

Python as a Calculator

Arithmetic Operators

Operator	Operation	Expression	English description	Result
+	addition	11 + 56	11 plus 56	67
-	subtraction	23 - 52	23 minus 52	-29
*	multiplication	4 * 5	4 multiplied by 5	20
**	exponentiation	2 ** 5	2 to the power of 5	32
/	division	9 / 2	9 divided by 2	4.5
//	integer division	9 // 2	9 divided by 2	4
%	modulo (remainder)	9 % 2	9 mod 2	1

Types `int` and `float`

A *type* is a set of values and operations that can be performed on those values.

Two of Python's numeric types:

- `int`: integer
For example: 3, 4, 894, 0, -3, -18
- `float`: floating point number (an approximation to a real number)
For example: 5.6, 7.342, 53452.0, 0.0, -89.34, -9.5

Arithmetic Operator Precedence

When multiple operators are combined in a single expression, the operations are evaluated in order of precedence.

Operator	Precedence
**	highest
- (negation)	
*, /, //, %	
+ (addition), - (subtraction)	lowest

Syntax and Semantics

Syntax: the rules that describe valid combinations of Python symbols

Semantics: the meaning of a combination of Python symbols is the meaning of an instruction — what a particular combination of symbols does when you execute it.

Errors

A syntax error occurs when an instruction with invalid syntax is executed. For example:

```
>>> 3) + 2 * 4
SyntaxError: invalid syntax
```

A semantic error occurs when an instruction with invalid semantics is executed. For example:

```
>>> 89.4 / 0
Traceback (most recent call last):
  File "", line 1, in
    89.4 / 0
ZeroDivisionError: float division by zero
```