Database Programming in python

Introduction:

- To build the real world applications, connecting with the databases is the necessity for the programming languages. However, python allows us to connect our application to the databases like MySQL, SQLite, MongoDB, and many others.
- Python also supports Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Statements. For database programming, the Python **DB-API** is a widely used module that provides a database application programming interface.

The Python Programming language has powerful features for database programming, those are

- Python is famous for its portability.
- It is platform independent.
- In many programming languages, the application developer needs to take care of the open and closed connections of the database, to avoid further exceptions and errors. In Python, these connections are taken care of.
- Python supports relational database systems.
- Python database APIs are compatible with various databases, so it is very easy to migrate and port database application interfaces.

Environment Setup:

- In this topic we will discuss Python-MySQL database connectivity, and we will perform the database operations in python.
- The Python DB API implementation for MySQL is possible by **MySQLdb** or **mysql.connector.**

<u>Note:</u> The Python DB-API implementation for MySQL is possible by **MySQLdb** in **python2.x** but it deprecated in python3.x.In **Python3.x**,DB -API implementation for MySQL is possible by **mysql.connector.**

- ***You should have MySQL installed on your computer***

<u>Windows:</u>

You can download a free MySQL database at

https://www.mysql.com/downloads/.

<u>Linux(Ubuntu):</u>

sudo apt-get install mysql-server

- ***You need to install*** **MySQLdb: (in case of Python2.x)**
- MySQLdb is an interface for connecting to a MySQL database server from Python. The **MySQLdb** is not a built-in module, We need to install it to get it working.
- Execute the following command to install it.

➢ For (Linux)Ubuntu, use the following command -

```
sudo apt-get install python2.7-mysqldb
```

➢ For Windows command prompt, use the following command -

```
pip install MySQL-python
```

- To test if the installation was successful, or if you already have "**MySQLdb**" installed, execute following python statement at terminal or CMD.

```
import MySQLdb
```

- If the above statement was executed with no errors, "MySQLdb " is installed and ready to be used.

(OR)

- ***You need to install mysql.connector*: (in case of Python3.x)**
- To connect the python application with the MySQL database, we must import the **mysql.connector** module in the program.
- The **mysql.connector** is not a built-in module, We need to install it to get it working.
- Execute the following command to install it using pip installer.
- ➢ For (Linux)Ubuntu, use the following command -

  pip install mysql-connector-python

- ➢ For Windows command prompt, use the following command -

  pip install mysql-connector

- To test if the installation was successful, or if you already have " ***mysql.connector*** " installed, execute following python statement at terminal or CMD.

  import mysql.connector

- If the above statement was executed with no errors, "***mysql.connector*** " is installed and ready to be used.

Python Database Application Programmer’s Interface (DB-API):

- Python DB-API is independent of any database engine, which enables you to write Python scripts to access any database engine.
- The Python DB API implementation for MySQL is possible by **MySQLdb** or **mysql.connector**.
- Using Python structure, **DB-API** provides standard and support for working with databases.

The API consists of:

1. Import module(**mysql.connector or MySQLdb)**
2. Create the connection object.
3. Create the cursor object
4. Execute the query
5. Close the connection

1. Import module(mysql.connector or MySQLdb):

- To interact with MySQL database using Python, you need first to import **mysql.connector or MySQLdb** module by using following statement.

- **MySQLdb(in python2.x)**

  import MySQLdb

- **mysql.connector(in python3.x)**

  import mysql.connector

2. Create the connection object:

- After importing **mysql.connector or MySQLdb** module, we need to create connection object, for that python DB-API supports one method i.e. **connect ()** method.
- It creates connection between MySQL database and Python Application.
- If you import **MySQLdb(**in python2.x**)** then we need to use following code to create connection.

Syntax:

*Conn-name=MySQLdb.**connect**(<hostname>,<username>,<password>,<database>)*

Example:

Myconn =*MySQLdb.**connect*** ("localhost","root","root",”emp”)

(Or)

- If you import **mysql.connector(**in python3.x**)** then we need to use following code to create connection.

Syntax:

*conn-name= mysql.connector.**connect** (**host**=<host-name>,*
*   **user**=<username>,**passwd**=<pwd>,**database**=<dbname>)*

Example:

```
myconn=mysql.connector.connect(host="localhost",user="root",
                              passwd="root",database=”emp”)
```

3. Create the cursor object:

- After creation of connection object we need to create cursor object to execute SQL queries in MySQL database.
- The cursor object facilitates us to have multiple separate working environments through the same connection to the database.
- The Cursor object can be created by using **cursor ()** method.

Syntax:

*cur_came = conn-name.**cursor()***

Example:

my_cur=myconn.**cursor()**

4. Execute the query:

- After creation of cursor object we need to execute required queries by using cursor object. To execute SQL queries, python DB-API supports following method i.e. **execute ().**

Syntax:

*cur-name.**execute(query)***

Example:

my_cur.**execute** (“select * from Employee”)

5. Close the connection:

- After completion of all required queries we need to close the connection.

Syntax:

*conn-name.**close()***

Example:

*conn-name.**close()***

MySQLdb(in python2.x):

- MySQLdb is an interface for connecting to a MySQL database server from Python. The following are example programs demonstrate interactions with MySQL database using **MySQLdb** module.

- **Note** − Make sure you have root privileges of MySQL database to interact with database.i.e. Userid and password of MySQL database.

- We are going to perform the following operations on MySQL database.

  - Show databases
  - Create database
  - Create table
  - To insert data into table
  - Read/Select data from table
  - Update data in table
  - Delete data from table

Example Programs:

To display databases :

We can get the list of all the databases by using the following MySQL query.

>show databases;

Example: showdb.py

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root")
#creating the cursor object
cur = myconn.cursor()
#executing the query
dbs = cur.execute("show databases")
#display the result
for x in cur:
    print(x)
#close the connection
myconn.close()
```

Output:

```
>>>python showdb.py
('information_schema',)
('mysql',)
('performance_schema',)
('phpmyadmin',)
('test',)
('Sampledb',)
```

To Create database :

The new database can be created by using the following SQL query.

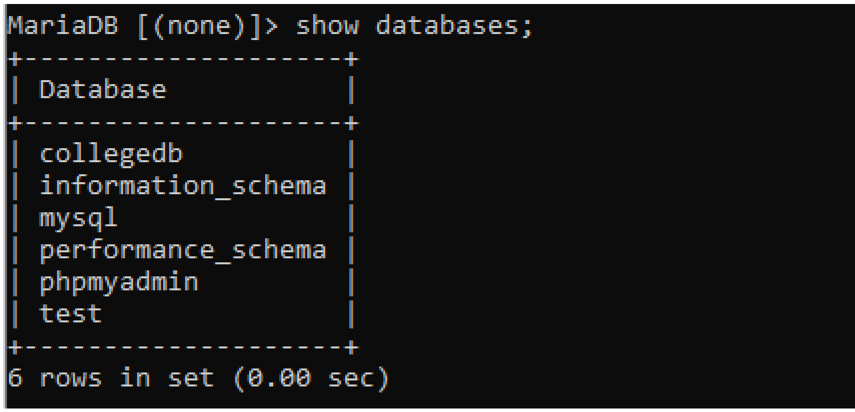
> create database <database-name>

Example: **createdb.py**

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root")
#creating the cursor object
cur = myconn.cursor()
#executing the query
cur.execute("create database Collegedb")
print("Database created successfully")
#close the connection
myconn.close()
```

Output:

```
>>>python createdb.py
Database created successfully
```



```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| collegedb          |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| test               |
+--------------------+
6 rows in set (0.00 sec)
```

To Create table :

The new table can be created by using the following SQL query.

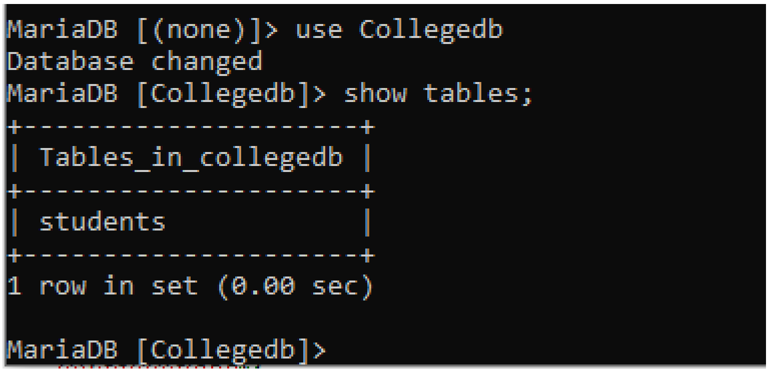
> create table <table-name> (column-name1 datatype, column-name2 datatype,…)

Example: **createtable.py**

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root",”Colleged”)
#creating the cursor object
cur = myconn.cursor()
#executing the query
cur.execute("create table students(sid varchar(20)primary key,sname varchar(25),age int(10))")
print("Table created successfully")
#close the connection
myconn.close()
```

Output:

```
>>>python createtable.py
Table created successfully
```



```
MariaDB [(none)]> use Collegedb
Database changed
MariaDB [Collegedb]> show tables;
+---------------------+
| Tables_in_collegedb |
+---------------------+
| students            |
+---------------------+
1 row in set (0.00 sec)

MariaDB [Collegedb]>
```

To Insert data into table :

The data can be inserted into table by using the following SQL query.

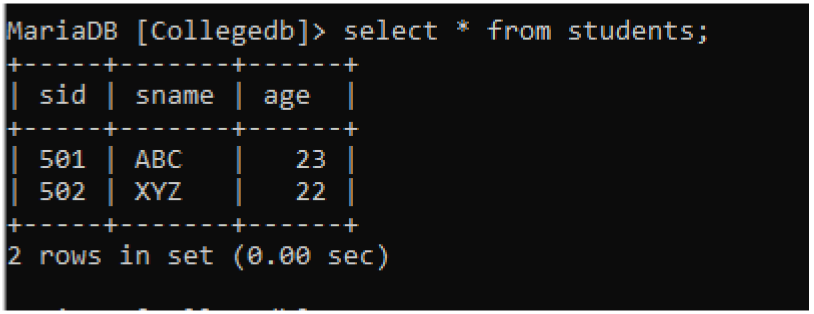
> insert into <table-name> values (value1, value2,…)

Example: **insertdata.py**

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root",”Colleged”)
#creating the cursor object
cur = myconn.cursor()
#executing the query
cur.execute("INSERT INTO students VALUES ('501', 'ABC', 23)")
cur.execute("INSERT INTO students VALUES ('502', 'XYZ', 22)")
#commit the transaction
myconn.commit()
print("Data inserted successfully")
#close the connection
myconn.close()
```

Output:

```
>>>python insertdata.py
Data inserted successfully
```



```
MariaDB [Collegedb]> select * from students;
+-----+-------+------+
| sid | sname | age  |
+-----+-------+------+
| 501 | ABC   |   23 |
| 502 | XYZ   |   22 |
+-----+-------+------+
2 rows in set (0.00 sec)
```

To Read/Select data from table ::

The data can be read/select data from table by using the following SQL query.

>select column-names from <table-name>

fetchall() method returns all rows in the table.
fetchone() method returns one row from table.

Example: **selectdata.py**

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root",”Colleged”)
#creating the cursor object
cur = myconn.cursor()
#executing the query
cur.execute("select * from students")
#fetching all the rows from the cursor object
result = cur.fetchall()
print("Student Details are :")
#printing the result
for x in result:
      print(x);
#close the connection
myconn.close()
```

Output:

```
>>>python selectdata.py

Student Details are:
('501', 'ABC', 23)
('502', 'XYZ', 22)
```

fetchall() method returns all rows in the table.
fetchone() method returns one row from table.

Example: **selectone.py**

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root",”Colleged”)
#creating the cursor object
cur = myconn.cursor()
#executing the query
cur.execute("select * from students")
#fetching all the rows from the cursor object
result = cur.fetchone()
print("One student Details are :")
 #printing the result
print(result)
#close the connection
myconn.close()
```

Output:

```
>>>python selectone.py

One student Details are:
('501', 'ABC', 23)
```

To Update data into table :

The data can be updated in table by using the following SQL query.

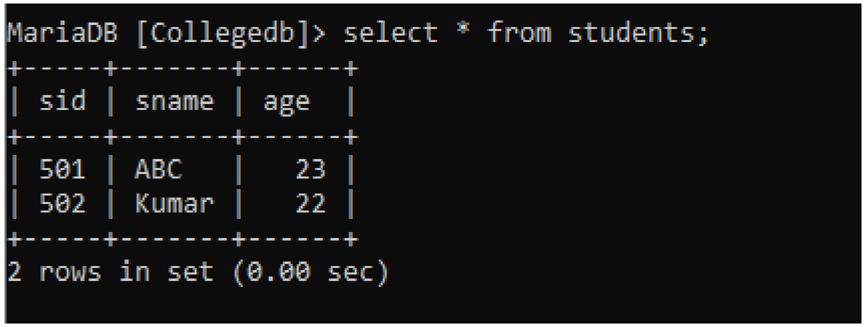
> update <table-name> set column-name=value where condition

Example: **updatedata.py**

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root",”Colleged”)
#creating the cursor object
cur = myconn.cursor()
#executing the query
cur.execute("update students set sname='Kumar' where sid='502'")
#commit the transaction
myconn.commit()
print("Data updated successfully")
#close the connection
myconn.close()
```

Output:

```
>>>python updatedata.py
Data updated successfully
```



```
MariaDB [Collegedb]> select * from students;
+-----+-------+------+
| sid | sname | age  |
+-----+-------+------+
| 501 | ABC   |   23 |
| 502 | Kumar |   22 |
+-----+-------+------+
2 rows in set (0.00 sec)
```

To Delete data from table :

The data can be deleted from table by using the following SQL query.

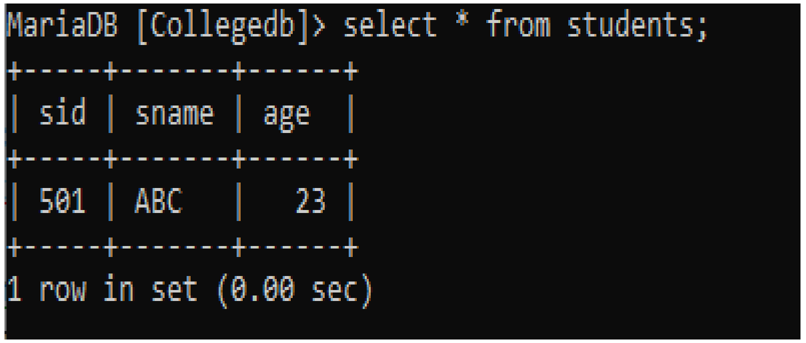
> delete from <table-name> where condition

Example: **deletedata.py**

```
import MySQLdb
 #Create the connection object
myconn = MySQLdb.connect("localhost","root","root",”Colleged”)
#creating the cursor object
cur = myconn.cursor()
#executing the query
cur.execute("delete from students where sid='502'")
#commit the transaction
myconn.commit()
print("Data deleted successfully")
#close the connection
myconn.close()
```

Output:

```
>>>python deletedata.py
Data deleted successfully
```



```
MariaDB [Collegedb]> select * from students;
+-----+-------+------+
| sid | sname | age  |
+-----+-------+------+
| 501 | ABC   |   23 |
+-----+-------+------+
1 row in set (0.00 sec)
```

DB-API for MySQL in Python

MySQLdb (python2.x)

```
#Import MySQLdb
import MySQLdb
 #Create the connection object
myconn =MySQLdb.connect
("localhost","root","root",”Colleged”)
```

Mysql.connector(python3.x)

```
#Import mysql.connector
import mysql.connector
 #Create the connection object
myconn=mysql.connector.connect
(host="localhost",user="root",
passwd="root",database="Colleged")
```

mysql.connector(in python3.x)::

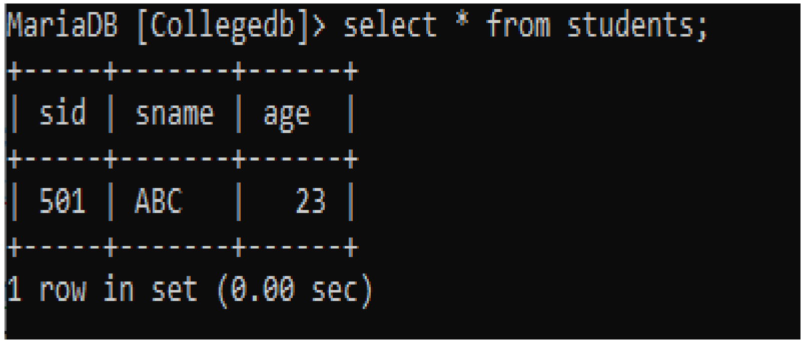
MySQL Connector enables Python programs to access MySQL databases.

Example: deletedata.py

```
import mysql.connector
 #Create the connection object
myconn=mysql.connector.connect(host="localhost",user="root",passwd="root",
database="Collegedb")
 #creating the cursor object
cur = myconn.cursor()
 #executing the querys
cur.execute("delete from students where sid='502'")
 #commit the transaction
myconn.commit()
print("Data deleted successfully")
 #close the connection
myconn.close()
```

Output:

```
>>>python deletedata.py
Data deleted successfully
```



```
MariaDB [Collegedb]> select * from students;
+-----+-------+------+
| sid | sname | age  |
+-----+-------+------+
| 501 | ABC   |   23 |
+-----+-------+------+
1 row in set (0.00 sec)
```