Operators

An introduction to basic operator types in Python, including their syntax and most common uses. This material will cover how operators can be used to perform various calculations and operations within a Python program.

`Operators` are special symbols in Python that carry out arithmetic computation or logical comparison. They operate on values and variables (referred to as operands). Let's explore various types of operators in Python.

Arithmetic Operators

You're already familiar with some basic arithmetic operators:

- + Addition operator
- – Subtraction operator
- * Multiplication operator
- / Division operator

Python also includes these commonly used arithmetic operators:

Modulus (Division Remainder Operator)

The modulus `%` returns the remainder of a division operation.

- `15 % 3` will result in `0`, as 15 is divisible by 3 without remainder.
- `15 % 2` will result in `1`, as 15 divided by 2 leaves a remainder of 1.

Exponentiation

The exponentiation operator `**` raises the first number to your chosen power.

- `2 ** 3` will result in 8, as it represents (2^3).

Floor Division (aka integer division)

The integer division operator `//` divides two numbers (not only integers) and returns an integer, rounding the result down to the nearest whole number (ignoring anything after the decimal point).

- `7 // 2` will result in `3`.

To sum it up, you can perform the following arithmetic operations:

Operator	Function	Example	Result
+	addition	5+2	7
–	subtraction	5-2	3
*	multiplication	5*2	10
/	division	5/2	2.5
%	modulus	5%2	1
**	exponentiation (power)	5**2	25
//	floor division	5//2	2

Assignment Operators

Assignment operators are used to assign values to variables. In addition to the basic = operator, Python includes operators that combine assignment with arithmetic operations:

- +=: Adds a value and assigns the result. `my_age += 1` is equivalent to `my_age = my_age + 1`.
- -=: Subtracts a value and assigns the result.
- *=: Multiplies by a value and assigns the result.
- /=: Divides by a value and assigns the result.

You can use all the arithmetic operations introduced earlier.

Comparison Operators

Comparison operators are used to compare values:

- ==: Equal to. `4 == 4` will be `True`.
- !=: Not equal to. `4 != 5` will be `True`.
- >: Greater than.
- <: Less than.
- >=: Greater than or equal to.
- <=: Less than or equal to.

Logical Operators

Logical operators are used to combine conditional statements:

- `and`: Returns `True` if both statements are true.
- `or`: Returns `True` if at least one of the statements is true.
- `not`: Reverses the result, returns `False` if the result is true.

Example: `4 == 4 and 3 == 3` will be `True`.

Comparison of Data Structures

In Python, comparing data structures like lists, tuples, or dictionaries checks whether they are the same object, not just if they have the same content:

- Comparing different objects with identical contents (like two different lists with the same elements) will result in `False`.

```
my_data = {"age": 18}
your_data = {"age": 18}

# This will be False
my_data == your_data
```

However, comparing the contents of these data structures, if they contain primitive types, can be `True`.

```
# This will be True
my_data["age"] == your_data["age"]
```

Complete Lesson