INTERFACE PYTHON WITH SQL

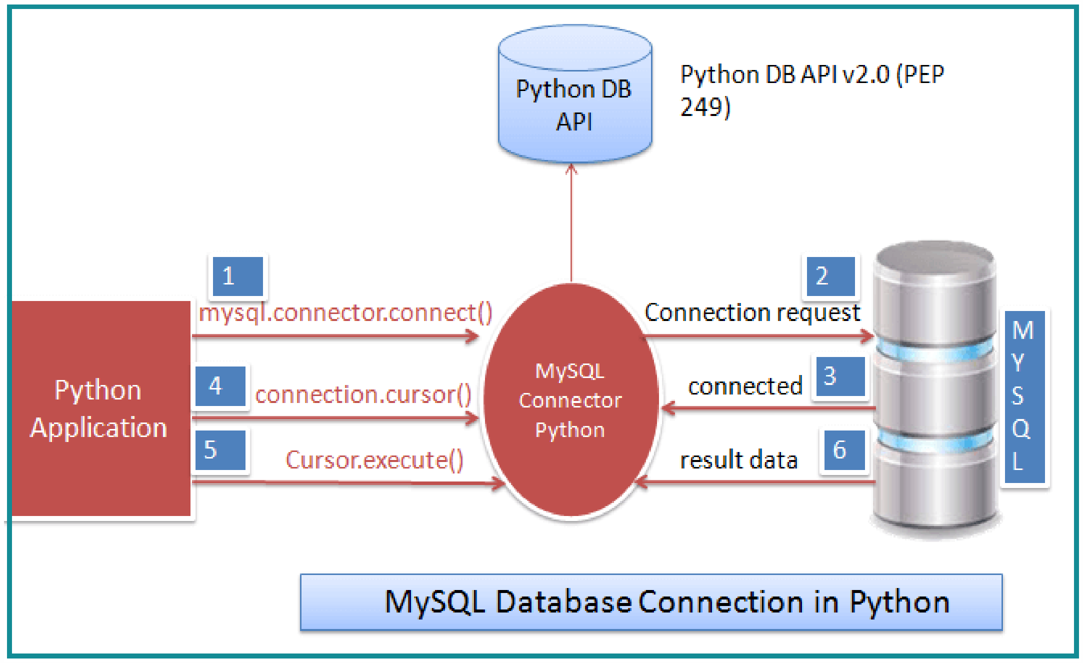
The Python programming language has powerful features for database programming. Python supports various databases like SQLite, MySQL, Oracle, Sybase, PostgreSQL, etc. Python also supports Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Statements. The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following −

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

Steps to connect MySQL database in Python using MySQL Connector Python

1. **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.



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.

mysql.connector.connect()

- Using this method we can connect the MySQL Database, this method accepts four required parameters: **Host, Database, User and Password**
  - **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.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.

- **A SQL cursor** is a database object that retrieves data from result sets one row at a time. The **cursor** in **SQL** can be used when the data needs to be updated row by row. A **SQL cursor** is a database object that is used to retrieve data from a result set one row at a time

READ Operation

READ Operation on any database means to fetch some useful information from the database.

Once our database connection is established, you are ready to make a query into this database. You can use either **fetchone()** method to fetch single record or **fetchall()** method to fetech multiple values from a database table.

- **fetchone()** − It fetches the next row of a query result set. A result set is an object that is returned when a cursor object is used to query a table.
- **fetchall()** − It fetches all the rows in a result set. If some rows have already been extracted from the result set, then it retrieves the remaining rows from the result set.
- **rowcount** − This is a read-only attribute and returns the number of rows that were affected by an execute() method.

Create Connection

Start by creating a connection to the database.

Use the username and password from your MySQL database:

```
import mysql.connector
dbCon = mysql.connector.connect(host="localhost",
                                user="root",
                                passwd="student",
                                database="school")
print("create connection")
print(dbCon)
```

Now you can start querying the database using SQL statements.

Creating a Database

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

Example

create a database named "RESULT":

```
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  passwd="student"
)
print("Successfully Connected")
print(mydb)
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE result")
```

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

```
mydb = mysql.connector.connect(
host="localhost",
user="root",
passwd="student",
database="result3"
)
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
```

#****************INSERT DATA INTO Table***********************

```
sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```

Select From a Table

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

```
mycursor.execute("SELECT * FROM customers")
myresult = mycursor.fetchall()
```

```
for x in myresult:

  print(x)
```

Selecting Columns

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

```
mycursor = mydb.cursor()

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

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

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

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:

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

myresult = mycursor.fetchone()

print("MY RESULT",myresult)
```

x---x---x