Various Python Operators

[Python operators](https://www.simplilearn.com/tutorials/python-tutorial/operators-in-python) are used to perform operations on variables and values. The main operators in Python are:

Assignment Operators

Assignment operators assign values to variables.

|  |  |  |
| --- | --- | --- |
| Operator | Example | Same As |
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 3 | x = x \* 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 | x = x // 3 |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |
| &= | x &= 3 | x = x & 3 |
| |= | x |= 3 | x = x | 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

|  |  |  |
| --- | --- | --- |
| Operator | Name | Example |
| + | Addition | x + y |
| - | Subtraction | x - y |
| \* | Multiplication | x \* y |
| / | Division | x / y |
| % | Modulus | x % y |
| \*\* | Exponentiation | x \*\* y |
| // | Floor division | x // y |

Comparison Operators

Comparison operators are used to compare two values.

|  |  |  |
| --- | --- | --- |
| Operator | Name | Example |
| == | Equal | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

Logical Operators

Logical operators are used to combine conditional statements.

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| and | Returns True if both statements are true | x < 5 and  x < 10 |
| or | Returns True if one of the statements is true | x < 5 or x < 4 |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |

Membership Operators

Membership operators are used to test if a sequence is presented in an object.

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| in | Returns True if a sequence with the specified value is present in the object | x in y |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y |

Identity Operators

Identity operators are used to compare the objects to see if they are actually the same object with the same memory location.

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| is | Returns True if both variables are the same object | x is y |
| is not | Returns True if both variables are not the same object | x is not y |

Bitwise Operators

Bitwise operators are used to compare binary numbers.

|  |  |  |
| --- | --- | --- |
| Operator | Name | Description |
| & | AND | Sets each bit to 1 if both bits are 1 |
| | | OR | Sets each bit to 1 if one of two bits is 1 |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 |
| ~ | NOT | Inverts all the bits |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off |

Data Types

In Python, every value is referred to as an [object](https://www.simplilearn.com/tutorials/python-tutorial/objects-and-classes-in-python) and every object has a specific [data type.](https://www.simplilearn.com/tutorials/python-tutorial/data-types-in-python) The three most used data types in Python are integers, floating-point numbers, and string.

Integer (int)

An integer is used to represent an object such as “number 5”.

Example: -2, -1, 0, 1, 2, 3

Floating-Point Number (float)

Float is used to represent floating-point numbers.

Example: -1.25, -1.0, – 0.5, 0.0, 0.5, 1.0, 1.25

String

A [string](https://www.simplilearn.com/tutorials/python-tutorial/python-strings) is used to modify a sequence of characters. For example, the word “hello”. Also, remember that strings are immutable in Python. This means that if you have already defined one, you cannot change it later on.

Example: ‘hello’, ‘bye’, hi’, ‘goodbye’

Other common data types in Python are lists, dictionaries, and tuples.

Variables

[Variables](https://www.simplilearn.com/tutorials/python-tutorial/python-variables) are used to temporarily store data in the computer’s memory.

Example:

price = 100

rating = 4.7

course\_name = ‘Python for Beginners’

published = True

In the above example, price is an integer (a whole number with no decimal points), rating is a floating-point number (a number with a decimal point), course\_name is a string (a sequence of characters), and published is a boolean.

Boolean values in Python can be True or False.

Functions

[Functions in Python](https://www.simplilearn.com/tutorials/python-tutorial/python-functions) are used to break up the code into small chunks. These chunks are easier to read and maintain. It is also easier to find bugs in a small chunk rather than going through the entire program. These chunks can also be reused.

Example:

def greet\_user(name):

print(f”Hi {name}”)

greet\_user(“John”)

Parameters are the placeholders for data to pass through functions. Arguments are the values passed.

There are two types of values - positional arguments (their position matters) and keyword arguments (their position doesn’t matter).

Positional Argument Example:

greet\_user(“John”, “Smith”)

Keyword Argument Example:

calculate\_total(order=30, shipping=8, tax=0.1)

Functions can return values. If in case you don’t use the return statement, None is returned by default. None represents the absence of a value.

Example:

def square(number):

return number \* number

result = square(2)

print(result)

The above function will print 4 as a result.

Flow Control

Comparison Operators

Comparison operators are used to compare two values and evaluate True or False depending on the values you give them.

|  |  |
| --- | --- |
| Operator | Meaning |
| == | Equal to |
| != | Not equal to |
| < | Less than |
| > | Greater Than |
| <= | Less than or Equal to |
| >= | Greater than or Equal to |

Example:

Python_Cheat_Sheet_1

Python_Cheat_Sheet_2

Boolean Evaluation

If you want to evaluate a boolean operation in Python, never use == or != operators.

Use the “is” or “is not” operators, or use implicit boolean evaluation.

Example:

Python_Cheat_Sheet_3

Python_Cheat_Sheet_4

Boolean Operators

There are three Boolean operators in Python - and, or, and not.

The “or” operator’s Truth Table:

|  |  |
| --- | --- |
| Expression | Value |
| True or True | True |
| True or False | True |
| False or True | True |
| False or False | False |

The “or not” operator’s Truth Table:

|  |  |
| --- | --- |
| Expression | Evaluates to |
| not True | False |
| not False | True |

The “and” operator’s Truth Table:

|  |  |
| --- | --- |
| Expression | Value |
| True and True | True |
| True and False | False |
| False and True | False |
| False and False | False |

Example:

Python_Cheat_Sheet_5

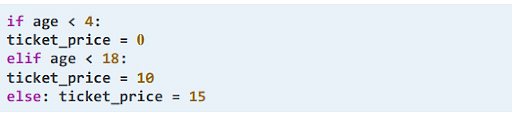
Python_Cheat_Sheet_6.

Python_Cheat_Sheet_7.

If-Else Statements

The if-else statement is used to execute both the true and false parts of a given condition. If the condition is true, the “if” block code is executed. If the condition is false, the “else” block code is executed. The elif keyword prompts your program to try another condition if the previous one is not true.

Example:



Python Loops

Python has two loop commands that are good to know - for loops and while loops

For Loop

The for loop is used to iterate over a sequence such as a list, string, tuple, etc.

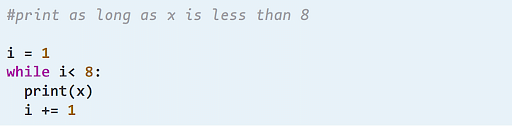
Example:

Python_Cheat_Sheet_9.

While Loop

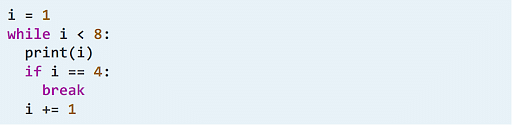
The while loop enables you to execute a set of statements as long as the condition is true.

Example:



Break and Continue

The break and continue statements are used to alter the loops even if the condition is met. It can be used in, both, for and while loops.



Loop and Range Functions

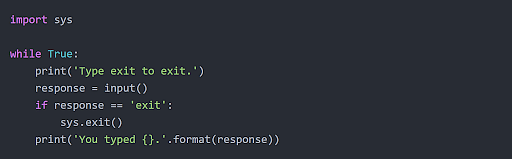
If you want to loop through a set of code a number of times, you can use the range() function. It returns a sequence of numbers, starting by 0 from default, increments by 1 (default), and ends at a specified number.

Python_Cheat_Sheet_12

Exit and Sys.exit

The exit() and sys.exit() both have the same functionality as they raise the SystemExit exception where the Python interpreter exits and no stack traceback is printed. The main difference is that exit is a helper for the interactive shell and sys.exit is used in programs.

Example:



Return Values and Return Statements

A return statement is used at the end of the function call and returns the result or the value of the expression to the caller. Any statement after the return statements is not executed. In case the return statement does not have any expression, then “None” is returned.

Keyword Print

The print() function in Python is used to print the specified message on the screen. This message can be a string or any other object. In case of an object, it will be converted into a string before being written to the screen.

Syntax:

print(object(s), sep=separator, end=end, file=file, flush=flush)

Example:

Python_Cheat_Sheet_14

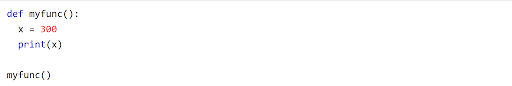
Local and Global Scope

Whenever we create a variable in PYthon, it is only available inside the region in which it is created. This is called the scope of the variable.

Local Scope

A variable created inside a function that can only be used in that function belongs to the local scope of that function.

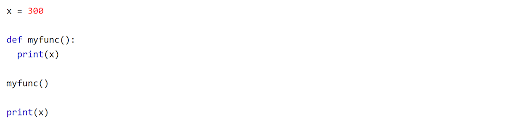
Example:



Global Scope

A variable created in the main body of the code is called a global variable. It belongs to the global scope and is available from within any scope, global and local.

Example:



Exception Handling

Exceptions are errors that crash programs. In Python, we can handle exceptions using try and except statements.

Example:

