Interface Python with MySQL

Steps for creating Database Connectivity Applications

1. import mysql.connector
2. open a connection to MySQL Database : connect() function is used to establish connection to a MySQL database

   connectionObject = mysql.connector.connect
   (host=hostname,user=username,passwd=paasword,[database=databasename])

   Where
   - hostname is the server name or IP address on which MySQL is running. If you are running on localhost, then you can use **localhost**.
   - username is the user name on MySQL. The default username for the MySQL database is "**root"**.
   - Password is the password of MySQL.
   - database parameter is optional which provides the database name of a MySQL database.

   e.g.

   import mysql.connector as my

   mycon=my.connect(host="localhost",user="root",passwd="yes1",database="test")

if Python reports no error which means connection is successfully established.

3. Create a cursor instance : A database cursor is an identifier that retrieves data from result sets one row at a time. When we connect to a database from within a program, the query gets sent to the server for execution and the resultset(the set of records retrieved as per query) is sent over the connection to program in one go. To access the retrieved data row by row, database object is created that gets access to resultset and allows to traverse the resultset row by row. cursor() function is used to create an instance of cursor.

   Syntax

   cursorObject = connectionObject.cursor()

   example :

   c = mycon.cursor()

4. Execute SQL query : execute() function is used to excute SQL query.

   Syntax :

   cursorObject.execute("SQL query")

   this executes the sql query passed as parameter and store the retrieved records in the cursor object.

   Example : c.execute("select * from employee")

5. Extract data from resultset : following are the functions to extract data from the resultset
   a. data = cursorObject.fetchall() : returns all the records of resultset in a tuple form.
   b. data = cursorObject.fetchone() : returns one record from the resultset in a tuple form. First time it returns first record, next time it will fetch the next record and so on. If there are no more records then it returns None.
   c. data = cursorObject.fetchmany(n) : it accepts number of records to fetch and a tuple where each record itself is a tuple. If there are not more records then it returns an empty tuple.
   d. variable = cursorObject.rowcount : rowcount is property of cursor object that returns the number of rows retrieved from the cursor so far.
6. Clean up the environment : to close the connection established

Syntax : connectionObject.close()
Example : mycon.close()

Note : we can use pymysql library for connecting with a MySQL database. The basic difference between mysql.connector and pymysql is that the mysql.connector is made available by oracle which owns MySQL and the pymysql is made available by python.
Syntax :
connectionObject = pymysql.connect("localhost",<username>, <password>,<database name>)
Example :
mycon = pymysql.connect("localhost", "root", "yes1", "test")

Parameterized queries : are queries based on some parameters or values provided from outside.

format() method : string template with{} formatting :
e.g.
"select * from student where marks > {} and section = '{}' ".format(70,'A')
The query stores as follows:
"select * from student where marks >70 and section = 'A' "

commit() : is used to permanently store changes in the database.
Syntax : connectionObject.commit()