / a = 2
%	Modulus Divides left hand operand by right	b % a = 0

hand operand and returns remainder		
** Exponent Performs exponential (power) a**b =10 to the power 20		calculation on operators
// Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) –		9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Example

Assume variable a holds 21 and variable b holds 10, then –

```
a = 21
b = 10
c = 0

c = a + b
print "Line 1 - Value of c is ", c
c = a - b
print "Line 2 - Value of c is ", c
c = a * b
print "Line 3 - Value of c is ", c
c = a / b
print "Line 4 - Value of c is ", c
c = a % b
print "Line 5 - Value of c is ", c
```

```
a = 2
b = 3
c = a**b
print "Line 6 - Value of c is ", c
a = 10
b = 5
c = a//b
print "Line 7 - Value of c is ", c
```

```
Line 1 - Value of c is 31
Line 2 - Value of c is 11
Line 3 - Value of c is 210
Line 4 - Value of c is 2
Line 5 - Value of c is 1
Line 6 - Value of c is 8
Line 7 - Value of c is 2
```

Python Comparison Operators

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.

Assume variable a holds 10 and variable b holds 20, then –

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.
<>	If values of two operands are not equal, then condition becomes true. This is similar to != operator.	(a <> b) is true.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Example

Assume variable a holds 10 and variable b holds 20, then –

a = 21
b = 10
c = 0

```
if ( a == b ):
print "Line 1 - a is equal to b"
else:
print "Line 1 - a is not equal to b"
if ( a != b ):
print "Line 2 - a is not equal to b"
else:
print "Line 2 - a is equal to b"
if ( a <> b ):
print "Line 3 - a is not equal to b"
else:
print "Line 3 - a is equal to b"
if ( a < b ):
print "Line 4 - a is less than b"
else:
print "Line 4 - a is not less than b"
if ( a > b ):
print "Line 5 - a is greater than b"
else:
print "Line 5 - a is not greater than b"
a = 5;
b = 20;
if ( a <= b ):
print "Line 6 - a is either less than or equal to b"
else:
    print "Line 6 - a is neither less than nor equal to b"
if ( b >= a ):
   print "Line 7 - b is either greater than or equal to b"
```

```
else:
   print "Line 7 - b is neither greater than nor equal to b"
```

Line 1 - a is not equal to b

Line 2 - a is not equal to b
Line 3 - a is not equal to b
Line 4 - a is not less than b
Line 5 - a is greater than b
Line 6 - a is either less than or equal to b
Line 7 - b is either greater than or equal to b

Python Assignment Operators

Assume variable a holds 10 and variable b holds 20, then –

Operator Description	Example
= Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%= Modulus AND It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Example

Assume variable a holds 10 and variable b holds 20, then –

```
a = 21
b = 10
c = 0

c = a + b
print "Line 1 - Value of c is ", c
c += a
print "Line 2 - Value of c is ", c
c *= a
print "Line 3 - Value of c is ", c
c /= a
print "Line 4 - Value of c is ", c
c = 2
c %= a
print "Line 5 - Value of c is ", c

c **= a
print "Line 6 - Value of c is ", c
c //= a
print "Line 7 - Value of c is ", c
```

```
Line 1 - Value of c is 31
Line 2 - Value of c is 52
Line 3 - Value of c is 1092
```

```
Line 4 - Value of c is 52
Line 5 - Value of c is 2
Line 6 - Value of c is 2097152
Line 7 - Value of c is 99864
```

Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation.

Assume if a = 60; and b = 13; Now in the binary format their values will be 0011 1100 and 0000 1101 respectively. Following table lists out the bitwise operators supported by Python language with an example each in those, we use the above two variables (a and b) as operands –

a = 0011 1100

b = 0000 1101

---

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

There are following Bitwise operators supported by Python language

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping'	(~a) = -61 (means 1100

		complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

Example

```
a = 60 # 60 = 0011 1100
b = 13 # 13 = 0000 1101
c = 0

c = a & b; # 12 = 0000 1100
print "Line 1 - Value of c is ", c
c = a | b; # 61 = 0011 1101
print "Line 2 - Value of c is ", c
c = a ^ b; # 49 = 0011 0001
print "Line 3 - Value of c is ", c
c = ~a; # -61 = 1100 0011
print "Line 4 - Value of c is ", c
c = a << 2; # 240 = 1111 0000
print "Line 5 - Value of c is ", c
c = a >> 2; # 15 = 0000 1111
```

```
print "Line 6 - Value of c is ", c
```

```
Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
Line 4 - Value of c is -61
Line 5 - Value of c is 240
Line 6 - Value of c is 15
```

Python Logical Operators

There are following logical operators supported by Python language.
Assume variable a holds 10 and variable b holds 20 then

Operator	Description	Example
and Logical	If both the operands are true then condition becomes true.	(a and b) is true. AND
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical	Used to reverse the logical state of its NOT operand.	Not(a and b) is false.

Python Membership Operators

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below –

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Example

```
a = 10
b = 20
list = [1, 2, 3, 4, 5 ];

if ( a in list ):
print "Line 1 - a is available in the given list"
else:
   print "Line 1 - a is not available in the given list"
if ( b not in list ):
   print "Line 2 - b is not available in the given list"
else:
print "Line 2 - b is available in the given list"
```

```
a = 2
if ( a in list ):
print "Line 3 - a is available in the given list"
else:
  print "Line 3 - a is not available in the given list"
```

Line 1 - a is not available in the given list

Line 2 - b is not available in the given list
Line 3 - a is available in the given list

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below –

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is** **not** results in 1 if id(x) is not equal to id(y).

```
a = 20
b = 20
if ( a is b ):
print "Line 1 - a and b have same identity"
else:
print "Line 1 - a and b do not have same identity"
if ( id(a) == id(b) ):
print "Line 2 - a and b have same identity"
else:
print "Line 2 - a and b do not have same identity"
b = 30
if ( a is b ):
print "Line 3 - a and b have same identity"
else:
```

```
   print "Line 3 - a and b do not have same identity"

if ( a is not b ):
   print "Line 4 - a and b do not have same identity"
else:
print "Line 4 - a and b have same identity"
```

```
Line 1 - a and b have same identity
Line 2 - a and b have same identity
Line 3 - a and b do not have same identity
Line 4 - a and b do not have same identity
```

Python Operators Precedence

The following table lists all operators from highest precedence to lowest.

Sr.No.	Operator & Description
1	**** Exponentiation (raise to the power)
2	**~ + -** Complement, unary plus and minus (method names for the last two are +@ and -@)
3	*** / % //** Multiply, divide, modulo and floor division
4	**+ -** Addition and subtraction
5	**>> <<** Right and left bitwise shift
6	**&** Bitwise 'AND'
7	**^ |** Bitwise exclusive `OR' and regular `OR'
8	**<= < > >=**

	Comparison operators
9	**<> == !=** Equality operators
10	**= %= /= //= -= += *= **=** Assignment operators
11	**is is not** Identity operators
12	**in not in** Membership operators
13	**not or and** Logical operators