Arithmetic Operators

In this lesson, we'll learn how to perform calculations using arithmetic operators.

We'll cover the following

Below, we can find the basic arithmetic operators in order of **precedence**. The operator listed higher will be computed first.

These operators allow us to perform arithmetic operations in Python.

Operator	Purpose	Notation
`()`	Parentheses	Encapsulates the Precedent Operation
`**`	Exponent	In-fix
`%`, `*`, `/`, `//`	Modulo, Multiplication, Division, Floor Division	In-fix
`+`, `-`	Addition, Subtraction	In-fix

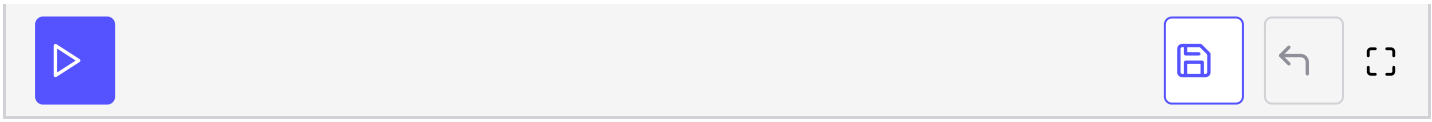
Addition

We can add two numbers using the `+` operator:

```
print(10 + 5)

float1 = 13.65
float2 = 3.40
print(float1 + float2)

num = 20
flt = 10.5
print(num + flt)
```



As we can see in **line 9**, summing an integer and floating-point number gives us a floating-point number.

Python automatically converts the integer to a floating-point number. This applies to all arithmetic operations.

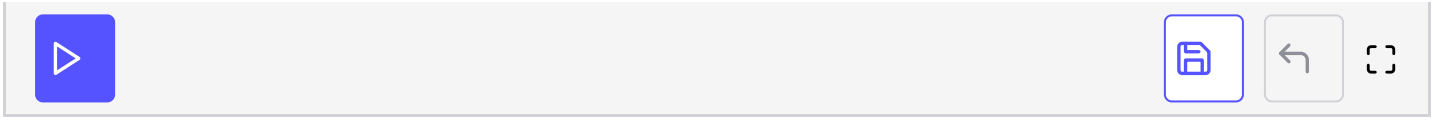
Subtraction

We can subtract one number from the other using the `-` operator:

```
print(10 - 5)

float1 = -18.678
float2 = 3.55
print(float1 - float2)

num = 20
flt = 10.5
print(num - flt)
```



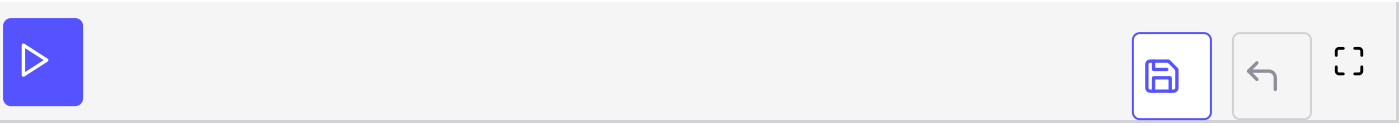
Multiplication

We can multiply two numbers using the `*` operator:

```
print(40 * 10)

float1 = 5.5
float2 = 4.5
print(float1 * float2)
```

```
print(10.2 * 3)
```

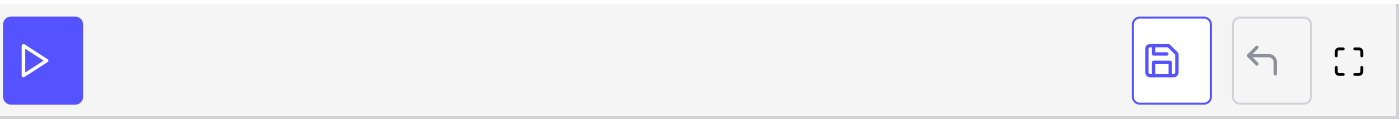


Division

We can divide one number by another using the `/` operator:

```
print(40 / 10)

float1 = 5.5
float2 = 4.5
print(float1 / float2)
print(12.4 / 2)
```



A division operation always results in a floating-point number.

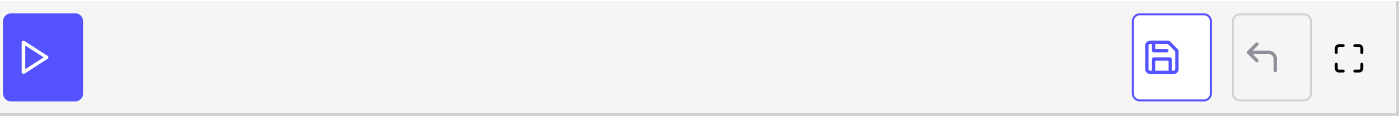
Floor Division

In floor division, the result is *floored* to the nearest smaller integer. It is also known as **integer division**.

For floor division, we must use the `//` operator:

```
print(43 // 10)

float1 = 5.5
float2 = 4.5
print(5.5 // 4.5)
print(12.4 // 2)
```



Unlike normal division, floor division between two integers results in an integer.

Modulo

A number's modulo with another number can be found using the `%` operator:

```
print(10 % 2)

twenty_eight = 28
print(twenty_eight % 10)
```

```
print(-28 % 10)  # The remainder is positive if the right-hand operand is positive


print(28 % -10)  # The remainder is negative if the right-hand operand is negative
print(34.4 % 2.5)  # The remainder can be a float
```

Precedence

An arithmetic expression containing different operators will be computed on the basis of **operator precedence**.

Whenever operators have equal precedence, the expression is computed from the left side:

```
# Different precedence
print(10 - 3 * 2)  # Multiplication computed first, followed by subtraction

# Same precedence
print(3 * 20 / 5)  # Multiplication computed first, followed by division
print(3 / 20 * 5)  # Division computed first, followed by multiplication
```

Parentheses

An expression which is enclosed inside parentheses will be computed first, regardless of operator precedence:

```
print((10 - 3) * 2)  # Subtraction occurs first
print((18 + 2) / (10 % 8))
```

Using all the operations above, we can compute complex mathematical expressions in Python!

Keep in mind that we are never restricted to these operators. There are countless more arithmetic utilities at our disposal, some of which will be discussed soon.

The next lesson will cover the comparison operators.