## SURVEY OF TECHNOLOGIES

***Introduction to Python:***   
**What is Python?**

Python is an interpreted high level general purpose programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

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

**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.
* Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.
* It supports multiple programming paradigms,including structured (particularly,procedural),object-oriented and functional programming.
* It is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

**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-orientated 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.

Main Concepts in Python Which are Used While Coding:-

# **Python Virtual Environment**

## Introduction

Developers often deal with python projects where they have to use module and packages which were not part of the python standard library and they need it for this particular application only. Consider a case, where you have installed the current version of python (let’s assume its python-3.6) but your project requires specific python version--2.7 for this particular application, so it's not just the new packages but requirement may come for a particular version of your already installed application. Then the requirements are in conflict and installing either version 2.7 or 3.6 will leave one application unable to run.

One solution to resolve these circumstances is to create a virtual environment for your project. Virutalenv is a kind of tool that allows us to create isolated python environments. Virtualenv creates a self-contained folder which contains all the required executables to use the packages that a Python project would require in its project.

## Why we want it?

The main purpose of using virtualenv (virutal environment tool) is to resolve the issues of dependencies, versions (of python packages) and indirectly permissions.

## Requirements for installing Virtual Environment

First thing first, you need to have the python installed in your machine (not necessarily the latest version) and pip package manager. However, if you are using python 3.4 version or higher, pip is included by default (comes as python standard library). In case you don’t have any one of them, it’s recommended to installing it first.

## Creating Virtual Environments

Open your command prompt(type cmd in your run terminal). Now go to the directory path(location), where you want to install the virtual environment.

In case you are not using python 3.x, then you need to install the virtualenv tool with pip.

**Shell**

pip install virtualenv

In case virtualenv is already installed (either you are using python 3 version or you’re not aware of virtualenv installed already), then running above command will give you message something like,

>pip install virtualenv

Requirement already satisfied: virtualenv in c:\python\python361\lib\site-packages (15.1.0)

Start by creating a new folder to work with

**Shell**

mkdir python-virtual-environments && cd python-virtual-environments

After running above command, you are inside the newly created folder. Now create a new virtual environment inside the directory (the directory you created above).

**Shell**

#For python 2.x version −

virtualenv myenv

# For python 3.x version −

python -m venv myenv

Above script will create a new folder name myenv with a couple of directories and lots of files, with a directory structure similar to below -

├── Include

│ ├── abstract.h

│ ├── accu.h

│ ├── asdl.h

│ ├── ast.h

│ ├── bitset.h

………

├── Lib

│ ├── \_\_future\_\_.py

│ ├── \_\_pycache\_\_

│ ├── \_bootlocale.py

│ ├── \_collections\_abc.py

│ ├── \_dummy\_thread.py

│ ├── \_weakrefset.py

│ ├── abc.py

│ ├── base64.py

│ ├── bisect.py

│ ├── codecs.py

……

├── pip-selfcheck.json

├── Scripts

│ ├── activate

│ ├── activate.bat

│ ├── activate.ps1

│ ├── activate\_this.py

│ ├── deactivate.bat

│ ├── easy\_install.exe

│ ├── easy\_install-3.6.exe

│ ├── pip.exe

│ ├── pip3.6.exe

│ ├── pip3.exe

│ ├── python.exe

│ ├── python36.dll

│ ├── pythonw.exe

│ └── wheel.exe

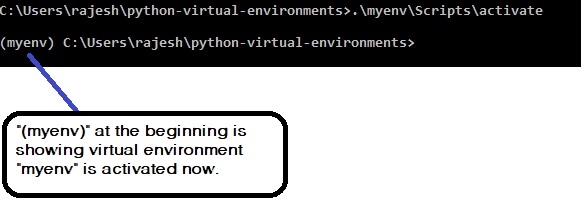
Where −

* Include (directory): C headers that compile the python package
* Scripts(directory): files that interact with the virtual environment
* Lib(directory): Contains the python version copy and site-packages directory where each dependency is installed.

## How to activate the virtual environment?

One of the interesting files is the activate scripts in the scripts directory. The activate scripts used the environment’s python executables and its site-package by default to set up your shell.

However, to use this virtual environment “myenv” packages or resources in isolation, you need to “activate” it first. To activate your virtual environment run the command as shown in the screenshot.



Once the virtual environment is active, we can install all the project related packages and other dependencies isolated from the outside world. For example, if we are working with a data science project, we can install all the required packages and their dependencies at once simply, by typing below command like below −

(myenv) C:\Users\rajesh\python-virtual-environments>pip install numpy scipy matplotlib ipython jupyter pandas

Once we are done with our project, we can come out of the virtual environment simply by deactivating it.

(myenv) C:\Users\rajesh\python-virtual-environments>deactivate

C:\Users\rajesh\python-virtual-environments>

Now we are back in the windows command shell (like above).

# **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!")

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  
"""

# **Python Variables**

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

## 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)

ADVERTISEMENT

## 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

## 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)

# **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)

# **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 |
| None Type: | NoneType |

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) |
| x = None | NoneType | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_nonetype) |

# **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)

ADVERTISEMENT

## 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, unchangeable\*, 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.

# **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)

ADVERTISEMENT

## 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)

# **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)

ADVERTISEMENT

## 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)

# **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)

ADVERTISEMENT

## 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

# **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.

ADVERTISEMENT

## 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)

# **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)

ADVERTISEMENT

## 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.

## 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!")

# **Python OOPs Concepts**

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In [Python](https://www.javatpoint.com/python-tutorial), we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Major principles of object-oriented programming system are given below.

* Class
* Object
* Method
* Inheritance
* Polymorphism
* Data Abstraction
* Encapsulation

## Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

51.9M

836

Hello Java Program for Beginners

**Syntax**

1. **class** ClassName:
2. <statement-1>
3. .
4. .
5. <statement-N>

## Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute \_\_doc\_\_, which returns the docstring defined in the function source code.

When we define a class, it needs to create an object to allocate the memory. Consider the following example.

**Example:**

1. **class** car:
2. **def** \_\_init\_\_(self,modelname, year):
3. self.modelname = modelname
4. self.year = year
5. **def** display(self):
6. **print**(self.modelname,self.year)
8. c1 = car("Toyota", 2016)
9. c1.display()

**Output:**

Toyota 2016

In the above example, we have created the class named car, and it has two attributes modelname and year. We have created a c1 object to access the class attribute. The c1 object will allocate memory for these values. We will learn more about class and object in the next tutorial.

## Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

## Inheritance

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

It provides the re-usability of the code.

## Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

## Encapsulation

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

## Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation.

Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

# **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.

ADVERTISEMENT

## 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)

# Python SQLite3 tutorial (Database programming)

[Ayesha Tariq](https://likegeeks.com/author/ayesha/) Published: January 24, 2019 Last updated: April 23, 2022

In this tutorial, we will work with the SQLite3 database programmatically using Python.

SQLite in general is a server-less database that you can use within almost all programming languages including Python. Server-less means there is no need to install a separate server to work with SQLite so you can connect directly with the database.

SQLite is a lightweight database that can provide a relational database management system with zero-configuration because there is no need to configure or set up anything to use it.

We will use SQLite version 3 or SQLite3, so let’s get started.

Table of Contents

* [Create Connection](https://likegeeks.com/python-sqlite3-tutorial/#Create_Connection)
* [SQLite3 Cursor](https://likegeeks.com/python-sqlite3-tutorial/#SQLite3_Cursor)
* [Create Database](https://likegeeks.com/python-sqlite3-tutorial/#Create_Database)
* [Create Table](https://likegeeks.com/python-sqlite3-tutorial/#Create_Table)
* [Insert in Table](https://likegeeks.com/python-sqlite3-tutorial/#Insert_in_Table)
* [Update Table](https://likegeeks.com/python-sqlite3-tutorial/#Update_Table)
* [Select statement](https://likegeeks.com/python-sqlite3-tutorial/#Select_statement)
* [Fetch all data](https://likegeeks.com/python-sqlite3-tutorial/#Fetch_all_data)
* [SQLite3 rowcount](https://likegeeks.com/python-sqlite3-tutorial/#SQLite3_rowcount)
* [List tables](https://likegeeks.com/python-sqlite3-tutorial/#List_tables)
* [Check if a table exists or not](https://likegeeks.com/python-sqlite3-tutorial/#Check_if_a_table_exists_or_not)
* [Drop table](https://likegeeks.com/python-sqlite3-tutorial/#Drop_table)
* [SQLite3 exceptions](https://likegeeks.com/python-sqlite3-tutorial/#SQLite3_exceptions)
  + [DatabaseError](https://likegeeks.com/python-sqlite3-tutorial/#DatabaseError)
  + [IntegrityError](https://likegeeks.com/python-sqlite3-tutorial/#IntegrityError)
  + [ProgrammingError](https://likegeeks.com/python-sqlite3-tutorial/#ProgrammingError)
  + [OperationalError](https://likegeeks.com/python-sqlite3-tutorial/#OperationalError)
  + [NotSupportedError](https://likegeeks.com/python-sqlite3-tutorial/#NotSupportedError)
* [SQLite3 Executemany (Bulk insert)](https://likegeeks.com/python-sqlite3-tutorial/#SQLite3_Executemany_Bulk_insert)
* [Close Connection](https://likegeeks.com/python-sqlite3-tutorial/#Close_Connection)
* [SQLite3 datetime](https://likegeeks.com/python-sqlite3-tutorial/#SQLite3_datetime)

## Create Connection

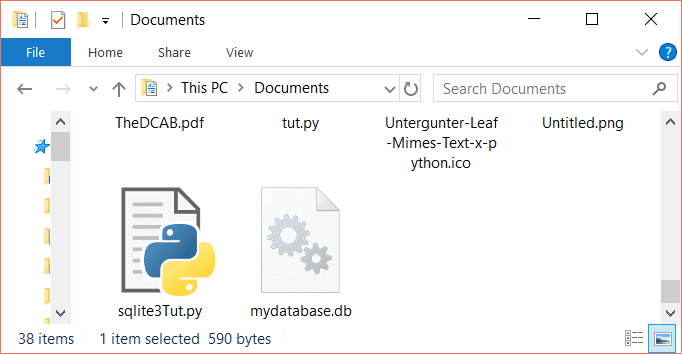
To use SQLite3 in Python, first of all, you will have to import the sqlite3 module and then create a connection object which will connect us to the database and will let us execute the SQL statements.

You can a connection object using the connect() function:

import sqlite3

con = sqlite3.connect('mydatabase.db')

That will create a new file with the name ‘mydatabase.db’.



## SQLite3 Cursor

To execute SQLite statements in Python, you need a cursor object. You can create it using the cursor() method.

The SQLite3 cursor is a method of the connection object. To execute the SQLite3 statements, you should establish a connection at first and then create an object of the cursor using the connection object as follows:

con = sqlite3.connect('mydatabase.db')

cursorObj = con.cursor()

Now we can use the cursor object to call the execute() method to execute any SQL queries.

## Create Database

When you create a connection with SQLite, that will create a database file automatically if it doesn’t already exist. This database file is created on disk; we can also create a database in RAM by using :memory: with the connect function. This database is called in-memory database.

Consider the code below in which we have created a database with a try, except and finally blocks to handle any exceptions:

import sqlite3

from sqlite3 import Error

def sql\_connection():

try:

con = sqlite3.connect(':memory:')

print("Connection is established: Database is created in memory")

except Error:

print(Error)

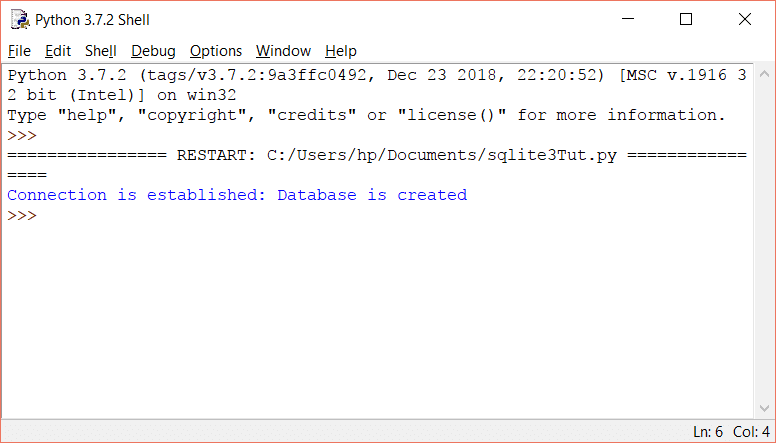
finally:

con.close()

sql\_connection()

First, we import the sqlite3 module, then we define a function sql\_connection. Inside this function, we have a try block where the connect() function is returning a connection object after establishing the connection.

Then we have except block, which in case of any exceptions prints the error message. If there are no errors, the connection will be established and will display a message as follows.



After that, we have closed our connection in the finally block. Closing a connection is optional, but it is a good programming practice, so you free the memory from any unused resources.

## Create Table

To create a table in SQLite3, you can use the Create Table query in the execute() method. Consider the following steps:

1. Create a connection object.
2. From the connection object, create a cursor object.
3. Using the cursor object, call the execute method with create table query as the parameter.

Let’s create employees with the following attributes:

employees (id, name, salary, department, position, hireDate)

The code will be like this:

import sqlite3

from sqlite3 import Error

def sql\_connection():

try:

con = sqlite3.connect('mydatabase.db')

return con

except Error:

print(Error)

def sql\_table(con):

cursorObj = con.cursor()

cursorObj.execute("CREATE TABLE employees(id integer PRIMARY KEY, name text, salary real, department text, position text, hireDate text)")

con.commit()

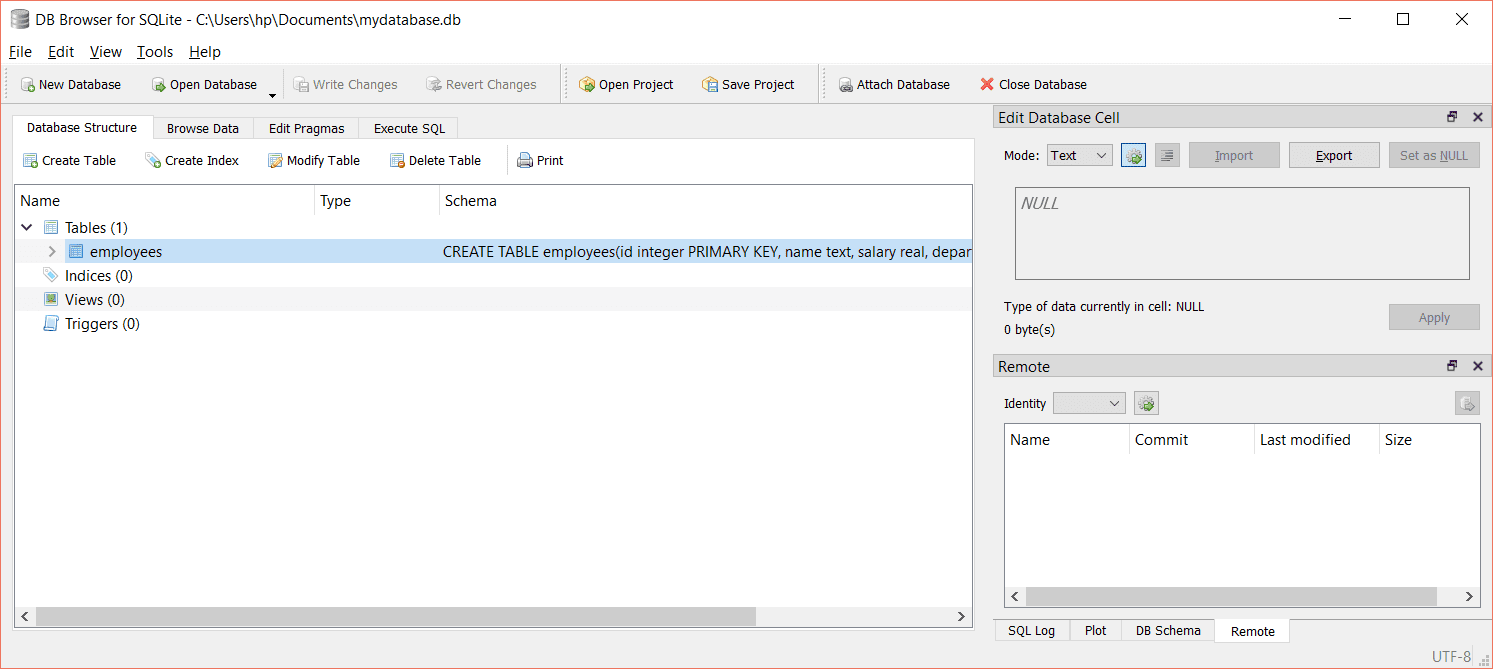
con = sql\_connection()

sql\_table(con)

In the above code, we have defined two methods, the first one establishes a connection and the second method creates a cursor object to execute the create table statement.

The commit() method saves all the changes we make. In the end, both methods are called.

To check if our table is created, you can use the [DB browser for SQLite](https://github.com/sqlitebrowser/sqlitebrowser/releases/tag/v3.11.0-beta3) to view your table. Open your mydatabase.db file with the program, and you should see your table:



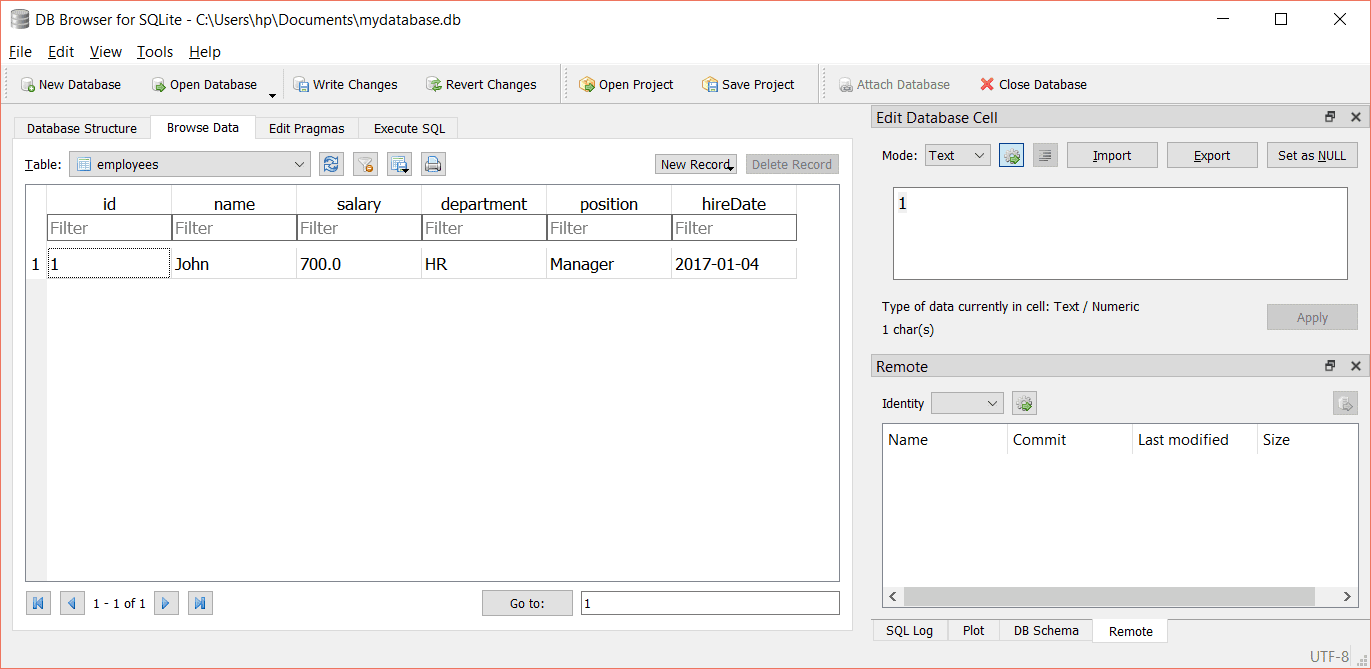
## Insert in Table

To insert data in a table, we use the INSERT INTO statement. Consider the following line of code:

cursorObj.execute("INSERT INTO employees VALUES(1, 'John', 700, 'HR', 'Manager', '2017-01-04')")

con.commit()

To check if the data is inserted, click on Browse Data in the DB Browser:



We can also pass values/arguments to an INSERT statement in the execute() method. You can use the question mark (?) as a placeholder for each value. The syntax of the INSERT will be like the following:

cursorObj.execute('''INSERT INTO employees(id, name, salary, department, position, hireDate) VALUES(?, ?, ?, ?, ?, ?)''', entities)

Where entities contain the values for the placeholders as follows:

entities = (2, 'Andrew', 800, 'IT', 'Tech', '2018-02-06')

The entire code is as follows:

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql\_insert(con, entities):

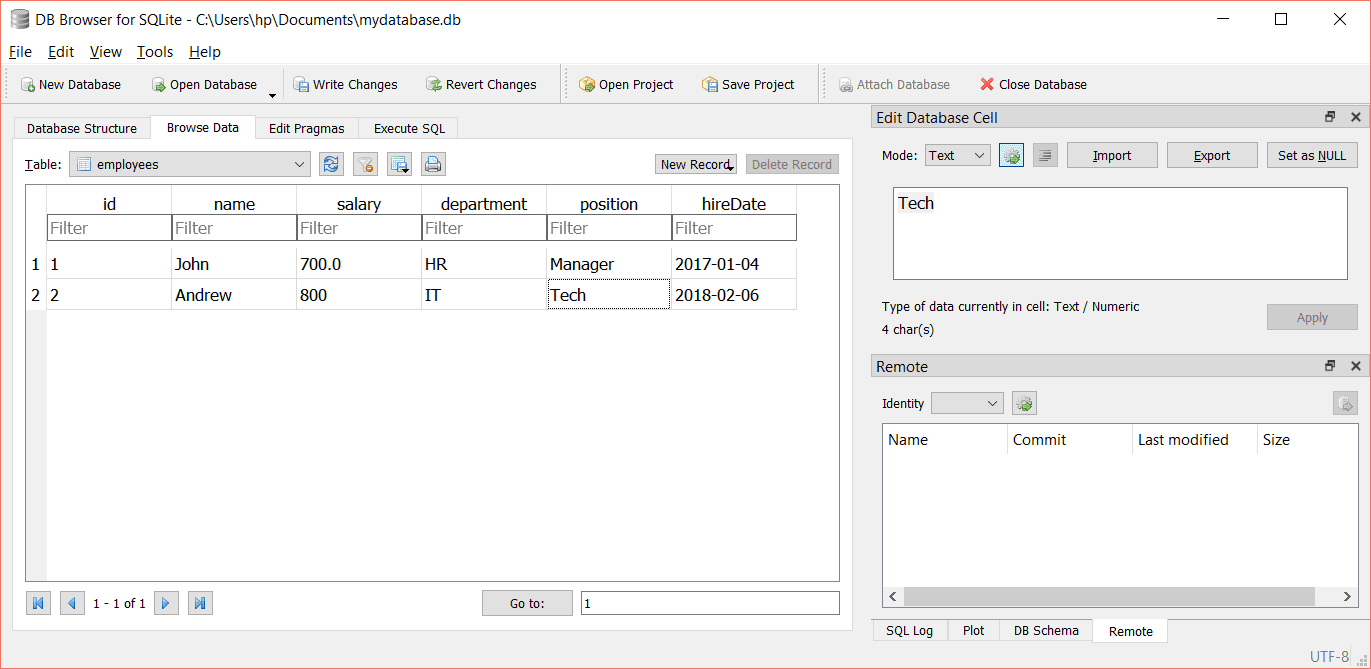
cursorObj = con.cursor()

cursorObj.execute('INSERT INTO employees(id, name, salary, department, position, hireDate) VALUES(?, ?, ?, ?, ?, ?)', entities)

con.commit()

entities = (2, 'Andrew', 800, 'IT', 'Tech', '2018-02-06')

sql\_insert(con, entities)



## Update Table

To update the table, simply create a connection, then create a cursor object using the connection and finally use the UPDATE statement in the execute() method.

Suppose that we want to update the name of the employee whose id equals 2. For updating, we will use the UPDATE statement and for the employee whose id equals 2. We will use the WHERE clause as a condition to select this employee.

Consider the following code:

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql\_update(con):

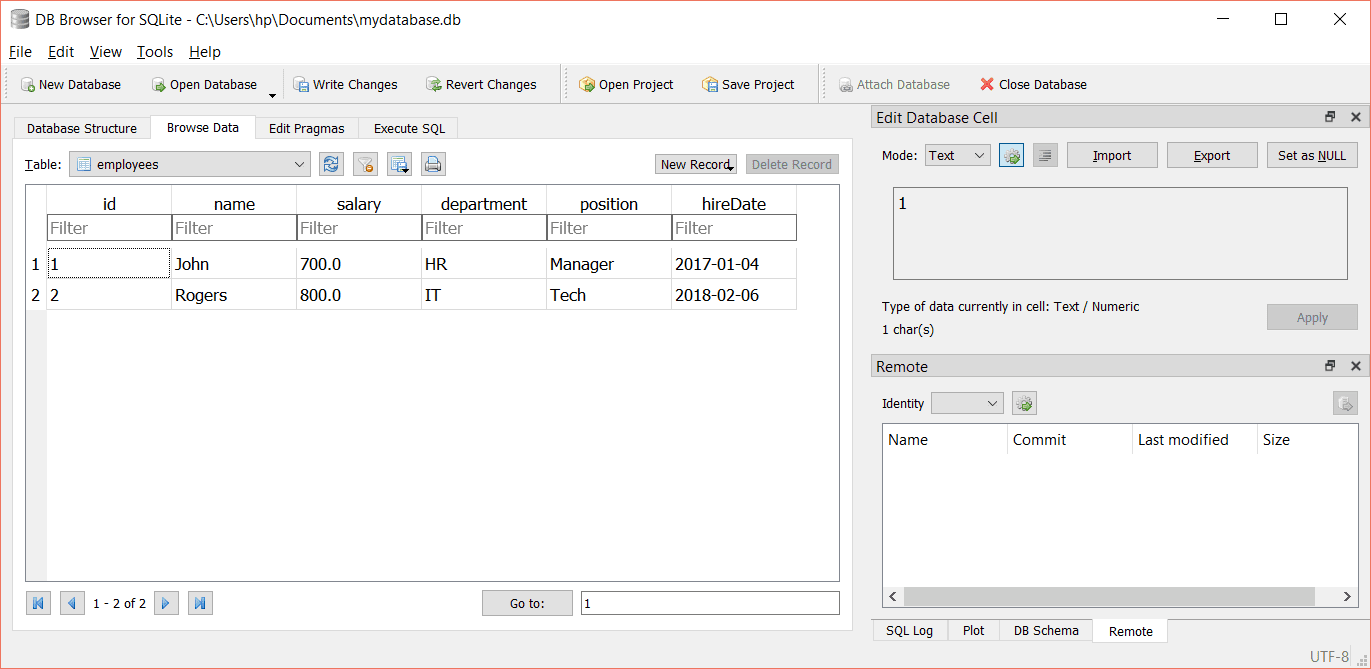
cursorObj = con.cursor()

cursorObj.execute('UPDATE employees SET name = "Rogers" where id = 2')

con.commit()

sql\_update(con)

This will change the name from Andrew to Rogers as follows:



## Select statement

You can use the select statement to select data from a particular table. If you want to select all the columns of the data from a table, you can use the asterisk (\*). The syntax for this will be as follows:

select \* from table\_name

In SQLite3, the SELECT statement is executed in the execute method of the cursor object. For example, select all the columns of the employees’ table, run the following code:

cursorObj.execute('SELECT \* FROM employees ')

If you want to select a few columns from a table, then specify the columns like the following:

select column1, column2 from tables\_name

For example,

cursorObj.execute('SELECT id, name FROM employees')

The select statement selects the required data from the database table, and if you want to fetch the selected data, the fetchall() method of the cursor object is used. We will demonstrate this in the next section.

## Fetch all data

To fetch the data from a database, we will execute the SELECT statement and then will use the fetchall() method of the cursor object to store the values into a variable. After that, we will loop through the variable and [print](https://likegeeks.com/python-print/) all values.

The code will be like this:

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql\_fetch(con):

cursorObj = con.cursor()

cursorObj.execute('SELECT \* FROM employees')

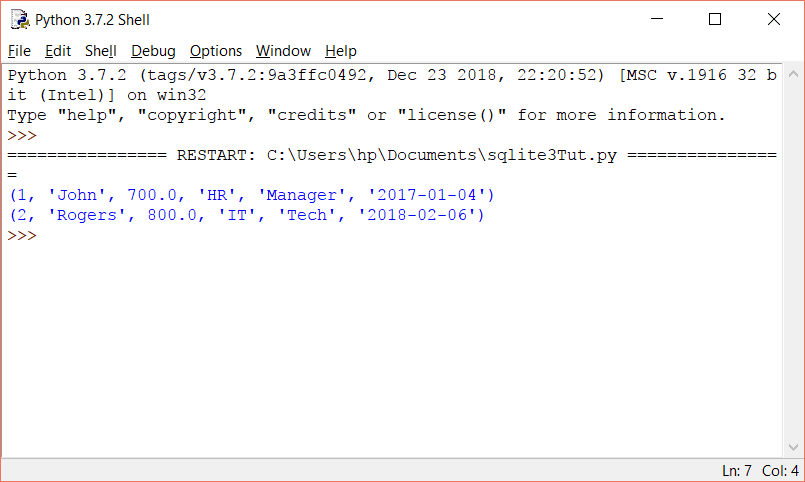
rows = cursorObj.fetchall()

for row in rows:

print(row)

sql\_fetch(con)

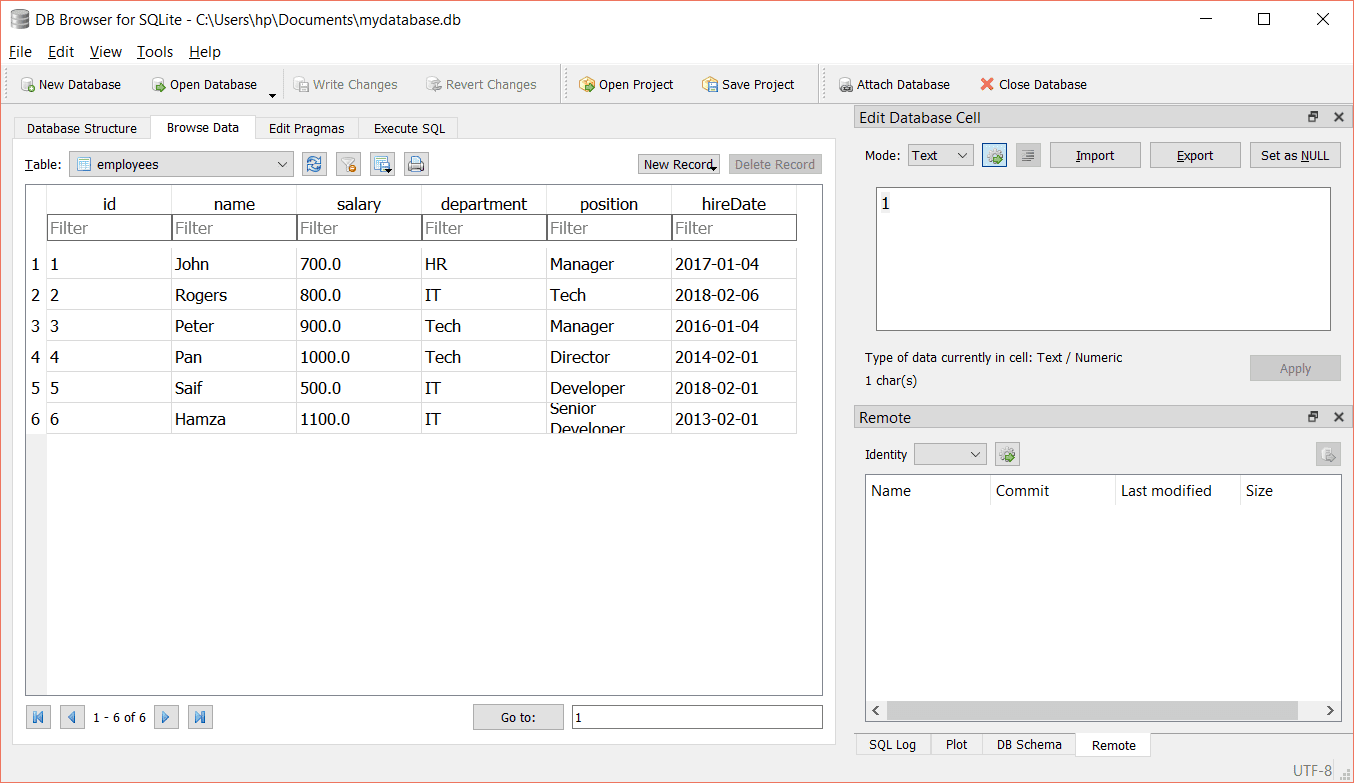
The above code will print out the records in our database as follows:



You can also use the fetchall() in one line as follows:

[print(row) for row in cursorObj.fetchall()]

If you want to fetch specific data from the database, you can use the WHERE clause. For example, we want to fetch the ids and names of those employees whose salary is greater than 800. For this, let’s populate our table with more rows, then execute our query.



You can use the insert statement to populate the data, or you can enter them manually in the DB browser program.

Now, to fetch id and names of those who have a salary greater than 800:

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql\_fetch(con):

cursorObj = con.cursor()

cursorObj.execute('SELECT id, name FROM employees WHERE salary > 800.0')

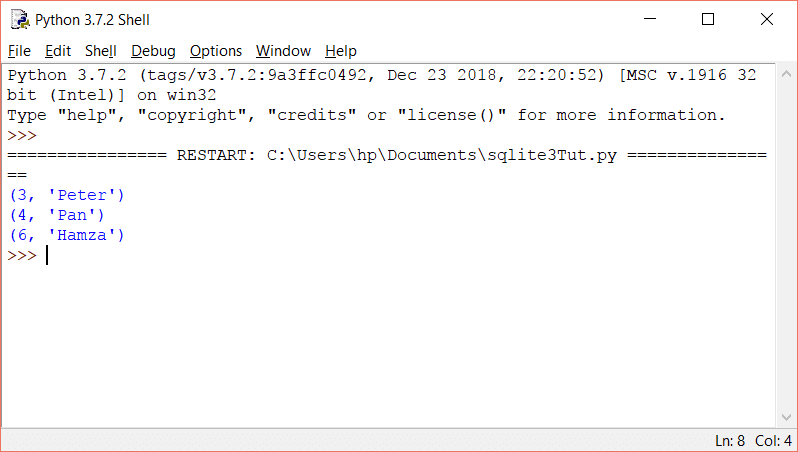
rows = cursorObj.fetchall()

for row in rows:

print(row)

sql\_fetch(con)

In the above SELECT statement, instead of using the asterisk (\*), we specified the id and name attributes. The result will be like the following:

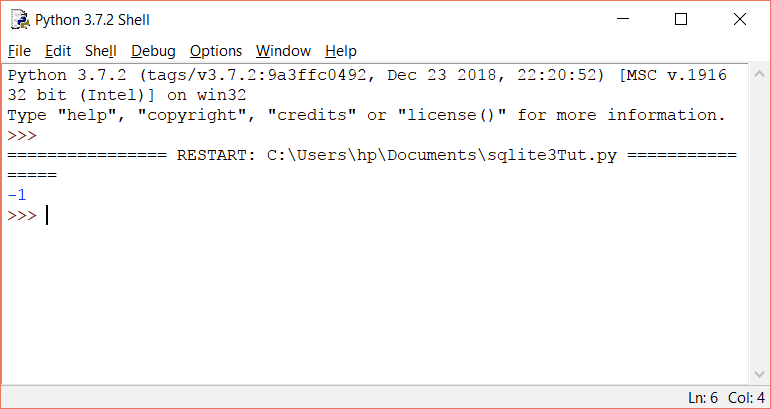


## SQLite3 rowcount

The SQLite3 rowcount is used to return the number of rows that are affected or selected by the latest executed SQL query.

When we use rowcount with the SELECT statement, -1 will be returned as how many rows are selected is unknown until they are all fetched. Consider the example below:

print(cursorObj.execute('SELECT \* FROM employees').rowcount)



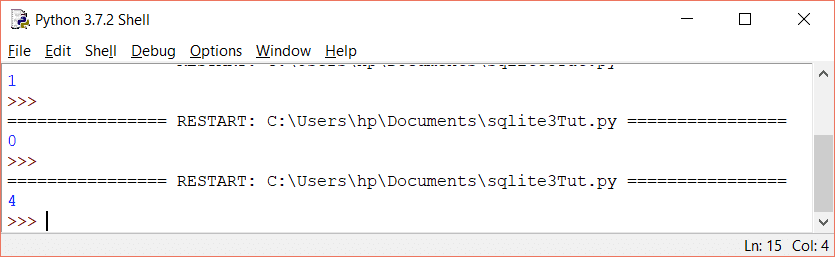
Therefore, to get the row count, you need to fetch all the data, and then get the length of the result:

rows = cursorObj.fetchall()

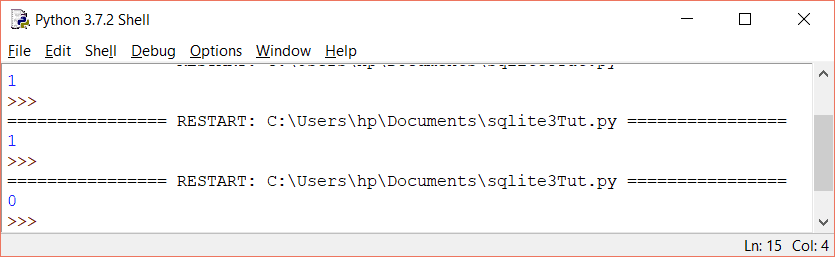
print len (rows)

When you use the DELETE statement without any condition (a where clause), that will delete all the rows in the table, and it will return the total number of deleted rows in rowcount.

print(cursorObj.execute('DELETE FROM employees').rowcount)



If no row is deleted, it will return zero.



## List tables

To list all tables in an SQLite3 database, you should query the sqlite\_master table and then use the fetchall() to fetch the results from the SELECT statement.

The sqlite\_master is the master table in SQLite3, which stores all tables.

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql\_fetch(con):

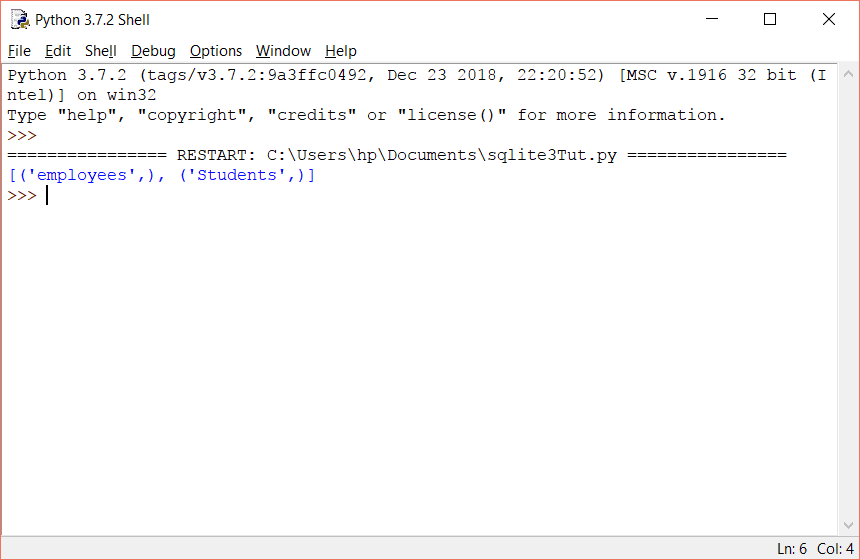
cursorObj = con.cursor()

cursorObj.execute('SELECT name from sqlite\_master where type= "table"')

print(cursorObj.fetchall())

sql\_fetch(con)

This will list all the tables as follows:



## Check if a table exists or not

When creating a table, we should make sure that the table is not already existed. Similarly, when removing/ deleting a table, the table should exist.

To check if the table doesn’t already exist, we use “if not exists” with the CREATE TABLE statement as follows:

create table if not exists table\_name (column1, column2, …, columnN)

For example:

import sqlite3

con = sqlite3.connect('mydatabase.db')

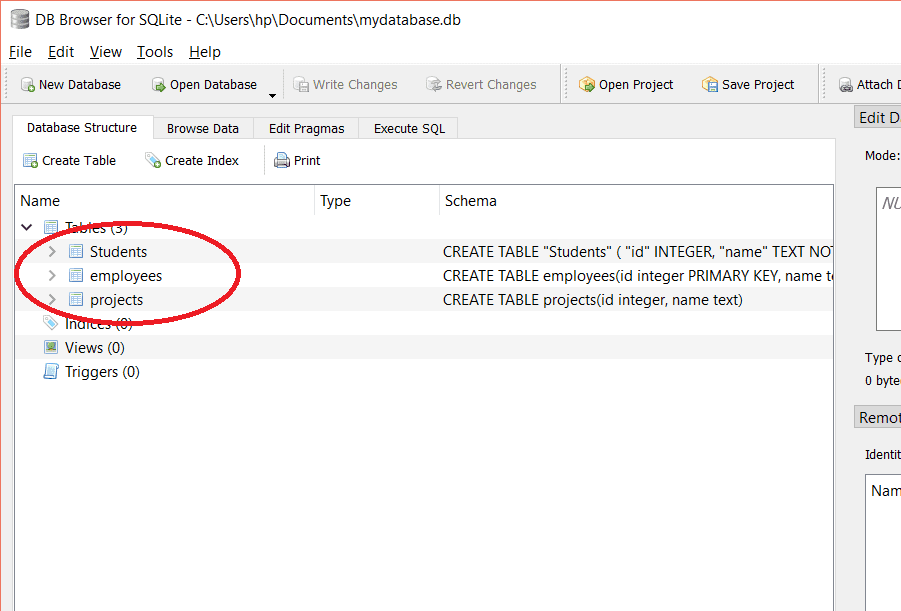
def sql\_fetch(con):

cursorObj = con.cursor()

cursorObj.execute('create table if not exists projects(id integer, name text)')

con.commit()

sql\_fetch(con)

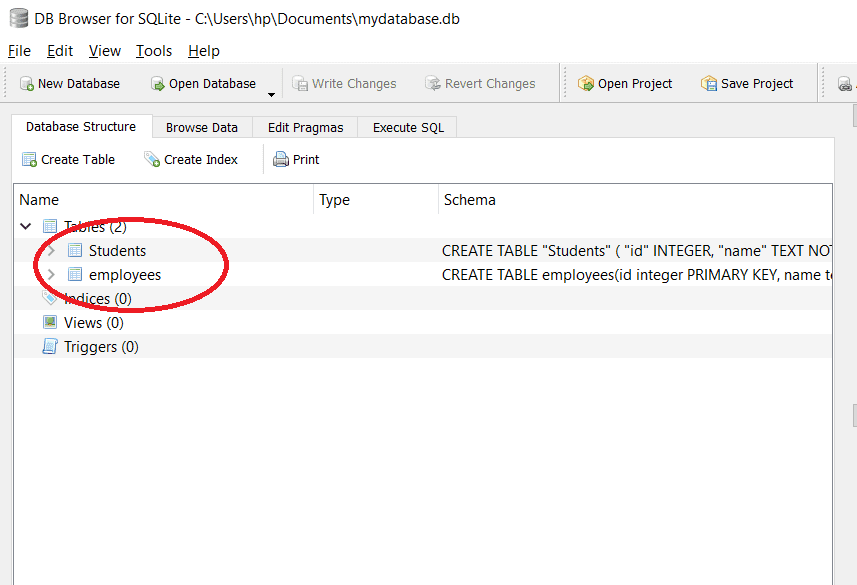


Similarly, to check if the table exists when deleting, we use “if exists” with the DROP TABLE statement as follows:

drop table if exists table\_name

For example,

cursorObj.execute('drop table if exists projects')

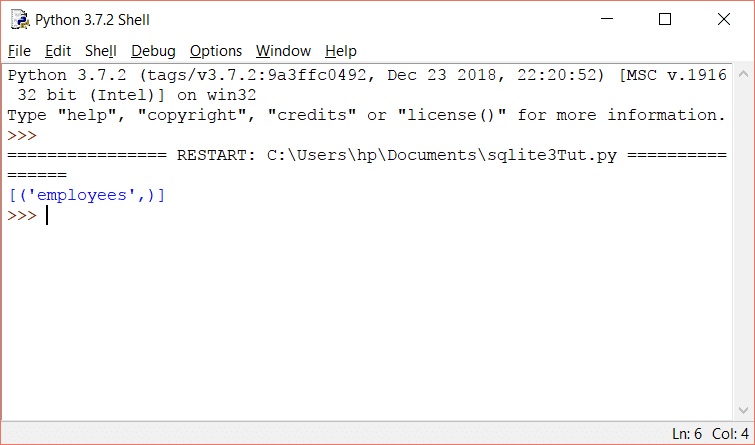


We can also check if the table we want to access exists or not by executing the following query:

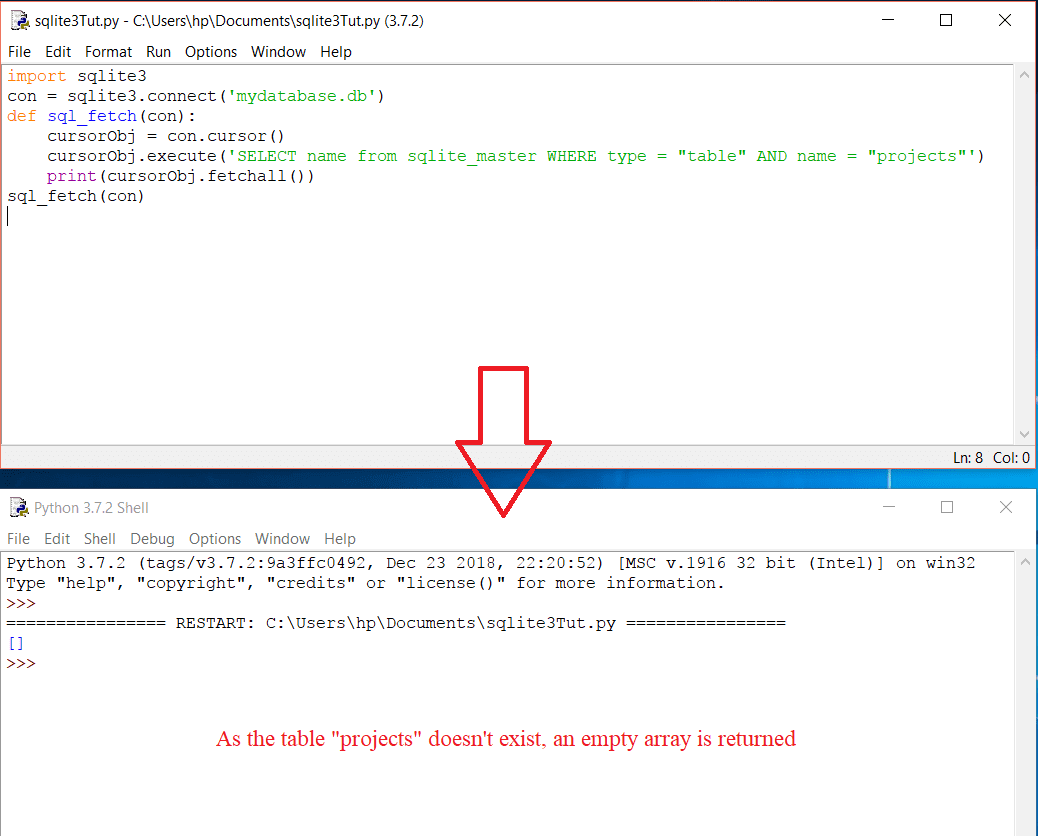
cursorObj.execute('SELECT name from sqlite\_master WHERE type = "table" AND name = "employees"')

print(cursorObj.fetchall())

If the employees’ table exists, it will return its name as follows:



If the table name we specified doesn’t exist, an empty array will be returned:



## Drop table

You can drop/delete a table using the DROP statement. The syntax of the DROP statement is as follows:

drop table table\_name

To drop a table, the table should exist in the database. Therefore, it is recommended to use “if exists” with the drop statement as follows:

drop table if exists table\_name

For example,

import sqlite3

con = sqlite3.connect('mydatabase.db')

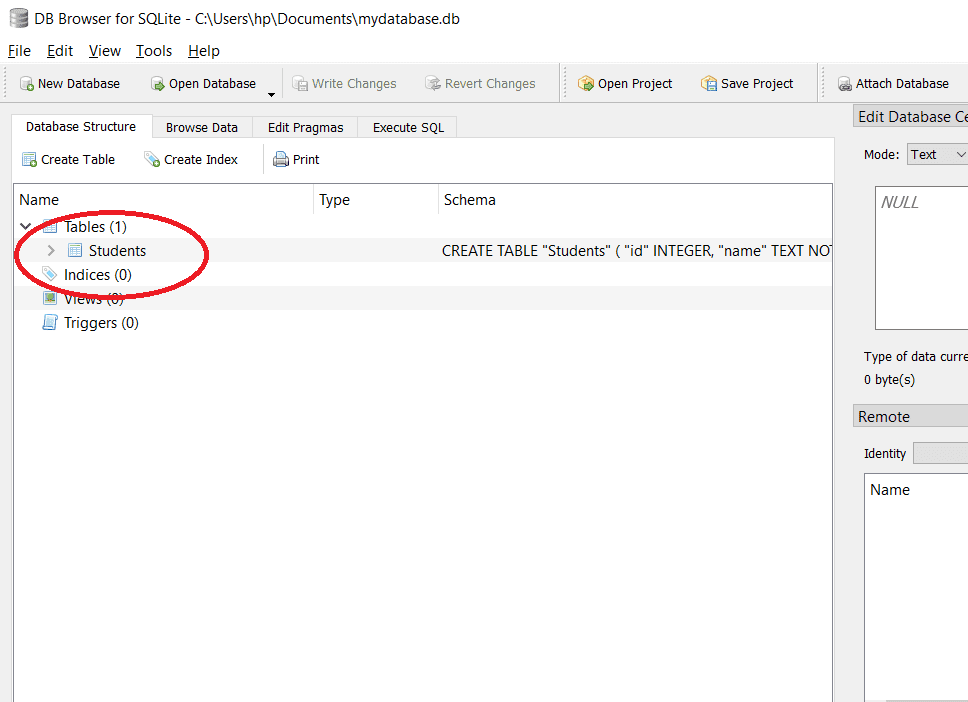
def sql\_fetch(con):

cursorObj = con.cursor()

cursorObj.execute('DROP table if exists employees')

con.commit()

sql\_fetch(con)



## SQLite3 exceptions

Exceptions are the run time errors. In [Python programming](https://likegeeks.com/python-programming-basics/), all exceptions are the instances of the class derived from the BaseException.

In SQLite3, we have the following main Python exceptions:

### DatabaseError

Any error related to the database raises the DatabaseError.

### IntegrityError

IntegrityError is a subclass of DatabaseError and is raised when there is a data integrity issue. For example, foreign data isn’t updated in all tables resulting in the inconsistency of the data.

### ProgrammingError

The exception ProgrammingError raises when there are syntax errors or table is not found or function is called with the wrong number of parameters/ arguments.

### OperationalError

This exception is raised when the database operations are failed, for example, unusual disconnection. This is not the fault of the programmers.

### NotSupportedError

When you use some methods that aren’t defined or supported by the database, that will raise the NotSupportedError exception.

## SQLite3 Executemany (Bulk insert)

You can use the executemany statement to insert multiple rows at once.

Consider the following code:

import sqlite3

con = sqlite3.connect('mydatabase.db')

cursorObj = con.cursor()

cursorObj.execute('create table if not exists projects(id integer, name text)')

data = [(1, "Ridesharing"), (2, "Water Purifying"), (3, "Forensics"), (4, "Botany")]

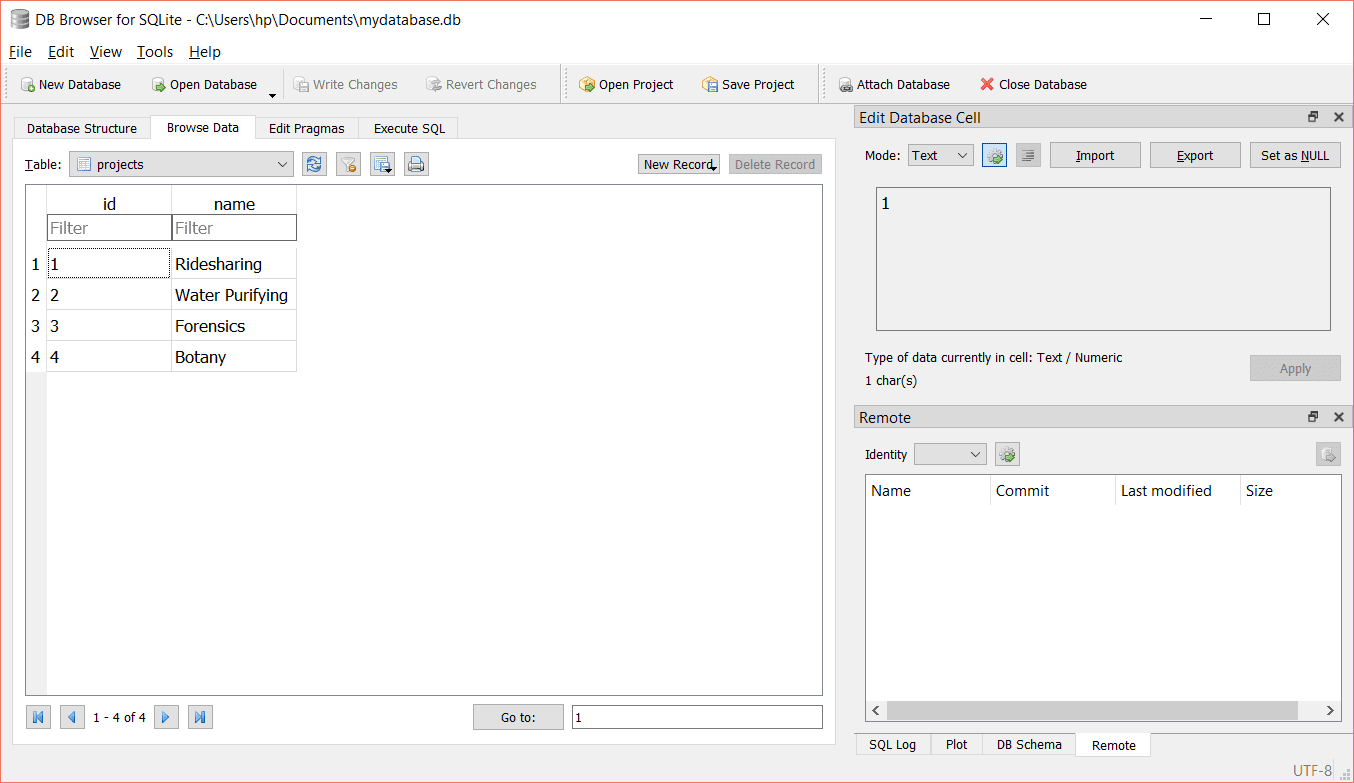
cursorObj.executemany("INSERT INTO projects VALUES(?, ?)", data)

con.commit()

Here we created a table with two columns, and “data” has four values for each column. We pass the variable to the executemany() method along with the query.

Note that we have used the placeholder to pass the values.

The above code will generate the following result:



## Close Connection

Once you are done with your database, it is a good practice to close the connection. You can close the connection by using the close() method.

To close a connection, use the connection object and call the close() method as follows:

con = sqlite3.connect('mydatabase.db')

#program statements

con.close()

## SQLite3 datetime

In the Python SQLite3 database, we can easily store date or time by importing the datatime module. The following formats are the most common formats you can use for datetime:

YYYY-MM-DD

YYYY-MM-DD HH:MM

YYYY-MM-DD HH:MM:SS

YYYY-MM-DD HH:MM:SS.SSS

HH:MM

HH:MM:SS

HH:MM:SS.SSS

now

Consider the following code:

import sqlite3

import datetime

con = sqlite3.connect('mydatabase.db')

cursorObj = con.cursor()

cursorObj.execute('create table if not exists assignments(id integer, name text, date date)')

data = [(1, "Ridesharing", datetime.date(2017, 1, 2)), (2, "Water Purifying", datetime.date(2018, 3, 4))]

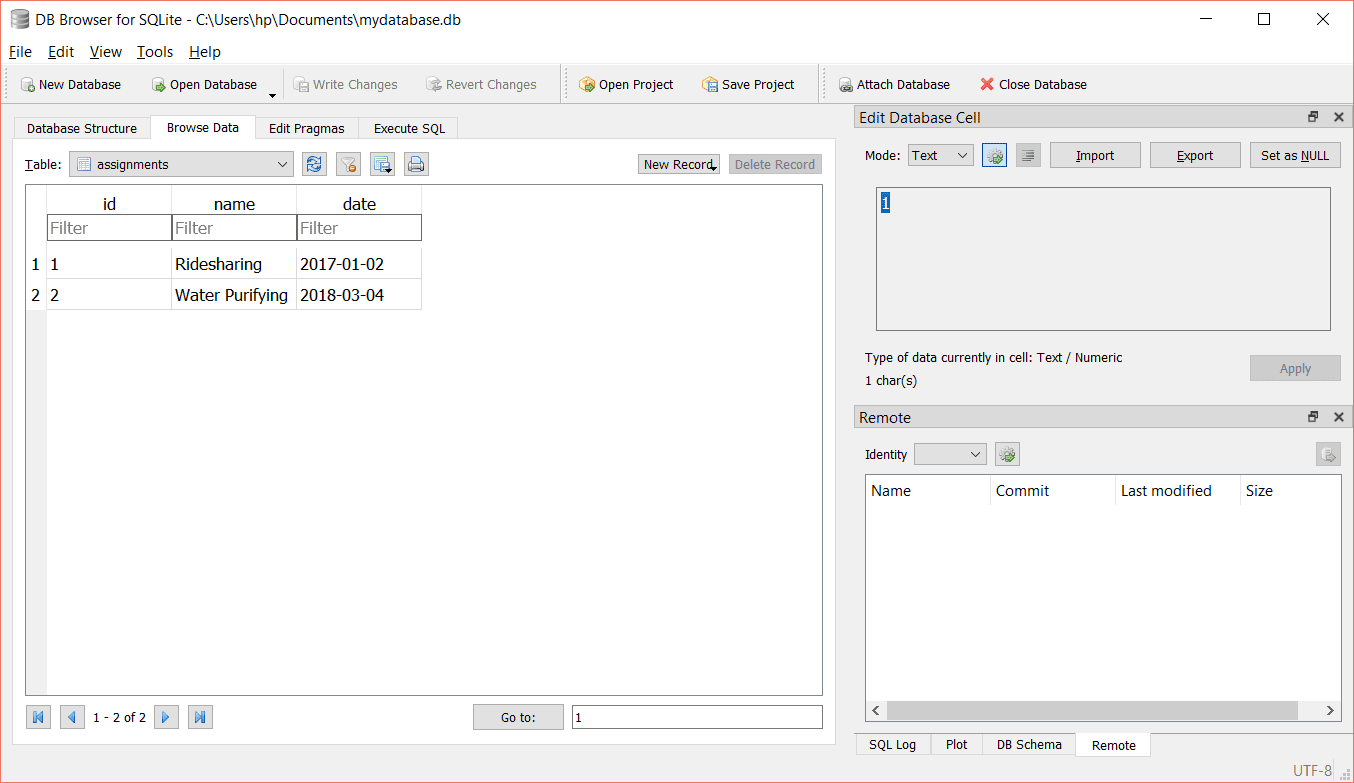
cursorObj.executemany("INSERT INTO assignments VALUES(?, ?, ?)", data)

con.commit()

In this code, we imported the datetime module first, and we have created a table named assignments with three columns.

The data type of the third column is a date. To insert the date in the column, we have used datetime.date. Similarly, we can use datetime.time to handle time.

The above code will generate the following output:



The great flexibility and mobility of the SQLite3 database make it the first choice for any developer to use it and ship it with any product he works with.

You can use SQLite3 databases in Windows, Linux, Mac OS, Android, and iOS projects due to their awesome portability. So you ship one file with your project and that’s it.

**Tikenter:-**

Tkinter is the Python interface to Tk, which is the GUI toolkit for Tcl/Tk.

Tcl (pronounced as tickle) is a scripting language often used in testing, prototyping, and GUI development. Tk is an open-source, cross-platform widget toolkit used by many different programming languages to build GUI programs.

Python implements the Tkinter as a module. Tkinter is a wrapper of C extensions that use Tcl/Tk libraries.

Tkinter allows you to develop desktop applications. It’s a very good tool for GUI programming in Python.

Tkinter is a good choice because of the following reasons:

* Easy to learn.
* Use very little code to make a functional desktop application.
* Layered design.
* Portable across all operating systems including Windows, macOS, and Linux.
* Pre-installed with the standard Python library.

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

## Tkinter widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

|  |  |  |
| --- | --- | --- |
| **SN** | **Widget** | **Description** |
| 1 | [Button](https://www.javatpoint.com/python-tkinter-button) | The Button is used to add various kinds of buttons to the python application. |
| 2 | [Canvas](https://www.javatpoint.com/python-tkinter-canvas) | The canvas widget is used to draw the canvas on the window. |
| 3 | [Checkbutton](https://www.javatpoint.com/python-tkinter-checkbutton) | The Checkbutton is used to display the CheckButton on the window. |
| 4 | [Entry](https://www.javatpoint.com/python-tkinter-entry) | The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values. |
| 5 | [Frame](https://www.javatpoint.com/python-tkinter-frame) | It can be defined as a container to which, another widget can be added and organized. |
| 6 | [Label](https://www.javatpoint.com/python-tkinter-label) | A label is a text used to display some message or information about the other widgets. |
| 7 | [ListBox](https://www.javatpoint.com/python-tkinter-listbox) | The ListBox widget is used to display a list of options to the user. |
| 8 | [Menubutton](https://www.javatpoint.com/python-tkinter-menubutton) | The Menubutton is used to display the menu items to the user. |
| 9 | [Menu](https://www.javatpoint.com/python-tkinter-menu) | It is used to add menu items to the user. |
| 10 | [Message](https://www.javatpoint.com/python-tkinter-message) | The Message widget is used to display the message-box to the user. |
| 11 | [Radiobutton](https://www.javatpoint.com/python-tkinter-radiobutton) | The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them. |
| 12 | [Scale](https://www.javatpoint.com/python-tkinter-scale) | It is used to provide the slider to the user. |
| 13 | [Scrollbar](https://www.javatpoint.com/python-tkinter-scrollbar) | It provides the scrollbar to the user so that the user can scroll the window up and down. |
| 14 | [Text](https://www.javatpoint.com/python-tkinter-text) | It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it. |
| 14 | [Toplevel](https://www.javatpoint.com/python-tkinter-toplevel) | It is used to create a separate window container. |
| 15 | [Spinbox](https://www.javatpoint.com/python-tkinter-spinbox) | It is an entry widget used to select from options of values. |
| 16 | [PanedWindow](https://www.javatpoint.com/python-tkinter-panedwindow) | It is like a container widget that contains horizontal or vertical panes. |
| 17 | [LabelFrame](https://www.javatpoint.com/python-tkinter-labelframe) | A LabelFrame is a container widget that acts as the container |
| 18 | [MessageBox](https://www.javatpoint.com/python-tkinter-messagebox) | This module is used to display the message-box in the desktop based applications. |

## Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method
2. The grid() method
3. The place() method

Let's discuss each one of them in detail.

### **Python Tkinter pack() method**

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

However, the controls are less and widgets are generally added in the less organized manner.

The syntax to use the pack() is given below.

### **syntax**

1. widget.pack(options)

A list of possible options that can be passed in pack() is given below.

* **expand:** If the expand is set to true, the widget expands to fill any space.
* **Fill:** By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.
* **size:** it represents the side of the parent to which the widget is to be placed on the window.

# **Python Tkinter Button**

The button widget is used to add various types of buttons to the python application. Python allows us to configure the look of the button according to our requirements. Various options can be set or reset depending upon the requirements.

We can also associate a method or function with a button which is called when the button is pressed.

The syntax to use the button widget is given below.

### **Syntax**

1. W = Button(parent, options)

# **Python Tkinter Entry**

The Entry widget is used to provde the single line text-box to the user to accept a value from the user. We can use the Entry widget to accept the text strings from the user. It can only be used for one line of text from the user. For multiple lines of text, we must use the text widget.

The syntax to use the Entry widget is given below.

### **Syntax**

1. w = Entry (parent, options)

# **Python Tkinter Frame**

Python Tkinter Frame widget is used to organize the group of widgets. It acts like a container which can be used to hold the other widgets. The rectangular areas of the screen are used to organize the widgets to the python application.

The syntax to use the Frame widget is given below.

### **Syntax**

1. w = Frame(parent,  options)

# **Python Tkinter Label**

The Label is used to specify the container box where we can place the text or images. This widget is used to provide the message to the user about other widgets used in the python application.

There are the various options which can be specified to configure the text or the part of the text shown in the Label.

The syntax to use the Label is given below.

### **Syntax**

1. w = Label (master, options)

# **Python Tkinter Scrollbar**

The scrollbar widget is used to scroll down the content of the other widgets like listbox, text, and canvas. However, we can also create the horizontal scrollbars to the Entry widget.

The syntax to use the Scrollbar widget is given below.

### **Syntax**

1. w = Scrollbar(top, options)

# **Tkinter messagebox**

The messagebox module is used to display the message boxes in the python applications. There are the various functions which are used to display the relevant messages depending upon the application requirements.

The syntax to use the messagebox is given below.

### **Syntax**

` messagebox.function\_name(title, message [, options])

# Python Pandas - Introduction

Advertisements

[**Previous Page**](https://www.tutorialspoint.com/python_pandas/index.htm)

[**Next Page**](https://www.tutorialspoint.com/python_pandas/python_pandas_environment_setup.htm)

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

## Key Features of Pandas

* Fast and efficient DataFrame object with default and customized indexing.
* Tools for loading data into in-memory data objects from different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of date sets.
* Label-based slicing, indexing and subsetting of large data sets.
* Columns from a data structure can be deleted or inserted.
* Group by data for aggregation and transformations.
* High performance merging and joining of data.
* Time Series functionality.

# **Write Pandas DataFrame to Excel Sheet?**

Contents

1. [Introduction](https://pythonexamples.org/pandas-write-dataframe-to-excel-sheet/#1)
2. [Example 1: Write DataFrame to Excel File](https://pythonexamples.org/pandas-write-dataframe-to-excel-sheet/#2)
3. [Example 2: Write DataFrame to a specific Excel Sheet](https://pythonexamples.org/pandas-write-dataframe-to-excel-sheet/#3)
4. [Summary](https://pythonexamples.org/pandas-write-dataframe-to-excel-sheet/#4)

## Pandas DataFrame to Excel

You can save or write a DataFrame to an Excel File or a specific Sheet in the Excel file using pandas.DataFrame.to\_excel() method of DataFrame class.

In this tutorial, we shall learn how to write a Pandas DataFrame to an Excel File, with the help of well detailed example Python programs.

**Prerequisite**

The prerequisite to work with Excel file functions in pandas is that, you have to install openpyxl module. To install openpyxl using pip, run the following pip command.

pip install openpyxl

### **Example 1: Write DataFrame to Excel File**

You can write the DataFrame to Excel File without mentioning any sheet name. The step by step process is given below:

1. Have your DataFrame ready. In this example we shall initialize a DataFrame with some rows and columns.
2. Create an Excel Writer with the name of the output excel file, to which you would like to write our DataFrame.
3. Call to\_excel() function on the DataFrame with the Excel Writer passed as argument.
4. Save the Excel file using save() method of Excel Writer.

**Python Program**

import pandas as pd

# create dataframe

df\_marks = pd.DataFrame({'name': ['Somu', 'Kiku', 'Amol', 'Lini'],

'physics': [68, 74, 77, 78],

'chemistry': [84, 56, 73, 69],

'algebra': [78, 88, 82, 87]})

# create excel writer object

writer = pd.ExcelWriter('output.xlsx')

# write dataframe to excel

df\_marks.to\_excel(writer)

# save the excel

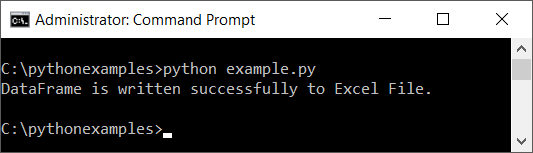
writer.save()

print('DataFrame is written successfully to Excel File.')

[Run](https://pythonexamples.org/run.php?pgm=import+pandas+as+pd%0A%0A%23+create+dataframe%0Adf_marks+%3D+pd.DataFrame%28%7B%27name%27%3A+%5B%27Somu%27%2C+%27Kiku%27%2C+%27Amol%27%2C+%27Lini%27%5D%2C%0A+++++%27physics%27%3A+%5B68%2C+74%2C+77%2C+78%5D%2C%0A+++++%27chemistry%27%3A+%5B84%2C+56%2C+73%2C+69%5D%2C%0A+++++%27algebra%27%3A+%5B78%2C+88%2C+82%2C+87%5D%7D%29%0A%0A%23+create+excel+writer+object%0Awriter+%3D+pd.ExcelWriter%28%27output.xlsx%27%29%0A%23+write+dataframe+to+excel%0Adf_marks.to_excel%28writer%29%0A%23+save+the+excel%0Awriter.save%28%29%0Aprint%28%27DataFrame+is+written+successfully+to+Excel+File.%27%29)

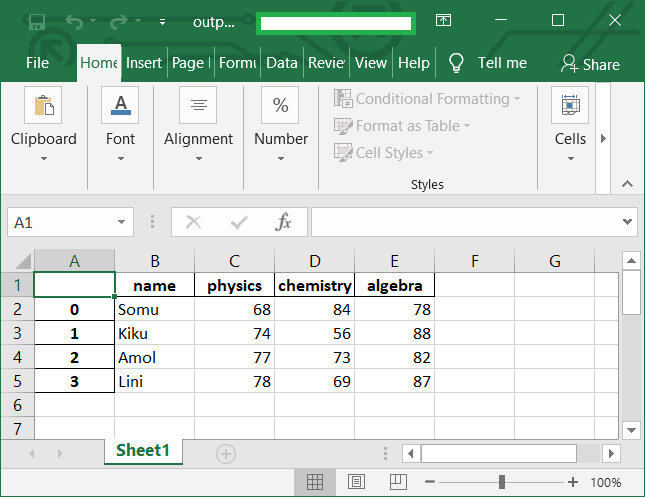
**Output**

Run the above program, and an excel file shall be created with the name specified while creating excel writer.



**Output Excel File**

Open the excel file, and you shall see the index, column labels and row data written to file.



### **Example 2: Write DataFrame to a specific Excel Sheet**

You can write the DataFrame to a specific Excel Sheet. The step by step process is:

1. Have your DataFrame ready.
2. Create an Excel Writer with the name of the desired output excel file.
3. Call to\_excel() function on the DataFrame with the writer and the name of the Excel Sheet passed as arguments.
4. Save the Excel file using save() method of Excel Writer.

**Python Program**

import pandas as pd

# create dataframe

df\_marks = pd.DataFrame({'name': ['Somu', 'Kiku', 'Amol', 'Lini'],

'physics': [68, 74, 77, 78],

'chemistry': [84, 56, 73, 69],

'algebra': [78, 88, 82, 87]})

# create excel writer

writer = pd.ExcelWriter('output.xlsx')

# write dataframe to excel sheet named 'marks'

df\_marks.to\_excel(writer, 'marks')

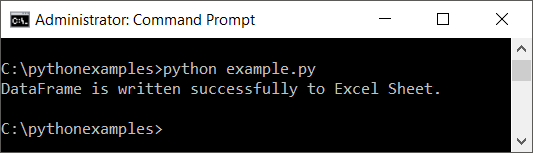
# save the excel file

writer.save()

print('DataFrame is written successfully to Excel Sheet.')

[Run](https://pythonexamples.org/run.php?pgm=import+pandas+as+pd%0A%0A%23+create+dataframe%0Adf_marks+%3D+pd.DataFrame%28%7B%27name%27%3A+%5B%27Somu%27%2C+%27Kiku%27%2C+%27Amol%27%2C+%27Lini%27%5D%2C%0A+++++%27physics%27%3A+%5B68%2C+74%2C+77%2C+78%5D%2C%0A+++++%27chemistry%27%3A+%5B84%2C+56%2C+73%2C+69%5D%2C%0A+++++%27algebra%27%3A+%5B78%2C+88%2C+82%2C+87%5D%7D%29%0A%0A%23+create+excel+writer%0Awriter+%3D+pd.ExcelWriter%28%27output.xlsx%27%29%0A%23+write+dataframe+to+excel+sheet+named+%27marks%27%0Adf_marks.to_excel%28writer%2C+%27marks%27%29%0A%23+save+the+excel+file%0Awriter.save%28%29%0Aprint%28%27DataFrame+is+written+successfully+to+Excel+Sheet.%27%29)

**Run Program**



**Output Excel File**

Open the excel file. Please note the name of the excel sheet. It is named to the string we specified as second argument to to\_excel() function.

