Python - Operators

Python Operators

Python operators are special symbols used to perform specific operations on one or more operands. The variables, values, or expressions can be used as operands. For example, Python's addition operator (**+**) is used to perform addition operations on two variables, values, or expressions.

The following are some of the terms related to **Python operators**:

- **Unary operators**: Python operators that require one operand to perform a specific operation are known as unary operators.
- **Binary operators**: Python operators that require two operands to perform a specific operation are known as binary operators.
- **Operands**: Variables, values, or expressions that are used with the operator to perform a specific operation.

Types of Python Operators

Python operators are categorized in the following categories –

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

Let us have a look at all the operators one by one.

Learn **Python** in-depth with real-world projects through our **Python certification course**. Enroll and become a certified expert to boost your career.

Python Arithmetic Operators

Python Arithmetic operators are used to perform basic mathematical operations such as addition, subtraction, multiplication, etc.

The following table contains all arithmetic operators with their symbols, names, and examples (assume that the values of **a** and **b** are 10 and 20, respectively) –

Operator	Name	Example
+	Addition	a + b = 30
-	Subtraction	a – b = -10
*	Multiplication	a * b = 200
/	Division	b / a = 2
%	Modulus	b % a = 0
**	Exponent	a**b =10**20
//	Floor Division	9//2 = 4

Example of Python Arithmetic Operators

</> Open Compiler

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

c = a + b
print ("a: {} b: {} a+b: {}".format(a,b,c))

c = a - b
print ("a: {} b: {} a-b: {}".format(a,b,c) )

c = a * b
print ("a: {} b: {} a*b: {}".format(a,b,c))

c = a / b
print ("a: {} b: {} a/b: {}".format(a,b,c))

c = a % b
print ("a: {} b: {} a%b: {}".format(a,b,c))

a = 2
b = 3
```

```
c = a**b
print ("a: {} b: {} a**b: {}".format(a,b,c))

a = 10
b = 5
c = a//b
print ("a: {} b: {} a//b: {}".format(a,b,c))
```

Output

```
a: 21 b: 10 a+b: 31
a: 21 b: 10 a-b: 11
a: 21 b: 10 a*b: 210
a: 21 b: 10 a/b: 2.1
a: 21 b: 10 a%b: 1
a: 2 b: 3 a**b: 8
a: 10 b: 5 a//b: 2
```

Python Comparison Operators

Python Comparison operators compare the values on either side of them and decide the relation among them. They are also called Relational operators.

The following table contains all comparison operators with their symbols, names, and examples (assume that the values of **a** and **b** are 10 and 20, respectively) –

Operator	Name	Example
==	Equal	(a == b) is not true.
!=	Not equal	(a != b) is true.
>	Greater than	(a > b) is not true.
<	Less than	(a < b) is true.
>=	Greater than or equal to	(a >= b) is not true.
<=	Less than or equal to	(a <= b) is true.

Example of Python Comparison Operators

</> Open Compiler

```
a = 21
b = 10
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 less than b" )
else:
   print ("Line 3 - a is not less than b")

if ( a > b ):
   print ("Line 4 - a is greater than b")
else:
   print ("Line 4 - a is not greater than b")

a,b=b,a #values of a and b swapped. a becomes 10, b becomes 21

if ( a <= b ):
   print ("Line 5 - a is either less than or equal to  b")
else:
   print ("Line 5 - a is neither less than nor equal to  b")

if ( b >= a ):
   print ("Line 6 - b is either greater than  or equal to b")
else:
   print ("Line 6 - b is neither greater than  nor equal to b")
```

Output

```
Line 1 - a is not equal to b
Line 2 - a is not equal to b
Line 3 - a is not less than b
Line 4 - a is greater than b
```

```
Line 5 - a is either less than or equal to  b
Line 6 - b is either greater than  or equal to b
```

Python Assignment Operators

Python Assignment operators are used to assign values to variables. Following is a table which shows all Python assignment operators.

The following table contains all assignment operators with their symbols, names, and examples –

Operator	Example	Same As
=	a = 10	a = 10
+=	a += 30	a = a + 30
-=	a -= 15	a = a - 15
*=	a *= 10	a = a * 10
/=	a /= 5	a = a / 5
%=	a %= 5	a = a % 5
**=	a **= 4	a = a ** 4
//=	a //= 5	a = a // 5
&=	a &= 5	a = a & 5
|=	a |= 5	a = a | 5
^=	a ^= 5	a = a ^ 5
>>=	a >>= 5	a = a >> 5
<<=	a <<= 5	a = a << 5

Example of Python Assignment Operators

</> Open Compiler

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

```
print ("a: {} b: {} c : {}".format(a,b,c))
c = a + b
print ("a: {}  c = a + b: {}".format(a,c))

c += a
print ("a: {} c += a: {}".format(a,c))

c *= a
print ("a: {} c *= a: {}".format(a,c))

c /= a
print ("a: {} c /= a : {}".format(a,c))

c  = 2
print ("a: {} b: {} c : {}".format(a,b,c))
c %= a
print ("a: {} c %= a: {}".format(a,c))

c **= a
print ("a: {} c **= a: {}".format(a,c))

c //= a
print ("a: {} c //= a: {}".format(a,c))
```

Output

```
a: 21 b: 10 c : 0
a: 21  c = a + b: 31
a: 21 c += a: 52
a: 21 c *= a: 1092
a: 21 c /= a : 52.0
a: 21 b: 10 c : 2
a: 21 c %= a: 2
a: 21 c **= a: 2097152
a: 21 c //= a: 99864
```

Python Bitwise Operators

Python Bitwise operator works on bits and performs bit by bit operation. These operators are used to compare binary numbers.

The following table contains all bitwise operators with their symbols, names, and examples –

Operator	Name	Example
&	AND	a & b
|	OR	a | b
^	XOR	a ^ b
~	NOT	~a
<<	Zero fill left shift	a << 3
>>	Signed right shift	a >> 3

Example of Python Bitwise Operators

</> Open Compiler

```
a = 20
b = 10

print ('a=',a,':',bin(a),'b=',b,':',bin(b))
c = 0

c = a & b;
print ("result of AND is ", c,':',bin(c))

c = a | b;
print ("result of OR is ", c,':',bin(c))

c = a ^ b;
print ("result of EXOR is ", c,':',bin(c))

c = ~a;
print ("result of COMPLEMENT is ", c,':',bin(c))

c = a << 2;
print ("result of LEFT SHIFT is ", c,':',bin(c))
```

```
c = a >> 2;
print ("result of RIGHT SHIFT is ", c,':',bin(c))
```

Output

```
a= 20 : 0b10100 b= 10 : 0b1010
result of AND is  0 : 0b0
result of OR is  30 : 0b11110
result of EXOR is  30 : 0b11110
result of COMPLEMENT is  -21 : -0b10101
result of LEFT SHIFT is  80 : 0b1010000
result of RIGHT SHIFT is  5 : 0b101
```

Python Logical Operators

Python logical operators are used to combile two or more conditions and check the final result. There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

The following table contains all logical operators with their symbols, names, and examples –

Operator	Name	Example
and	AND	a and b
or	OR	a or b
not	NOT	not(a)

Example of Python Logical Operators

</> Open Compiler

```
var = 5

print(var > 3 and var < 10)
print(var > 3 or var < 4)
print(not (var > 3 and var < 10))
```

Output

```
True
True
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	Returns True if it finds a variable in the specified sequence, false otherwise.	a in b
not in	returns True if it does not finds a variable in the specified sequence and false otherwise.	a not in b

Example of Python Membership Operators

</> Open Compiler

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

print ("a:", a, "b:", b, "list:", list)

if ( a in list ):
   print ("a is present in the given list")
else:
   print ("a is not present in the given list")

if ( b not in list ):
   print ("b is not present in the given list")
else:
   print ("b is present in the given list")
```

```
c=b/a
print ("c:", c, "list:", list)
if ( c in list ):
   print ("c is available in the given list")
else:
    print ("c is not available in the given list")
```

Output

```
a: 10 b: 20 list: [1, 2, 3, 4, 5]
a is not present in the given list
b is not present in the given list
c: 2.0 list: [1, 2, 3, 4, 5]
c is available in the given list
```

Python Identity Operators

Python identity operators compare the memory locations of two objects.

There are two Identity operators explained below –

Operator	Description	Example
is	Returns True if both variables are the same object and false otherwise.	a is b
is not	Returns True if both variables are not the same object and false otherwise.	a is not b

Example of Python Identity Operators

</> Open Compiler

```
a = [1, 2, 3, 4, 5]
b = [1, 2, 3, 4, 5]
c = a

print(a is c)
print(a is b)
```

```
print(a is not c)
print(a is not b)
```

Output

```
True
False
False
True
```

Python Operators Precedence

Operators precedence decides the order of the evaluation in which an operator is evaluated. Python operators have different levels of precedence. The following table contains the list of operators having highest to lowest 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

Read more about the Python operators precedence here: Python operators precedence