BCA – 502: Python Programming

Rahul Kumar Singh

In today's Class we have discussed on SQL Database connection using python.

Database Connection:-

MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

How do I Install MySQLdb?

Before proceeding, you make sure you have MySQLdb installed on your machine. Just type the following in your Python script and execute it −

```
#!/usr/bin/python

import MySQLdb
```

If it produces the following result, then it means MySQLdb module is not installed −

```
Traceback (most recent call last):

  File "test.py", line 3, in <module>

    import MySQLdb

ImportError: No module named MySQLdb
```

To install MySQLdb module, use the following command –

For Python command prompt, use the following command -

pip install MySQL-python

Before connecting to a MySQL database, make sure of the followings –

- You have created a database TESTDB.
- You have created a table EMPLOYEE in TESTDB.
- This table has fields FIRST_NAME, LAST_NAME, AGE, SEX and INCOME.
- User ID "testuser" and password "test123" are set to access TESTDB.
- Python module MySQLdb is installed properly on your machine.
- You have gone through MySQL tutorial to understand MySQL Basics.

Example:-

Following is the example of connecting with MySQL database "TESTDB"

```
#!/usr/bin/python

import MySQLdb

# Open database connection
```

```
db =
MySQLdb.connect("localhost","testuser","test123","TESTDB"
)

# prepare a cursor object using cursor() method

cursor = db.cursor()

# execute SQL query using execute() method.

cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.

data = cursor.fetchone()

print "Database version : %s " % data

# disconnect from server

db.close()
```

Output:-

While running this script, it is producing the following result.

```
Database version : 5.0.45
```

If a connection is established with the datasource, then a Connection Object is returned and saved into **db** for further use, otherwise **db** is set to None. Next, **db** object is used to create a **cursor** object, which in turn is used to execute SQL queries. Finally, before coming out, it ensures that database connection is closed and resources are released.

Creating Database Table:-

Once a database connection is established, we are ready to create tables or records into the database tables using execute method of the created cursor.

Example:-

Let us create Database table EMPLOYEE –

```
#!/usr/bin/python

import MySQLdb

# Open database connection

db =
MySQLdb.connect("localhost","testuser","test123","TESTDB"
)

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Drop table if it already exist using execute() method.

cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement

sql = """CREATE TABLE EMPLOYEE (

        FIRST_NAME  CHAR(20) NOT NULL,

        LAST_NAME  CHAR(20),

        AGE INT,

        SEX CHAR(1),
        INCOME FLOAT )"""
cursor.execute(sql)
# disconnect from server
db.close()
```

INSERT Operation:-

It is required when you want to create your records into a database table.

Example:-

The following example, executes SQL INSERT statement to create a record into EMPLOYEE table –

```
#!/usr/bin/python
import MySQLdb
# Open database connection
db =
MySQLdb.connect("localhost","testuser","test123","TESTDB"
)
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to INSERT a record into the database.
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
        LAST_NAME, AGE, SEX, INCOME)
        VALUES ('Rahul', 'Singh', 29, 'M', 25000)"""
try:
  # Execute the SQL command
  cursor.execute(sql)
  # Commit your changes in the database
  db.commit()
except:
  # Rollback in case there is any error
  db.rollback()
# disconnect from server
db.close()
```

Above example can be written as follows to create SQL queries dynamically –

```
#!/usr/bin/python
import MySQLdb
# Open database connection
db =
MySQLdb.connect("localhost","testuser","test123","TESTDB"
 )
```

```
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to INSERT a record into the database.
sql = "INSERT INTO EMPLOYEE(FIRST_NAME, \
       LAST_NAME, AGE, SEX, INCOME) \
       VALUES ('%s', '%s', '%d', '%c', '%d' )" % \
       ('Rahul', 'Singh', 29, 'M', 25000)
try:
  # Execute the SQL command
  cursor.execute(sql)
  # Commit your changes in the database
  db.commit()
except:
  # Rollback in case there is any error
  db.rollback()
# disconnect from server
db.close()
```

COMMIT Operation:-

Commit is the operation, which gives a green signal to database to finalize the changes, and after this operation, no change can be reverted back.

Here is a simple example to call commit method.

```
db.commit()
```

ROLLBACK Operation:-

If you are not satisfied with one or more of the changes and you want to revert back those changes completely, then use rollback() method.

Here is a simple example to call rollback() method.

```
db.rollback()
```

Disconnecting Database:-

To disconnect Database connection, use close() method.

```
db.close()
```

If the connection to a database is closed by the user with the close() method, any outstanding transactions are rolled back by the DB. However, instead of depending on any of DB lower level implementation details, your application would be better off calling commit or rollback explicitly.