**Python MySQL Database Connection Explained with Examples**

Note: We are using **MySQL Connector Python** to connect MySQL.

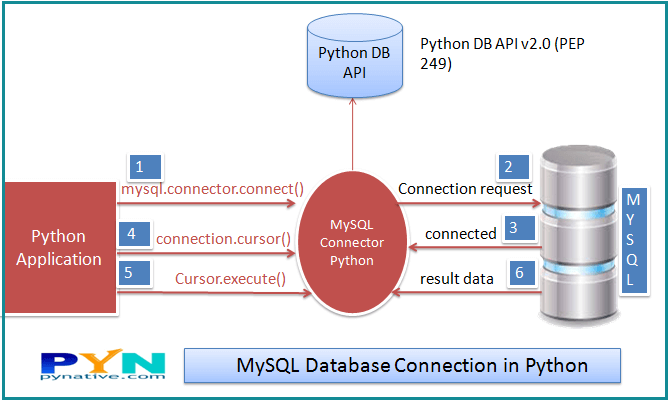
**Arguments required to connect MySQL from Python**

* **Username** –  i.e., the username that you use to work with MySQL Server. The default username for the MySQL database is a **root**
* **Password** – Password is given by the user at the time of installing the MySQL database. If you are using root then you won’t need the password.
* **Host Name**  – is the server name or Ip address on which MySQL is running. if you are running on localhost, then you can use localhost, or it’s IP, i.e. 127.0.0.0
* **Database Name** – Database name to which you want to connect. Here we are using Database named ‘**Electronics**‘  because we have already created this for our example.

**Note**: We are using the **MySQL Connector Python** module to communicate with MySQL Serv

## Steps to connect MySQL database in Python using MySQL Connector Python

1. [**Install MySQL Connector Python using pip**](https://pynative.com/install-mysql-connector-python/).
2. Use the  **mysql.connector.connect()**  method of MySQL Connector Python with required parameters to connect MySQL.
3. Use the connection object returned by a  connect()  method to create a **cursor** object to perform Database Operations.
4. The **cursor.execute()** to execute SQL queries from Python.
5. Close the Cursor object using a **cursor.close()** and MySQL database connection using **connection.close()** after your work completes.
6. Catch Exception if any that may occur during this process.



### Python Example to connect MySQL Database

To connect the MySQL database, you must know the database name you want to connect. Run below query on MySQL console if you have not created any database in MySQL. Otherwise, you can skip the below query.

**Create Database in MySQL**

Create database Electronics;

Below is the last step, i.e. using methods of MySQL Connector Python to connect MySQL database. Let see the example now.

import mysql.connector

from mysql.connector import Error

try:

connection = mysql.connector.connect(host='localhost',

database='Electronics',

user='pynative',

password='pynative@#29')

if connection.is\_connected():

db\_Info = connection.get\_server\_info()

print("Connected to MySQL Server version ", db\_Info)

cursor = connection.cursor()

cursor.execute("select database();")

record = cursor.fetchone()

print("You're connected to database: ", record)

except Error as e:

print("Error while connecting to MySQL", e)

finally:

if (connection.is\_connected()):

cursor.close()

connection.close()

print("MySQL connection is closed")

After connecting to MySQL Server, you should get below output.

Connected to MySQL Server version 5.7.19

You're connected to database: ('electronics',)

MySQL connection is closed

### Understand the Python MySQL Database connection program

**import mysql.connector**

* This line imports the MySQL Connector Python module in your program so you can use this module’s API to connect MySQL.

**from mysql.connector import Error**

* **mysql connector Error object** is used to show us an error when we failed to connect Databases or if any other database error occurred while working with the database. Example ACCESS DENIED ERROR when username or password is wrong.

**mysql.connector.connect()**

* Using this method we can connect the MySQL Database, this method accepts four required parameters: **Host, Database, User and Password** that we already discussed.
* **connect()** method established a connection to the  MySQL database from Python application and returned a MySQLConnection object.  Then we can use MySQLConnection object to perform various operations on the MySQL Database.
* The**Connect()**  method can throw an exception, i.e. Database error if one of the required parameters is wrong. For example, if you provide a database name that is not present in MySQL, then Python application throws an exception. So check the arguments that you are passing to this method.

**connection.is\_connected()**

* is\_connected() is the method of the MySQLConnection class through which we can verify is our python application connected to MySQL.

**connection.cursor()**

* This method returns a cursor object. Using a cursor object, we can execute SQL queries.
* The MySQLCursor class instantiates objects that can execute operations such as SQL statements.  
  Cursor objects interact with the MySQL server using a MySQLConnection object.

**cursor.close()**

* Using the cursor’s close method we can close the cursor object. Once we close the cursor object, we can not execute any SQL statement.

**connection.close()**

* At last, we are closing the MySQL database connection using a close() method of MySQLConnection class.

Now you know how to connect to MySQL server from python let’s proceed with creating a table from Python.

## Python MySQL Create Table

In this section, we will learn how to create a table in MySQL from Python. In this example, I am creating a **Laptop** table under the Electronics database.

import mysql.connector

from mysql.connector import Error

try:

connection = mysql.connector.connect(host='localhost',

database='Electronics',

user='pynative',

password='pynative@#29')

mySql\_Create\_Table\_Query = """CREATE TABLE Laptop (

Id int(11) NOT NULL,

Name varchar(250) NOT NULL,

Price float NOT NULL,

Purchase\_date Date NOT NULL,

PRIMARY KEY (Id)) """

cursor = connection.cursor()

result = cursor.execute(mySql\_Create\_Table\_Query)

print("Laptop Table created successfully ")

except mysql.connector.Error as error:

print("Failed to create table in MySQL: {}".format(error))

finally:

if (connection.is\_connected()):

cursor.close()

connection.close()

print("MySQL connection is closed")

**Packages to Install**

mysql-connector-python

mysql-python

If using anaconda

conda install -c anaconda mysql-python

conda install -c anaconda mysql-connector-python

else

pip install MySQL-python

pip install MySQL-python-connector

**Import Package**

import MYSQLdb

**Oracle Database**  
For communicating any database with our Python program, then we required some connector which is nothing but the *cx\_Oracle* module.

**For installing cx\_Oracle :**

pip install cx\_Oracle

By this command, you can install cx\_Oracle package but it is required to install Oracle database first in your PC.

**How to use this module for connection**

* **Import database specific module**  
  Ex. import cx\_Oracle
* **connect():** Now Establish a connection between Python program and Oracle database by using connect() function.

con = cx\_Oracle.connect('username/password@localhost')

* **cursor():** To execute sql query and to provide result some special object required is nothing but cursor() object

cursor = cx\_Oracle.cursor()

* **execute method :**

*cursor.execute(sqlquery) – – – -> to execute single query.  
cursor.execute(sqlqueries) – – – -> to execute a group of multiple sqlquery seperated by “;”*

* **commit():** For DML(Data Manuplate Language) query in this query you have (update, insert, delete) operation we need to commit() then only the result reflecte in database.
* **Fetch():** This retrieves the next row of a query result set and returns a single sequence, or None if no more rows are available.
* **close():** After all done mendentory to close all operation
* cursor.close()
* con.close()

**Creting table:**

import cx\_Oracle

# Create a table in Oracle database

try:

    con = cx\_Oracle.connect('scott/tiger@localhost')

    # Now execute the sqlquery

    cursor = con.cursor()

    # Creating a table srollno heading which is number

    cursor.execute("create table student(srollno number, \

                    name varchar2(10), efees number(10, 2)")

    print("Table Created successful")

except cx\_Oracle.DatabaseError as e:

    print("There is a problem with Oracle", e)

# by writing finally if any error occurs

# then also we can close the all database operation

finally:

    if cursor:

        cursor.close()

    if con:

        con.close()

**Inserting into table:**

import cx\_Oracle

try:

    con = cx\_Oracle.connect('scott/tiger@localhost')

    # Now execute the sqlquery

    cursor = con.cursor()

    cursor.execute("insert into student values(19585, Niranjan Shukla, 72000")

    # commit that insert the provided data

    con.commit()

    print("value inserted successful")

except cx\_Oracle.DatabaseError as e:

    print("There is a problem with Oracle", e)

# by writing finally if any error occurs

# then also we can close the all database operation

finally:

    if cursor:

        cursor.close()

    if con:

        con.close()