Reference guide: Python operators

You've encountered many Python operators already. Many of them likely feel very familiar to you. After all, there's nothing novel about addition and subtraction in Python. But there are many more operators than the ones used for basic arithmetic! Operators are characters that enact specific arithmetic, logical actions, or processes. Data professionals use operators all the time in their work, and they're a rudimentary part of Python programming, so it's important to learn them. This reading is a guide to the various operators available to you in Python.

Save this course item

You may want to save a copy of this guide for future reference. You can use it as a resource for additional practice or in your future professional projects. To access a downloadable version of this course item, click the following link and select "Use Template."

Reference guide: Python operators

OR

If you don't have a Google account, you can download the item directly from the following attachment.

Reference guide_ Python operators
DOCX File

Comparators

In Python, you can use comparison operators to compare values. When a comparison is made, Python returns a Boolean result—True or False. Python uses the following comparators:

Operation	Operator
greater than	`>`
greater than or equal to	`>=`
less than	`<`
less than or equal to	`<=`
not equal to	`!=`
equal to	`==`

Notes:

- The single equals sign (`=`) is reserved for assignment statements. If you use a single equal sign to make a comparison, the computer will return a **SyntaxError**.
- If you try to compare data types that aren't compatible, like checking if a string is greater than an integer, Python will throw a **TypeError**.

Logical operators

Python also has three logical operators that can be combined with comparators to create more complex statements.

These operators are:

- **`and`**
  - evaluates to True only if both statements are true
- **`or`**
  - evaluates to True if one or both of the statements are true
- **`not`**
  - reverses the evaluation
  - If the statement evaluates to True, returns False; if the statement evaluates to False, returns True

Examples:

```
x = 3
my_list = [3, 4, 6, 10]
print(x < 3 and x != 0)
print(x >= len(my_list) or x == min(my_list))
print(x not in my_list)
```

RunReset

Arithmetic operators

Python is also capable of performing mathematical operations using a set of built-in operators. These arithmetic operators are:

Operation	Operator	Example
Addition	`+`	`[IN] 5 + 2` `[OUT] 7`
Subtraction	`-`	`[IN] 5 - 2` `[OUT] 3`
Multiplication	`*`	`[IN] 5 * 2` `[OUT] 10`
Division	`/`	`[IN] 5 / 2`

Operation	Operator	Example
Modulo (the remainder of a division)	`%`	`[OUT] 2.5` `[IN] 5 % 2`
Exponentiation	`**`	`[OUT] 1` `[IN] 5 ** 2`

Floor division
(the number of times the denominator can fully go into the numerator)
	`//`	`[OUT] 25`
		`[IN] 5 // 2`
		`[OUT] 2`

There are many other mathematical operations that can be performed in Python using functions from special libraries, which you'll learn about later. Python uses a core set of operators to make comparisons, perform logical operations, and compute arithmetic operations. These operators can be combined in statements to perform an infinite number of tasks and operations.