PYTHON OOPs

Database Programming :

・Python Database API supports a wide range of database servers such as −

GadFly

mSQL

MySQL

PostgreSQL

Microsoft SQL Server 2000

Informix

Interbase

Oracle

・You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

・DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible.

This API include following :

・Importing the API module.

・Acquiring a connection with the database.

・Issuing SQL statements and stored procedures.

・Closing the connection.

・We would learn all the concepts using MySQL, so let us talk about MySQLdb module.

What is MySQLdb?

· 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 To Install MySQLdb?

· Before proceeding, you make sure you have MySQLdb installed on your machine.

```
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
```

Database Connection :

Before connecting to a MySQL database,see following:

· 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

Example :

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

```
import MySQLdb

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

cursor = db.cursor()

cursor.execute("SELECT VERSION()")

data = cursor.fetchone()

print ("Database version : %s " % data)

db.close()
```

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

```
import MySQLdb

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

cursor = db.cursor()

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

sql = """CREATE TABLE EMPLOYEE (

         FIRST_NAME CHAR(20) NOT NULL,

         LAST_NAME CHAR(20),

         AGE INT,

         SEX CHAR(1),

         INCOME FLOAT)"""

cursor.execute(sql)

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.

```
import MySQLdb

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

cursor = db.cursor()

sql = "INSERT INTO EMPLOYEE(FIRST_NAME, LAST_NAME, AGE, SEX, INCOME)
VALUES('%s','%s','%d','%c','%d') % ('Mac','Mohan',20,'M',2000)"

try:

        cursor.execute(sql)

        db.commit()

except:

        db.rollback()

db.close()
```

READ operation :

· 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.

```
import MySQLdb

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

cursor = db.cursor()

sql = "SELECT * FROM EMPLOYEE WHERE INCOME > '%d'" % (1000)

try:

        cursor.execute(sql)

        results = cursor.fetchall()

        for row in results:

                fname = row[0]

                lname = row[1]

                age = row[2]

                sex = row[3]

                income = row[4]

                print ("fname=%s,lname=%s,age=%d,sex=%s,income=%d" % (fname,
lname, age, sex, income))

except:

        print ("Error: unable to fetch data")

db.close()
```

UPDATE Operation : UPDATE Operation on any database means to update one or more records, which are already available in the database.

```
import MySQLdb

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

cursor = db.cursor()

sql = "UPDATE EMPLOYEE SET AGE = AGE + 1 WHERE SEX = '%c'" % ('M')

try:

        cursor.execute(sql)
```

```
        db.commit()
    except:
        db.rollback()
    db.close()
```

DELETE Operation : DELETE operation is required when you want to delete some records from your database.

```
import MySQLdb
db = MySQLdb.connect("localhost","testuser","test123","TESTDB")
cursor = db.cursor()
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
    cursor.execute(sql)
    db.commit()
except:
    db.rollback()
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.

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.

Disconnecting Database :

- To disconnect Database connection, use close() method.