## What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

### What can Python do?

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

### Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-oriented way or a functional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

### Example

print("Hello, World!")

P-2

# Python Getting Started

[❮ Previous](https://www.w3schools.com/python/python_intro.asp)[Next ❯](https://www.w3schools.com/python/python_syntax.asp)

## Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\*Your Name*>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

## Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\*Your Name*>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_helloworld)

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\*Your Name*>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

## The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\*Your Name*>python

Or, if the "python" command did not work, you can try "py":

C:\Users\*Your Name*>py

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\*Your Name*>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> print("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\*Your Name*>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> print("Hello, World!")

Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

P-3

# Python Syntax

[❮ Previous](https://www.w3schools.com/python/python_getstarted.asp)[Next ❯](https://www.w3schools.com/python/python_comments.asp)

## On this page

[Execute Python Syntax](https://www.w3schools.com/python/python_syntax.asp#execute_python_syntax)

[Python Indentation](https://www.w3schools.com/python/python_syntax.asp#python_indentation)

[Python Variables](https://www.w3schools.com/python/python_syntax.asp#python_variables)

[Python Comments](https://www.w3schools.com/python/python_syntax.asp#python_comments)

[Exercises](https://www.w3schools.com/python/python_syntax.asp#exercises)

## Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

>>> print("Hello, World!")

Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users\*Your Name*>python myfile.py

## Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

### Example

if 5 > 2:

print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation)

Python will give you an error if you skip the indentation:

### Example

Syntax Error:

if 5 > 2:

print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation_test)

The number of spaces is up to you as a programmer, but it has to be at least one.

### Example

if 5 > 2:

print("Five is greater than two!")

if 5 > 2:

print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation2)

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

### Example

Syntax Error:

if 5 > 2:

print("Five is greater than two!")

print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation2_error)

## Python Variables

In Python, variables are created when you assign a value to it:

### Example

Variables in Python:

x = 5

y = "Hello, World!"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_syntax_variables)

Python has no command for declaring a variable.

You will learn more about variables in the [Python Variables](https://www.w3schools.com/python/python_variables.asp) chapter.

## Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

### Example

Comments in Python:

#This is a comment.

print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment)

## Test Yourself With Exercises

## Exercise:

Insert the missing part of the code below to output "Hello World".

("Hello World")

Submit Answer »

[Start the](https://www.w3schools.com/python/exercise.asp?filename=exercise_syntax1) Exercisep

# Python Comments

[❮ Previous](https://www.w3schools.com/python/python_syntax.asp)[Next ❯](https://www.w3schools.com/python/python_variables.asp)

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

## Creating a Comment

Comments starts with a #, and Python will ignore them:

### Example

#This is a comment

print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment1)

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

### Example

print("Hello, World!") #This is a comment

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment2)

A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code:

### Example

#print("Hello, World!")

print("Cheers, Mate!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment3)

## Multi Line Comments

Python does not really have a syntax for multi line comments.

To add a multiline comment you could insert a # for each line:

### Example

#This is a comment

#written in

#more than just one line

print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment4)

Or, not quite as intended, you can use a multiline string.

Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

### Example

"""

This is a comment

written in

more than just one line

"""

print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment5)

As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multiline comment.

## Test Yourself With Exercises

## Exercise:

Comments in Python are written with a special character, which one?

This is a comment

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_comments1)

## Variables

Variables are containers for storing data values.

## Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

### Example

x = 5

y = "John"

print(x)

print(y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables1)

Variables do not need to be declared with any particular *type*, and can even change type after they have been set.

### Example

x = 4 # x is of type int

x = "Sally" # x is now of type str

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables2)

## Casting

If you want to specify the data type of a variable, this can be done with casting.

### Example

x = str(3) # x will be '3'

y = int(3) # y will be 3

z = float(3) # z will be 3.0

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_casting)

## Get the Type

You can get the data type of a variable with the type() function.

### Example

x = 5

y = "John"

print(type(x))

print(type(y))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_type)

You will learn more about [data types](https://www.w3schools.com/python/python_datatypes.asp) and [casting](https://www.w3schools.com/python/python_casting.asp) later in this tutorial.

## Single or Double Quotes?

String variables can be declared either by using single or double quotes:

### Example

x = "John"

# is the same as

x = 'John'

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables7)

## Case-Sensitive

Variable names are case-sensitive.

### Example

This will create two variables:

a = 4

A = "Sally"

#A will not overwrite a

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_cs)

[❮ Previous](https://www.w3schools.com/python/python_comments.asp)[Next ❯](https://www.w3schools.com/python/python_variables_names.asp)

# Python - Variable Names

[❮ Previous](https://www.w3schools.com/python/python_variables.asp)[Next ❯](https://www.w3schools.com/python/python_variables_multiple.asp)

## Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)

### Example

Legal variable names:

myvar = "John"

my\_var = "John"

\_my\_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variable_names_legal)

### Example

Illegal variable names:

2myvar = "John"

my-var = "John"

my var = "John"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variable_names_error)

Remember that variable names are case-sensitive

## Multi Words Variable Names

Variable names with more than one word can be difficult to read.

There are several techniques you can use to make them more readable:

## Camel Case

Each word, except the first, starts with a capital letter:

myVariableName = "John"

## Pascal Case

Each word starts with a capital letter:

MyVariableName = "John"

## Snake Case

Each word is separated by an underscore character:

my\_variable\_name = "John"

# Python Variables - Assign Multiple Values

[❮ Previous](https://www.w3schools.com/python/python_variables_names.asp)[Next ❯](https://www.w3schools.com/python/python_variables_output.asp)

## Many Values to Multiple Variables

Python allows you to assign values to multiple variables in one line:

### Example

x, y, z = "Orange", "Banana", "Cherry"

print(x)

print(y)

print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables8)

Note: Make sure the number of variables matches the number of values, or else you will get an error.

## One Value to Multiple Variables

And you can assign the *same* value to multiple variables in one line:

### Example

x = y = z = "Orange"

print(x)

print(y)

print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables6)

## Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you extract the values into variables. This is called *unpacking*.

### Example

Unpack a list:

fruits = ["apple", "banana", "cherry"]

x, y, z = fruits

print(x)

print(y)

print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_unpack)

Learn more about unpacking in our [Unpack Tuples](https://www.w3schools.com/python/python_tuples_unpack.asp) Chapter.

# Python - Output Variables

[❮ Previous](https://www.w3schools.com/python/python_variables_multiple.asp)[Next ❯](https://www.w3schools.com/python/python_variables_global.asp)

## Output Variables

The Python print statement is often used to output variables.

To combine both text and a variable, Python uses the + character:

### Example

x = "awesome"

print("Python is " + x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables3)

You can also use the + character to add a variable to another variable:

### Example

x = "Python is "

y = "awesome"

z = x + y

print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables4)

For numbers, the + character works as a mathematical operator:

### Example

x = 5

y = 10

print(x + y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables5)

If you try to combine a string and a number, Python will give you an error:

### Example

x = 5

y = "John"

print(x + y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_test)

# Python - Global Variables

[❮ Previous](https://www.w3schools.com/python/python_variables_output.asp)[Next ❯](https://www.w3schools.com/python/python_variables_exercises.asp)

## Global Variables

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

### Example

Create a variable outside of a function, and use it inside the function

x = "awesome"

def myfunc():

print("Python is " + x)

myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global)

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

### Example

Create a variable inside a function, with the same name as the global variable

x = "awesome"

def myfunc():

x = "fantastic"

print("Python is " + x)

myfunc()

print("Python is " + x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global2)

## The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

### Example

If you use the global keyword, the variable belongs to the global scope:

def myfunc():

global x

x = "fantastic"

myfunc()

print("Python is " + x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global3)

Also, use the global keyword if you want to change a global variable inside a function.

### Example

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = "awesome"

def myfunc():

global x

x = "fantastic"

myfunc()

print("Python is " + x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global4)

# Python - Variable Exercises

[❮ Previous](https://www.w3schools.com/python/python_variables_global.asp)[Next ❯](https://www.w3schools.com/python/python_datatypes.asp)

## Test Yourself With Exercises

Now you have learned a lot about variables, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

## Exercise:

Create a variable named carname and assign the value Volvo to it.

= ""

Submit Answer »

Go to the Exercise section and test all of our Python Variable Exercises:

[Python Variable Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_variables1)

# Python Data Types

[❮ Previous](https://www.w3schools.com/python/python_variables_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_numbers.asp)

## Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

| Text Type: | str |
| --- | --- |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |

## Getting the Data Type

You can get the data type of any object by using the type() function:

### Example

Print the data type of the variable x:

x = 5

print(type(x))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_type)

## Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

| **Example** | **Data Type** | **Try it** |
| --- | --- | --- |
| x = "Hello World" | str | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_str) |
| x = 20 | int | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int) |
| x = 20.5 | float | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float) |
| x = 1j | complex | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex) |
| x = ["apple", "banana", "cherry"] | list | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list) |
| x = ("apple", "banana", "cherry") | tuple | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range) |
| x = {"name" : "John", "age" : 36} | dict | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict) |
| x = {"apple", "banana", "cherry"} | set | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set) |
| x = frozenset({"apple", "banana", "cherry"}) | frozenset | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset) |
| x = True | bool | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool) |
| x = b"Hello" | bytes | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray) |
| x = memoryview(bytes(5)) | memoryview | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview) |

## Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions:

| **Example** | **Data Type** | **Try it** |
| --- | --- | --- |
| x = str("Hello World") | str | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_str2) |
| x = int(20) | int | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int2) |
| x = float(20.5) | float | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float2) |
| x = complex(1j) | complex | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex2) |
| x = list(("apple", "banana", "cherry")) | list | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list2) |
| x = tuple(("apple", "banana", "cherry")) | tuple | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple2) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range2) |
| x = dict(name="John", age=36) | dict | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict2) |
| x = set(("apple", "banana", "cherry")) | set | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set2) |
| x = frozenset(("apple", "banana", "cherry")) | frozenset | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset2) |
| x = bool(5) | bool | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool2) |
| x = bytes(5) | bytes | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes2) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray2) |
| x = memoryview(bytes(5)) | memoryview | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview2) |

## Test Yourself With Exercises

## Exercise:

The following code example would print the data type of x, what data type would that be?

x = 5

print(type(x))

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_datatypes1)

# Python Numbers

[❮ Previous](https://www.w3schools.com/python/python_datatypes.asp)[Next ❯](https://www.w3schools.com/python/python_casting.asp)

## Python Numbers

There are three numeric types in Python:

* int
* float
* complex

Variables of numeric types are created when you assign a value to them:

### Example

x = 1 # int

y = 2.8 # float

z = 1j # complex

To verify the type of any object in Python, use the type() function:

### Example

print(type(x))

print(type(y))

print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers)

## Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

### Example

Integers:

x = 1

y = 35656222554887711

z = -3255522

print(type(x))

print(type(y))

print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_int)

## Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

### Example

Floats:

x = 1.10

y = 1.0

z = -35.59

print(type(x))

print(type(y))

print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_float)

Float can also be scientific numbers with an "e" to indicate the power of 10.

### Example

Floats:

x = 35e3

y = 12E4

z = -87.7e100

print(type(x))

print(type(y))

print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_float2)

## Complex

Complex numbers are written with a "j" as the imaginary part:

### Example

Complex:

x = 3+5j

y = 5j

z = -5j

print(type(x))

print(type(y))

print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_complex)

## Type Conversion

You can convert from one type to another with the int(), float(), and complex() methods:

### Example

Convert from one type to another:

x = 1 # int

y = 2.8 # float

z = 1j # complex

#convert from int to float:

a = float(x)

#convert from float to int:

b = int(y)

#convert from int to complex:

c = complex(x)

print(a)

print(b)

print(c)

print(type(a))

print(type(b))

print(type(c))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_convert)

Note: You cannot convert complex numbers into another number type.

## Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

### Example

Import the random module, and display a random number between 1 and 9:

import random

print(random.randrange(1, 10))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_random)

In our [Random Module Reference](https://www.w3schools.com/python/module_random.asp) you will learn more about the Random module.

## Test Yourself With Exercises

## Exercise:

Insert the correct syntax to convert x into a floating point number.

x = 5

x = (x)

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_numbers1)

# Python Casting

[❮ Previous](https://www.w3schools.com/python/python_numbers.asp)[Next ❯](https://www.w3schools.com/python/python_strings.asp)

## Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

* int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
* float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
* str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

### Example

Integers:

x = int(1) # x will be 1

y = int(2.8) # y will be 2

z = int("3") # z will be 3

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_casting_int)

### Example

Floats:

x = float(1) # x will be 1.0

y = float(2.8) # y will be 2.8

z = float("3") # z will be 3.0

w = float("4.2") # w will be 4.2

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_float)

### Example

Strings:

x = str("s1") # x will be 's1'

y = str(2) # y will be '2'

z = str(3.0) # z will be '3.0'

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string)

# Python Strings

[❮ Previous](https://www.w3schools.com/python/python_casting.asp)[Next ❯](https://www.w3schools.com/python/python_strings_slicing.asp)

## Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the print() function:

### Example

print("Hello")

print('Hello')

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_literal)

## Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

### Example

a = "Hello"

print(a)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_var)

## Multiline Strings

You can assign a multiline string to a variable by using three quotes:

### Example

You can use three double quotes:

a = """Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua."""

print(a)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_multi)

Or three single quotes:

### Example

a = '''Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua.'''

print(a)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_multi2)

Note: in the result, the line breaks are inserted at the same position as in the code.

## Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

### Example

Get the character at position 1 (remember that the first character has the position 0):

a = "Hello, World!"

print(a[1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string1)

## Looping Through a String

Since strings are arrays, we can loop through the characters in a string, with a for loop.

### Example

Loop through the letters in the word "banana":

for x in "banana":

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_string)

Learn more about For Loops in our [Python For Loops](https://www.w3schools.com/python/python_for_loops.asp) chapter.

## String Length

To get the length of a string, use the len() function.

### Example

The len() function returns the length of a string:

a = "Hello, World!"

print(len(a))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_len)

## Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

### Example

Check if "free" is present in the following text:

txt = "The best things in life are free!"

print("free" in txt)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_is_in)

Use it in an if statement:

### Example

Print only if "free" is present:

txt = "The best things in life are free!"

if "free" in txt:

print("Yes, 'free' is present.")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_if_in)

Learn more about If statements in our [Python If...Else](https://www.w3schools.com/python/python_conditions.asp) chapter.

## Check if NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.

### Example

Check if "expensive" is NOT present in the following text:

txt = "The best things in life are free!"

print("expensive" not in txt)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_is_not_in)

Use it in an if statement:

### Example

print only if "expensive" is NOT present:

txt = "The best things in life are free!"

if "expensive" not in txt:

print("Yes, 'expensive' is NOT present.")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_if_not_in)

# Python - Slicing Strings

[❮ Previous](https://www.w3schools.com/python/python_strings.asp)[Next ❯](https://www.w3schools.com/python/python_strings_modify.asp)

## Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

### Example

Get the characters from position 2 to position 5 (not included):

b = "Hello, World!"

print(b[2:5])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string2)

Note: The first character has index 0.

## Slice From the Start

By leaving out the start index, the range will start at the first character:

### Example

Get the characters from the start to position 5 (not included):

b = "Hello, World!"

print(b[:5])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_slice_start)

## Slice To the End

By leaving out the *end* index, the range will go to the end:

### Example

Get the characters from position 2, and all the way to the end:

b = "Hello, World!"

print(b[2:])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_slice_end)

## Negative Indexing

Use negative indexes to start the slice from the end of the string:

### Example

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

b = "Hello, World!"

print(b[-5:-2])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_negativeindex)

# Python - Modify Strings

[❮ Previous](https://www.w3schools.com/python/python_strings_slicing.asp)[Next ❯](https://www.w3schools.com/python/python_strings_concatenate.asp)

Python has a set of built-in methods that you can use on strings.

## Upper Case

### Example

The upper() method returns the string in upper case:

a = "Hello, World!"

print(a.upper())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_upper)

## Lower Case

### Example

The lower() method returns the string in lower case:

a = "Hello, World!"

print(a.lower())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_lower)

## Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

### Example

The strip() method removes any whitespace from the beginning or the end:

a = " Hello, World! "

print(a.strip()) # returns "Hello, World!"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_strip)

## Replace String

### Example

The replace() method replaces a string with another string:

a = "Hello, World!"

print(a.replace("H", "J"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_replace)

## Split String

The split() method returns a list where the text between the specified separator becomes the list items.

### Example

The split() method splits the string into substrings if it finds instances of the separator:

a = "Hello, World!"

print(a.split(",")) # returns ['Hello', ' World!']

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_split)

Learn more about Lists in our [Python Lists](https://www.w3schools.com/python/python_lists.asp) chapter.

## String Methods

Learn more about String Methods with our [String Methods Reference](https://www.w3schools.com/python/python_ref_string.asp)

# Python - String Concatenation

[❮ Previous](https://www.w3schools.com/python/python_strings_modify.asp)[Next ❯](https://www.w3schools.com/python/python_strings_format.asp)

## String Concatenation

To concatenate, or combine, two strings you can use the + operator.

### Example

Merge variable a with variable b into variable c:

a = "Hello"

b = "World"

c = a + b

print(c)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_concat)

### Example

To add a space between them, add a " ":

a = "Hello"

b = "World"

c = a + " " + b

print(c)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_concat2)

[❮ Previous](https://www.w3schools.com/python/python_strings_modify.asp)[Next ❯](https://www.w3schools.com/python/python_strings_format.asp)

# Python - Format - Strings

[❮ Previous](https://www.w3schools.com/python/python_strings_concatenate.asp)[Next ❯](https://www.w3schools.com/python/python_strings_escape.asp)

## String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

### Example

age = 36

txt = "My name is John, I am " + age

print(txt)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format_error)

But we can combine strings and numbers by using the format() method!

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

### Example

Use the format() method to insert numbers into strings:

age = 36

txt = "My name is John, and I am {}"

print(txt.format(age))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format1)

The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

### Example

quantity = 3

itemno = 567

price = 49.95

myorder = "I want {} pieces of item {} for {} dollars."

print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format2)

You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

### Example

quantity = 3

itemno = 567

price = 49.95

myorder = "I want to pay {2} dollars for {0} pieces of item {1}."

print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format3)

Learn more about String Formatting in our [String Formatting](https://www.w3schools.com/python/python_string_formatting.asp) chapter.

# Python - Escape Characters

[❮ Previous](https://www.w3schools.com/python/python_strings_format.asp)[Next ❯](https://www.w3schools.com/python/python_strings_methods.asp)

## Escape Character

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

### Example

You will get an error if you use double quotes inside a string that is surrounded by double quotes:

txt = "We are the so-called "Vikings" from the north."

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape_error)

To fix this problem, use the escape character \":

### Example

The escape character allows you to use double quotes when you normally would not be allowed:

txt = "We are the so-called \"Vikings\" from the north."

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape)

## Escape Characters

Other escape characters used in Python:

| **Code** | **Result** | **Try it** |
| --- | --- | --- |
| \' | Single Quote | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape2) |
| \\ | Backslash | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_backslash) |
| \n | New Line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_newline) |
| \r | Carriage Return | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_r) |
| \t | Tab | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_t) |
| \b | Backspace | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_b) |
| \f | Form Feed |  |
| \ooo | Octal value | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_octal) |
| \xhh | Hex value | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_hex) |

[❮ Previous](https://www.w3schools.com/python/python_strings_format.asp)[Next ❯](https://www.w3schools.com/python/python_strings_methods.asp)

# Python - String Methods

[❮ Previous](https://www.w3schools.com/python/python_strings_escape.asp)[Next ❯](https://www.w3schools.com/python/python_strings_exercises.asp)

## String Methods

Python has a set of built-in methods that you can use on strings.

Note: All string methods returns new values. They do not change the original string.

| **Method** | **Description** |
| --- | --- |
| [capitalize()](https://www.w3schools.com/python/ref_string_capitalize.asp) | Converts the first character to upper case |
| [casefold()](https://www.w3schools.com/python/ref_string_casefold.asp) | Converts string into lower case |
| [center()](https://www.w3schools.com/python/ref_string_center.asp) | Returns a centered string |
| [count()](https://www.w3schools.com/python/ref_string_count.asp) | Returns the number of times a specified value occurs in a string |
| [encode()](https://www.w3schools.com/python/ref_string_encode.asp) | Returns an encoded version of the string |
| [endswith()](https://www.w3schools.com/python/ref_string_endswith.asp) | Returns true if the string ends with the specified value |
| [expandtabs()](https://www.w3schools.com/python/ref_string_expandtabs.asp) | Sets the tab size of the string |
| [find()](https://www.w3schools.com/python/ref_string_find.asp) | Searches the string for a specified value and returns the position of where it was found |
| [format()](https://www.w3schools.com/python/ref_string_format.asp) | Formats specified values in a string |
| format\_map() | Formats specified values in a string |
| [index()](https://www.w3schools.com/python/ref_string_index.asp) | Searches the string for a specified value and returns the position of where it was found |
| [isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp) | Returns True if all characters in the string are alphanumeric |
| [isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp) | Returns True if all characters in the string are in the alphabet |
| [isdecimal()](https://www.w3schools.com/python/ref_string_isdecimal.asp) | Returns True if all characters in the string are decimals |
| [isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp) | Returns True if all characters in the string are digits |
| [isidentifier()](https://www.w3schools.com/python/ref_string_isidentifier.asp) | Returns True if the string is an identifier |
| [islower()](https://www.w3schools.com/python/ref_string_islower.asp) | Returns True if all characters in the string are lower case |
| [isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp) | Returns True if all characters in the string are numeric |
| [isprintable()](https://www.w3schools.com/python/ref_string_isprintable.asp) | Returns True if all characters in the string are printable |
| [isspace()](https://www.w3schools.com/python/ref_string_isspace.asp) | Returns True if all characters in the string are whitespaces |
| [istitle()](https://www.w3schools.com/python/ref_string_istitle.asp) | Returns True if the string follows the rules of a title |
| [isupper()](https://www.w3schools.com/python/ref_string_isupper.asp) | Returns True if all characters in the string are upper case |
| [join()](https://www.w3schools.com/python/ref_string_join.asp) | Joins the elements of an iterable to the end of the string |
| [ljust()](https://www.w3schools.com/python/ref_string_ljust.asp) | Returns a left justified version of the string |
| [lower()](https://www.w3schools.com/python/ref_string_lower.asp) | Converts a string into lower case |
| [lstrip()](https://www.w3schools.com/python/ref_string_lstrip.asp) | Returns a left trim version of the string |
| [maketrans()](https://www.w3schools.com/python/ref_string_maketrans.asp) | Returns a translation table to be used in translations |
| [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Returns a tuple where the string is parted into three parts |
| [replace()](https://www.w3schools.com/python/ref_string_replace.asp) | Returns a string where a specified value is replaced with a specified value |
| [rfind()](https://www.w3schools.com/python/ref_string_rfind.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rjust()](https://www.w3schools.com/python/ref_string_rjust.asp) | Returns a right justified version of the string |
| [rpartition()](https://www.w3schools.com/python/ref_string_rpartition.asp) | Returns a tuple where the string is parted into three parts |
| [rsplit()](https://www.w3schools.com/python/ref_string_rsplit.asp) | Splits the string at the specified separator, and returns a list |
| [rstrip()](https://www.w3schools.com/python/ref_string_rstrip.asp) | Returns a right trim version of the string |
| [split()](https://www.w3schools.com/python/ref_string_split.asp) | Splits the string at the specified separator, and returns a list |
| [splitlines()](https://www.w3schools.com/python/ref_string_splitlines.asp) | Splits the string at line breaks and returns a list |
| [startswith()](https://www.w3schools.com/python/ref_string_startswith.asp) | Returns true if the string starts with the specified value |
| [strip()](https://www.w3schools.com/python/ref_string_strip.asp) | Returns a trimmed version of the string |
| [swapcase()](https://www.w3schools.com/python/ref_string_swapcase.asp) | Swaps cases, lower case becomes upper case and vice versa |
| [title()](https://www.w3schools.com/python/ref_string_title.asp) | Converts the first character of each word to upper case |
| [translate()](https://www.w3schools.com/python/ref_string_translate.asp) | Returns a translated string |
| [upper()](https://www.w3schools.com/python/ref_string_upper.asp) | Converts a string into upper case |
| [zfill()](https://www.w3schools.com/python/ref_string_zfill.asp) | Fills the string with a specified number of 0 values at the beginning |

[❮ Previous](https://www.w3schools.com/python/python_strings_escape.asp)[Next ❯](https://www.w3schools.com/python/python_strings_exercises.asp)

# Python - String Exercises

[❮ Previous](https://www.w3schools.com/python/python_strings_methods.asp)[Next ❯](https://www.w3schools.com/python/python_booleans.asp)

## Test Yourself With Exercises

Now you have learned a lot about Strings, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

## Test Yourself With Exercises

## Exercise:

Use the len method to print the length of the string.

x = "Hello World"

print()

Submit Answer »

Go to the Exercise section and test all of our Python Strings Exercises:

[Python String Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_strings1)

[❮ Previous](https://www.w3schools.com/python/python_strings_methods.asp)[Next ❯](https://www.w3schools.com/python/python_booleans.asp)

# Python Booleans

[❮ Previous](https://www.w3schools.com/python/python_strings_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_operators.asp)

Booleans represent one of two values: True or False.

## Boolean Values

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

### Example

print(10 > 9)

print(10 == 9)

print(10 < 9)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans1)

When you run a condition in an if statement, Python returns True or False:

### Example

Print a message based on whether the condition is True or False:

a = 200

b = 33

if b > a:

print("b is greater than a")

else:

print("b is not greater than a")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans2)

## Evaluate Values and Variables

The bool() function allows you to evaluate any value, and give you True or False in return,

### Example

Evaluate a string and a number:

print(bool("Hello"))

print(bool(15))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans3)

### Example

Evaluate two variables:

x = "Hello"

y = 15

print(bool(x))

print(bool(y))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans4)

## Most Values are True

Almost any value is evaluated to True if it has some sort of content.

Any string is True, except empty strings.

Any number is True, except 0.

Any list, tuple, set, and dictionary are True, except empty ones.

### Example

The following will return True:

bool("abc")

bool(123)

bool(["apple", "cherry", "banana"])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans5)

## Some Values are False

In fact, there are not many values that evaluate to False, except empty values, such as (), [], {}, "", the number 0, and the value None. And of course the value False evaluates to False.

### Example

The following will return False:

bool(False)

bool(None)

bool(0)

bool("")

bool(())

bool([])

bool({})

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans6)

One more value, or object in this case, evaluates to False, and that is if you have an object that is made from a class with a \_\_len\_\_ function that returns 0 or False:

### Example

class myclass():

def \_\_len\_\_(self):

return 0

myobj = myclass()

print(bool(myobj))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans7)

## Functions can Return a Boolean

You can create functions that returns a Boolean Value:

### Example

Print the answer of a function:

def myFunction() :

return True

print(myFunction())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_boolean_return)

You can execute code based on the Boolean answer of a function:

### Example

Print "YES!" if the function returns True, otherwise print "NO!":

def myFunction() :

return True

if myFunction():

print("YES!")

else:

print("NO!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_boolean_return2)

Python also has many built-in functions that return a boolean value, like the isinstance() function, which can be used to determine if an object is of a certain data type:

### Example

Check if an object is an integer or not:

x = 200

print(isinstance(x, int))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans8)

## Test Yourself With Exercises

## Exercise:

The statement below would print a Boolean value, which one?

print(10 > 9)

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_booleans1)

# Python Operators

[❮ Previous](https://www.w3schools.com/python/python_booleans.asp)[Next ❯](https://www.w3schools.com/python/python_lists.asp)

## Python Operators

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

### Example

print(10 + 5)

[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper)

Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

## Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

| **Operator** | **Name** | **Example** | **Try it** |
| --- | --- | --- | --- |
| + | Addition | x + y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_add) |
| - | Subtraction | x - y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_sub) |
| \* | Multiplication | x \* y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mult) |
| / | Division | x / y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_div) |
| % | Modulus | x % y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mod) |
| \*\* | Exponentiation | x \*\* y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_exp) |
| // | Floor division | x // y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_floordiv) |

## Python Assignment Operators

Assignment operators are used to assign values to variables:

| **Operator** | **Example** | **Same As** | **Try it** |
| --- | --- | --- | --- |
| = | x = 5 | x = 5 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass1) |
| += | x += 3 | x = x + 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass2) |
| -= | x -= 3 | x = x - 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass3) |
| \*= | x \*= 3 | x = x \* 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass4) |
| /= | x /= 3 | x = x / 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass5) |
| %= | x %= 3 | x = x % 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass6) |
| //= | x //= 3 | x = x // 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass7) |
| \*\*= | x \*\*= 3 | x = x \*\* 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass8) |
| &= | x &= 3 | x = x & 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass9) |
| |= | x |= 3 | x = x | 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass10) |
| ^= | x ^= 3 | x = x ^ 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass11) |
| >>= | x >>= 3 | x = x >> 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass12) |
| <<= | x <<= 3 | x = x << 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass13) |

## Python Comparison Operators

Comparison operators are used to compare two values:

| **Operator** | **Name** | **Example** | **Try it** |
| --- | --- | --- | --- |
| == | Equal | x == y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare1) |
| != | Not equal | x != y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare2) |
| > | Greater than | x > y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare4) |
| < | Less than | x < y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare5) |
| >= | Greater than or equal to | x >= y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare6) |
| <= | Less than or equal to | x <= y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare7) |

## Python Logical Operators

Logical operators are used to combine conditional statements:

| **Operator** | **Description** | **Example** | **Try it** |
| --- | --- | --- | --- |
| and | Returns True if both statements are true | x < 5 and x < 10 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_logical1) |
| or | Returns True if one of the statements is true | x < 5 or x < 4 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_logical2) |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_logical3) |

## Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

| **Operator** | **Description** | **Example** | **Try it** |
| --- | --- | --- | --- |
| is | Returns True if both variables are the same object | x is y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_identity1) |
| is not | Returns True if both variables are not the same object | x is not y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_identity2) |

## Python Membership Operators

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

| **Operator** | **Description** | **Example** | **Try it** |
| --- | --- | --- | --- |
| in | Returns True if a sequence with the specified value is present in the object | x in y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_membership1) |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_membership2) |

## Python 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 |

## Test Yourself With Exercises

## Exercise:

Multiply 10 with 5, and print the result.

print(10 5)

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_operators1)

[❮ Previous](https://www.w3schools.com/python/python_booleans.asp)[Next ❯](https://www.w3schools.com/python/python_lists.asp)

# Python Lists

[❮ Previous](https://www.w3schools.com/python/python_operators.asp)[Next ❯](https://www.w3schools.com/python/python_lists_access.asp)

mylist = ["apple", "banana", "cherry"]

## List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets:

### Example

Create a List:

thislist = ["apple", "banana", "cherry"]

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list)

## List Items

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

## Ordered

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

Note: There are some [list methods](https://www.w3schools.com/python/python_lists_methods.asp) that will change the order, but in general: the order of the items will not change.

## Changeable

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

## Allow Duplicates

Since lists are indexed, lists can have items with the same value:

### Example

Lists allow duplicate values:

thislist = ["apple", "banana", "cherry", "apple", "cherry"]

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_duplicates)

## List Length

To determine how many items a list has, use the len() function:

### Example

Print the number of items in the list:

thislist = ["apple", "banana", "cherry"]

print(len(thislist))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_len)

## List Items - Data Types

List items can be of any data type:

### Example

String, int and boolean data types:

list1 = ["apple", "banana", "cherry"]

list2 = [1, 5, 7, 9, 3]

list3 = [True, False, False]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_datatypes)

A list can contain different data types:

### Example

A list with strings, integers and boolean values:

list1 = ["abc", 34, True, 40, "male"]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_datatypes2)

## type()

From Python's perspective, lists are defined as objects with the data type 'list':

<class 'list'>

### Example

What is the data type of a list?

mylist = ["apple", "banana", "cherry"]

print(type(mylist))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_datatypes3)

## The list() Constructor

It is also possible to use the list() constructor when creating a new list.

### Example

Using the list() constructor to make a List:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_constructor)

## Python Collections (Arrays)

There are four collection data types in the Python programming language:

* List is a collection which is ordered and changeable. Allows duplicate members.
* [Tuple](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [Set](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered and unindexed. No duplicate members.
* [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\* and changeable. No duplicate members.

\*As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

# Python - Access List Items

[❮ Previous](https://www.w3schools.com/python/python_lists.asp)[Next ❯](https://www.w3schools.com/python/python_lists_change.asp)

## Access Items

List items are indexed and you can access them by referring to the index number:

### Example

Print the second item of the list:

thislist = ["apple", "banana", "cherry"]

print(thislist[1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_access)

Note: The first item has index 0.

### Negative Indexing

Negative indexing means start from the end

-1 refers to the last item, -2 refers to the second last item etc.

### Example

Print the last item of the list:

thislist = ["apple", "banana", "cherry"]

print(thislist[-1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_negative_index)

### Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new list with the specified items.

### Example

Return the third, fourth, and fifth item:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[2:5])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range)

Note: The search will start at index 2 (included) and end at index 5 (not included).

Remember that the first item has index 0.

By leaving out the start value, the range will start at the first item:

### Example

This example returns the items from the beginning to, but NOT including, "kiwi":

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[:4])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range2)

By leaving out the end value, the range will go on to the end of the list:

### Example

This example returns the items from "cherry" to the end:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[2:])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range3)

### Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the list:

### Example

This example returns the items from "orange" (-4) to, but NOT including "mango" (-1):

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[-4:-1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range_negative)

## Check if Item Exists

To determine if a specified item is present in a list use the in keyword:

### Example

Check if "apple" is present in the list:

thislist = ["apple", "banana", "cherry"]

if "apple" in thislist:

print("Yes, 'apple' is in the fruits list")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_in)

# Python - Change List Items

[❮ Previous](https://www.w3schools.com/python/python_lists_access.asp)[Next ❯](https://www.w3schools.com/python/python_lists_add.asp)

## Change Item Value

To change the value of a specific item, refer to the index number:

### Example

Change the second item:

thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_change)

## Change a Range of Item Values

To change the value of items within a specific range, define a list with the new values, and refer to the range of index numbers where you want to insert the new values:

### Example

Change the values "banana" and "cherry" with the values "blackcurrant" and "watermelon":

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]

thislist[1:3] = ["blackcurrant", "watermelon"]

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_change_range)

If you insert *more* items than you replace, the new items will be inserted where you specified, and the remaining items will move accordingly:

### Example

Change the second value by replacing it with *two* new values:

thislist = ["apple", "banana", "cherry"]

thislist[1:2] = ["blackcurrant", "watermelon"]

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_change2)

Note: The length of the list will change when the number of items inserted does not match the number of items replaced.

If you insert *less* items than you replace, the new items will be inserted where you specified, and the remaining items will move accordingly:

### Example

Change the second and third value by replacing it with *one* value:

thislist = ["apple", "banana", "cherry"]

thislist[1:3] = ["watermelon"]

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_change3)

## Insert Items

To insert a new list item, without replacing any of the existing values, we can use the insert() method.

The insert() method inserts an item at the specified index:

### Example

Insert "watermelon" as the third item:

thislist = ["apple", "banana", "cherry"]

thislist.insert(2, "watermelon")

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_change2_insert)

Note: As a result of the example above, the list will now contain 4 items.

# Python - Add List Items

[❮ Previous](https://www.w3schools.com/python/python_lists_change.asp)[Next ❯](https://www.w3schools.com/python/python_lists_remove.asp)

## Append Items

To add an item to the end of the list, use the append() method:

### Example

Using the append() method to append an item:

thislist = ["apple", "banana", "cherry"]

thislist.append("orange")

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_append)

## Insert Items

To insert a list item at a specified index, use the insert() method.

The insert() method inserts an item at the specified index:

### Example

Insert an item as the second position:

thislist = ["apple", "banana", "cherry"]

thislist.insert(1, "orange")

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_insert)

Note: As a result of the examples above, the lists will now contain 4 items.

## Extend List

To append elements from *another list* to the current list, use the extend() method.

### Example

Add the elements of tropical to thislist:

thislist = ["apple", "banana", "cherry"]

tropical = ["mango", "pineapple", "papaya"]

thislist.extend(tropical)

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_add_extend)

The elements will be added to the *end* of the list.

## Add Any Iterable

The extend() method does not have to append *lists*, you can add any iterable object (tuples, sets, dictionaries etc.).

### Example

Add elements of a tuple to a list:

thislist = ["apple", "banana", "cherry"]

thistuple = ("kiwi", "orange")

thislist.extend(thistuple)

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_add_extend_tuple)

[❮ Previous](https://www.w3schools.com/python/python_lists_change.asp)[Next ❯](https://www.w3schools.com/python/python_lists_remove.asp)

# Python - Remove List Items

[❮ Previous](https://www.w3schools.com/python/python_lists_add.asp)[Next ❯](https://www.w3schools.com/python/python_lists_loop.asp)

## Remove Specified Item

The remove() method removes the specified item.

### Example

Remove "banana":

thislist = ["apple", "banana", "cherry"]

thislist.remove("banana")

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_remove)

## Remove Specified Index

The pop() method removes the specified index.

### Example

Remove the second item:

thislist = ["apple", "banana", "cherry"]

thislist.pop(1)

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_pop2)

If you do not specify the index, the pop() method removes the last item.

### Example

Remove the last item:

thislist = ["apple", "banana", "cherry"]

thislist.pop()

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_pop)

The del keyword also removes the specified index:

### Example

Remove the first item:

thislist = ["apple", "banana", "cherry"]

del thislist[0]

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_del)

The del keyword can also delete the list completely.

### Example

Delete the entire list:

thislist = ["apple", "banana", "cherry"]

del thislist

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_del2)

## Clear the List

The clear() method empties the list.

The list still remains, but it has no content.

### Example

Clear the list content:

thislist = ["apple", "banana", "cherry"]

thislist.clear()

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_clear)

# Python - Loop Lists

[❮ Previous](https://www.w3schools.com/python/python_lists_remove.asp)[Next ❯](https://www.w3schools.com/python/python_lists_comprehension.asp)

## Loop Through a List

You can loop through the list items by using a for loop:

### Example

Print all items in the list, one by one:

thislist = ["apple", "banana", "cherry"]

for x in thislist:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_loop)

Learn more about for loops in our [Python For Loops](https://www.w3schools.com/python/python_for_loops.asp) Chapter.

## Loop Through the Index Numbers

You can also loop through the list items by referring to their index number.

Use the range() and len() functions to create a suitable iterable.

### Example

Print all items by referring to their index number:

thislist = ["apple", "banana", "cherry"]

for i in range(len(thislist)):

print(thislist[i])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_loop2)

The iterable created in the example above is [0, 1, 2].

## Using a While Loop

You can loop through the list items by using a while loop.

Use the len() function to determine the length of the list, then start at 0 and loop your way through the list items by refering to their indexes.

Remember to increase the index by 1 after each iteration.

### Example

Print all items, using a while loop to go through all the index numbers

thislist = ["apple", "banana", "cherry"]

i = 0

while i < len(thislist):

print(thislist[i])

i = i + 1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_while_loop)

Learn more about while loops in our [Python While Loops](https://www.w3schools.com/python/python_while_loops.asp) Chapter.

## Looping Using List Comprehension

List Comprehension offers the shortest syntax for looping through lists:

### Example

A short hand for loop that will print all items in a list:

thislist = ["apple", "banana", "cherry"]

[print(x) for x in thislist]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_loop_comprehensive)

Learn more about list comprehension in the next chapter: [List Comprehension](https://www.w3schools.com/python/python_lists_comprehension.asp).

# Python - List Comprehension

[❮ Previous](https://www.w3schools.com/python/python_lists_loop.asp)[Next ❯](https://www.w3schools.com/python/python_lists_sort.asp)

## List Comprehension

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

Example:

Based on a list of fruits, you want a new list, containing only the fruits with the letter "a" in the name.

Without list comprehension you will have to write a for statement with a conditional test inside:

### Example

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = []

for x in fruits:

if "a" in x:

newlist.append(x)

print(newlist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_without_comprehension)

With list comprehension you can do all that with only one line of code:

### Example

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension)

## The Syntax

newlist = [*expression* for *item* in *iterable* if *condition* == True]

The return value is a new list, leaving the old list unchanged.

### Condition

The *condition* is like a filter that only accepts the items that valuate to True.

### Example

Only accept items that are not "apple":

newlist = [x for x in fruits if x != "apple"]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension_if)

The condition if x != "apple" will return True for all elements other than "apple", making the new list contain all fruits except "apple".

The *condition* is optional and can be omitted:

### Example

With no if statement:

newlist = [x for x in fruits]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension_noif)

### Iterable

The *iterable* can be any iterable object, like a list, tuple, set etc.

### Example

You can use the range() function to create an iterable:

newlist = [x for x in range(10)]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension_range)

Same example, but with a condition:

### Example

Accept only numbers lower than 5:

newlist = [x for x in range(10) if x < 5]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension_range2)

### Expression

The *expression* is the current item in the iteration, but it is also the outcome, which you can manipulate before it ends up like a list item in the new list:

### Example

Set the values in the new list to upper case:

newlist = [x.upper() for x in fruits]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension_upper)

You can set the outcome to whatever you like:

### Example

Set all values in the new list to 'hello':

newlist = ['hello' for x in fruits]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension_hello)

The *expression* can also contain conditions, not like a filter, but as a way to manipulate the outcome:

### Example

Return "orange" instead of "banana":

newlist = [x if x != "banana" else "orange" for x in fruits]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_comprehension_if_else)

The *expression* in the example above says:

*"Return the item if it is not banana, if it is banana return orange".*

# Python - Sort Lists

[❮ Previous](https://www.w3schools.com/python/python_lists_comprehension.asp)[Next ❯](https://www.w3schools.com/python/python_lists_copy.asp)

## Sort List Alphanumerically

List objects have a sort() method that will sort the list alphanumerically, ascending, by default:

### Example

Sort the list alphabetically:

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]

thislist.sort()

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort)

### Example

Sort the list numerically:

thislist = [100, 50, 65, 82, 23]

thislist.sort()

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort_num)

## Sort Descending

To sort descending, use the keyword argument reverse = True:

### Example

Sort the list descending:

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]

thislist.sort(reverse = True)

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort_desc)

### Example

Sort the list descending:

thislist = [100, 50, 65, 82, 23]

thislist.sort(reverse = True)

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort_num_desc)

## Customize Sort Function

You can also customize your own function by using the keyword argument key = *function*.

The function will return a number that will be used to sort the list (the lowest number first):

### Example

Sort the list based on how close the number is to 50:

def myfunc(n):

return abs(n - 50)

thislist = [100, 50, 65, 82, 23]

thislist.sort(key = myfunc)

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort_key)

## Case Insensitive Sort

By default the sort() method is case sensitive, resulting in all capital letters being sorted before lower case letters:

### Example

Case sensitive sorting can give an unexpected result:

thislist = ["banana", "Orange", "Kiwi", "cherry"]

thislist.sort()

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort_case_sensitive)

Luckily we can use built-in functions as key functions when sorting a list.

So if you want a case-insensitive sort function, use str.lower as a key function:

### Example

Perform a case-insensitive sort of the list:

thislist = ["banana", "Orange", "Kiwi", "cherry"]

thislist.sort(key = str.lower)

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort_lower)

## Reverse Order

What if you want to reverse the order of a list, regardless of the alphabet?

The reverse() method reverses the current sorting order of the elements.

### Example

Reverse the order of the list items:

thislist = ["banana", "Orange", "Kiwi", "cherry"]

thislist.reverse()

print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_sort_reverse)

# Python - Copy Lists

[❮ Previous](https://www.w3schools.com/python/python_lists_sort.asp)[Next ❯](https://www.w3schools.com/python/python_lists_join.asp)

## Copy a List

You cannot copy a list simply by typing list2 = list1, because: list2 will only be a *reference* to list1, and changes made in list1 will automatically also be made in list2.

There are ways to make a copy, one way is to use the built-in List method copy().

### Example

Make a copy of a list with the copy() method:

thislist = ["apple", "banana", "cherry"]

mylist = thislist.copy()

print(mylist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_copy)

Another way to make a copy is to use the built-in method list().

### Example

Make a copy of a list with the list() method:

thislist = ["apple", "banana", "cherry"]

mylist = list(thislist)

print(mylist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_copy2)

[❮ Previous](https://www.w3schools.com/python/python_lists_sort.asp)[Next ❯](https://www.w3schools.com/python/python_lists_join.asp)

# Python - Join Lists

[❮ Previous](https://www.w3schools.com/python/python_lists_copy.asp)[Next ❯](https://www.w3schools.com/python/python_lists_methods.asp)

## Join Two Lists

There are several ways to join, or concatenate, two or more lists in Python.

One of the easiest ways are by using the + operator.

### Example

Join two list:

list1 = ["a", "b", "c"]

list2 = [1, 2, 3]

list3 = list1 + list2

print(list3)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_concat1)

Another way to join two lists is by appending all the items from list2 into list1, one by one:

### Example

Append list2 into list1:

list1 = ["a", "b" , "c"]

list2 = [1, 2, 3]

for x in list2:

list1.append(x)

print(list1)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_concat2)

Or you can use the extend() method, which purpose is to add elements from one list to another list:

### Example

Use the extend() method to add list2 at the end of list1:

list1 = ["a", "b" , "c"]

list2 = [1, 2, 3]

list1.extend(list2)

print(list1)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_concat3)

[❮ Previous](https://www.w3schools.com/python/python_lists_copy.asp)[Next ❯](https://www.w3schools.com/python/python_lists_methods.asp)

# Python - List Methods

[❮ Previous](https://www.w3schools.com/python/python_lists_join.asp)[Next ❯](https://www.w3schools.com/python/python_lists_exercises.asp)

## List Methods

Python has a set of built-in methods that you can use on lists.

| **Method** | **Description** |
| --- | --- |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

[❮ Previous](https://www.w3schools.com/python/python_lists_join.asp)[Next ❯](https://www.w3schools.com/python/python_lists_exercises.asp)

# Python List Exercises

[❮ Previous](https://www.w3schools.com/python/python_lists_methods.asp)[Next ❯](https://www.w3schools.com/python/python_tuples.asp)

## Test Yourself With Exercises

Now you have learned a lot about lists, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

## Exercise:

Print the second item in the fruits list.

fruits = ["apple", "banana", "cherry"]

print()

Submit Answer »

Go to the Exercise section and test all of our Python List Exercises:

[Python List Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_lists1)

[❮ Previous](https://www.w3schools.com/python/python_lists_methods.asp)[Next ❯](https://www.w3schools.com/python/python_tuples.asp)

# Python Tuples

[❮ Previous](https://www.w3schools.com/python/python_lists_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_access.asp)

mytuple = ("apple", "banana", "cherry")

## Tuple

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

A tuple is a collection which is ordered and unchangeable.

Tuples are written with round brackets.

### Example

Create a Tuple:

thistuple = ("apple", "banana", "cherry")

print(thistuple)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple)

## Tuple Items

Tuple items are ordered, unchangeable, and allow duplicate values.

Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

## Ordered

When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.

## Unchangeable

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

## Allow Duplicates

Since tuples are indexed, they can have items with the same value:

### Example

Tuples allow duplicate values:

thistuple = ("apple", "banana", "cherry", "apple", "cherry")

print(thistuple)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_duplicates)

## Tuple Length

To determine how many items a tuple has, use the len() function:

### Example

Print the number of items in the tuple:

thistuple = ("apple", "banana", "cherry")

print(len(thistuple))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_len)

## Create Tuple With One Item

To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.

### Example

One item tuple, remember the comma:

thistuple = ("apple",)

print(type(thistuple))

#NOT a tuple

thistuple = ("apple")

print(type(thistuple))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_one_item)

## Tuple Items - Data Types

Tuple items can be of any data type:

### Example

String, int and boolean data types:

tuple1 = ("apple", "banana", "cherry")

tuple2 = (1, 5, 7, 9, 3)

tuple3 = (True, False, False)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_datatypes)

A tuple can contain different data types:

### Example

A tuple with strings, integers and boolean values:

tuple1 = ("abc", 34, True, 40, "male")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_datatypes2)

## type()

From Python's perspective, tuples are defined as objects with the data type 'tuple':

<class 'tuple'>

### Example

What is the data type of a tuple?

mytuple = ("apple", "banana", "cherry")

print(type(mytuple))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_datatypes3)

## The tuple() Constructor

It is also possible to use the tuple() constructor to make a tuple.

### Example

Using the tuple() method to make a tuple:

thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets

print(thistuple)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple3)

## Python Collections (Arrays)

There are four collection data types in the Python programming language:

* [List](https://www.w3schools.com/python/python_lists.asp) is a collection which is ordered and changeable. Allows duplicate members.
* Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
* [Set](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered and unindexed. No duplicate members.
* [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\* and changeable. No duplicate members.

\*As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

# Python - Access Tuple Items

[❮ Previous](https://www.w3schools.com/python/python_tuples.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_update.asp)

## Access Tuple Items

You can access tuple items by referring to the index number, inside square brackets:

### Example

Print the second item in the tuple:

thistuple = ("apple", "banana", "cherry")

print(thistuple[1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple1)

Note: The first item has index 0.

## Negative Indexing

Negative indexing means start from the end.

-1 refers to the last item, -2 refers to the second last item etc.

### Example

Print the last item of the tuple:

thistuple = ("apple", "banana", "cherry")

print(thistuple[-1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_negative_index)

## Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new tuple with the specified items.

### Example

Return the third, fourth, and fifth item:

thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")

print(thistuple[2:5])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_range)

Note: The search will start at index 2 (included) and end at index 5 (not included).

Remember that the first item has index 0.

By leaving out the start value, the range will start at the first item:

### Example

This example returns the items from the beginning to, but NOT included, "kiwi":

thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")

print(thistuple[:4])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_range2)

By leaving out the end value, the range will go on to the end of the list:

### Example

This example returns the items from "cherry" and to the end:

thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")

print(thistuple[2:])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_range3)

## Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the tuple:

### Example

This example returns the items from index -4 (included) to index -1 (excluded)

thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")

print(thistuple[-4:-1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_range_negative)

## Check if Item Exists

To determine if a specified item is present in a tuple use the in keyword:

### Example

Check if "apple" is present in the tuple:

thistuple = ("apple", "banana", "cherry")

if "apple" in thistuple:

print("Yes, 'apple' is in the fruits tuple")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_in)

# Python - Update Tuples

[❮ Previous](https://www.w3schools.com/python/python_tuples_access.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_unpack.asp)

Tuples are unchangeable, meaning that you cannot change, add, or remove items once the tuple is created.

But there are some workarounds.

## Change Tuple Values

Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.

But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

### Example

Convert the tuple into a list to be able to change it:

x = ("apple", "banana", "cherry")

y = list(x)

y[1] = "kiwi"

x = tuple(y)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_change)

## Add Items

Once a tuple is created, you cannot add items to it.

### Example

You cannot add items to a tuple:

thistuple = ("apple", "banana", "cherry")

thistuple.append("orange") # This will raise an error

print(thistuple)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_add_error)

Just like the workaround for *changing* a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

### Example

Convert the tuple into a list, add "orange", and convert it back into a tuple:

thistuple = ("apple", "banana", "cherry")

y = list(thistuple)

y.append("orange")

thistuple = tuple(y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_add_orange)

## Remove Items

Note: You cannot remove items in a tuple.

Tuples are unchangeable, so you cannot remove items from it, but you can use the same workaround as we used for changing and adding tuple items:

### Example

Convert the tuple into a list, remove "apple", and convert it back into a tuple:

thistuple = ("apple", "banana", "cherry")

y = list(thistuple)

y.remove("apple")

thistuple = tuple(y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_remove)

Or you can delete the tuple completely:

### Example

The del keyword can delete the tuple completely:

thistuple = ("apple", "banana", "cherry")

del thistuple

print(thistuple) #this will raise an error because the tuple no longer exists

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_del)

# Python - Unpack Tuples

[❮ Previous](https://www.w3schools.com/python/python_tuples_update.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_loop.asp)

## Unpacking a Tuple

When we create a tuple, we normally assign values to it. This is called "packing" a tuple:

### Example

Packing a tuple:

fruits = ("apple", "banana", "cherry")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_packing)

But, in Python, we are also allowed to extract the values back into variables. This is called "unpacking":

### Example

Unpacking a tuple:

fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)

print(yellow)

print(red)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_unpacking)

Note: The number of variables must match the number of values in the tuple, if not, you must use an asterisk to collect the remaining values as a list.

## Using Asterisk\*

If the number of variables is less than the number of values, you can add an \* to the variable name and the values will be assigned to the variable as a list:

### Example

Assign the rest of the values as a list called "red":

fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")

(green, yellow, \*red) = fruits

print(green)

print(yellow)

print(red)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_unpacking_asterix1)

If the asterisk is added to another variable name than the last, Python will assign values to the variable until the number of values left matches the number of variables left.

### Example

Add a list of values the "tropic" variable:

fruits = ("apple", "mango", "papaya", "pineapple", "cherry")

(green, \*tropic, red) = fruits

print(green)

print(tropic)

print(red)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_unpacking_asterix2)

# Python - Loop Tuples

[❮ Previous](https://www.w3schools.com/python/python_tuples_unpack.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_join.asp)

## Loop Through a Tuple

You can loop through the tuple items by using a for loop.

### Example

Iterate through the items and print the values:

thistuple = ("apple", "banana", "cherry")

for x in thistuple:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_loop)

Learn more about for loops in our [Python For Loops](https://www.w3schools.com/python/python_for_loops.asp) Chapter.

## Loop Through the Index Numbers

You can also loop through the tuple items by referring to their index number.

Use the range() and len() functions to create a suitable iterable.

### Example

Print all items by referring to their index number:

thistuple = ("apple", "banana", "cherry")

for i in range(len(thistuple)):

print(thistuple[i])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_loop_index)

## Using a While Loop

You can loop through the list items by using a while loop.

Use the len() function to determine the length of the tuple, then start at 0 and loop your way through the tuple items by refering to their indexes.

Remember to increase the index by 1 after each iteration.

### Example

Print all items, using a while loop to go through all the index numbers:

thistuple = ("apple", "banana", "cherry")

i = 0

while i < len(thistuple):

print(thistuple[i])

i = i + 1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_while)

Learn more about while loops in our [Python While Loops](https://www.w3schools.com/python/python_while_loops.asp) Chapter.

# Python - Join Tuples

[❮ Previous](https://www.w3schools.com/python/python_tuples_loop.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_methods.asp)

## Join Two Tuples

To join two or more tuples you can use the + operator:

### Example

Join two tuples:

tuple1 = ("a", "b" , "c")

tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2

print(tuple3)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_concat)

## Multiply Tuples

If you want to multiply the content of a tuple a given number of times, you can use the \* operator:

### Example

Multiply the fruits tuple by 2:

fruits = ("apple", "banana", "cherry")

mytuple = fruits \* 2

print(mytuple)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_multiply)

[❮ Previous](https://www.w3schools.com/python/python_tuples_loop.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_methods.asp)

# Python - Tuple Methods

[❮ Previous](https://www.w3schools.com/python/python_tuples_join.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_exercises.asp)

## Tuple Methods

Python has two built-in methods that you can use on tuples.

| **Method** | **Description** |
| --- | --- |
| [count()](https://www.w3schools.com/python/ref_tuple_count.asp) | Returns the number of times a specified value occurs in a tuple |
| [index()](https://www.w3schools.com/python/ref_tuple_index.asp) | Searches the tuple for a specified value and returns the position of where it was found |

[❮ Previous](https://www.w3schools.com/python/python_tuples_join.asp)[Next ❯](https://www.w3schools.com/python/python_tuples_exercises.asp)

# Python - Tuple Exercises

[❮ Previous](https://www.w3schools.com/python/python_tuples_methods.asp)[Next ❯](https://www.w3schools.com/python/python_sets.asp)

## Test Yourself With Exercises

Now you have learned a lot about tuples, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

## Exercise:

Print the first item in the fruits tuple.

fruits = ("apple", "banana", "cherry")

print()

Submit Answer »

Go to the Exercise section and test all of our Python Tuple Exercises:

[Python Tuple Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_tuples1)

# Python Sets

[❮ Previous](https://www.w3schools.com/python/python_tuples_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_sets_access.asp)

myset = {"apple", "banana", "cherry"}

## Set

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Tuple](https://www.w3schools.com/python/python_tuples.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

A set is a collection which is both *unordered* and *unindexed*.

Sets are written with curly brackets.

### Example

Create a Set:

thisset = {"apple", "banana", "cherry"}

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set)

Note: Sets are unordered, so you cannot be sure in which order the items will appear.

## Set Items

Set items are unordered, unchangeable, and do not allow duplicate values.

## Unordered

Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

## Unchangeable

Sets are unchangeable, meaning that we cannot change the items after the set has been created.

Once a set is created, you cannot change its items, but you can add new items.

## Duplicates Not Allowed

Sets cannot have two items with the same value.

### Example

Duplicate values will be ignored:

thisset = {"apple", "banana", "cherry", "apple"}

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_sets_duplicate_error)

## Get the Length of a Set

To determine how many items a set has, use the len() method.

### Example

Get the number of items in a set:

thisset = {"apple", "banana", "cherry"}

print(len(thisset))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_length)

## Set Items - Data Types

Set items can be of any data type:

### Example

String, int and boolean data types:

set1 = {"apple", "banana", "cherry"}

set2 = {1, 5, 7, 9, 3}

set3 = {True, False, False}

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_datatypes)

A set can contain different data types:

### Example

A set with strings, integers and boolean values:

set1 = {"abc", 34, True, 40, "male"}

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_datatypes2)

## type()

From Python's perspective, sets are defined as objects with the data type 'set':

<class 'set'>

### Example

What is the data type of a set?

myset = {"apple", "banana", "cherry"}

print(type(myset))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_datatypes3)

## The set() Constructor

It is also possible to use the set() constructor to make a set.

### Example

Using the set() constructor to make a set:

thisset = set(("apple", "banana", "cherry")) # note the double round-brackets

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set2)

## Python Collections (Arrays)

There are four collection data types in the Python programming language:

* [List](https://www.w3schools.com/python/python_lists.asp) is a collection which is ordered and changeable. Allows duplicate members.
* [Tuple](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* Set is a collection which is unordered and unindexed. No duplicate members.
* [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\* and changeable. No duplicate members.

\*As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

# Python - Access Set Items

[❮ Previous](https://www.w3schools.com/python/python_sets.asp)[Next ❯](https://www.w3schools.com/python/python_sets_add.asp)

## Access Items

You cannot access items in a set by referring to an index or a key.

But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

### Example

Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}

for x in thisset:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_loop)

### Example

Check if "banana" is present in the set:

thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_in)

## Change Items

Once a set is created, you cannot change its items, but you can add new items.

[❮ Previous](https://www.w3schools.com/python/python_sets.asp)[Next ❯](https://www.w3schools.com/python/python_sets_add.asp)

# Python - Add Set Items

[❮ Previous](https://www.w3schools.com/python/python_sets_access.asp)[Next ❯](https://www.w3schools.com/python/python_sets_remove.asp)

## Add Items

Once a set is created, you cannot change its items, but you can add new items.

To add one item to a set use the add() method.

### Example

Add an item to a set, using the add() method:

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_add)

## Add Sets

To add items from another set into the current set, use the update() method.

### Example

Add elements from tropical into thisset:

thisset = {"apple", "banana", "cherry"}

tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_sets_update)

## Add Any Iterable

The object in the update() method does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.).

### Example

Add elements of a list to at set:

thisset = {"apple", "banana", "cherry"}

mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_sets_update2)

# Python - Remove Set Items

[❮ Previous](https://www.w3schools.com/python/python_sets_add.asp)[Next ❯](https://www.w3schools.com/python/python_sets_loop.asp)

## Remove Item

To remove an item in a set, use the remove(), or the discard() method.

### Example

Remove "banana" by using the remove() method:

thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_remove)

Note: If the item to remove does not exist, remove() will raise an error.

### Example

Remove "banana" by using the discard() method:

thisset = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_discard)

Note: If the item to remove does not exist, discard() will NOT raise an error.

You can also use the pop() method to remove an item, but this method will remove the *last* item. Remember that sets are unordered, so you will not know what item that gets removed.

The return value of the pop() method is the removed item.

### Example

Remove the last item by using the pop() method:

thisset = {"apple", "banana", "cherry"}

x = thisset.pop()

print(x)

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_pop)

Note: Sets are *unordered*, so when using the pop() method, you do not know which item that gets removed.

### Example

The clear() method empties the set:

thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_clear)

### Example

The del keyword will delete the set completely:

thisset = {"apple", "banana", "cherry"}

del thisset

print(thisset)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_del)

# Python - Loop Sets

[❮ Previous](https://www.w3schools.com/python/python_sets_remove.asp)[Next ❯](https://www.w3schools.com/python/python_sets_join.asp)

## Loop Items

You can loop through the set items by using a for loop:

### Example

Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}

for x in thisset:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_loop)

# Python - Join Sets

[❮ Previous](https://www.w3schools.com/python/python_sets_loop.asp)[Next ❯](https://www.w3schools.com/python/python_sets_methods.asp)

## Join Two Sets

There are several ways to join two or more sets in Python.

You can use the union() method that returns a new set containing all items from both sets, or the update() method that inserts all the items from one set into another:

### Example

The union() method returns a new set with all items from both sets:

set1 = {"a", "b" , "c"}

set2 = {1, 2, 3}

set3 = set1.union(set2)

print(set3)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_union)

### Example

The update() method inserts the items in set2 into set1:

set1 = {"a", "b" , "c"}

set2 = {1, 2, 3}

set1.update(set2)

print(set1)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_set_update2)

Note: Both union() and update() will exclude any duplicate items.

## Keep ONLY the Duplicates

The intersection\_update() method will keep only the items that are present in both sets.

### Example

Keep the items that exist in both set x, and set y:

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.intersection\_update(y)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_sets_intersection_update)

The intersection() method will return a *new* set, that only contains the items that are present in both sets.

### Example

Return a set that contains the items that exist in both set x, and set y:

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.intersection(y)

print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_sets_intersection)

## Keep All, But NOT the Duplicates

The symmetric\_difference\_update() method will keep only the elements that are NOT present in both sets.

### Example

Keep the items that are not present in both sets:

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.symmetric\_difference\_update(y)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_sets_symmetric_difference_update)

The symmetric\_difference() method will return a new set, that contains only the elements that are NOT present in both sets.

### Example

Return a set that contains all items from both sets, except items that are present in both:

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.symmetric\_difference(y)

print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_sets_symmetric_difference)

# Python - Set Methods

[❮ Previous](https://www.w3schools.com/python/python_sets_join.asp)[Next ❯](https://www.w3schools.com/python/python_sets_exercises.asp)

## Set Methods

Python has a set of built-in methods that you can use on sets.

| **Method** | **Description** |
| --- | --- |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | Adds an element to the set |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) | Remove the specified item |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two other sets |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Removes an element from the set |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specified element |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | inserts the symmetric differences from this set and another |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with the union of this set and others |

[❮ Previous](https://www.w3schools.com/python/python_sets_join.asp)[Next ❯](https://www.w3schools.com/python/python_sets_exercises.asp)

# Python - Set Exercises

[❮ Previous](https://www.w3schools.com/python/python_sets_methods.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries.asp)

## Test Yourself With Exercises

Now you have learned a lot about sets, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

## Exercise:

Check if "apple" is present in the fruits set.

fruits = {"apple", "banana", "cherry"}

if "apple" fruits:

print("Yes, apple is a fruit!")

Submit Answer »

Go to the Exercise section and test all of our Python Set Exercises:

[Python Set Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_sets1)

[❮ Previous](https://www.w3schools.com/python/python_sets_methods.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries.asp)

# Python Dictionaries

[❮ Previous](https://www.w3schools.com/python/python_sets_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_access.asp)

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

## Dictionary

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered\*, changeable and does not allow duplicates.

As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

Dictionaries are written with curly brackets, and have keys and values:

### Example

Create and print a dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print(thisdict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary)

## Dictionary Items

Dictionary items are ordered, changeable, and does not allow duplicates.

Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

### Example

Print the "brand" value of the dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print(thisdict["brand"])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_brand)

## Ordered or Unordered?

As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change.

Unordered means that the items does not have a defined order, you cannot refer to an item by using an index.

## Changeable

Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

## Duplicates Not Allowed

Dictionaries cannot have two items with the same key:

### Example

Duplicate values will overwrite existing values:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964,

"year": 2020

}

print(thisdict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_duplicates)

## Dictionary Length

To determine how many items a dictionary has, use the len() function:

### Example

Print the number of items in the dictionary:

print(len(thisdict))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_len)

## Dictionary Items - Data Types

The values in dictionary items can be of any data type:

### Example

String, int, boolean, and list data types:

thisdict = {

"brand": "Ford",

"electric": False,

"year": 1964,

"colors": ["red", "white", "blue"]

}

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_datatypes)

## type()

From Python's perspective, dictionaries are defined as objects with the data type 'dict':

<class 'dict'>

### Example

Print the data type of a dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print(type(thisdict))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_datatypes2)

## Python Collections (Arrays)

There are four collection data types in the Python programming language:

* [List](https://www.w3schools.com/python/python_lists.asp) is a collection which is ordered and changeable. Allows duplicate members.
* [Tuple](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [Set](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered and unindexed. No duplicate members.
* Dictionary is a collection which is ordered and changeable. No duplicate members.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

# Python - Access Dictionary Items

[❮ Previous](https://www.w3schools.com/python/python_dictionaries.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_change.asp)

## Accessing Items

You can access the items of a dictionary by referring to its key name, inside square brackets:

### Example

Get the value of the "model" key:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = thisdict["model"]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_access)

There is also a method called get() that will give you the same result:

### Example

Get the value of the "model" key:

x = thisdict.get("model")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_get)

## Get Keys

The keys() method will return a list of all the keys in the dictionary.

### Example

Get a list of the keys:

x = thisdict.keys()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_keys)

The list of the keys is a *view* of the dictionary, meaning that any changes done to the dictionary will be reflected in the keys list.

### Example

Add a new item to the original dictionary, and see that the keys list gets updated as well:

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.keys()

print(x) #before the change

car["color"] = "white"

print(x) #after the change

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_keys2)

## Get Values

The values() method will return a list of all the values in the dictionary.

### Example

Get a list of the values:

x = thisdict.values()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_values)

The list of the values is a *view* of the dictionary, meaning that any changes done to the dictionary will be reflected in the values list.

### Example

Make a change in the original dictionary, and see that the values list gets updated as well:

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.values()

print(x) #before the change

car["year"] = 2020

print(x) #after the change

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_values2)

### Example

Add a new item to the original dictionary, and see that the values list gets updated as well:

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.values()

print(x) #before the change

car["color"] = "red"

print(x) #after the change

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_values3)

## Get Items

The items() method will return each item in a dictionary, as tuples in a list.

### Example

Get a list of the key:value pairs

x = thisdict.items()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_items)

The returned list is a *view* of the items of the dictionary, meaning that any changes done to the dictionary will be reflected in the items list.

### Example

Make a change in the original dictionary, and see that the items list gets updated as well:

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.items()

print(x) #before the change

car["year"] = 2020

print(x) #after the change

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_items2)

### Example

Add a new item to the original dictionary, and see that the items list gets updated as well:

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = car.items()

print(x) #before the change

car["color"] = "red"

print(x) #after the change

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_items3)

## Check if Key Exists

To determine if a specified key is present in a dictionary use the in keyword:

### Example

Check if "model" is present in the dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

if "model" in thisdict:

print("Yes, 'model' is one of the keys in the thisdict dictionary")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_in)

# Python - Change Dictionary Items

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_access.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_add.asp)

## Change Values

You can change the value of a specific item by referring to its key name:

### Example

Change the "year" to 2018:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict["year"] = 2018

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_change)

## Update Dictionary

The update() method will update the dictionary with the items from the given argument.

The argument must be a dictionary, or an iterable object with key:value pairs.

### Example

Update the "year" of the car by using the update() method:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.update({"year": 2020})

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_update)

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_access.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_add.asp)

The update() method will update the dictionary with the items from a given argument. If the item does not exist, the item will be added.

The argument must be a dictionary, or an iterable object with key:value pairs.

### Example

Add a color item to the dictionary by using the update() method:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.update({"color": "red"})

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dict_add2)

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_change.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_remove.asp)

# Python - Remove Dictionary Items

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_add.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_loop.asp)

## Removing Items

There are several methods to remove items from a dictionary:

### Example

The pop() method removes the item with the specified key name:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.pop("model")

print(thisdict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_pop)

### Example

The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.popitem()

print(thisdict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_popitem)

### Example

The del keyword removes the item with the specified key name:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

del thisdict["model"]

print(thisdict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_del2)

### Example

The del keyword can also delete the dictionary completely:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

del thisdict

print(thisdict) #this will cause an error because "thisdict" no longer exists.

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_del3)

### Example

The clear() method empties the dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.clear()

print(thisdict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_clear)

# Python - Loop Dictionaries

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_remove.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_copy.asp)

## Loop Through a Dictionary

You can loop through a dictionary by using a for loop.

When looping through a dictionary, the return value are the *keys* of the dictionary, but there are methods to return the *values* as well.

### Example

Print all key names in the dictionary, one by one:

for x in thisdict:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_loop)

### Example

Print all *values* in the dictionary, one by one:

for x in thisdict:

print(thisdict[x])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_loop2)

### Example

You can also use the values() method to return values of a dictionary:

for x in thisdict.values():

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_loop_values)

### Example

You can use the keys() method to return the keys of a dictionary:

for x in thisdict.keys():

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_loop_keys)

### Example

Loop through both *keys* and *values*, by using the items() method:

for x, y in thisdict.items():

print(x, y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_loop_items)

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_remove.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_copy.asp)

# Python - Copy Dictionaries

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_loop.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_nested.asp)

## Copy a Dictionary

You cannot copy a dictionary simply by typing dict2 = dict1, because: dict2 will only be a *reference* to dict1, and changes made in dict1 will automatically also be made in dict2.

There are ways to make a copy, one way is to use the built-in Dictionary method copy().

### Example

Make a copy of a dictionary with the copy() method:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

mydict = thisdict.copy()

print(mydict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_copy)

Another way to make a copy is to use the built-in function dict().

### Example

Make a copy of a dictionary with the dict() function:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

mydict = dict(thisdict)

print(mydict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_copy2)

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_loop.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_nested.asp)

# Python - Nested Dictionaries

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_copy.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_methods.asp)

## Nested Dictionaries

A dictionary can contain dictionaries, this is called nested dictionaries.

### Example

Create a dictionary that contain three dictionaries:

myfamily = {

"child1" : {

"name" : "Emil",

"year" : 2004

},

"child2" : {

"name" : "Tobias",

"year" : 2007

},

"child3" : {

"name" : "Linus",

"year" : 2011

}

}

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_nested)

Or, if you want to add three dictionaries into a new dictionary:

### Example

Create three dictionaries, then create one dictionary that will contain the other three dictionaries:

child1 = {

"name" : "Emil",

"year" : 2004

}

child2 = {

"name" : "Tobias",

"year" : 2007

}

child3 = {

"name" : "Linus",

"year" : 2011

}

myfamily = {

"child1" : child1,

"child2" : child2,

"child3" : child3

}

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_nested2)

# Python Dictionary Methods

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_nested.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_exercises.asp)

## Dictionary Methods

Python has a set of built-in methods that you can use on dictionaries.

| **Method** | **Description** |
| --- | --- |
| [clear()](https://www.w3schools.com/python/ref_dictionary_clear.asp) | Removes all the elements from the dictionary |
| [copy()](https://www.w3schools.com/python/ref_dictionary_copy.asp) | Returns a copy of the dictionary |
| [fromkeys()](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | Returns a dictionary with the specified keys and value |
| [get()](https://www.w3schools.com/python/ref_dictionary_get.asp) | Returns the value of the specified key |
| [items()](https://www.w3schools.com/python/ref_dictionary_items.asp) | Returns a list containing a tuple for each key value pair |
| [keys()](https://www.w3schools.com/python/ref_dictionary_keys.asp) | Returns a list containing the dictionary's keys |
| [pop()](https://www.w3schools.com/python/ref_dictionary_pop.asp) | Removes the element with the specified key |
| [popitem()](https://www.w3schools.com/python/ref_dictionary_popitem.asp) | Removes the last inserted key-value pair |
| [setdefault()](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| [update()](https://www.w3schools.com/python/ref_dictionary_update.asp) | Updates the dictionary with the specified key-value pairs |
| [values()](https://www.w3schools.com/python/ref_dictionary_values.asp) | Returns a list of all the values in the dictionary |

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_nested.asp)[Next ❯](https://www.w3schools.com/python/python_dictionaries_exercises.asp)

# Python Dictionary Exercises

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_methods.asp)[Next ❯](https://www.w3schools.com/python/python_conditions.asp)

## Test Yourself With Exercises

Now you have learned a lot about dictionaries, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

## Test Yourself With Exercises

## Exercise:

Use the get method to print the value of the "model" key of the car dictionary.

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print()

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_dictionaries1)

Go to the Exercise section and test all of our Python Dictionary Exercises:

[Python Dictionary Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_dictionaries1)

# Python If ... Else

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_while_loops.asp)

## Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

### Example

If statement:

a = 33

b = 200

if b > a:

print("b is greater than a")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if2)

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

## Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

### Example

If statement, without indentation (will raise an error):

a = 33

b = 200

if b > a:

print("b is greater than a") # you will get an error

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_error)

## Elif

The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

### Example

a = 33

b = 33

if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_elif)

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

## Else

The else keyword catches anything which isn't caught by the preceding conditions.

### Example

a = 200

b = 33

if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

else:

print("a is greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else)

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

### Example

a = 200

b = 33

if b > a:

print("b is greater than a")

else:

print("b is not greater than a")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else2)

## Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

### Example

One line if statement:

if a > b: print("a is greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_short)

## Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

### Example

One line if else statement:

a = 2

b = 330

print("A") if a > b else print("B")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else_short)

This technique is known as Ternary Operators, or Conditional Expressions.

You can also have multiple else statements on the same line:

### Example

One line if else statement, with 3 conditions:

a = 330

b = 330

print("A") if a > b else print("=") if a == b else print("B")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else_short2)

## And

The and keyword is a logical operator, and is used to combine conditional statements:

### Example

Test if a is greater than b, AND if c is greater than a:

a = 200

b = 33

c = 500

if a > b and c > a:

print("Both conditions are True")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_and)

## Or

The or keyword is a logical operator, and is used to combine conditional statements:

### Example

Test if a is greater than b, OR if a is greater than c:

a = 200

b = 33

c = 500

if a > b or a > c:

print("At least one of the conditions is True")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_or)

## Nested If

You can have if statements inside if statements, this is called *nested* if statements.

### Example

x = 41

if x > 10:

print("Above ten,")

if x > 20:

print("and also above 20!")

else:

print("but not above 20.")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_nested)

## The pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

### Example

a = 33

b = 200

if b > a:

pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_pass)

## Test Yourself With Exercises

## Exercise:

Print "Hello World" if a is greater than b.

a = 50

b = 10

a b

print("Hello World")

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_ifelse1)

# Python While Loops

[❮ Previous](https://www.w3schools.com/python/python_conditions.asp)[Next ❯](https://www.w3schools.com/python/python_for_loops.asp)

## Python Loops

Python has two primitive loop commands:

* while loops
* for loops

## The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

### Example

Print i as long as i is less than 6:

i = 1

while i < 6:

print(i)

i += 1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while)

Note: remember to increment i, or else the loop will continue forever.

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

## The break Statement

With the break statement we can stop the loop even if the while condition is true:

### Example

Exit the loop when i is 3:

i = 1

while i < 6:

print(i)

if i == 3:

break

i += 1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while_break)

## The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

### Example

Continue to the next iteration if i is 3:

i = 0

while i < 6:

i += 1

if i == 3:

continue

print(i)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while_continue)

## The else Statement

With the else statement we can run a block of code once when the condition no longer is true:

### Example

Print a message once the condition is false:

i = 1

while i < 6:

print(i)

i += 1

else:

print("i is no longer less than 6")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while_else)

## Test Yourself With Exercises

## Exercise:

Print i as long as i is less than 6.

i = 1

i < 6

print(i)

i += 1

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_while_loops1)

# Python For Loops

[❮ Previous](https://www.w3schools.com/python/python_while_loops.asp)[Next ❯](https://www.w3schools.com/python/python_functions.asp)

## Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

### Example

Print each fruit in a fruit list:

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for)

The for loop does not require an indexing variable to set beforehand.

## Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

### Example

Loop through the letters in the word "banana":

for x in "banana":

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_string)

## The break Statement

With the break statement we can stop the loop before it has looped through all the items:

### Example

Exit the loop when x is "banana":

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

if x == "banana":

break

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_break)

### Example

Exit the loop when x is "banana", but this time the break comes before the print:

fruits = ["apple", "banana", "cherry"]

for x in fruits:

if x == "banana":

break

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_break2)

## The continue Statement

With the continue statement we can stop the current iteration of the loop, and continue with the next:

### Example

Do not print banana:

fruits = ["apple", "banana", "cherry"]

for x in fruits:

if x == "banana":

continue

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_continue)

## The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

### Example

Using the range() function:

for x in range(6):

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_range)

Note that range(6) is not the values of 0 to 6, but the values 0 to 5.

The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

### Example

Using the start parameter:

for x in range(2, 6):

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_range2)

The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, 3):

### Example

Increment the sequence with 3 (default is 1):

for x in range(2, 30, 3):

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_range3)

## Else in For Loop

The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

### Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

for x in range(6):

print(x)

else:

print("Finally finished!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_else)

Note: The else block will NOT be executed if the loop is stopped by a break statement.

### Example

Break the loop when x is 3, and see what happens with the else block:

for x in range(6):

if x == 3: break

print(x)

else:

print("Finally finished!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_else_break)

## Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

### Example

Print each adjective for every fruit:

adj = ["red", "big", "tasty"]

fruits = ["apple", "banana", "cherry"]

for x in adj:

for y in fruits:

print(x, y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_nested)

## The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

### Example

for x in [0, 1, 2]:

pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_pass)

## Test Yourself With Exercises

## Exercise:

Loop through the items in the fruits list.

fruits = ["apple", "banana", "cherry"]

x fruits

print(x)

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_for_loops1)

# Python Functions

[❮ Previous](https://www.w3schools.com/python/python_for_loops.asp)[Next ❯](https://www.w3schools.com/python/python_lambda.asp)

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

## Creating a Function

In Python a function is defined using the def keyword:

### Example

def my\_function():

print("Hello from a function")

## Calling a Function

To call a function, use the function name followed by parenthesis:

### Example

def my\_function():

print("Hello from a function")

my\_function()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function)

## Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

### Example

def my\_function(fname):

print(fname + " Refsnes")

my\_function("Emil")

my\_function("Tobias")

my\_function("Linus")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_param)

*Arguments* are often shortened to *args* in Python documentations.

## Parameters or Arguments?

The terms *parameter* and *argument* can be used for the same thing: information that are passed into a function.

From a function's perspective:

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that is sent to the function when it is called.

## Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

### Example

This function expects 2 arguments, and gets 2 arguments:

def my\_function(fname, lname):

print(fname + " " + lname)

my\_function("Emil", "Refsnes")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_args_n)

If you try to call the function with 1 or 3 arguments, you will get an error:

### Example

This function expects 2 arguments, but gets only 1:

def my\_function(fname, lname):

print(fname + " " + lname)

my\_function("Emil")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_args_error)

## Arbitrary Arguments, \*args

If you do not know how many arguments that will be passed into your function, add a \* before the parameter name in the function definition.

This way the function will receive a *tuple* of arguments, and can access the items accordingly:

### Example

If the number of arguments is unknown, add a \* before the parameter name:

def my\_function(\*kids):

print("The youngest child is " + kids[2])

my\_function("Emil", "Tobias", "Linus")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_args)

*Arbitrary Arguments* are often shortened to *\*args* in Python documentations.

## Keyword Arguments

You can also send arguments with the *key* = *value* syntax.

This way the order of the arguments does not matter.

### Example

def my\_function(child3, child2, child1):

print("The youngest child is " + child3)

my\_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_kwargs)

The phrase *Keyword Arguments* are often shortened to *kwargs* in Python documentations.

## Arbitrary Keyword Arguments, \*\*kwargs

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: \*\* before the parameter name in the function definition.

This way the function will receive a *dictionary* of arguments, and can access the items accordingly:

### Example

If the number of keyword arguments is unknown, add a double \*\* before the parameter name:

def my\_function(\*\*kid):

print("His last name is " + kid["lname"])

my\_function(fname = "Tobias", lname = "Refsnes")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_kwargs_n)

*Arbitrary Kword Arguments* are often shortened to *\*\*kwargs* in Python documentations.

## Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without argument, it uses the default value:

### Example

def my\_function(country = "Norway"):

print("I am from " + country)

my\_function("Sweden")

my\_function("India")

my\_function()

my\_function("Brazil")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_param2)

## Passing a List as an Argument

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as an argument, it will still be a List when it reaches the function:

### Example

def my\_function(food):

for x in food:

print(x)

fruits = ["apple", "banana", "cherry"]

my\_function(fruits)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_param3)

## Return Values

To let a function return a value, use the return statement:

### Example

def my\_function(x):

return 5 \* x

print(my\_function(3))

print(my\_function(5))

print(my\_function(9))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_return)

## The pass Statement

function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

### Example

def myfunction():

pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_pass)

## Recursion

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

In this example, tri\_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out how exactly this works, best way to find out is by testing and modifying it.

### Example

Recursion Example

def tri\_recursion(k):

if(k > 0):

result = k + tri\_recursion(k - 1)

print(result)

else:

result = 0

return result

print("\n\nRecursion Example Results")

tri\_recursion(6)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_recursion)

## Test Yourself With Exercises

## Exercise:

Create a function named my\_function.

:

print("Hello from a function")

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_functions1)

# Python Lambda

[❮ Previous](https://www.w3schools.com/python/python_functions.asp)[Next ❯](https://www.w3schools.com/python/python_arrays.asp)

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

## Syntax

lambda *arguments* : *expression*

The expression is executed and the result is returned:

### Example

Add 10 to argument a, and return the result:

x = lambda a : a + 10

print(x(5))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda)

Lambda functions can take any number of arguments:

### Example

Multiply argument a with argument b and return the result:

x = lambda a, b : a \* b

print(x(5, 6))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda2)

### Example

Summarize argument a, b, and c and return the result:

x = lambda a, b, c : a + b + c

print(x(5, 6, 2))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda3)

## Why Use Lambda Functions?

The power of lambda is better shown when you use them as an anonymous function inside another function.

Say you have a function definition that takes one argument, and that argument will be multiplied with an unknown number:

def myfunc(n):

return lambda a : a \* n

Use that function definition to make a function that always doubles the number you send in:

### Example

def myfunc(n):

return lambda a : a \* n

mydoubler = myfunc(2)

print(mydoubler(11))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda_double)

Or, use the same function definition to make a function that always *triples* the number you send in:

### Example

def myfunc(n):

return lambda a : a \* n

mytripler = myfunc(3)

print(mytripler(11))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda_triple)

Or, use the same function definition to make both functions, in the same program:

### Example

def myfunc(n):

return lambda a : a \* n

mydoubler = myfunc(2)

mytripler = myfunc(3)

print(mydoubler(11))

print(mytripler(11))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda_both)

Use lambda functions when an anonymous function is required for a short period of time.

## Test Yourself With Exercises

## Exercise:

Create a lambda function that takes one parameter (a) and returns it.

x =

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_lambda1)

# Python Arrays

[❮ Previous](https://www.w3schools.com/python/python_lambda.asp)[Next ❯](https://www.w3schools.com/python/python_classes.asp)

Note: Python does not have built-in support for Arrays, but [Python Lists](https://www.w3schools.com/python/python_lists.asp) can be used instead.

## Arrays

Note: This page shows you how to use LISTS as ARRAYS, however, to work with arrays in Python you will have to import a library, like the [NumPy library](https://www.w3schools.com/python/numpy/default.asp).

Arrays are used to store multiple values in one single variable:

### Example

Create an array containing car names:

cars = ["Ford", "Volvo", "BMW"]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array1)

## What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

car1 = "Ford"

car2 = "Volvo"

car3 = "BMW"

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

## Access the Elements of an Array

You refer to an array element by referring to the *index number*.

### Example

Get the value of the first array item:

x = cars[0]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array2)

### Example

Modify the value of the first array item:

cars[0] = "Toyota"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array3)

## The Length of an Array

Use the len() method to return the length of an array (the number of elements in an array).

### Example

Return the number of elements in the cars array:

x = len(cars)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array4)

Note: The length of an array is always one more than the highest array index.

## Looping Array Elements

You can use the for in loop to loop through all the elements of an array.

### Example

Print each item in the cars array:

for x in cars:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array5)

## Adding Array Elements

You can use the append() method to add an element to an array.

### Example

Add one more element to the cars array:

cars.append("Honda")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array6)

## Removing Array Elements

You can use the pop() method to remove an element from the array.

### Example

Delete the second element of the cars array:

cars.pop(1)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array7)

You can also use the remove() method to remove an element from the array.

### Example

Delete the element that has the value "Volvo":

cars.remove("Volvo")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array8)

Note: The list's remove() method only removes the first occurrence of the specified value.

## Array Methods

Python has a set of built-in methods that you can use on lists/arrays.

| **Method** | **Description** |
| --- | --- |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

Note: Python does not have built-in support for Arrays, but Python Lists can be used instead.

# Python Classes and Objects

[❮ Previous](https://www.w3schools.com/python/python_arrays.asp)[Next ❯](https://www.w3schools.com/python/python_inheritance.asp)

## Python Classes/Objects

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

## Create a Class

To create a class, use the keyword class:

### Example

Create a class named MyClass, with a property named x:

class MyClass:

x = 5

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class1)

## Create Object

Now we can use the class named MyClass to create objects:

### Example

Create an object named p1, and print the value of x:

p1 = MyClass()

print(p1.x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class2)

## The \_\_init\_\_() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function.

All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.

Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

### Example

Create a class named Person, use the \_\_init\_\_() function to assign values for name and age:

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

p1 = Person("John", 36)

print(p1.name)

print(p1.age)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class3)

Note: The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

## Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Person class:

### Example

Insert a function that prints a greeting, and execute it on the p1 object:

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def myfunc(self):

print("Hello my name is " + self.name)

p1 = Person("John", 36)

p1.myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class4)

Note: The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

## The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

### Example

Use the words *mysillyobject* and *abc* instead of *self*:

class Person:

def \_\_init\_\_(mysillyobject, name, age):

mysillyobject.name = name

mysillyobject.age = age

def myfunc(abc):

print("Hello my name is " + abc.name)

p1 = Person("John", 36)

p1.myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class5)

## Modify Object Properties

You can modify properties on objects like this:

### Example

Set the age of p1 to 40:

p1.age = 40

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class6)

## Delete Object Properties

You can delete properties on objects by using the del keyword:

### Example

Delete the age property from the p1 object:

del p1.age

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class7)

## Delete Objects

You can delete objects by using the del keyword:

### Example

Delete the p1 object:

del p1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class8)

## The pass Statement

class definitions cannot be empty, but if you for some reason have a class definition with no content, put in the pass statement to avoid getting an error.

### Example

class Person:

pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class_pass)

## Test Yourself With Exercises

## Exercise:

Create a class named MyClass:

MyClass:

x = 5

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_classes1)

# Python Inheritance

[❮ Previous](https://www.w3schools.com/python/python_classes.asp)[Next ❯](https://www.w3schools.com/python/python_iterators.asp)

## Python Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

## Create a Parent Class

Any class can be a parent class, so the syntax is the same as creating any other class:

### Example

Create a class named Person, with firstname and lastname properties, and a printname method:

class Person:

def \_\_init\_\_(self, fname, lname):

self.firstname = fname

self.lastname = lname

def printname(self):

print(self.firstname, self.lastname)

#Use the Person class to create an object, and then execute the printname method:

x = Person("John", "Doe")

x.printname()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_parent)

## Create a Child Class

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class:

### Example

Create a class named Student, which will inherit the properties and methods from the Person class:

class Student(Person):

pass

Note: Use the pass keyword when you do not want to add any other properties or methods to the class.

Now the Student class has the same properties and methods as the Person class.

### Example

Use the Student class to create an object, and then execute the printname method:

x = Student("Mike", "Olsen")

x.printname()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_child)

## Add the \_\_init\_\_() Function

So far we have created a child class that inherits the properties and methods from its parent.

We want to add the \_\_init\_\_() function to the child class (instead of the pass keyword).

Note: The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

### Example

Add the \_\_init\_\_() function to the Student class:

class Student(Person):

def \_\_init\_\_(self, fname, lname):

#add properties etc.

When you add the \_\_init\_\_() function, the child class will no longer inherit the parent's \_\_init\_\_() function.

Note: The child's \_\_init\_\_() function overrides the inheritance of the parent's \_\_init\_\_() function.

To keep the inheritance of the parent's \_\_init\_\_() function, add a call to the parent's \_\_init\_\_() function:

### Example

class Student(Person):

def \_\_init\_\_(self, fname, lname):

Person.\_\_init\_\_(self, fname, lname)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_init)

Now we have successfully added the \_\_init\_\_() function, and kept the inheritance of the parent class, and we are ready to add functionality in the \_\_init\_\_() function.

## Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

### Example

class Student(Person):

def \_\_init\_\_(self, fname, lname):

super().\_\_init\_\_(fname, lname)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_super)

By using the super() function, you do not have to use the name of the parent element, it will automatically inherit the methods and properties from its parent.

## Add Properties

### Example

Add a property called graduationyear to the Student class:

class Student(Person):

def \_\_init\_\_(self, fname, lname):

super().\_\_init\_\_(fname, lname)

self.graduationyear = 2019

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_add_prop)

In the example below, the year 2019 should be a variable, and passed into the Student class when creating student objects. To do so, add another parameter in the \_\_init\_\_() function:

### Example

Add a year parameter, and pass the correct year when creating objects:

class Student(Person):

def \_\_init\_\_(self, fname, lname, year):

super().\_\_init\_\_(fname, lname)

self.graduationyear = year

x = Student("Mike", "Olsen", 2019)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_add_prop2)

## Add Methods

### Example

Add a method called welcome to the Student class:

class Student(Person):

def \_\_init\_\_(self, fname, lname, year):

super().\_\_init\_\_(fname, lname)

self.graduationyear = year

def welcome(self):

print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_add_method)

If you add a method in the child class with the same name as a function in the parent class, the inheritance of the parent method will be overridden.

## Test Yourself With Exercises

## Exercise:

What is the correct syntax to create a class named Student that will inherit properties and methods from a class named Person?

class :

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_inheritance1)

# Python Iterators

[❮ Previous](https://www.w3schools.com/python/python_inheritance.asp)[Next ❯](https://www.w3schools.com/python/python_scope.asp)

## Python Iterators

An iterator is an object that contains a countable number of values.

An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.

Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods \_\_iter\_\_() and \_\_next\_\_().

## Iterator vs Iterable

Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable *containers* which you can get an iterator from.

All these objects have a iter() method which is used to get an iterator:

### Example

Return an iterator from a tuple, and print each value:

mytuple = ("apple", "banana", "cherry")

myit = iter(mytuple)

print(next(myit))

print(next(myit))

print(next(myit))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator)

Even strings are iterable objects, and can return an iterator:

### Example

Strings are also iterable objects, containing a sequence of characters:

mystr = "banana"

myit = iter(mystr)

print(next(myit))

print(next(myit))

print(next(myit))

print(next(myit))

print(next(myit))

print(next(myit))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator2)

## Looping Through an Iterator

We can also use a for loop to iterate through an iterable object:

### Example

Iterate the values of a tuple:

mytuple = ("apple", "banana", "cherry")

for x in mytuple:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_loop)

### Example

Iterate the characters of a string:

mystr = "banana"

for x in mystr:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_loop2)

The for loop actually creates an iterator object and executes the next() method for each loop.

## Create an Iterator

To create an object/class as an iterator you have to implement the methods \_\_iter\_\_() and \_\_next\_\_() to your object.

As you have learned in the [Python Classes/Objects](https://www.w3schools.com/python/python_classes.asp) chapter, all classes have a function called \_\_init\_\_(), which allows you to do some initializing when the object is being created.

The \_\_iter\_\_() method acts similar, you can do operations (initializing etc.), but must always return the iterator object itself.

The \_\_next\_\_() method also allows you to do operations, and must return the next item in the sequence.

### Example

Create an iterator that returns numbers, starting with 1, and each sequence will increase by one (returning 1,2,3,4,5 etc.):

class MyNumbers:

def \_\_iter\_\_(self):

self.a = 1

return self

def \_\_next\_\_(self):

x = self.a

self.a += 1

return x

myclass = MyNumbers()

myiter = iter(myclass)

print(next(myiter))

print(next(myiter))

print(next(myiter))

print(next(myiter))

print(next(myiter))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_create)

## StopIteration

The example above would continue forever if you had enough next() statements, or if it was used in a for loop.

To prevent the iteration to go on forever, we can use the StopIteration statement.

In the \_\_next\_\_() method, we can add a terminating condition to raise an error if the iteration is done a specified number of times:

### Example

Stop after 20 iterations:

class MyNumbers:

def \_\_iter\_\_(self):

self.a = 1

return self

def \_\_next\_\_(self):

if self.a <= 20:

x = self.a

self.a += 1

return x

else:

raise StopIteration

myclass = MyNumbers()

myiter = iter(myclass)

for x in myiter:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_create2)

# Python Scope

[❮ Previous](https://www.w3schools.com/python/python_iterators.asp)[Next ❯](https://www.w3schools.com/python/python_modules.asp)

A variable is only available from inside the region it is created. This is called scope.

## Local Scope

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

### Example

A variable created inside a function is available inside that function:

def myfunc():

x = 300

print(x)

myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope1)

### Function Inside Function

As explained in the example above, the variable x is not available outside the function, but it is available for any function inside the function:

### Example

The local variable can be accessed from a function within the function:

def myfunc():

x = 300

def myinnerfunc():

print(x)

myinnerfunc()

myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope2)

## Global Scope

A variable created in the main body of the Python code is a global variable and belongs to the global scope.

Global variables are available from within any scope, global and local.

### Example

A variable created outside of a function is global and can be used by anyone:

x = 300

def myfunc():

print(x)

myfunc()

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope3)

### Naming Variables

If you operate with the same variable name inside and outside of a function, Python will treat them as two separate variables, one available in the global scope (outside the function) and one available in the local scope (inside the function):

### Example

The function will print the local x, and then the code will print the global x:

x = 300

def myfunc():

x = 200

print(x)

myfunc()

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope4)

## Global Keyword

If you need to create a global variable, but are stuck in the local scope, you can use the global keyword.

The global keyword makes the variable global.

### Example

If you use the global keyword, the variable belongs to the global scope:

def myfunc():

global x

x = 300

myfunc()

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope5)

Also, use the global keyword if you want to make a change to a global variable inside a function.

### Example

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = 300

def myfunc():

global x

x = 200

myfunc()

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope6)

# Python Modules

[❮ Previous](https://www.w3schools.com/python/python_scope.asp)[Next ❯](https://www.w3schools.com/python/python_datetime.asp)

## What is a Module?

Consider a module to be the same as a code library.

A file containing a set of functions you want to include in your application.

## Create a Module

To create a module just save the code you want in a file with the file extension .py:

### Example

Save this code in a file named mymodule.py

def greeting(name):

print("Hello, " + name)

## Use a Module

Now we can use the module we just created, by using the import statement:

### Example

Import the module named mymodule, and call the greeting function:

import mymodule

mymodule.greeting("Jonathan")

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module1)

Note: When using a function from a module, use the syntax: *module\_name.function\_name*.

## Variables in Module

The module can contain functions, as already described, but also variables of all types (arrays, dictionaries, objects etc):

### Example

Save this code in the file mymodule.py

person1 = {

"name": "John",

"age": 36,

"country": "Norway"

}

### Example

Import the module named mymodule, and access the person1 dictionary:

import mymodule

a = mymodule.person1["age"]

print(a)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module2)

## Naming a Module

You can name the module file whatever you like, but it must have the file extension .py

## Re-naming a Module

You can create an alias when you import a module, by using the as keyword:

### Example

Create an alias for mymodule called mx:

import mymodule as mx

a = mx.person1["age"]

print(a)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module3)

## Built-in Modules

There are several built-in modules in Python, which you can import whenever you like.

### Example

Import and use the platform module:

import platform

x = platform.system()

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_module4)

## Using the dir() Function

There is a built-in function to list all the function names (or variable names) in a module. The dir() function:

### Example

List all the defined names belonging to the platform module:

import platform

x = dir(platform)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_module5)

Note: The dir() function can be used on *all* modules, also the ones you create yourself.

## Import From Module

You can choose to import only parts from a module, by using the from keyword.

### Example

The module named mymodule has one function and one dictionary:

def greeting(name):

print("Hello, " + name)

person1 = {

"name": "John",

"age": 36,

"country": "Norway"

}

### Example

Import only the person1 dictionary from the module:

from mymodule import person1

print (person1["age"])

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module6)

Note: When importing using the from keyword, do not use the module name when referring to elements in the module. Example: person1["age"], not ~~mymodule.person1["age"]~~

## Test Yourself With Exercises

## Exercise:

What is the correct syntax to import a module named "mymodule"?

mymodule

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_modules1)

# Python Datetime

[❮ Previous](https://www.w3schools.com/python/python_modules.asp)[Next ❯](https://www.w3schools.com/python/python_math.asp)

## Python Dates

A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

### Example

Import the datetime module and display the current date:

import datetime

x = datetime.datetime.now()

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime1)

## Date Output

When we execute the code from the example above the result will be:

2021-06-16 11:47:26.677810

The date contains year, month, day, hour, minute, second, and microsecond.

The datetime module has many methods to return information about the date object.

Here are a few examples, you will learn more about them later in this chapter:

### Example

Return the year and name of weekday:

import datetime

x = datetime.datetime.now()

print(x.year)

print(x.strftime("%A"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime2)

## Creating Date Objects

To create a date, we can use the datetime() class (constructor) of the datetime module.

The datetime() class requires three parameters to create a date: year, month, day.

### Example

Create a date object:

import datetime

x = datetime.datetime(2020, 5, 17)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime3)

The datetime() class also takes parameters for time and timezone (hour, minute, second, microsecond, tzone), but they are optional, and has a default value of 0, (None for timezone).

## The strftime() Method

The datetime object has a method for formatting date objects into readable strings.

The method is called strftime(), and takes one parameter, format, to specify the format of the returned string:

### Example

Display the name of the month:

import datetime

x = datetime.datetime(2018, 6, 1)

print(x.strftime("%B"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime)

A reference of all the legal format codes:

| **Directive** | **Description** | **Example** | **Try it** |
| --- | --- | --- | --- |
| %a | Weekday, short version | Wed | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a) |
| %A | Weekday, full version | Wednesday | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a2) |
| %w | Weekday as a number 0-6, 0 is Sunday | 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w) |
| %d | Day of month 01-31 | 31 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_d) |
| %b | Month name, short version | Dec | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b) |
| %B | Month name, full version | December | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b2) |
| %m | Month as a number 01-12 | 12 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m) |
| %y | Year, short version, without century | 18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y) |
| %Y | Year, full version | 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y2) |
| %H | Hour 00-23 | 17 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_h2) |
| %I | Hour 00-12 | 05 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_i2) |
| %p | AM/PM | PM | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_p) |
| %M | Minute 00-59 | 41 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m2) |
| %S | Second 00-59 | 08 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_s2) |
| %f | Microsecond 000000-999999 | 548513 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_f) |
| %z | UTC offset | +0100 |  |
| %Z | Timezone | CST |  |
| %j | Day number of year 001-366 | 365 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_j) |
| %U | Week number of year, Sunday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_u2) |
| %W | Week number of year, Monday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w2) |
| %c | Local version of date and time | Mon Dec 31 17:41:00 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_c) |
| %x | Local version of date | 12/31/18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x) |
| %X | Local version of time | 17:41:00 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x2) |
| %% | A % character | % | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_percent) |
| %G | ISO 8601 year | 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_g) |
| %u | ISO 8601 weekday (1-7) | 1 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_u) |
| %V | ISO 8601 weeknumber (01-53) | 01 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_v) |

# Python Math

[❮ Previous](https://www.w3schools.com/python/python_datetime.asp)[Next ❯](https://www.w3schools.com/python/python_json.asp)

Python has a set of built-in math functions, including an extensive math module, that allows you to perform mathematical tasks on numbers.

## Built-in Math Functions

The min() and max() functions can be used to find the lowest or highest value in an iterable:

### Example

x = min(5, 10, 25)

y = max(5, 10, 25)

print(x)

print(y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_math_min_max)

The abs() function returns the absolute (positive) value of the specified number:

### Example

x = abs(-7.25)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_math_abs)

The pow(*x*, *y*) function returns the value of x to the power of y (xy).

### Example

Return the value of 4 to the power of 3 (same as 4 \* 4 \* 4):

x = pow(4, 3)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_math_pow)

## The Math Module

Python has also a built-in module called math, which extends the list of mathematical functions.

To use it, you must import the math module:

import math

When you have imported the math module, you can start using methods and constants of the module.

The math.sqrt() method for example, returns the square root of a number:

### Example

import math

x = math.sqrt(64)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_math_sqrt)

The math.ceil() method rounds a number upwards to its nearest integer, and the math.floor() method rounds a number downwards to its nearest integer, and returns the result:

### Example

import math

x = math.ceil(1.4)

y = math.floor(1.4)

print(x) # returns 2

print(y) # returns 1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_math_ceil_floor)

The math.pi constant, returns the value of PI (3.14...):

### Example

import math

x = math.pi

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_math_pi)

## Complete Math Module Reference

In our [Math Module Reference](https://www.w3schools.com/python/module_math.asp) you will find a complete reference of all methods and constants that belongs to the Math module.

# Python JSON

[❮ Previous](https://www.w3schools.com/python/python_math.asp)[Next ❯](https://www.w3schools.com/python/python_regex.asp)

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

## JSON in Python

Python has a built-in package called json, which can be used to work with JSON data.

### Example

Import the json module:

import json

## Parse JSON - Convert from JSON to Python

If you have a JSON string, you can parse it by using the json.loads() method.

The result will be a [Python dictionary](https://www.w3schools.com/python/python_dictionaries.asp).

### Example

Convert from JSON to Python:

import json

# some JSON:

x = '{ "name":"John", "age":30, "city":"New York"}'

# parse x:

y = json.loads(x)

# the result is a Python dictionary:

print(y["age"])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json)

## Convert from Python to JSON

If you have a Python object, you can convert it into a JSON string by using the json.dumps() method.

### Example

Convert from Python to JSON:

import json

# a Python object (dict):

x = {

"name": "John",

"age": 30,

"city": "New York"

}

# convert into JSON:

y = json.dumps(x)

# the result is a JSON string:

print(y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python)

You can convert Python objects of the following types, into JSON strings:

* dict
* list
* tuple
* string
* int
* float
* True
* False
* None

### Example

Convert Python objects into JSON strings, and print the values:

import json

print(json.dumps({"name": "John", "age": 30}))

print(json.dumps(["apple", "bananas"]))

print(json.dumps(("apple", "bananas")))

print(json.dumps("hello"))

print(json.dumps(42))

print(json.dumps(31.76))

print(json.dumps(True))

print(json.dumps(False))

print(json.dumps(None))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_all)

When you convert from Python to JSON, Python objects are converted into the JSON (JavaScript) equivalent:

| **Python** | **JSON** |
| --- | --- |
| dict | Object |
| list | Array |
| tuple | Array |
| str | String |
| int | Number |
| float | Number |
| True | true |
| False | false |
| None | null |

### Example

Convert a Python object containing all the legal data types:

import json

x = {

"name": "John",

"age": 30,

"married": True,

"divorced": False,

"children": ("Ann","Billy"),

"pets": None,

"cars": [

{"model": "BMW 230", "mpg": 27.5},

{"model": "Ford Edge", "mpg": 24.1}

]

}

print(json.dumps(x))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_all_in_one)

## Format the Result

The example above prints a JSON string, but it is not very easy to read, with no indentations and line breaks.

The json.dumps() method has parameters to make it easier to read the result:

### Example

Use the indent parameter to define the numbers of indents:

json.dumps(x, indent=4)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_indent)

You can also define the separators, default value is (", ", ": "), which means using a comma and a space to separate each object, and a colon and a space to separate keys from values:

### Example

Use the separators parameter to change the default separator:

json.dumps(x, indent=4, separators=(". ", " = "))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_separators)

## Order the Result

The json.dumps() method has parameters to order the keys in the result:

### Example

Use the sort\_keys parameter to specify if the result should be sorted or not:

json.dumps(x, indent=4, sort\_keys=True)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_sort_keys)

# Python RegEx

[❮ Previous](https://www.w3schools.com/python/python_json.asp)[Next ❯](https://www.w3schools.com/python/python_pip.asp)

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

## RegEx Module

Python has a built-in package called re, which can be used to work with Regular Expressions.

Import the re module:

import re

## RegEx in Python

When you have imported the re module, you can start using regular expressions:

### Example

Search the string to see if it starts with "The" and ends with "Spain":

import re

txt = "The rain in Spain"

x = re.search("^The.\*Spain$", txt)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex)

## RegEx Functions

The re module offers a set of functions that allows us to search a string for a match:

| **Function** | **Description** |
| --- | --- |
| [findall](https://www.w3schools.com/python/python_regex.asp#findall) | Returns a list containing all matches |
| [search](https://www.w3schools.com/python/python_regex.asp#search) | Returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match anywhere in the string |
| [split](https://www.w3schools.com/python/python_regex.asp#split) | Returns a list where the string has been split at each match |
| [sub](https://www.w3schools.com/python/python_regex.asp#sub) | Replaces one or many matches with a string |

## Metacharacters

Metacharacters are characters with a special meaning:

| **Character** | **Description** | **Example** | **Try it** |
| --- | --- | --- | --- |
| [] | A set of characters | "[a-m]" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta1) |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta2) |
| . | Any character (except newline character) | "he..o" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta3) |
| ^ | Starts with | "^hello" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta4) |
| $ | Ends with | "world$" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta5) |
| \* | Zero or more occurrences | "aix\*" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta6) |
| + | One or more occurrences | "aix+" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta7) |
| {} | Exactly the specified number of occurrences | "al{2}" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta8) |
| | | Either or | "falls|stays" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta9) |
| () | Capture and group |  |  |

## Special Sequences

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

| **Character** | **Description** | **Example** | **Try it** |
| --- | --- | --- | --- |
| \A | Returns a match if the specified characters are at the beginning of the string | "\AThe" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq1) |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word  (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\bain"  r"ain\b" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq2)  [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq2-2) |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word  (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\Bain"  r"ain\B" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq3)  [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq3-2) |
| \d | Returns a match where the string contains digits (numbers from 0-9) | "\d" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq4) |
| \D | Returns a match where the string DOES NOT contain digits | "\D" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq5) |
| \s | Returns a match where the string contains a white space character | "\s" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq6) |
| \S | Returns a match where the string DOES NOT contain a white space character | "\S" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq7) |
| \w | Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore \_ character) | "\w" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq8) |
| \W | Returns a match where the string DOES NOT contain any word characters | "\W" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq9) |
| \Z | Returns a match if the specified characters are at the end of the string | "Spain\Z" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq10) |

## Sets

A set is a set of characters inside a pair of square brackets [] with a special meaning:

| **Set** | **Description** | **Try it** |
| --- | --- | --- |
| [arn] | Returns a match where one of the specified characters (a, r, or n) are present | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set1) |
| [a-n] | Returns a match for any lower case character, alphabetically between a and n | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set2) |
| [^arn] | Returns a match for any character EXCEPT a, r, and n | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set3) |
| [0123] | Returns a match where any of the specified digits (0, 1, 2, or 3) are present | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set4) |
| [0-9] | Returns a match for any digit between 0 and 9 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set5) |
| [0-5][0-9] | Returns a match for any two-digit numbers from 00 and 59 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set6) |
| [a-zA-Z] | Returns a match for any character alphabetically between a and z, lower case OR upper case | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set7) |
| [+] | In sets, +, \*, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set8) |

## The findall() Function

The findall() function returns a list containing all matches.

### Example

Print a list of all matches:

import re

txt = "The rain in Spain"

x = re.findall("ai", txt)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_findall)

The list contains the matches in the order they are found.

If no matches are found, an empty list is returned:

### Example

Return an empty list if no match was found:

import re

txt = "The rain in Spain"

x = re.findall("Portugal", txt)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_findall2)

## The search() Function

The search() function searches the string for a match, and returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

### Example

Search for the first white-space character in the string:

import re

txt = "The rain in Spain"

x = re.search("\s", txt)

print("The first white-space character is located in position:", x.start())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_search)

If no matches are found, the value None is returned:

### Example

Make a search that returns no match:

import re

txt = "The rain in Spain"

x = re.search("Portugal", txt)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_search2)

## The split() Function

The split() function returns a list where the string has been split at each match:

### Example

Split at each white-space character:

import re

txt = "The rain in Spain"

x = re.split("\s", txt)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_split)

You can control the number of occurrences by specifying the maxsplit parameter:

### Example

Split the string only at the first occurrence:

import re

txt = "The rain in Spain"

x = re.split("\s", txt, 1)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_split2)

## The sub() Function

The sub() function replaces the matches with the text of your choice:

### Example

Replace every white-space character with the number 9:

import re

txt = "The rain in Spain"

x = re.sub("\s", "9", txt)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_sub)

You can control the number of replacements by specifying the count parameter:

### Example

Replace the first 2 occurrences:

import re

txt = "The rain in Spain"

x = re.sub("\s", "9", txt, 2)

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_sub2)

## Match Object

A Match Object is an object containing information about the search and the result.

Note: If there is no match, the value None will be returned, instead of the Match Object.

### Example

Do a search that will return a Match Object:

import re

txt = "The rain in Spain"

x = re.search("ai", txt)

print(x) #this will print an object

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match)

The Match object has properties and methods used to retrieve information about the search, and the result:

.span() returns a tuple containing the start-, and end positions of the match.

.string returns the string passed into the function

.group() returns the part of the string where there was a match

### Example

Print the position (start- and end-position) of the first match occurrence.

The regular expression looks for any words that starts with an upper case "S":

import re

txt = "The rain in Spain"

x = re.search(r"\bS\w+", txt)

print(x.span())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match_span)

### Example

Print the string passed into the function:

import re

txt = "The rain in Spain"

x = re.search(r"\bS\w+", txt)

print(x.string)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match_string)

### Example

Print the part of the string where there was a match.

The regular expression looks for any words that starts with an upper case "S":

import re

txt = "The rain in Spain"

x = re.search(r"\bS\w+", txt)

print(x.group())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match_group)

Note: If there is no match, the value None will be returned, instead of the Match Object.

# Python PIP

[❮ Previous](https://www.w3schools.com/python/python_regex.asp)[Next ❯](https://www.w3schools.com/python/python_try_except.asp)

## What is PIP?

PIP is a package manager for Python packages, or modules if you like.

Note: If you have Python version 3.4 or later, PIP is included by default.

## What is a Package?

A package contains all the files you need for a module.

Modules are Python code libraries you can include in your project.

## Check if PIP is Installed

Navigate your command line to the location of Python's script directory, and type the following:

### Example

Check PIP version:

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>pip --version

## Install PIP

If you do not have PIP installed, you can download and install it from this page: <https://pypi.org/project/pip/>

## Download a Package

Downloading a package is very easy.

Open the command line interface and tell PIP to download the package you want.

Navigate your command line to the location of Python's script directory, and type the following:

### Example

Download a package named "camelcase":

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>pip install camelcase

Now you have downloaded and installed your first package!

## Using a Package

Once the package is installed, it is ready to use.

Import the "camelcase" package into your project.

### Example

Import and use "camelcase":

import camelcase

c = camelcase.CamelCase()

txt = "hello world"

print(c.hump(txt))

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_camelcase)

## Find Packages

Find more packages at <https://pypi.org/>.

## Remove a Package

Use the uninstall command to remove a package:

### Example

Uninstall the package named "camelcase":

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>pip uninstall camelcase

The PIP Package Manager will ask you to confirm that you want to remove the camelcase package:

Uninstalling camelcase-02.1:

Would remove:

c:\users\*Your Name*\appdata\local\programs\python\python36-32\lib\site-packages\camecase-0.2-py3.6.egg-info

c:\users\*Your Name*\appdata\local\programs\python\python36-32\lib\site-packages\camecase\\*

Proceed (y/n)?

Press y and the package will be removed.

## List Packages

Use the list command to list all the packages installed on your system:

### Example

List installed packages:

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>pip list

Result:

Package Version

-----------------------

camelcase 0.2

mysql-connector 2.1.6

pip 18.1

pymongo 3.6.1

setuptools 39.0.1

# Python Try Except

[❮ Previous](https://www.w3schools.com/python/python_pip.asp)[Next ❯](https://www.w3schools.com/python/python_user_input.asp)

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

## Exception Handling

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

These exceptions can be handled using the try statement:

### Example

The try block will generate an exception, because x is not defined:

try:

print(x)

except:

print("An exception occurred")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except)

Since the try block raises an error, the except block will be executed.

Without the try block, the program will crash and raise an error:

### Example

This statement will raise an error, because x is not defined:

print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except_error)

## Many Exceptions

You can define as many exception blocks as you want, e.g. if you want to execute a special block of code for a special kind of error:

### Example

Print one message if the try block raises a NameError and another for other errors:

try:

print(x)

except NameError:

print("Variable x is not defined")

except:

print("Something else went wrong")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except2)

## Else

You can use the else keyword to define a block of code to be executed if no errors were raised:

### Example

In this example, the try block does not generate any error:

try:

print("Hello")

except:

print("Something went wrong")

else:

print("Nothing went wrong")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except3)

## Finally

The finally block, if specified, will be executed regardless if the try block raises an error or not.

### Example

try:

print(x)

except:

print("Something went wrong")

finally:

print("The 'try except' is finished")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except4)

This can be useful to close objects and clean up resources:

### Example

Try to open and write to a file that is not writable:

try:

f = open("demofile.txt")

f.write("Lorum Ipsum")

except:

print("Something went wrong when writing to the file")

finally:

f.close()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except5)

The program can continue, without leaving the file object open.

## Raise an exception

As a Python developer you can choose to throw an exception if a condition occurs.

To throw (or raise) an exception, use the raise keyword.

### Example

Raise an error and stop the program if x is lower than 0:

x = -1

if x < 0:

raise Exception("Sorry, no numbers below zero")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ref_keyword_raise)

The raise keyword is used to raise an exception.

You can define what kind of error to raise, and the text to print to the user.

### Example

Raise a TypeError if x is not an integer:

x = "hello"

if not type(x) is int:

raise TypeError("Only integers are allowed")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ref_keyword_raise2)

# Python User Input

[❮ Previous](https://www.w3schools.com/python/python_try_except.asp)[Next ❯](https://www.w3schools.com/python/python_string_formatting.asp)

## User Input

Python allows for user input.

That means we are able to ask the user for input.

The method is a bit different in Python 3.6 than Python 2.7.

Python 3.6 uses the input() method.

Python 2.7 uses the raw\_input() method.

The following example asks for the username, and when you entered the username, it gets printed on the screen:

### Python 3.6

username = input("Enter username:")

print("Username is: " + username)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_user_input3)

### Python 2.7

username = raw\_input("Enter username:")

print("Username is: " + username)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_user_input2)

Python stops executing when it comes to the input() function, and continues when the user has given some input.

# Python String Formatting

[❮ Previous](https://www.w3schools.com/python/python_user_input.asp)[Next ❯](https://www.w3schools.com/python/python_file_handling.asp)

To make sure a string will display as expected, we can format the result with the format() method.

## String format()

The format() method allows you to format selected parts of a string.

Sometimes there are parts of a text that you do not control, maybe they come from a database, or user input?

To control such values, add placeholders (curly brackets {}) in the text, and run the values through the format() method:

### Example

Add a placeholder where you want to display the price:

price = 49

txt = "The price is {} dollars"

print(txt.format(price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting1)

You can add parameters inside the curly brackets to specify how to convert the value:

### Example

Format the price to be displayed as a number with two decimals:

txt = "The price is {:.2f} dollars"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting2)

Check out all formatting types in our [String format() Reference](https://www.w3schools.com/python/ref_string_format.asp).

## Multiple Values

If you want to use more values, just add more values to the format() method:

print(txt.format(price, itemno, count))

And add more placeholders:

### Example

quantity = 3

itemno = 567

price = 49

myorder = "I want {} pieces of item number {} for {:.2f} dollars."

print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting3)

## Index Numbers

You can use index numbers (a number inside the curly brackets {0}) to be sure the values are placed in the correct placeholders:

### Example

quantity = 3

itemno = 567

price = 49

myorder = "I want {0} pieces of item number {1} for {2:.2f} dollars."

print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting4)

Also, if you want to refer to the same value more than once, use the index number:

### Example

age = 36

name = "John"

txt = "His name is {1}. {1} is {0} years old."

print(txt.format(age, name))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting5)

## Named Indexes

You can also use named indexes by entering a name inside the curly brackets {carname}, but then you must use names when you pass the parameter values txt.format(carname = "Ford"):

### Example

myorder = "I have a {carname}, it is a {model}."

print(myorder.format(carname = "Ford", model = "Mustang"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting6)

# Python File Open

[❮ Previous](https://www.w3schools.com/python/python_string_formatting.asp)[Next ❯](https://www.w3schools.com/python/python_file_open.asp)

File handling is an important part of any web application.

Python has several functions for creating, reading, updating, and deleting files.

## File Handling

The key function for working with files in Python is the open() function.

The open() function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

## Syntax

To open a file for reading it is enough to specify the name of the file:

f = open("demofile.txt")

The code above is the same as:

f = open("demofile.txt", "rt")

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

Note: Make sure the file exists, or else you will get an error.

# Python File Open

[❮ Previous](https://www.w3schools.com/python/python_file_handling.asp)[Next ❯](https://www.w3schools.com/python/python_file_write.asp)

## Open a File on the Server

Assume we have the following file, located in the same folder as Python:

demofile.txt

Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!

To open the file, use the built-in open() function.

The open() function returns a file object, which has a read() method for reading the content of the file:

### Example

f = open("demofile.txt", "r")

print(f.read())

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_open)

If the file is located in a different location, you will have to specify the file path, like this:

### Example

Open a file on a different location:

f = open("D:\\myfiles\welcome.txt", "r")

print(f.read())

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_open_d)

## Read Only Parts of the File

By default the read() method returns the whole text, but you can also specify how many characters you want to return:

### Example

Return the 5 first characters of the file:

f = open("demofile.txt", "r")

print(f.read(5))

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_open2)

## Read Lines

You can return one line by using the readline() method:

### Example

Read one line of the file:

f = open("demofile.txt", "r")

print(f.readline())

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_readline)

By calling readline() two times, you can read the two first lines:

### Example

Read two lines of the file:

f = open("demofile.txt", "r")

print(f.readline())

print(f.readline())

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_readline2)

By looping through the lines of the file, you can read the whole file, line by line:

### Example

Loop through the file line by line:

f = open("demofile.txt", "r")

for x in f:

print(x)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_readline3)

## Close Files

It is a good practice to always close the file when you are done with it.

### Example

Close the file when you are finish with it:

f = open("demofile.txt", "r")

print(f.readline())

f.close()

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_close)

Note: You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

# Python File Write

[❮ Previous](https://www.w3schools.com/python/python_file_open.asp)[Next ❯](https://www.w3schools.com/python/python_file_remove.asp)

## Write to an Existing File

To write to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

### Example

Open the file "demofile2.txt" and append content to the file:

f = open("demofile2.txt", "a")

f.write("Now the file has more content!")

f.close()

#open and read the file after the appending:

f = open("demofile2.txt", "r")

print(f.read())

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_append)

### Example

Open the file "demofile3.txt" and overwrite the content:

f = open("demofile3.txt", "w")

f.write("Woops! I have deleted the content!")

f.close()

#open and read the file after the appending:

f = open("demofile3.txt", "r")

print(f.read())

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_file_write)

Note: the "w" method will overwrite the entire file.

## Create a New File

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

### Example

Create a file called "myfile.txt":

f = open("myfile.txt", "x")

Result: a new empty file is created!

### Example

Create a new file if it does not exist:

f = open("myfile.txt", "w")

[❮ Previous](https://www.w3schools.com/python/python_file_open.asp)[Next ❯](https://www.w3schools.com/python/python_file_remove.asp)

# Python Delete File

[❮ Previous](https://www.w3schools.com/python/python_file_write.asp)[Next ❯](https://www.w3schools.com/python/matplotlib_intro.asp)

## Delete a File

To delete a file, you must import the OS module, and run its os.remove() function:

### Example

Remove the file "demofile.txt":

import os

os.remove("demofile.txt")

## Check if File exist:

To avoid getting an error, you might want to check if the file exists before you try to delete it:

### Example

Check if file exists, *then* delete it:

import os

if os.path.exists("demofile.txt"):

os.remove("demofile.txt")

else:

print("The file does not exist")

## Delete Folder

To delete an entire folder, use the os.rmdir() method:

### Example

Remove the folder "myfolder":

import os

os.rmdir("myfolder")

Note: You can only remove *empty* folders.

# Python MySQL

[❮ Previous](https://www.w3schools.com/python/python_ml_decision_tree.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_create_db.asp)

Python can be used in database applications.

One of the most popular databases is MySQL.

## MySQL Database

To be able to experiment with the code examples in this tutorial, you should have MySQL installed on your computer.

You can download a free MySQL database at <https://www.mysql.com/downloads/>.

## Install MySQL Driver

Python needs a MySQL driver to access the MySQL database.

In this tutorial we will use the driver "MySQL Connector".

We recommend that you use PIP to install "MySQL Connector".

PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following:

Download and install "MySQL Connector":

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install mysql-connector-python

Now you have downloaded and installed a MySQL driver.

## Test MySQL Connector

To test if the installation was successful, or if you already have "MySQL Connector" installed, create a Python page with the following content:

demo\_mysql\_test.py:

import mysql.connector

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_test)

If the above code was executed with no errors, "MySQL Connector" is installed and ready to be used.

## Create Connection

Start by creating a connection to the database.

Use the username and password from your MySQL database:

demo\_mysql\_connection.py:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*"

)

print(mydb)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_connection)

# Python MySQL Create Database

[❮ Previous](https://www.w3schools.com/python/python_mysql_getstarted.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_create_table.asp)

## Creating a Database

To create a database in MySQL, use the "CREATE DATABASE" statement:

### Example

create a database named "mydatabase":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*"

)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE mydatabase")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_create_db)

If the above code was executed with no errors, you have successfully created a database.

## Check if Database Exists

You can check if a database exist by listing all databases in your system by using the "SHOW DATABASES" statement:

### Example

Return a list of your system's databases:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*"

)

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

for x in mycursor:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_show_databases)

Or you can try to access the database when making the connection:

### Example

Try connecting to the database "mydatabase":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_db_exist)

# Python MySQL Create Table

[❮ Previous](https://www.w3schools.com/python/python_mysql_create_db.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_insert.asp)

## Creating a Table

To create a table in MySQL, use the "CREATE TABLE" statement.

Make sure you define the name of the database when you create the connection

### Example

Create a table named "customers":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_create_table)

If the above code was executed with no errors, you have now successfully created a table.

## Check if Table Exists

You can check if a table exist by listing all tables in your database with the "SHOW TABLES" statement:

### Example

Return a list of your system's databases:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

for x in mycursor:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_show_tables)

## Primary Key

When creating a table, you should also create a column with a unique key for each record.

This can be done by defining a PRIMARY KEY.

We use the statement "INT AUTO\_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.

### Example

Create primary key when creating the table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255))")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_primary_key)

If the table already exists, use the ALTER TABLE keyword:

### Example

Create primary key on an existing table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO\_INCREMENT PRIMARY KEY")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_alter_primary_key)

[❮ Previous](https://www.w3schools.com/python/python_mysql_create_db.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_insert.asp)

# Python MySQL Insert Into Table

[❮ Previous](https://www.w3schools.com/python/python_mysql_create_table.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_select.asp)

## Insert Into Table

To fill a table in MySQL, use the "INSERT INTO" statement.

### Example

Insert a record in the "customers" table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = ("John", "Highway 21")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_insert)

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

## Insert Multiple Rows

To insert multiple rows into a table, use the executemany() method.

The second parameter of the executemany() method is a list of tuples, containing the data you want to insert:

### Example

Fill the "customers" table with data:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = [

('Peter', 'Lowstreet 4'),

('Amy', 'Apple st 652'),

('Hannah', 'Mountain 21'),

('Michael', 'Valley 345'),

('Sandy', 'Ocean blvd 2'),

('Betty', 'Green Grass 1'),

('Richard', 'Sky st 331'),

('Susan', 'One way 98'),

('Vicky', 'Yellow Garden 2'),

('Ben', 'Park Lane 38'),

('William', 'Central st 954'),

('Chuck', 'Main Road 989'),

('Viola', 'Sideway 1633')

]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_insert_many)

## Get Inserted ID

You can get the id of the row you just inserted by asking the cursor object.

Note: If you insert more than one row, the id of the last inserted row is returned.

### Example

Insert one row, and return the ID:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = ("Michelle", "Blue Village")

mycursor.execute(sql, val)

mydb.commit()

print("1 record inserted, ID:", mycursor.lastrowid)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_insert_id)

# Python MySQL Select From

[❮ Previous](https://www.w3schools.com/python/python_mysql_insert.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_where.asp)

## Select From a Table

To select from a table in MySQL, use the "SELECT" statement:

### Example

Select all records from the "customers" table, and display the result:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_select)

Note: We use the fetchall() method, which fetches all rows from the last executed statement.

## Selecting Columns

To select only some of the columns in a table, use the "SELECT" statement followed by the column name(s):

### Example

Select only the name and address columns:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_select_columns)

## Using the fetchone() Method

If you are only interested in one row, you can use the fetchone() method.

The fetchone() method will return the first row of the result:

### Example

Fetch only one row:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers")

myresult = mycursor.fetchone()

print(myresult)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_select_fetchone)

# Python MySQL Where

[❮ Previous](https://www.w3schools.com/python/python_mysql_select.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_orderby.asp)

## Select With a Filter

When selecting records from a table, you can filter the selection by using the "WHERE" statement:

### Example

Select record(s) where the address is "Park Lane 38": result:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers WHERE address ='Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_where)

## Wildcard Characters

You can also select the records that starts, includes, or ends with a given letter or phrase.

Use the % to represent wildcard characters:

### Example

Select records where the address contains the word "way":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers WHERE address LIKE '%way%'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_where_wildcard)

## Prevent SQL Injection

When query values are provided by the user, you should escape the values.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module has methods to escape query values:

### Example

Escape query values by using the placholder %s method:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers WHERE address = %s"

adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_where_escape)

# Python MySQL Order By

[❮ Previous](https://www.w3schools.com/python/python_mysql_where.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_delete.asp)

## Sort the Result

Use the ORDER BY statement to sort the result in ascending or descending order.

The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword.

### Example

Sort the result alphabetically by name: result:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_orderby)

## ORDER BY DESC

Use the DESC keyword to sort the result in a descending order.

### Example

Sort the result reverse alphabetically by name:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers ORDER BY name DESC"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_orderby_desc)

# Python MySQL Delete From By

[❮ Previous](https://www.w3schools.com/python/python_mysql_orderby.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_drop_table.asp)

## Delete Record

You can delete records from an existing table by using the "DELETE FROM" statement:

### Example

Delete any record where the address is "Mountain 21":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_delete)

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted!

## Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in delete statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module uses the placeholder %s to escape values in the delete statement:

### Example

Escape values by using the placeholder %s method:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = %s"

adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_delete_escape)

[❮ Previous](https://www.w3schools.com/python/python_mysql_orderby.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_drop_table.asp)

# Python MySQL Drop Table

[❮ Previous](https://www.w3schools.com/python/python_mysql_delete.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_update.asp)

## Delete a Table

You can delete an existing table by using the "DROP TABLE" statement:

### Example

Delete the table "customers":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_drop_table)

## Drop Only if Exist

If the the table you want to delete is already deleted, or for any other reason does not exist, you can use the IF EXISTS keyword to avoid getting an error.

### Example

Delete the table "customers" if it exists:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DROP TABLE IF EXISTS customers"

mycursor.execute(sql)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_drop_table2)

# Python MySQL Update Table

[❮ Previous](https://www.w3schools.com/python/python_mysql_drop_table.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_limit.asp)

## Update Table

You can update existing records in a table by using the "UPDATE" statement:

### Example

Overwrite the address column from "Valley 345" to "Canyoun 123":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_update)

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

## Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in update statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module uses the placeholder %s to escape values in the delete statement:

### Example

Escape values by using the placholder %s method:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = %s WHERE address = %s"

val = ("Valley 345", "Canyon 123")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_update_escape)

# Python MySQL Limit

[❮ Previous](https://www.w3schools.com/python/python_mysql_update.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_join.asp)

## Limit the Result

You can limit the number of records returned from the query, by using the "LIMIT" statement:

### Example

Select the 5 first records in the "customers" table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers LIMIT 5")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_limit)

## Start From Another Position

If you want to return five records, starting from the third record, you can use the "OFFSET" keyword:

### Example

Start from position 3, and return 5 records:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers LIMIT 5 OFFSET 2")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_limit_offset)

[❮ Previous](https://www.w3schools.com/python/python_mysql_update.asp)[Next ❯](https://www.w3schools.com/python/python_mysql_join.asp)

# Python MySQL Join

[❮ Previous](https://www.w3schools.com/python/python_mysql_limit.asp)[Next ❯](https://www.w3schools.com/python/python_mongodb_getstarted.asp)

## Join Two or More Tables

You can combine rows from two or more tables, based on a related column between them, by using a JOIN statement.

Consider you have a "users" table and a "products" table:

### users

{ id: 1, name: 'John', fav: 154},

{ id: 2, name: 'Peter', fav: 154},

{ id: 3, name: 'Amy', fav: 155},

{ id: 4, name: 'Hannah', fav:},

{ id: 5, name: 'Michael', fav:}

### products

{ id: 154, name: 'Chocolate Heaven' },

{ id: 155, name: 'Tasty Lemons' },

{ id: 156, name: 'Vanilla Dreams' }

These two tables can be combined by using users' fav field and products' id field.

### Example

Join users and products to see the name of the users favorite product:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \

users.name AS user, \

products.name AS favorite \

FROM users \

INNER JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_inner_join)

Note: You can use JOIN instead of INNER JOIN. They will both give you the same result.

## LEFT JOIN

In the example above, Hannah, and Michael were excluded from the result, that is because INNER JOIN only shows the records where there is a match.

If you want to show all users, even if they do not have a favorite product, use the LEFT JOIN statement:

### Example

Select all users and their favorite product:

sql = "SELECT \

users.name AS user, \

products.name AS favorite \

FROM users \

LEFT JOIN products ON users.fav = products.id"

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_left_join)

## RIGHT JOIN

If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the RIGHT JOIN statement:

### Example

Select all products, and the user(s) who have them as their favorite:

sql = "SELECT \

users.name AS user, \

products.name AS favorite \

FROM users \

RIGHT JOIN products ON users.fav = products.id"

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_right_join)

Note: Hannah and Michael, who have no favorite product, are not included in the result.

[❮ Previous](https://www.w3schools.com/python/python_mysql_limit.asp)[Next ❯](https://www.w3schools.com/python/python_mongodb_getstarted.asp)