Python Iterators

[❮ Previous](https://www.w3schools.com/python/python_inheritance.asp)[Next ❯](https://www.w3schools.com/python/python_polymorphism.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[Get your own Python Server](https://www.w3schools.com/spaces/)

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.

ADVERTISEMENT

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 from going 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)

Python Polymorphism

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

The word "polymorphism" means "many forms", and in programming it refers to methods/functions/operators with the same name that can be executed on many objects or classes.

Function Polymorphism

An example of a Python function that can be used on different objects is the len() function.

String

For strings len() returns the number of characters:

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

x = "Hello World!"  
  
print(len(x))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_1)

Tuple

For tuples len() returns the number of items in the tuple:

Example

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_2)

Dictionary

For dictionaries len() returns the number of key/value pairs in the dictionary:

Example

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
  
print(len(thisdict))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_3)

ADVERTISEMENT

Class Polymorphism

Polymorphism is often used in Class methods, where we can have multiple classes with the same method name.

For example, say we have three classes: Car, Boat, and Plane, and they all have a method called move():

Example

Different classes with the same method:

class Car:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Drive!")  
  
class Boat:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Sail!")  
  
class Plane:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Fly!")  
  
car1 = Car("Ford", "Mustang")       #Create a Car class  
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat class  
plane1 = Plane("Boeing", "747")     #Create a Plane class  
  
for x in (car1, boat1, plane1):  
  x.move()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_4)

Look at the for loop at the end. Because of polymorphism we can execute the same method for all three classes.

Inheritance Class Polymorphism

What about classes with child classes with the same name? Can we use polymorphism there?

Yes. If we use the example above and make a parent class called Vehicle, and make Car, Boat, Plane child classes of Vehicle, the child classes inherits the Vehicle methods, but can override them:

Example

Create a class called Vehicle and make Car, Boat, Plane child classes of Vehicle:

class Vehicle:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Move!")  
  
class Car(Vehicle):  
  pass  
  
class Boat(Vehicle):  
  def move(self):  
    print("Sail!")  
  
class Plane(Vehicle):  
  def move(self):  
    print("Fly!")  
  
car1 = Car("Ford", "Mustang") #Create a Car object  
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat object  
plane1 = Plane("Boeing", "747") #Create a Plane object  
  
for x in (car1, boat1, plane1):  
  print(x.brand)  
  print(x.model)  
  x.move()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_5)

Child classes inherits the properties and methods from the parent class.

In the example above you can see that the Car class is empty, but it inherits brand, model, and move() from Vehicle.

The Boat and Plane classes also inherit brand, model, and move() from Vehicle, but they both override the move() method.

Because of polymorphism we can execute the same method for all classes.

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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:

2023-09-15 13:15:01.792721

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).

ADVERTISEMENT

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) |
| %C | Century | 20 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_century) |
| %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 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[Get your own Python Server](https://www.w3schools.com/spaces/)

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)

ADVERTISEMENT

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 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[Get your own Python Server](https://www.w3schools.com/spaces/)

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"**  
)

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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)

ADVERTISEMENT

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")

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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)

ADVERTISEMENT

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)

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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)

ADVERTISEMENT

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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)

ADVERTISEMENT

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)

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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!

ADVERTISEMENT

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")

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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 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)

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[Get your own Python Server](https://www.w3schools.com/spaces/)

Overwrite the address column from "Valley 345" to "Canyon 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!

ADVERTISEMENT

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 placeholder %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[Get your own Python Server](https://www.w3schools.com/spaces/)

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)