**Here will use mysql-connector-python library to connecting to mysql DB from pythons**

pip install mysql-connector-python

**Connecting to mysql database**

The **connect()** constructor from 'mysql.connector' package creates a connection to the MySQL server and **returns a MySQLConnection object**

*connect(user=?, password=?, port=?, database=?, \*\*kargs)*

import mysql.connector

try:

    cnx = mysql.connector.connect(user='root', password='admin',host='127.0.0.1',database='projectalpha')

    print(cnx)

except Exception as e:

    print(e)

finally:

    cnx.close()

Output:

<mysql.connector.connection\_cext.CMySQLConnection object at 0x000001C0B5D088E0>

connect() can take below option argument

1. user/username ---- user name for connection to mysql server
2. password/passwd ---- password for connecting to mysql server
3. database(db) ------ database name to use when connecting to mysql server
4. host ---- default 127.0.0.1 ---- host ip addres where mysql server running
5. port --- default 3306 --- port of mysql server
6. autocommit ---- False --- whether to autocommit transaction
7. time\_zone ---- set the time\_zone session variable at connection time
8. buffered ---- if true then created cursor will be buffered cursor
9. raw= Boolean---default to False ---- if False then returned resultset are automatically converted into python data types

Methods of MySQLConnection class

**Cursor in mysql**

A cursor allows you to iterate a set of rows returned by a query and process each row individually.

*Cursor can be created using connection object*.

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 = cnx.cursor([arg=value[, arg=value]...])

This method returns a MySQLCursor() object, or a subclass of it depending on the passed arguments. The returned object is a cursor.CursorBase instance.

Cursor are various types based on arguments passed to cursor() few of them are –

1. “cursor.MySQLCursorBuffered Class” ------ buffered = True
2. “cursor.MySQLCursorRaw Class” ------- raw=True( can be created during connect. creation)
3. “cursor.MySQLCursorBufferedRaw Class” --- buffered = True, raw= True
4. “cursor.MySQLCursorDict Class” --------- dictonary=True
5. “cursor.MySQLCursorNamedTuple Class” --- named\_tupe=True
6. “cursor.MySQLCursorBufferedDict Class” ---- dictionary=True, buffered=True
7. “cursor.MySQLCursorPrepared Class” ------ prepared=True

* If buffered is True, the cursor fetches all rows from the server after an operation is executed. This is useful when queries return small result sets.
* dictionary=True , then creates a MySQLCursorDict cursor that returns rows as dictionaries.
* named\_tuple=True, then creates a MySQLCursorNamedTuple cursor that returns rows as named tuples

**Executing query/sql command**

For executing query of any DB operation we can use execute() or executemany() based on requirements.

**execute(operation/sql\_query, params=None, multi=False)**

This method executes the given database operation (query or command).

The parameters found in the tuple or dictionary params are bound to the variables in the operation.

Specify variables using %s or %(name)s parameter style (that is, using format or pyformat style).

params ---- it specifies the data what want to insert. If fetching data not it’s not required

*execute() returns an iterator if multi is True.*

cursor.execute(operation, params=None, multi=False)

iterator = cursor.execute(operation, params=None, multi=True)

**Note:**

*execute() can insert/update/select only one row at a time, for more rows use executemany()*

Example 1:

Insert data into woodshop\_employee table of projectalpha database, where data is in tuple

import mysql.connector

try:

    cnx = mysql.connector.connect(user='root', password='admin',host='127.0.0.1',database='projectalpha')

    cursor=cnx.cursor()

except Exception as e:

    print(e)

insert\_stmt =   "INSERT INTO woodshophome\_employee (id,first\_name, last\_name) VALUES (%s, %s, %s)"

data = (11,'Brush','for painting')

data=cursor.execute(insert\_stmt, data)

Example 2:

Data is in dictionary style

import mysql.connector

try:

    cnx = mysql.connector.connect(user='root', password='admin',host='127.0.0.1',database='projectalpha')

    cursor=cnx.cursor()

except Exception as e:

    print(e)

select\_stmt = "SELECT \* FROM employees WHERE emp\_no = %(emp\_no)s"

cursor.execute(select\_stmt, { 'emp\_no': 2 })

**executemany(operations/sql\_query, seq\_of\_params)**

This method prepares a database operation (query or command) and executes it against all parameter sequences or mappings found in the sequence seq\_of\_params.

*This method runs sql query against all set of seq\_of\_params*

operations --- this is the sql query

seq\_of\_data --- this is the data as list of tupple

Example1:

import mysql.connector

try:

    cnx = mysql.connector.connect(user='root', password='admin',host='127.0.0.1',database='projectalpha')

    cursor=cnx.cursor()

except Exception as e:

    print(e)

insert\_stmt =   "INSERT INTO woodshophome\_employee (id,first\_name, last\_name) VALUES (%s, %s, %s)"

data = [(12,'Brush','for painting'),(13,'Brush','for painting')]

cursor.executemany(insert\_stmt,data )

cnx.commit()

cursor.close()

cnx.close()

if we run the same query using execute() then we get error as there are multiple data against query

**Fetching rows/data from DB**

**We have below methods in cursor class which we can use to retrieve data**

For fetching data/rows from DB we have three methods and all are executed on cursor object.

1. MySQLCursor.fetchall() Method
2. MySQLCursor.fetchmany() Method
3. MySQLCursor.fetchone() Method

**MySQLCursor.fetchmany(size=integer)**

This method fetches the next set of rows of a query result and returns a list of tuples. If no more rows are available, it returns an empty list

size --- it represent number of rows want to fetch, this default to 1 or if not specified

How to fetch data:

Step 1 --- execute sql statement using cursor, here dataset are not required

Step 2 --- using cursor run execute fetchone() or fetchmany() or fetchall()

query="select \* from woodshophome\_stock"

cursor.execute(query)

data=cursor.fetchmany(size=2) # size=2 means fetch 2 rowss

for each in data:

    print(each)

Output:

(1, 'Chair', 'King design', 0, 10, 'number')

(2, 'Bed', 'queen sze', 0, 12, 'unit')

**MySQLCursor.fetchall() Method**

This method fetched all records from DB or remaining records and returns in list of tuple.

If no more rows are available, it returns an empty list.

Example 1 ---- Fetch all records from DB

query="select \* from woodshophome\_stock"

cursor.execute(query)

data=cursor.fetchall()

for each in data:

    print(each)

Example 1 ---- Fetch first 2 rows and then fetch remaining records

query="select \* from woodshophome\_stock"

cursor.execute(query)

data=cursor.fetchmany(size=2)

for each in data:

    print(each)

print('now getting all remaning records')

data=cursor.fetchall()

for each in data:

    print(each)

output-

(1, 'Chair', 'King design', 0, 10, 'number')

(2, 'Bed', 'queen sze', 0, 12, 'unit')

now getting all remaning records

(3, 'plywood', 'general purpose', 0, 1000, 'unit')

(4, 'Paint', 'Asian paint', 0, 123,

**MySQLCursor.fetchone() Method**

This method retrieves the next row of a query result set and returns a single sequence, or None if no more rows are available. By default, the returned tuple consists of data returned by the MySQL server, converted to Python objects

*The fetchone() method is used by fetchall() and fetchmany(). It is also used when a cursor is used as an iterator* *as you can see in below example.*

**Note:**

We can directly fetch records from cursor and number of data will be based on SQL query.

Records fetched will be stored in cursor object no need to store in any variable

Example:

Below code will run all records from specified table and database

query="select \* from woodshophome\_stock"

cursor.execute(query)

#data=cursor.fetchmany(size=2)

for each in cursor:

    print(each)

Output: it fetched all records

(1, 'Chair', 'King design', 0, 10, 'number')

(2, 'Bed', 'queen sze', 0, 12, 'unit')

(3, 'plywood', 'general purpose', 0, 1000, 'unit')

(4, 'Paint', 'Asian paint', 0, 123, 'unit')

**Some other methods of cursor class**

**MySQLCursor.fetchwarnings()**

fetchwarnings()--- used for fetching warning of last executed query*. To use this methods we should use value of get\_warnings parameter as True while creating connection object.*

**MySQLCursor.stored\_results()**

This method returns a list iterator object that can be used to process result sets produced by a stored procedure executed using the callproc() method.

**Attributes of cursor class**

1. cursor.column\_names ----- This read-only property returns the column names of a result set as sequence of Unicode strings.
2. cursor.description
3. cursor.lastrowid
4. cursor.rowcount
5. cursor.with\_rows
6. cursor.statement