|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Symbol** |  |  |  |  |
| Additioin | + | 5+9 | 14 |  |  |
| Subtraction | - | 4-3 | 1 |  |  |
| Mutliplication | \* | 4\*3 | 12 |  |  |
| Exponentiation | \*\* | 3\*\*3 | 27 |  |  |
| Division | / | 6/2 | 3.0 |  |  |
| Integer division | // | 5//3 | 1 |  |  |

**Two of Python’s numeric types**

Int: integer

Float: floating point number

An approximation to a real number

**Operator precedence**

\*\* highest precedence

- (negation)

\* / // %

+ - lowest precedence

*Syntax:* the rules that describe valid combinations of Python symbols.

*Semantics:* the meaning of a combination of Python symbols.

**Order of operations**

3 + 4 - 5 (right to left)

4 + 5 \* 3 (multiply, addition)

-10 \* 3 + 5 \*\* 3 = (5 \*\*3) + (-10 \* 3)

Multiplication and division have higher precedence than addition and subtraction; operators are otherwise applied left to right

7\*3+4/2 = (7\*3)+\*4/2)

**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 we 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

<https://d3c33hcgiwev3.cloudfront.net/_b57eab3bf52075f38d322bb0e4b43ee1_calc.html?Expires=1508371200&Signature=WfAQDsoOQykkxKxrm6TVFMObf4ZKO8acYLUcRI9veT-OWxc97w61wpopzSUGoM3zbjg6PBn0xQgfMs6pz2mUeRWhmA~2hcB6EshQTSVhDUYUdj1XB1RF845zLb1isXQcBDQJFDe4vq0UnzyaCbrMIRYeE4hIhGqDaEDpxYYLQ6c_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A>

abs(-1000) = 1000 = abs(-90) = 90

max() for know which bigger number

eg: max(78, 39) = 78

dir (\_\_builtins\_\_) = for builtin functions list

help (function name)