# Python MongoDB

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

Python can be used in database applications.

One of the most popular NoSQL database is MongoDB.

## MongoDB

MongoDB stores data in JSON-like documents, which makes the database very flexible and scalable.

To be able to experiment with the code examples in this tutorial, you will need access to a MongoDB database.

You can download a free MongoDB database at [https://www.mongodb.com](https://www.mongodb.com/).

Or get started right away with a MongoDB cloud service at <https://www.mongodb.com/cloud/atlas>.

## PyMongo

Python needs a MongoDB driver to access the MongoDB database.

In this tutorial we will use the MongoDB driver "PyMongo".

We recommend that you use PIP to install "PyMongo".

PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following:

Download and install "PyMongo":

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install pymongo

Now you have downloaded and installed a mongoDB driver.

## Test PyMongo

To test if the installation was successful, or if you already have "pymongo" installed, create a Python page with the following content:

demo\_mongodb\_test.py:

import pymongo

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

If the above code was executed with no errors, "pymongo" is installed and ready to be used.

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

# Python MongoDB Create Database

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

## Creating a Database

To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database you want to create.

MongoDB will create the database if it does not exist, and make a connection to it.

### Example

Create a database called "mydatabase":

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

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

Important: In MongoDB, a database is not created until it gets content!

MongoDB waits until you have created a collection (table), with at least one document (record) before it actually creates the database (and collection).

## Check if Database Exists

Remember: In MongoDB, a database is not created until it gets content, so if this is your first time creating a database, you should complete the next two chapters (create collection and create document) before you check if the database exists!

You can check if a database exist by listing all databases in you system:

### Example

Return a list of your system's databases:

print(myclient.list\_database\_names())

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

Or you can check a specific database by name:

### Example

Check if "mydatabase" exists:

dblist = myclient.list\_database\_names()

if "mydatabase" in dblist:

print("The database exists.")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mongodb_check_db2)

# Python MongoDB Create Collection

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

A collection in MongoDB is the same as a table in SQL databases.

## Creating a Collection

To create a collection in MongoDB, use database object and specify the name of the collection you want to create.

MongoDB will create the collection if it does not exist.

### Example

Create a collection called "customers":

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

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

Important: In MongoDB, a collection is not created until it gets content!

MongoDB waits until you have inserted a document before it actually creates the collection.

## Check if Collection Exists

Remember: In MongoDB, a collection is not created until it gets content, so if this is your first time creating a collection, you should complete the next chapter (create document) before you check if the collection exists!

You can check if a collection exist in a database by listing all collections:

### Example

Return a list of all collections in your database:

print(mydb.list\_collection\_names())

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

Or you can check a specific collection by name:

### Example

Check if the "customers" collection exists:

collist = mydb.list\_collection\_names()

if "customers" in collist:

print("The collection exists.")

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mongodb_check_collection2)

# Python MongoDB Insert Document

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

A document in MongoDB is the same as a record in SQL databases.

## Insert Into Collection

To insert a record, or *document* as it is called in MongoDB, into a collection, we use the insert\_one() method.

The first parameter of the insert\_one() method is a dictionary containing the name(s) and value(s) of each field in the document you want to insert.

### Example

Insert a record in the "customers" collection:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mydict = { "name": "John", "address": "Highway 37" }

x = mycol.insert\_one(mydict)

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

## Return the \_id Field

The insert\_one() method returns a InsertOneResult object, which has a property, inserted\_id, that holds the id of the inserted document.

### Example

Insert another record in the "customers" collection, and return the value of the \_id field:

mydict = { "name": "Peter", "address": "Lowstreet 27" }

x = mycol.insert\_one(mydict)

print(x.inserted\_id)

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

If you do not specify an \_id field, then MongoDB will add one for you and assign a unique id for each document.

In the example above no \_id field was specified, so MongoDB assigned a unique \_id for the record (document).

## Insert Multiple Documents

To insert multiple documents into a collection in MongoDB, we use the insert\_many() method.

The first parameter of the insert\_many() method is a list containing dictionaries with the data you want to insert:

### Example

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mylist = [

{ "name": "Amy", "address": "Apple st 652"},

{ "name": "Hannah", "address": "Mountain 21"},

{ "name": "Michael", "address": "Valley 345"},

{ "name": "Sandy", "address": "Ocean blvd 2"},

{ "name": "Betty", "address": "Green Grass 1"},

{ "name": "Richard", "address": "Sky st 331"},

{ "name": "Susan", "address": "One way 98"},

{ "name": "Vicky", "address": "Yellow Garden 2"},

{ "name": "Ben", "address": "Park Lane 38"},

{ "name": "William", "address": "Central st 954"},

{ "name": "Chuck", "address": "Main Road 989"},

{ "name": "Viola", "address": "Sideway 1633"}

]

x = mycol.insert\_many(mylist)

#print list of the \_id values of the inserted documents:

print(x.inserted\_ids)

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

The insert\_many() method returns a InsertManyResult object, which has a property, inserted\_ids, that holds the ids of the inserted documents.

## Insert Multiple Documents, with Specified IDs

If you do not want MongoDB to assign unique ids for you document, you can specify the \_id field when you insert the document(s).

Remember that the values has to be unique. Two documents cannot have the same \_id.

### Example

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mylist = [

{ "\_id": 1, "name": "John", "address": "Highway 37"},

{ "\_id": 2, "name": "Peter", "address": "Lowstreet 27"},

{ "\_id": 3, "name": "Amy", "address": "Apple st 652"},

{ "\_id": 4, "name": "Hannah", "address": "Mountain 21"},

{ "\_id": 5, "name": "Michael", "address": "Valley 345"},

{ "\_id": 6, "name": "Sandy", "address": "Ocean blvd 2"},

{ "\_id": 7, "name": "Betty", "address": "Green Grass 1"},

{ "\_id": 8, "name": "Richard", "address": "Sky st 331"},

{ "\_id": 9, "name": "Susan", "address": "One way 98"},

{ "\_id": 10, "name": "Vicky", "address": "Yellow Garden 2"},

{ "\_id": 11, "name": "Ben", "address": "Park Lane 38"},

{ "\_id": 12, "name": "William", "address": "Central st 954"},

{ "\_id": 13, "name": "Chuck", "address": "Main Road 989"},

{ "\_id": 14, "name": "Viola", "address": "Sideway 1633"}

]

x = mycol.insert\_many(mylist)

#print list of the \_id values of the inserted documents:

print(x.inserted\_ids)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mongodb_insert_many2)

# Python MongoDB Find

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

In MongoDB we use the find and findOne methods to find data in a collection.

Just like the SELECT statement is used to find data in a table in a MySQL database.

## Find One

To select data from a collection in MongoDB, we can use the find\_one() method.

The find\_one() method returns the first occurrence in the selection.

### Example

Find the first document in the customers collection:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

x = mycol.find\_one()

print(x)

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

## Find All

To select data from a table in MongoDB, we can also use the find() method.

The find() method returns all occurrences in the selection.

The first parameter of the find() method is a query object. In this example we use an empty query object, which selects all documents in the collection.

No parameters in the find() method gives you the same result as SELECT \* in MySQL.

### Example

Return all documents in the "customers" collection, and print each document:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

for x in mycol.find():

print(x)

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

## Return Only Some Fields

The second parameter of the find() method is an object describing which fields to include in the result.

This parameter is optional, and if omitted, all fields will be included in the result.

### Example

Return only the names and addresses, not the \_ids:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

for x in mycol.find({},{ "\_id": 0, "name": 1, "address": 1 }):

print(x)

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

You are not allowed to specify both 0 and 1 values in the same object (except if one of the fields is the \_id field). If you specify a field with the value 0, all other fields get the value 1, and vice versa:

### Example

This example will exclude "address" from the result:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

for x in mycol.find({},{ "address": 0 }):

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mongodb_find_some2)

### Example

You get an error if you specify both 0 and 1 values in the same object (except if one of the fields is the \_id field):

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

for x in mycol.find({},{ "name": 1, "address": 0 }):

print(x)

# Python MongoDB Query

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

## Filter the Result

When finding documents in a collection, you can filter the result by using a query object.

The first argument of the find() method is a query object, and is used to limit the search.

### Example

Find document(s) with the address "Park Lane 38":

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": "Park Lane 38" }

mydoc = mycol.find(myquery)

for x in mydoc:

print(x)

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

## Advanced Query

To make advanced queries you can use modifiers as values in the query object.

E.g. to find the documents where the "address" field starts with the letter "S" or higher (alphabetically), use the greater than modifier: {"$gt": "S"}:

### Example

Find documents where the address starts with the letter "S" or higher:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": { "$gt": "S" } }

mydoc = mycol.find(myquery)

for x in mydoc:

print(x)

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

## Filter With Regular Expressions

You can also use regular expressions as a modifier.

Regular expressions can only be used to query *strings*.

To find only the documents where the "address" field starts with the letter "S", use the regular expression {"$regex": "^S"}:

### Example

Find documents where the address starts with the letter "S":

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": { "$regex": "^S" } }

mydoc = mycol.find(myquery)

for x in mydoc:

print(x)

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

# Python MongoDB Sort

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

## Sort the Result

Use the sort() method to sort the result in ascending or descending order.

The sort() method takes one parameter for "fieldname" and one parameter for "direction" (ascending is the default direction).

### Example

Sort the result alphabetically by name:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mydoc = mycol.find().sort("name")

for x in mydoc:

print(x)

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

## Sort Descending

Use the value -1 as the second parameter to sort descending.

sort("name", 1) #ascending

sort("name", -1) #descending

### Example

Sort the result reverse alphabetically by name:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mydoc = mycol.find().sort("name", -1)

for x in mydoc:

print(x)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mongodb_sort2)

# Python MongoDB Delete Document

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

## Delete Document

To delete one document, we use the delete\_one() method.

The first parameter of the delete\_one() method is a query object defining which document to delete.

Note: If the query finds more than one document, only the first occurrence is deleted.

### Example

Delete the document with the address "Mountain 21":

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": "Mountain 21" }

mycol.delete\_one(myquery)

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

## Delete Many Documents

To delete more than one document, use the delete\_many() method.

The first parameter of the delete\_many() method is a query object defining which documents to delete.

### Example

Delete all documents were the address starts with the letter S:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": {"$regex": "^S"} }

x = mycol.delete\_many(myquery)

print(x.deleted\_count, " documents deleted.")

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

## Delete All Documents in a Collection

To delete all documents in a collection, pass an empty query object to the delete\_many() method:

### Example

Delete all documents in the "customers" collection:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

x = mycol.delete\_many({})

print(x.deleted\_count, " documents deleted.")

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

# Python MongoDB Drop Collection

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

## Delete Collection

You can delete a table, or collection as it is called in MongoDB, by using the drop() method.

### Example

Delete the "customers" collection:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

mycol.drop()

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

The drop() method returns true if the collection was dropped successfully, and false if the collection does not exist.

# Python MongoDB Update

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

## Update Collection

You can update a record, or document as it is called in MongoDB, by using the update\_one() method.

The first parameter of the update\_one() method is a query object defining which document to update.

Note: If the query finds more than one record, only the first occurrence is updated.

The second parameter is an object defining the new values of the document.

### Example

Change the address from "Valley 345" to "Canyon 123":

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": "Valley 345" }

newvalues = { "$set": { "address": "Canyon 123" } }

mycol.update\_one(myquery, newvalues)

#print "customers" after the update:

for x in mycol.find():

print(x)

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

## Update Many

To update *all* documents that meets the criteria of the query, use the update\_many() method.

### Example

Update all documents where the address starts with the letter "S":

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myquery = { "address": { "$regex": "^S" } }

newvalues = { "$set": { "name": "Minnie" } }

x = mycol.update\_many(myquery, newvalues)

print(x.modified\_count, "documents updated.")

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

# Python MongoDB Limit

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

## Limit the Result

To limit the result in MongoDB, we use the limit() method.

The limit() method takes one parameter, a number defining how many documents to return.

Consider you have a "customers" collection:

### Customers

{'\_id': 1, 'name': 'John', 'address': 'Highway37'}

{'\_id': 2, 'name': 'Peter', 'address': 'Lowstreet 27'}

{'\_id': 3, 'name': 'Amy', 'address': 'Apple st 652'}

{'\_id': 4, 'name': 'Hannah', 'address': 'Mountain 21'}

{'\_id': 5, 'name': 'Michael', 'address': 'Valley 345'}

{'\_id': 6, 'name': 'Sandy', 'address': 'Ocean blvd 2'}

{'\_id': 7, 'name': 'Betty', 'address': 'Green Grass 1'}

{'\_id': 8, 'name': 'Richard', 'address': 'Sky st 331'}

{'\_id': 9, 'name': 'Susan', 'address': 'One way 98'}

{'\_id': 10, 'name': 'Vicky', 'address': 'Yellow Garden 2'}

{'\_id': 11, 'name': 'Ben', 'address': 'Park Lane 38'}

{'\_id': 12, 'name': 'William', 'address': 'Central st 954'}

{'\_id': 13, 'name': 'Chuck', 'address': 'Main Road 989'}

{'\_id': 14, 'name': 'Viola', 'address': 'Sideway 1633'}

### Example

Limit the result to only return 5 documents:

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]

mycol = mydb["customers"]

myresult = mycol.find().limit(5)

#print the result:

for x in myresult:

print(x)

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

# Python MySQL

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

Python can be used in database applications.

One of the most popular databases is MySQL.

## MySQL Database

To be able to experiment with the code examples in this tutorial, you should have MySQL installed on your computer.

You can download a free MySQL database at <https://www.mysql.com/downloads/>.

## Install MySQL Driver

Python needs a MySQL driver to access the MySQL database.

In this tutorial we will use the driver "MySQL Connector".

We recommend that you use PIP to install "MySQL Connector".

PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following:

Download and install "MySQL Connector":

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install mysql-connector-python

Now you have downloaded and installed a MySQL driver.

## Test MySQL Connector

To test if the installation was successful, or if you already have "MySQL Connector" installed, create a Python page with the following content:

demo\_mysql\_test.py:

import mysql.connector

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

If the above code was executed with no errors, "MySQL Connector" is installed and ready to be used.

## Create Connection

Start by creating a connection to the database.

Use the username and password from your MySQL database:

demo\_mysql\_connection.py:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*"

)

print(mydb)

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

# Python MySQL Create Database

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

## Creating a Database

To create a database in MySQL, use the "CREATE DATABASE" statement:

### Example

create a database named "mydatabase":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*"

)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE mydatabase")

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

If the above code was executed with no errors, you have successfully created a database.

## Check if Database Exists

You can check if a database exist by listing all databases in your system by using the "SHOW DATABASES" statement:

### Example

Return a list of your system's databases:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*"

)

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

for x in mycursor:

print(x)

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

Or you can try to access the database when making the connection:

### Example

Try connecting to the database "mydatabase":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

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

# Python MySQL Create Table

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

## Creating a Table

To create a table in MySQL, use the "CREATE TABLE" statement.

Make sure you define the name of the database when you create the connection

### Example

Create a table named "customers":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")

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

If the above code was executed with no errors, you have now successfully created a table.

## Check if Table Exists

You can check if a table exist by listing all tables in your database with the "SHOW TABLES" statement:

### Example

Return a list of your system's databases:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

for x in mycursor:

print(x)

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

## Primary Key

When creating a table, you should also create a column with a unique key for each record.

This can be done by defining a PRIMARY KEY.

We use the statement "INT AUTO\_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.

### Example

Create primary key when creating the table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255))")

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

If the table already exists, use the ALTER TABLE keyword:

### Example

Create primary key on an existing table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO\_INCREMENT PRIMARY KEY")

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

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

# Python MySQL Insert Into Table

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

## Insert Into Table

To fill a table in MySQL, use the "INSERT INTO" statement.

### Example

Insert a record in the "customers" table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = ("John", "Highway 21")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")

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

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

## Insert Multiple Rows

To insert multiple rows into a table, use the executemany() method.

The second parameter of the executemany() method is a list of tuples, containing the data you want to insert:

### Example

Fill the "customers" table with data:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = [

('Peter', 'Lowstreet 4'),

('Amy', 'Apple st 652'),

('Hannah', 'Mountain 21'),

('Michael', 'Valley 345'),

('Sandy', 'Ocean blvd 2'),

('Betty', 'Green Grass 1'),

('Richard', 'Sky st 331'),

('Susan', 'One way 98'),

('Vicky', 'Yellow Garden 2'),

('Ben', 'Park Lane 38'),

('William', 'Central st 954'),

('Chuck', 'Main Road 989'),

('Viola', 'Sideway 1633')

]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")

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

## Get Inserted ID

You can get the id of the row you just inserted by asking the cursor object.

Note: If you insert more than one row, the id of the last inserted row is returned.

### Example

Insert one row, and return the ID:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = ("Michelle", "Blue Village")

mycursor.execute(sql, val)

mydb.commit()

print("1 record inserted, ID:", mycursor.lastrowid)

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

# Python MySQL Select From

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

## Select From a Table

To select from a table in MySQL, use the "SELECT" statement:

### Example

Select all records from the "customers" table, and display the result:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

Note: We use the fetchall() method, which fetches all rows from the last executed statement.

## Selecting Columns

To select only some of the columns in a table, use the "SELECT" statement followed by the column name(s):

### Example

Select only the name and address columns:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

## Using the fetchone() Method

If you are only interested in one row, you can use the fetchone() method.

The fetchone() method will return the first row of the result:

### Example

Fetch only one row:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers")

myresult = mycursor.fetchone()

print(myresult)

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

# Python MySQL Where

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

## Select With a Filter

When selecting records from a table, you can filter the selection by using the "WHERE" statement:

### Example

Select record(s) where the address is "Park Lane 38": result:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers WHERE address ='Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

## Wildcard Characters

You can also select the records that starts, includes, or ends with a given letter or phrase.

Use the % to represent wildcard characters:

### Example

Select records where the address contains the word "way":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers WHERE address LIKE '%way%'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

## Prevent SQL Injection

When query values are provided by the user, you should escape the values.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module has methods to escape query values:

### Example

Escape query values by using the placholder %s method:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers WHERE address = %s"

adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

# Python MySQL Order By

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

## Sort the Result

Use the ORDER BY statement to sort the result in ascending or descending order.

The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword.

### Example

Sort the result alphabetically by name: result:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

## ORDER BY DESC

Use the DESC keyword to sort the result in a descending order.

### Example

Sort the result reverse alphabetically by name:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \* FROM customers ORDER BY name DESC"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

# Python MySQL Delete From By

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

## Delete Record

You can delete records from an existing table by using the "DELETE FROM" statement:

### Example

Delete any record where the address is "Mountain 21":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")

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

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted!

## Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in delete statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module uses the placeholder %s to escape values in the delete statement:

### Example

Escape values by using the placeholder %s method:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = %s"

adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")

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

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

# Python MySQL Drop Table

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

## Delete a Table

You can delete an existing table by using the "DROP TABLE" statement:

### Example

Delete the table "customers":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)

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

## Drop Only if Exist

If the the table you want to delete is already deleted, or for any other reason does not exist, you can use the IF EXISTS keyword to avoid getting an error.

### Example

Delete the table "customers" if it exists:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DROP TABLE IF EXISTS customers"

mycursor.execute(sql)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_drop_table2)

# Python MySQL Update Table

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

## Update Table

You can update existing records in a table by using the "UPDATE" statement:

### Example

Overwrite the address column from "Valley 345" to "Canyoun 123":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")

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

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

## Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in update statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module uses the placeholder %s to escape values in the delete statement:

### Example

Escape values by using the placholder %s method:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = %s WHERE address = %s"

val = ("Valley 345", "Canyon 123")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")

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

# Python MySQL Limit

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

## Limit the Result

You can limit the number of records returned from the query, by using the "LIMIT" statement:

### Example

Select the 5 first records in the "customers" table:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers LIMIT 5")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

## Start From Another Position

If you want to return five records, starting from the third record, you can use the "OFFSET" keyword:

### Example

Start from position 3, and return 5 records:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT \* FROM customers LIMIT 5 OFFSET 2")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

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

# Python MySQL Join

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

## Join Two or More Tables

You can combine rows from two or more tables, based on a related column between them, by using a JOIN statement.

Consider you have a "users" table and a "products" table:

### users

{ id: 1, name: 'John', fav: 154},

{ id: 2, name: 'Peter', fav: 154},

{ id: 3, name: 'Amy', fav: 155},

{ id: 4, name: 'Hannah', fav:},

{ id: 5, name: 'Michael', fav:}

### products

{ id: 154, name: 'Chocolate Heaven' },

{ id: 155, name: 'Tasty Lemons' },

{ id: 156, name: 'Vanilla Dreams' }

These two tables can be combined by using users' fav field and products' id field.

### Example

Join users and products to see the name of the users favorite product:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="*yourusername*",

password="*yourpassword*",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "SELECT \

users.name AS user, \

products.name AS favorite \

FROM users \

INNER JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:

print(x)

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

Note: You can use JOIN instead of INNER JOIN. They will both give you the same result.

## LEFT JOIN

In the example above, Hannah, and Michael were excluded from the result, that is because INNER JOIN only shows the records where there is a match.

If you want to show all users, even if they do not have a favorite product, use the LEFT JOIN statement:

### Example

Select all users and their favorite product:

sql = "SELECT \

users.name AS user, \

products.name AS favorite \

FROM users \

LEFT JOIN products ON users.fav = products.id"

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

## RIGHT JOIN

If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the RIGHT JOIN statement:

### Example

Select all products, and the user(s) who have them as their favorite:

sql = "SELECT \

users.name AS user, \

products.name AS favorite \

FROM users \

RIGHT JOIN products ON users.fav = products.id"

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

Note: Hannah and Michael, who have no favorite product, are not included in the result.

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