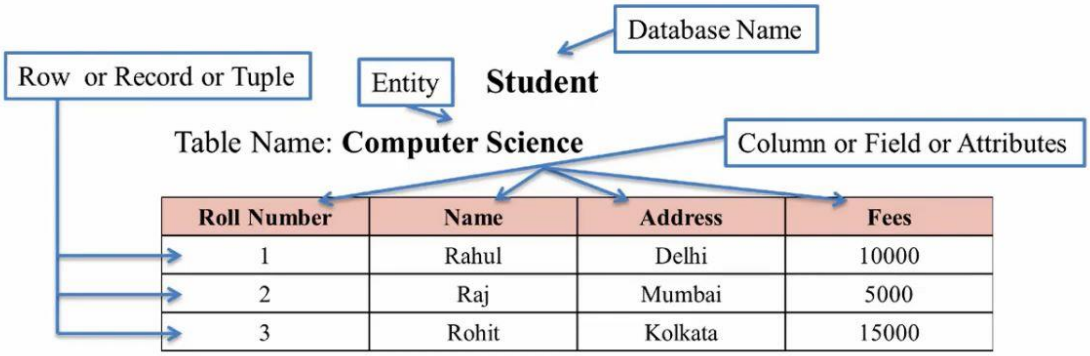
Getting Started MySQL with Python

Database

Database is integrated collection of related information along with the details so that it is available to the several user for the different application.



Roll Number	Name	Address	Fees
1	Rahul	Delhi	10000
2	Raj	Mumbai	5000
3	Rohit	Kolkata	15000

Python Supports various Databases

- MySQL
- MS-SQL
- SQLite
- MongoDB
- Oracle OCI8
- PostgreSQL
- Firebird
- MS Access

Muhammad Hashim

Getting Started MySQL with Python

Creating Connection

connect() – This method is used to open or establish a new connection. It returns an object representing the connection.

Syntax: -

```
connection_object = connect(user='username', password='pass' host='localhost', port=3306);
```

eg: -

```
import mysql.connector
conn = mysql.connector.connect(user='root', password= `root`  host='localhost', port=3306)
```

Creating connection using Dictionary

```
import mysql.connector
config = {
        'user': 'root',
        'password': `root`
        'host' : 'localhost',
        'port': 3307
}
conn = mysql.connector.connect(**config)
```

Muhammad Hashim

Getting Started MySQL with Python

Check Connection

is_connected() – This method is used to check if the connection to MySQL is established or not. It returns True if the connection is established successfully.

Syntax:- connection_object.is_connected()

eg:-

```
import mysql.connector
conn = mysql.connector.connect(user='root', password= `root` ,
host='localhost')
print(conn.is_connected())
```

Connection Code

```
1  # Creating Connection
2  import mysql.connector
3
4  config = {
5      'user':'root',
6      'password': `root` ,
7      'host':'localhost',
8      'port':3306
9  }
10 try:
11     conn = mysql.connector.connect(**config)
12     if(conn.is_connected()):
13         print('Connected')
14 except:
15     print('Unable to Connect')
```

Muhammad Hashim

Getting Started MySQL with Python

Closing a connection

```
14 print('Before Close:', conn.is_connected())
15 conn.close()
16 print('After Close:', conn.is_connected())
```

cursor() Method

This method is used to create cursor class object.

We need cursor object so we can call execute() method.

Syntax:- cursor_object = connection_object.cursor()

Arguments may be passed to the cursor() method to control what type of cursor to create:

- 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. buffered can be used alone, or in combination with the dictionary or named_tuple argument.
- If *dictionary* is *True*, the cursor returns rows as dictionaries.
- If *named_tuple* is *True*, the cursor returns rows as named tuples.
- If *prepared* is *True*, the cursor is used for executing prepared statements.

Getting Started MySQL with Python

cursor() Method

- if *raw* is *True*, the cursor skips the conversion from MySQL data types to Python types when fetching rows. A raw cursor is usually used to get better performance or when you want to do the conversion yourself.
- The *cursor_class* argument can be used to pass a class to use for instantiating a new cursor. It must be a subclass of cursor.CursorBase.

The returned object depends on the combination of the arguments. Examples:

- If not buffered and not raw: MySQLCursor
- If buffered and not raw: MySQLCursorBuffered
- If not buffered and raw: MySQLCursorRaw
- If buffered and raw: MySQLCursorBufferedRaw

cursor() Method

The returned object depends on the combination of the arguments. Examples:

- If not buffered and not raw: MySQLCursor
- If buffered and not raw: MySQLCursorBuffered
- If not buffered and raw: MySQLCursorRaw
- If buffered and raw: MySQLCursorBufferedRaw

eg:- myc = conn.cursor() ← MySQLCursor

eg:- myc = conn.cursor(buffered=True) ← MySQLCursorBuffered

eg:- myc = conn.cursor(prepared=True) ← MySQLCursorPrepared

Getting Started MySQL with Python

execute() Method

This method is used to execute given SQL queries.

We need cursor object so we can call execute() method.

Syntax:- cursor_object.execute(sql, param=None, multi=False)

Sql – It is sql query.

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

Multi – execute() returns an iterator if multi is True.

eg:-

```
myc = conn.cursor()
myc.execute('SELECT * FROM student')

sql ='SELECT * FROM student'
myc = conn.cursor()
myc.execute(sql)
```

Close Cursor

close () method closes the cursor, resets all results, and ensures that the cursor object has no reference to its original connection object.

Syntax:- cursor_object.close()

eg:- myc.close()

Muhammad Hashim

Getting Started MySQL with Python

Now Remind all previous step and let's dive in with Muhammad Hashim

Step #1: Create Database!

```
3  try:
4      conn = mysql.connector.connect(
5          user='root'.
6          password= `root` ,
7          host='localhost',
8          port=3306
9      )
10     if(conn.is_connected()):
11         print('Connected')
12 except:
13     print('Unable to Connect')
14
15 sql = 'CREATE DATABASE pdb'
16 myc = conn.cursor()
17 myc.execute(sql)
18
19 myc.close()
20 conn.close()
```

Step #2: Now connect to Database and create a Table!

```
4      conn = mysql.connector.connect(
5          user='root'.
6          password= `root` ,
7          host='localhost',
8          database='pdb',
9          port=3306
10     )
11     if(conn.is_connected()):
12         print('Connected')
13 except:
14     print('Unable to Connect')
15 sql = 'CREATE TABLE student1(stuid INT AUTO_INCREMENT PRIMARY KEY, name
   VARCHAR(20), roll INT, fees FLOAT)'
16 myc = conn.cursor()
17 myc.execute(sql)
18 myc.close()
19 conn.close()
```

Getting Started MySQL with Python

Step #3: Now Show Tables!

```
sql = 'SHOW TABLES'
myc = conn.cursor()
myc.execute(sql)
for t in myc:
    print(t)
myc.close()
conn.close()
```

commit() Method

This method is used to save inserted row in the table. It is required to make the changes, otherwise no changes are made to the table.

This method sends a COMMIT statement to the MySQL server, committing the current transaction. Since by default Connector/Python does not autocommit, it is important to call this method after every transaction that modifies data for tables that use transactional storage engines.

Syntax:- connection_object.commit()

eg:- conn.commit()

Getting Started MySQL with Python

rollback() Method

This method is used to un-save row, if there is an error.

This method sends a ROLLBACK statement to the MySQL server, undoing all data changes from the current transaction. By default, Connector/Python does not autocommit, so it is possible to cancel transactions when using transactional storage engines such as InnoDB.

Syntax:- connection_object.rollback()

eg:- conn.rollback()

```
try:
        myc.execute(sql)
        conn.commit()
except:
        conn.rollback()
```

Insert Records

```
        host='localhost',
        database='pdb',
        port=3306
    )
    if(conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')
sql = 'INSERT INTO student(name, roll, fees) VALUES("Sumit", 101, 10000.52)'
myc = conn.cursor()
myc.execute(sql)
conn.commit()

myc.close()
conn.close()
```

Getting Started MySQL with Python

rowcount Property

This read-only property returns the number of rows returned for SELECT statements, or the number of rows affected by DML statements such as INSERT or UPDATE.

Syntax:- cursor_object.rowcount

eg:- myc.rowcount

Row count Property

```
        host='localhost',
        database='pdb',
        port=3306
    )
    if(conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')
sql = 'INSERT INTO student(name, roll, fees) VALUES("Jyoti", 107, 70000.2),
("Aman", 108, 8000.4), ("Raj", 109, 45654)'
myc = conn.cursor()
try:
    myc.execute(sql)
    conn.commit()
    print(myc.rowcount, 'Row Inserted')
except:
    conn.rollback()
    print('Unable to Insert Data')

myc.close()
```


Muhammad Hashim

Getting Started MySQL with Python

lastrowid Property

This read-only property returns the value generated for an AUTO_INCREMENT column by the previous INSERT or UPDATE statement or None when there is no such value available.

If you perform an INSERT into a table that contains an AUTO_INCREMENT column, lastrowid returns the AUTO_INCREMENT value for the new row.

If you insert multiple rows into a table using a single INSERT statement, the lastrowid property contains the last insert id of the first row.

Syntax:- cursor_object.lastrowid

eg:- myc.lastrowid

```
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect'
sql = 'INSERT INTO student(na
myc = conn.cursor()
try:
    myc.execute(sql)
    conn.commit()       # Com
    print(myc.lastrowid)
except:
    conn.rollback()     # Rol
    print('Unable to Insert D
myc.close()     # Close Curso
conn.close()    # Close Conne
```

Muhammad Hashim

Getting Started MySQL with Python

Delete Record

```
        port=3306
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'DELETE FROM student WHERE stuid=55'
myc = conn.cursor()
try:
    myc.execute(sql)
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Inserted')
except:
    conn.rollback()     # Rollback the change

myc.close()     # Close Cursor
conn.close()    # Close Connection
```

Update Record

```
 9          port=3306
10      )
11      if (conn.is_connected()):
12          print('Connected')
13  except:
14      print('Unable to Connect')
15
16  sql = 'UPDATE student SET fees=200 WHERE stuid=56'
17  myc = conn.cursor()
18  try:
19      myc.execute(sql)
20      conn.commit()       # Committing the change
21      print(myc.rowcount, 'Row Deleted')
22  except:
23      conn.rollback()     # Rollback the change
24      print('Unable to Delete Data')
25  myc.close()     # Close Cursor
26  conn.close()    # Close Connection
27
```

Muhammad Hashim

Getting Started MySQL with Python

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. If the cursor is a raw cursor, no such conversion occurs.

You must fetch all rows for the current query before executing new statements using the same connection.

Syntax:- row = cursor_object.fetchone()

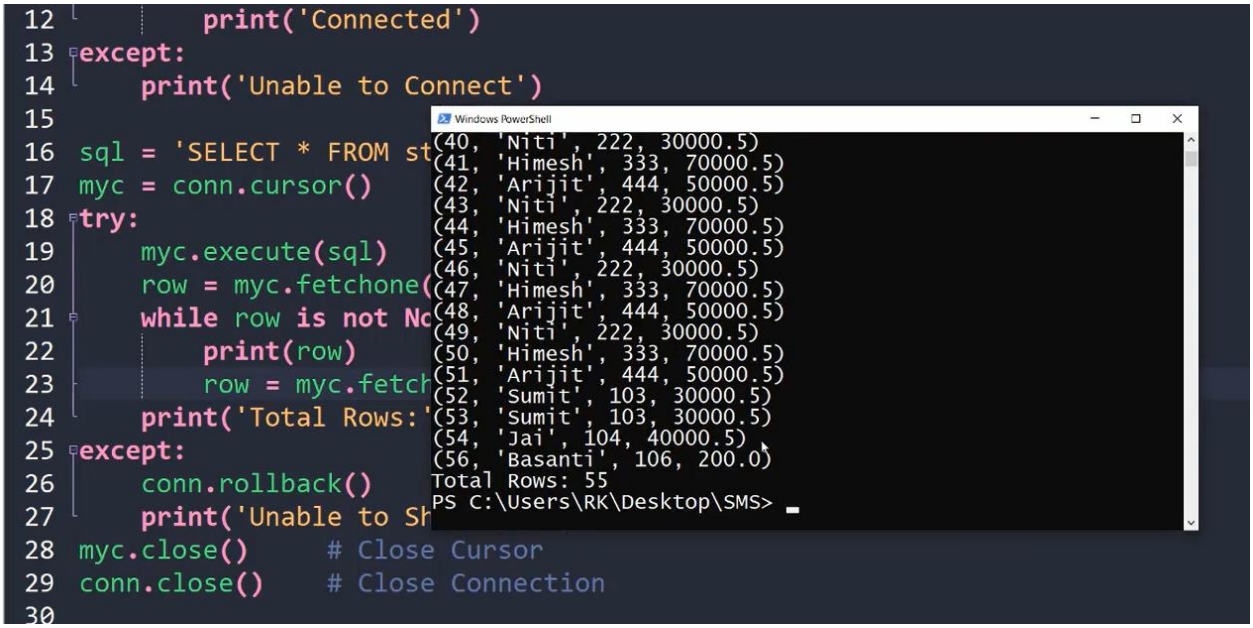
eg:- row = myc.fetchone()

```
        print('Connected')
except:
    print('Unable to Connect')

sql = 'SELECT * FROM student'
myc = conn.cursor()
try:
    myc.execute(sql)
    row = myc.fetchone()
    while row is not None:
        print(row)
        row = myc.fetchone()
    print('Total Rows:', myc.rowcount)
except:
    conn.rollback()     # Rollback the change
    print('Unable to Show Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```

Muhammad Hashim

Getting Started MySQL with Python

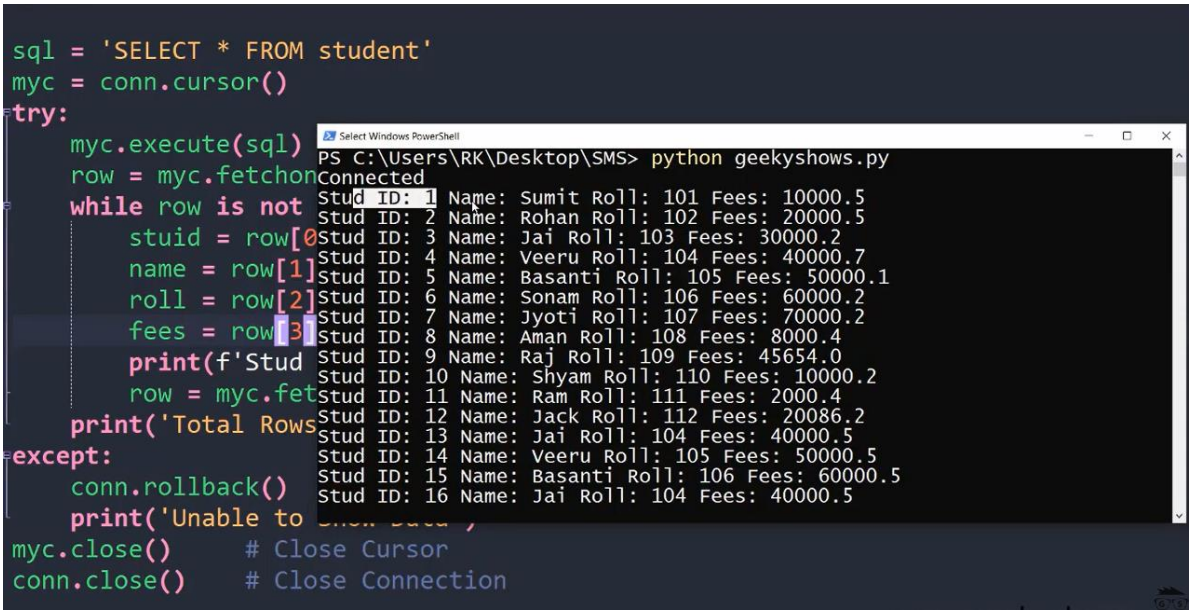


Fetch using format string

```
sql = 'SELECT * FROM student'
myc = conn.cursor()
try:
    myc.execute(sql)
    row = myc.fetchone()
    while row is not None:
        stuid = row[0]
        name = row[1]
        roll = row[2]
        fees = row[3]
        print(f'Stud ID: {stuid} Name: {name} Roll: {roll} Fees: {fees}')
        row = myc.fetchone()
    print('Total Rows:', myc.rowcount)
except:
    conn.rollback()     # Rollback the change
    print('Unable to Show Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```

Muhammad Hashim

Getting Started MySQL with Python



fetchall() Method

This method fetches all (or all remaining) rows of a query result set and returns a list of tuples. If no more rows are available, it returns an empty list.

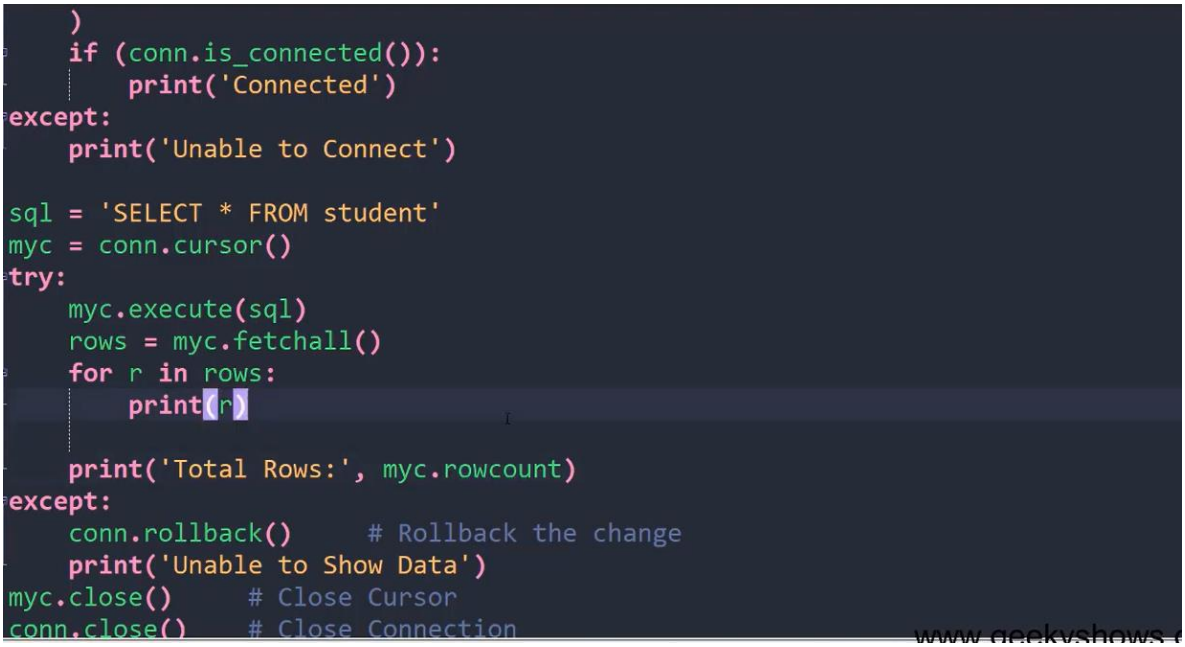
You must fetch all rows for the current query before executing new statements using the same connection.

Syntax:- rows = cursor_object.fetchall()

eg:- rows = myc.fetchall()

Muhammad Hashim

Getting Started MySQL with Python



```
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'SELECT * FROM student'
myc = conn.cursor()
try:
    myc.execute(sql)
    rows = myc.fetchall()
    for r in rows:
        print(r)

    print('Total Rows:', myc.rowcount)
except:
    conn.rollback()     # Rollback the change
    print('Unable to Show Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```


fetchmany() Method

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.

The number of rows returned can be specified using the size argument, which defaults to one. Fewer rows are returned if fewer rows are available than specified.

You must fetch all rows for the current query before executing new statements using the same connection.

Syntax:- rows = cursor_object.fetchmany(size=1)

eg:- rows = myc.fetchmany(3)

Muhammad Hashim

Getting Started MySQL with Python

```
16 sql = 'SELECT * FROM student'
17 myc = conn.cursor(buffered=True)
18 try:
19     myc.execute(sql)
20     rows = myc.fetchmany(size=5)
21     for r in rows:
22         stuid = r[0]
23         name = r[1]
24         roll = r[2]
25         fees = r[3]
26         print(f'StuID: {stuid} Name: {name} Roll: {roll} Fees:{fees}')
27 
28     print('Total Rows:', myc.rowcount)
29 except:
30     conn.rollback()     # Rollback the change
31     print('Unable to Show Data')
32 myc.close()     # Close Cursor
33 conn.close()    # Close Connection
```

fetchmany() with Where

```
 8         database='pdb',
 9         port=3306
10     )
11     if (conn.is_connected()):
12         print('Connected')
13 except:
14     print('Unable to Connect')
15 
16 sql = 'SELECT * FROM student WHERE stuid=4'
17 myc = conn.cursor()
18 try:
19     myc.execute(sql)
20     row = myc.fetchone()
21     print(row)
22     print(myc.rowcount)
23 
24 except:
25     conn.rollback()     # Rollback the change
26     print('Unable to Show Data')
27 myc.close()     # Close Cursor
```


Muhammad Hashim

Getting Started MySQL with Python

Parameterized Query

A parameterized query is a query which can use the format or pyformat parameterization style for parameters and the parameter values supplied at execution.

These executed with MySQLCursor can use the %s and %(key)s format style.

%s is used as format style in the sql queries, while using tuple parameters.

%(key)s is used as format style in the sql queries, while using dictionary parameters.

```
myc = conn.cursor()
```

Tuple Parameters

```
sql = 'INSERT INTO student(name, roll, fees) VALUES(%s, %s, %s)‘
myc = conn.cursor()
myc.execute(sql, ("Rohan", 111, 60000.50))
```

```
sql = 'INSERT INTO student(name, roll, fees) VALUES(%s, %s, %s)'
myc = conn.cursor()
params = ("Rohan", 111, 60000.50)
myc.execute(sql, params)
```

Getting Started MySQL with Python

Dictionary Parameters

```
sql = 'INSERT INTO student(name, roll, fees) VALUES(%(name)s, %(roll)s,
%(fees)s)‘
myc = conn.cursor()
myc.execute(sql, {'name':'Kajal', 'roll':777, 'fees': 54100})
```

```
sql = 'INSERT INTO student(name, roll, fees) VALUES(%(name)s, %(roll)s,
%(fees)s)‘
myc = conn.cursor()
params = {'name':'Kajal', 'roll':777, 'fees': 54100}
myc.execute(sql, params)
```

Using tuple

```
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'INSERT INTO student(name, roll, fees) VALUES(%s, %s, %s)'
myc = conn.cursor()
params = ("Rahul", 102, 50000.50)
try:
    myc.execute(sql, params)
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Inserted')     # Number of Row Inserted
    print(f'Stu ID: {myc.lastrowid} Inserted')  # Last inserted ID
except:
    conn.rollback()     # Rollback the change
    print('Unable to Insert Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```

Muhammad Hashim

Getting Started MySQL with Python

Insert multiple Data

```
    )
    if(conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')
sql = 'INSERT INTO student(name, roll, fees) VALUES("Jai", 103, 30000.2),
("Veeru", 104, 40000.7), ("Basanti", 105, 50000.1)'
myc = conn.cursor()
try:
    myc.execute(sql)
    conn.commit()
    print('Row Inserted')
except:
    conn.rollback()
    print('Unable to Insert Data')

myc.close()
conn.close()
```

executemany() Method

This method is used to prepare given SQL query and executes it against all parameter sequences or mappings found in the sequence seq_of_params.

With the executemany() method, it is not possible to specify multiple statements to execute in the sql argument.

Syntax:- cursor_object.executemany(sql, seq_of_params)

sql – It is sql qrery

seq_of_params – It is a list of tuples, containing the data to insert.

Muhammad Hashim

Getting Started MySQL with Python

```
        port=3306
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'INSERT INTO student(name, roll, fees) VALUES(%s, %s, %s)'
myc = conn.cursor()
seq_of_params = [("Jai", 104, 40000.50), ("Veeru", 105, 50000.50), ("Basanti",
106, 60000.50)]
try:
    myc.executemany(sql, seq_of_params)
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Inserted')     # Number of Row Inserted
except:
    conn.rollback()     # Rollback the change
    print('Unable to Insert Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```

Input from user and pass as a object

```
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'INSERT INTO student(name, roll, fees) VALUES(%s, %s, %s)'
myc = conn.cursor()
# Input from user
nm = input('Enter Name: ')
ro = int(input('Enter Roll: '))
fe = float(input('Enter Fees: '))
params = (nm, ro, fe)
try:
    myc.execute(sql, params)
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Inserted')     # Number of Row Inserted
    print(f'Stu ID: {myc.lastrowid} Inserted')  # Last inserted ID
except:
    conn.rollback()     # Rollback the change
    print('Unable to Insert Data')
```

Muhammad Hashim

Getting Started MySQL with Python

Function example in student Data

```python
# Insert Single Row - Input from User - Tuple
import mysql.connector
def student_data(n, r, f):
    try:
        conn= mysql.connector.connect(
            user='root',
            password `root` ,
            host='localhost',
            database='pdb',
            port=3306
        )
        if (conn.is_connected()):
            print('Connected')
    except:
        print('Unable to Connect')

    sql = 'INSERT INTO student(name, roll, fees) VALUES(%s, %s, %s)'
    myc = conn.cursor()
    params = (nm, ro, fe)
    try:
```



```python
    except:
        conn.rollback()     # Rollback the change
        print('Unable to Insert Data')
    myc.close()     # Close Cursor
    conn.close()    # Close Connection

while True:
    # Input from user
    nm = input('Enter Name: ')
    ro = int(input('Enter Roll: '))
    fe = float(input('Enter Fees: '))
    def student_data(nm, ro, fe)
    ans = input('Do You want to exit?(y/n)')
    if(ans == 'y'):
        break
```

Insert data using parameterized dictionary

Muhammad Hashim

Getting Started MySQL with Python

```
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'INSERT INTO student(name, roll, fees) VALUES(%(n)s, %(r)s, %(f)s)'
myc = conn.cursor()
try:
    myc.execute(sql, {'n':'Sameer', 'r':777, 'f':54100})
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Inserted')     # Number of Row Inserted
    print(f'Stu ID: {myc.lastrowid} Inserted')  # Last inserted ID
except:
    conn.rollback()     # Rollback the change
    print('Unable to Insert Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```

Insert multiple data using parameterized dictionary

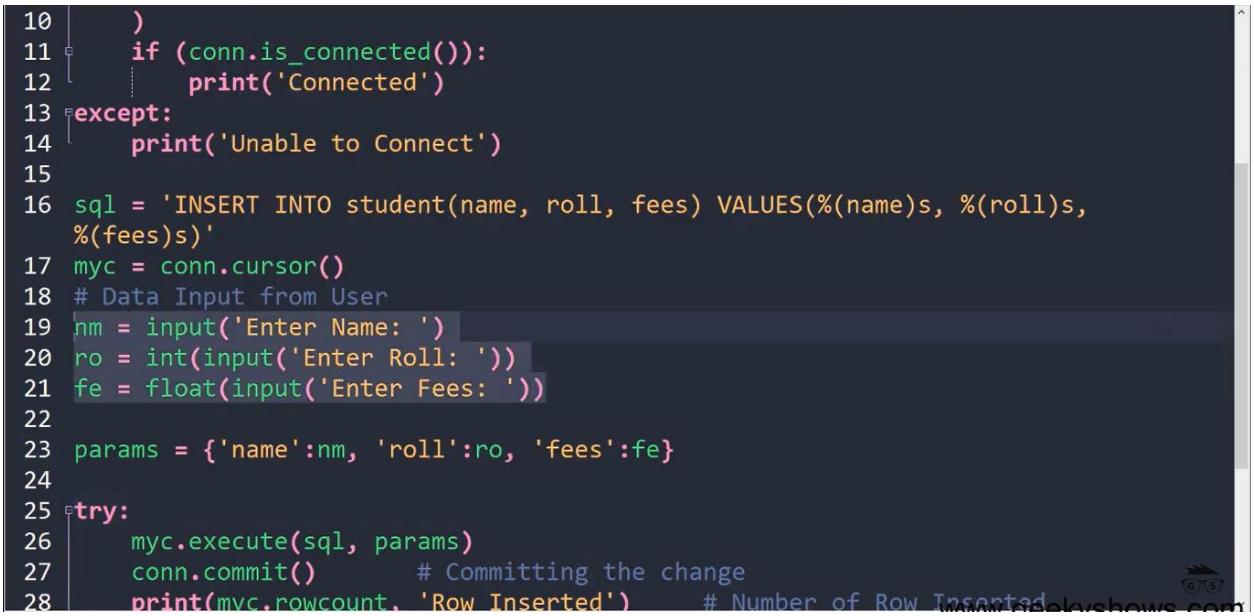
```
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'INSERT INTO student(name, roll, fees) VALUES(%(name)s, %(roll)s,
%(fees)s)'
myc = conn.cursor()
params = [                                  # List of Dictionaries
    {'name':'Ajay', 'roll':124, 'fees': 5426.23},
    {'name':'Rani', 'roll':845, 'fees': 845621.23},
    {'name':'Rohit', 'roll':659, 'fees': 47426.23} ]
try:
    myc.execute(sql, params)
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Inserted')     # Number of Row Inserted
    print(f'Stu ID: {myc.lastrowid} Inserted')  # Last inserted ID
except:
    conn.rollback()     # Rollback the change
    print('Unable to Insert Data')
```


Muhammad Hashim

Getting Started MySQL with Python

Input from user using parameterized dictionary



```
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'INSERT INTO student(name, roll, fees) VALUES(%(name)s, %(roll)s,
%(fees)s)'
myc = conn.cursor()
# Data Input from User
nm = input('Enter Name: ')
ro = int(input('Enter Roll: '))
fe = float(input('Enter Fees: '))

params = {'name':nm, 'roll':ro, 'fees':fe}

try:
    myc.execute(sql, params)
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Inserted')     # Number of Row Inserted
```

Function example in student Data

```
# Insert Multiple Rows - Input from User - Dictionary
import mysql.connector
def student_data(nm, ro, fe)
    try:
        conn= mysql.connector.connect(
            user='root',
            password: `root` ,
            host='localhost',
            database='pdb',
            port=3306
        )
        if (conn.is_connected()):
            print('Connected')
    except:
        print('Unable to Connect')

    sql = 'INSERT INTO student(name, roll, fees) VALUES(%(name)s, %(roll)s,
    %(fees)s)'
    myc = conn.cursor()
```

Muhammad Hashim

Getting Started MySQL with Python

```
16
17      sql = 'INSERT INTO student(name, roll, fees) VALUES(%(name)s, %(roll)s,
        %(fees)s)'
18      myc = conn.cursor()
19
20      params = {'name':nm, 'roll':ro, 'fees':fe}
21
22      try:
23          myc.execute(sql, params)
24          conn.commit()       # Committing the change
25          print(myc.rowcount, 'Row Inserted')     # Number of Row Inserted
26          print(f'Stu ID: {myc.lastrowid} Inserted')  # Last inserted ID
27      except:
28          conn.rollback()     # Rollback the change
29          print('Unable to Insert Data')
30      myc.close()     # Close Cursor
31      conn.close()    # Close Connection
32
33  while True:
```

```
27      except:
28          conn.rollback()     # Rollback the change
29          print('Unable to Insert Data')
30      myc.close()     # Close Cursor
31      conn.close()    # Close Connection
32
33  while True:
34      # Data Input from User
35      nm = input('Enter Name: ')
36      ro = int(input('Enter Roll: '))
37      fe = float(input('Enter Fees: '))
38      student_data(nm, ro, fe)
39      ans = input('Do you want exit(y/n)?')
40      if(ans=='y'):
41          break
42
```

Delete Record using tuple

Muhammad Hashim

Getting Started MySQL with Python

```
        port=3306
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'DELETE FROM student WHERE stuid=%s'
myc = conn.cursor()
del_value = (5,)
try:
    myc.execute(sql, del_value)
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Deleted')
except:
    conn.rollback()     # Rollback the change
    print('Unable to Delete Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```

Delete Record using dictionary

```
        port=3306
    )
    if (conn.is_connected()):
        print('Connected')
except:
    print('Unable to Connect')

sql = 'DELETE FROM student WHERE stuid=%(id)s'
myc = conn.cursor()
#n = int(input('Enter ID to Delete: '))
#del_value = (n,)
try:
    myc.execute(sql, {'id': 4})
    conn.commit()       # Committing the change
    print(myc.rowcount, 'Row Deleted')
except:
    conn.rollback()     # Rollback the change
    print('Unable to Delete Data')
myc.close()     # Close Cursor
conn.close()    # Close Connection
```


Muhammad Hashim