Operator precedence

The following table summarizes the operator precedence in Python, from lowest precedence (least binding) to highest precedence (most binding). Operators in the same box have the same precedence. Unless the syntax is explicitly given, operators are binary. Operators in the same box group left to right (except for exponentiation, which groups from right to left).

Note that comparisons, membership tests, and identity tests, all have the same precedence and have a left-to-right chaining feature as described in the [Comparisons](https://docs.python.org/3/reference/expressions.html#comparisons) section.

| **Operator** | **Description** |
| --- | --- |
| := | Assignment expression |
| [lambda](https://docs.python.org/3/reference/expressions.html#lambda) | Lambda expression |
| [if](https://docs.python.org/3/reference/expressions.html#if-expr) – else | Conditional expression |
| [or](https://docs.python.org/3/reference/expressions.html#or) | Boolean OR |
| [and](https://docs.python.org/3/reference/expressions.html#and) | Boolean AND |
| [not](https://docs.python.org/3/reference/expressions.html#not) x | Boolean NOT |
| [in](https://docs.python.org/3/reference/expressions.html#in), [not in](https://docs.python.org/3/reference/expressions.html#not-in), [is](https://docs.python.org/3/reference/expressions.html#is), [is not](https://docs.python.org/3/reference/expressions.html#is-not), <, <=, >, >=, !=, == | Comparisons, including membership tests and identity tests |
| | | Bitwise OR |
| ^ | Bitwise XOR |
| & | Bitwise AND |
| <<, >> | Shifts |
| +, - | Addition and subtraction |
| \*, @, /, //, % | Multiplication, matrix multiplication, division, floor division, remainder [5](https://docs.python.org/3/reference/expressions.html#id21) |
| +x, -x, ~x | Positive, negative, bitwise NOT |
| \*\* | Exponentiation [6](https://docs.python.org/3/reference/expressions.html#id22) |
| [await](https://docs.python.org/3/reference/expressions.html#await) x | Await expression |
| x[index], x[index:index], x(arguments...),  , x.attribute | Subscription, slicing, call, attribute reference |
| (expressions...),  [expressions...], {key: value...}, {expressions...} | Binding or parenthesized expression, list display, dictionary display, set display |

 Lambdas

**lambda\_expr**  ::= "lambda" [[parameter\_list](https://docs.python.org/3/reference/compound_stmts.html" \l "grammar-token-parameter-list)] ":" [expression](https://docs.python.org/3/reference/expressions.html#grammar-token-expression)

**lambda\_expr\_nocond** ::= "lambda" [[parameter\_list](https://docs.python.org/3/reference/compound_stmts.html" \l "grammar-token-parameter-list)] ":" [expression\_nocond](https://docs.python.org/3/reference/expressions.html#grammar-token-expression-nocond)

Lambda expressions (sometimes called lambda forms) are used to create anonymous functions. The expression lambda parameters: expression yields a function object. The unnamed object behaves like a function object defined with:

def <lambda>(parameters):

return expression

See section [Function definitions](https://docs.python.org/3/reference/compound_stmts.html#function) for the syntax of parameter lists. Note that functions created with lambda expressions cannot contain statements or annotations.