# Python Operators

‹ Previous Next ›

## Python Operators

Operators are used to perform operations on variables and values.

In the example below, we use the `+` operator to add together two values:

### Example

Get your own Python Server

```
print(10 + 5)
```

Run example »

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

## Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

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

## Python Assignment Operators

Assignment operators are used to assign values to variables:

| Operator | Example | Same As | Try it |
|---|---|---|---|

| | | | |
|---|---|---|---|
| -= | x -= 3 | x = x - 3 | Try it » |
| *= | x *= 3 | x = x * 3 | Try it » |
| /= | x /= 3 | x = x / 3 | Try it » |
| %= | x %= 3 | x = x % 3 | Try it » |
| //= | x //= 3 | x = x // 3 | Try it » |
| **= | x **= 3 | x = x ** 3 | Try it » |
| &= | x &= 3 | x = x & 3 | Try it » |
| \|= | x \|= 3 | x = x \| 3 | Try it » |
| ^= | x ^= 3 | x = x ^ 3 | Try it » |
| >>= | x >>= 3 | x = x >> 3 | Try it » |
| <<= | x <<= 3 | x = x << 3 | Try it » |
| := | print(x := 3) | x = 3<br>print(x) | Try it » |

## Python Comparison Operators

Comparison operators are used to compare two values:

| Operator | Name | Example | Try it |
|---|---|---|---|
| == | Equal | x == y | Try it » |
| != | Not equal | x != y | Try it » |
| > | Greater than | x > y | Try it » |
| < | Less than | x < y | Try it » |
| >= | Greater than or equal to | x >= y | Try it » |
| <= | Less than or equal to | x <= y | Try it » |

## Python Logical Operators

Logical operators are used to combine conditional statements:

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

## Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

| | | | |
|---|---|---|---|
| is not | Returns True if both variables are not the same object | x is not y | Try it » |

## Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

| Operator | Description | Example | Try it |
|---|---|---|---|
| in | Returns True if a sequence with the specified value is present in the object | x in y | Try it » |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y | Try it » |

## Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

| Operator | Name | Description | Example | Try it |
|---|---|---|---|---|
| & | AND | Sets each bit to 1 if both bits are 1 | x & y | Try it » |
| \| | OR | Sets each bit to 1 if one of two bits is 1 | x \| y | Try it » |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 | x ^ y | Try it » |
| ~ | NOT | Inverts all the bits | ~x | Try it » |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off | x << 2 | Try it » |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off | x >> 2 | Try it » |

## Operator Precedence

Operator precedence describes the order in which operations are performed.

### Example

Parentheses has the highest precedence, meaning that expressions inside parentheses must be evaluated first:

```
print((6 + 3) - (6 + 3))
```

Run example »

### Example

Multiplication `*` has higher precedence than addition `+`, and therefor multiplications are evaluated before additions:

```
print(100 + 5 * 3)
```

Run example »

| | | |
|---|---|---|
| `()` | Parentheses | Try it » |
| `**` | Exponentiation | Try it » |
| `+x -x ~x` | Unary plus, unary minus, and bitwise NOT | Try it » |
| `* / // %` | Multiplication, division, floor division, and modulus | Try it » |
| `+ -` | Addition and subtraction | Try it » |
| `<< >>` | Bitwise left and right shifts | Try it » |
| `&` | Bitwise AND | Try it » |
| `^` | Bitwise XOR | Try it » |
| `\|` | Bitwise OR | Try it » |
| `== != > >= < <= is is not in not in` | Comparisons, identity, and membership operators | Try it » |
| `not` | Logical NOT | Try it » |
| `and` | AND | Try it » |
| `or` | OR | Try it » |

If two operators have the same precedence, the expression is evaluated from left to right.

## Example

Addition `+` and subtraction `-` has the same precedence, and therefor we evaluate the expression from left to right:

```
print(5 + 4 - 7 + 3)
```

Run example »

## Test Yourself With Exercises

### Exercise:

Multiply `10` with `5`, and print the result.

```
print(10   5)
```

Submit Answer »

Start the Exercise