Operators in Python

Python is a versatile language with a wide range of operators to perform various tasks. Whether you're working on simple arithmetic calculations or complex logical operations, Python operators have got you covered. In this article, we'll explore the different types of operators in Python and their usage.

Arithmetic Operators in Python

Addition (+)

The + operator adds two operands. For example, 2 + 3 equals 5.

Subtraction (-)

The - operator subtracts the second operand from the first. For example, 5 - 2 equals 3.

Multiplication (*)

The * operator multiplies two operands. For example, 2 * 3 equals 6.

Division (/)

The / operator divides the first operand by the second operand. For example, 6 / 3 equals 2.

Python also has additional arithmetic operators:

- Modulus (%): Returns the remainder of a division operation.

For example, 5 % 2 equals 1.

- Exponent (**): Raises the first operand to the power of the second operand.

For example, 2 ** 3 equals 8.

- Floor Division (//): Performs integer division and rounds down to the nearest integer.

For example, 7 // 2 equals 3.

Comparison Operators in Python

Operator	Description
==	Returns True if the two operands are equal.
!=	Returns True if the two operands are not equal.
>	Returns True if the first operand is greater than the second.
<	Returns True if the first operand is less than the second.
>=	Returns True if the first operand is greater than or equal to the second.
<=	Returns True if the first operand is less than or equal to the second.

Assignment Operators in Python

Assignment operators are used to assign values to variables.

=

Assigns the value on the right to the variable on the left.

+=

Adds the value on the right to the variable on the left.

-=

Subtracts the value on the right from the variable on the left.

*=

Multiplies the variable on the left by the value on the right.

Logical Operators

> "The logical operators in Python allow you to check if two or more conditions are true at the same time. Logical operators are especially useful when working with control flow statements such as if-else statements."
>
> —Python documentation

and

Returns True if both operands are True.

or

Returns True if either operand is True.

not

Returns True if the operand is False.

Membership Operators in Python

Membership operators are used to test whether a value is a member of a sequence or not.

Operator	Description
`in`	Returns `True` if a value is found in a sequence.
`not in`	Returns `True` if a value is not found in a sequence.

Bitwise Operators in Python

Bitwise operators are used to manipulate the bits in an integer. They are used to perform tasks such as setting, clearing, or checking specific bits in an integer.

1

AND (&)

Returns a 1 in each bit position where both operands have a 1.

2

OR (|)

Returns a 1 in each bit position where at least one operand has a 1.

3

XOR (^)

Returns a 1 in each bit position where only one operand has a 1.

4

NOT (~)

Inverts all the bits of the operand.

5

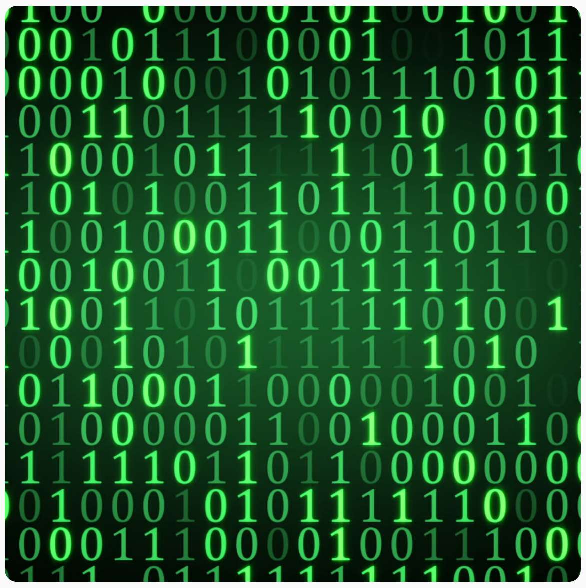
Left Shift (<<)

Shifts the bits of the first operand to the left by the number of positions specified by the second operand.

6

Right Shift (>>)

Shifts the bits of the first operand to the right by the number of positions specified by the second operand.



Binary code is at the heart of bitwise operators.

Bitwise operations are implemented in computer hardware to perform operations at a low level.

Conclusion

1

Operators in Python

Python operators are easy to use and offer a wide range of functionality. Whether you're working with numbers, lists, or more complex data types, Python operators make it easy to perform tasks quickly and efficiently.

2

Mastering Python Operators

By using these different types of operators correctly, you'll be able to write code that is concise, maintainable, and efficient.