

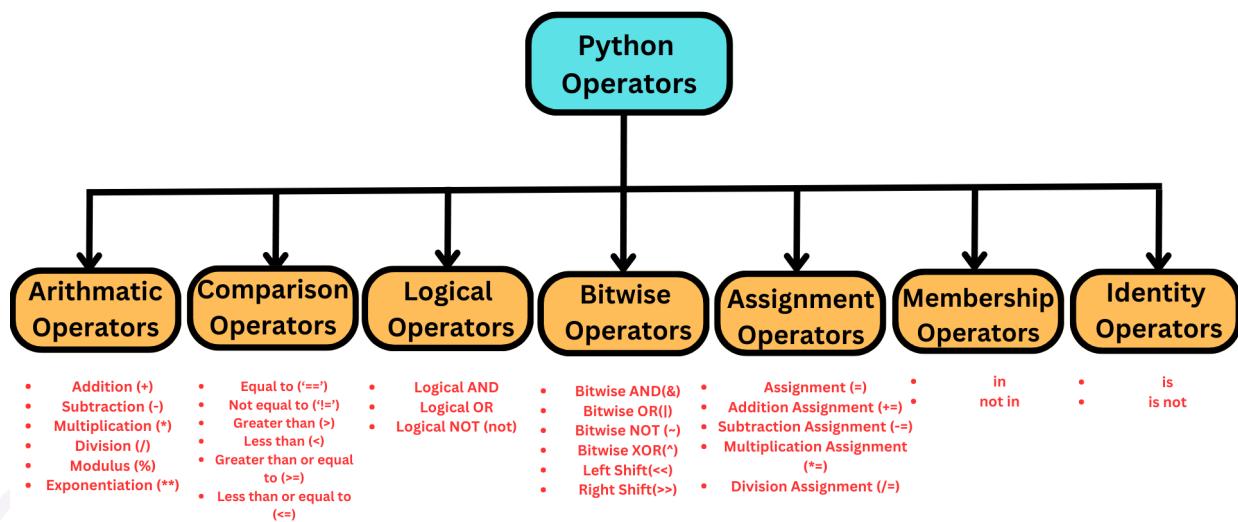
Operators in python

Lesson Plan



Today's Checklist:

- 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:

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

Python Assignment Operators

Assignment operators are used to assign values to variables:

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

Python Comparison Operators

Comparison operators are used to compare two values:

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

Python Logical Operators

Logical operators are used to combine conditional statements:

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

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:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

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

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

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

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))
```

Example

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

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

The precedence order is described in the table below, starting with the highest precedence at the top:

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

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

Example

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

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