Python-Data Base Conmmunications(PDBC):-
It stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. RDBMS store and manage huge volume of data.

You can choose the right database for your application. Python Database API supports a wide range of database servers:p
1. MySQL 2. PostgreSQL 3. Microsoft SQL Server
4. Informix 5. Oracle 6. Sybase 7. SQLite
8. MongoDB 9.AnyBigData

What Can SQL do?
1. SQL can execute queries, retrieve data, insert, update and delete records.
2. SQL can create new databases tables, SP, views and set permissions

Most Important SQL Commands:
SELECT, UPDATE, DELETE, INSERT INTO, ALTER DB, Table CREATE TABLE, DROP TABLE, INDEX..!!

Step1: Install PYTHON
Step2: Install MySQL (8.0 or later)
https://dev.mysql.com/downloads/mysql/
https://sourceforge.net/projects/mysql-python/files/mysql-python/1.2.3/MySQL-python-1.2.3.win32-py2.7.msi/download

Step3: MySQL Driver
$pip install mysql-connector
conda install -c anaconda mysql-cnnector-python

Test MySQL Connector
To test if the installation was successfull,
Example:
```
import mysql.connector
print(dir(mysql.connector))
```

Syntax:
```
mydb = mysql.connector.connect(host="localhost",
  user="yourusername", password="yourpassword",database="YourDBName")
```

Syntax:
```
mydb = mysql.connector.connect(host="127.0.0.1",
  user="yourusername", password="yourpassword",database="YourDBName")
```

Create Connection
Start by creating a connection to the database.
Use the username and password from your MySQL database:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",password="root")
print("MySQL-Server-ConnectedSuccessfully")
```

Python MySQL Create Database:
To create a database in MySQL, use the "CREATE DATABASE" statement:

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",password="root")
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE STUDENTS")
print("DataBaseCreatedSuccessfully")
```

NOTE:
A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.

Check if Database Exists
Check if a database exist by listing all databases, by using the "SHOW DATABASES" statement:

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root")
mycursor = mydb.cursor()
mycursor.execute("SHOW DATABASES")
for data in mycursor:
  print(data)
```

To use particular database:

```
mysql> use student
OR
mysql> connect student
```

Python MySQL Create Table
Creating a Table
To create a table in MySQL, use the "CREATE TABLE" statement.

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE Students (name VARCHAR(255), address
VARCHAR(255))")
print("Table Created Successfully")
```

```
mysql> show tables;
mysql> desc student;
```

Check if Table Exists
You can check if a database exist by listing all tables in your database by using the "SHOW TABLES" statement:

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
```

```
mycursor = mydb.cursor()
mycursor.execute("SHOW TABLES")
for data in mycursor:
  print(data)
```

Python MySQL Insert Into Table
To fill a table in MySQL, use the "INSERT INTO" statement.

Example:

```
import mysql.connector
mydb=mysql.connector.connect(host="127.0.0.1",
                             user="root",
                             passwd="root",
                             database='student')
mycursor=mydb.cursor()
mycursor.execute("insert into students values(1,'SARA')")
print("DataInsertedSuccessfully")
```

NOTE:
The above script executed successfully but data not saved in the table, that time we must 'commit' the table..!!

Example:

```
import mysql.connector
mydb=mysql.connector.connect(host="127.0.0.1",
                             user="root",
                             passwd="root",
                             database='std')
mycursor=mydb.cursor()
mycursor.execute("insert into emp values(1,'SARA')")
mydb.commit()
print("DataInsertedSuccessfully")
```

Insert Data in Multiple Rows
To insert multiple rows into a table, use the executemany() method.

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
sql = "INSERT INTO students (name, address) VALUES (%s, %s)"
val = [
  ('Peter', 'Lowstreet 4'),
  ('Amy', 'Apple st 652'),
  ('Hannah', 'Mountain 21'),
  ('Michael', 'Valley 345'),
  ('Sandy', 'Ocean blvd 2'),
  ('Betty', 'Green Grass 1'),
  ('Richard', 'Sky st 331'),
  ('Susan', 'One way 98'),
  ('Vicky', 'Yellow Garden 2'),
  ('Ben', 'Park Lane 38'),
  ('William', 'Central st 954'),
  ('Chuck', 'Main Road 989'),
  ('Viola', 'Sideway 1633')
```

```
]
mycursor.executemany(sql, val)
mydb.commit()
print("Data Inserted Successfully")
```

Python MySQL Select From:
To select from a table in MySQL, use the "SELECT" statement:
Select all records from the "students" table, and display the result:

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM students")
for data in mycursor:
  print(data)
```

Selecting Columns
To select only some of the columns in a table, use the "SELECT" statement followed by the column name(s):

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
mycursor.execute("SELECT name, address FROM students")
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

Using the fetchone() Method:
If you are only interested in one row, you can use the fetchone() method. It will return the first row.

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM students")
myresult = mycursor.fetchone()
print(myresult)
```

Python MySQL Where:
Select With a Filter, When selecting records from a table, you can filter the selection by using the "WHERE" statement:

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
```

```
mycursor = mydb.cursor()
sql = "SELECT * FROM students WHERE address ='Park Lane 38'"
mycursor.execute(sql)
myresult = mycursor.fetchall()
for data in myresult:
  print(data)
```

Python MySQL Order By
The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword.

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
sql = "SELECT * FROM students ORDER BY name"
mycursor.execute(sql)
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

ORDER BY DESC: Use the DESC keyword to sort the result in a descending order
Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
sql = "SELECT * FROM students ORDER BY name DESC"
mycursor.execute(sql)
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

Python MySQL Delete From By
You can delete records from an existing table by using the "DELETE FROM" statement:

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
sql = "DELETE FROM students WHERE address = 'Mountain 21'"
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) deleted")
```

Python MySQL Update Table:
You can update existing records in a table by using the "UPDATE" statement:
Example:

```
import mysql.connector
```

```
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
sql = "UPDATE students SET address = 'Canyon 123' WHERE address =
'Valley 345'"
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) affected")
```

Python MySQL Drop Table
You can delete an existing table by using the "DROP TABLE" statement
Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
sql = "DROP TABLE customers"
mycursor.execute(sql)
```

Drop Only if Exist
If the the table you want to delete is already deleted, or for any other reason does not exist, you can use the IF EXISTS keyword to avoid getting an error.

Example:

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
mycursor = mydb.cursor()
sql = "DROP TABLE IF EXISTS customers"
mycursor.execute(sql)
```

SQLite3
It can be integrated with Python using sqlite3 module, It is default along with Python-2.5.x. It is famous for its great feature zero-configuration, which means no complex setup or administration is needed.

Connect To Database
Example:

```
import sqlite3
conn = sqlite3.connect('test.db')
print("Opened database successfully")
```

Create a Table

```
import sqlite3
conn = sqlite3.connect('test.db')
conn.execute('''CREATE TABLE COMPANY
         (ID INT PRIMARY KEY     NOT NULL,
         NAME           TEXT    NOT NULL,
         SALARY         REAL);''')
print("Table created successfully")
conn.close()
```

INSERT Operation

```
import sqlite3
conn = sqlite3.connect('test.db')
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (1, 'Paul', 32, 'California', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");
conn.commit()
print("Records created successfully")
conn.close()
```

ORACLE:
It is an ORDBMS developed and marketed by Oracle Corporation.

Following are the four Editions of the Oracle:
Enterprise Edition: It is the most robust and secure edition.

Standard Edition: It provides the base functionality for users that do not require Enterprise Edition's robust package.

Express Edition (XE): It is the lightweight, free and limited Windows and Linux edition.

Oracle Lite: It is designed for mobile devices.

Step1: Install PYTHON
Step2:Intall Oracle Any Edition (XE Prefered)
Step3:Install Oracle Driver

```
$pip install cx_Oracle
```

Update Oracle

```
$pip install -U cx_Oracle
```

To View Version of Oracle

```
$pip show cx_Oracle
```

Syntax:

```
import cx_Oracle
```

Example:

```
import cx_Oracle
print(dir(cx_Oracle))
```

Example:

```
import cx_Oracle
con=Oracle.connect('scott/tiger@localhost')
cursor=con.cursor()
cursor.execute("select * from students")
print(cursor.fetchone())
cursor.close()
con.close()
```

Example:

```
import cx_Oracle
con=Oracle.connect('scott/tiger@localhost')
cursor=con.cursor()
```

```
cursor.execute("select * from students")
records=cursor.fetchall()
for record in records:
   print("Student Name: ",record[0])
   print("Address: ",record[1])
   print()
cursor.close()
con.close()
```

Example:

```
import cx_Oracle
con=Oracle.connect('scott/tiger@localhost')
cursor=con.cursor()
cursor.execute("select * from students")
records=cursor.fetchall()
print(records)
```

Example:

```
import mysql.connector
con =
mysql.connector.connect(host="localhost",user="root",passwd="root",
database='student')
cursor = con.cursor()
cursor.execute("SELECT * FROM students")
records = cursor.fetchall()
print(records)
```