mysql

- **MySQL** :
- MySQL is an open-source relational database management system(RDBMS) based on Structured Query Language (SQL). It is currently developed and managed by Oracle Corporation, since acquisition by Sun Microsystems on 27 January 2010, which had itself acquired MySQL in 2008. It was initially released internally on 23 may 1995 by David Axmark, Allan Larsson and Michael “Monty” Widenius of MySQL AB from Sweden, though development for personal use by Widenius and Axmark began in 1994. Since acquisition of Sun by Oracle, the original developers left MySQL to work on a fork known as MariaDB. It is widely being used in many small and large scale industrial applications and capable of handling a large volume of data.
- .

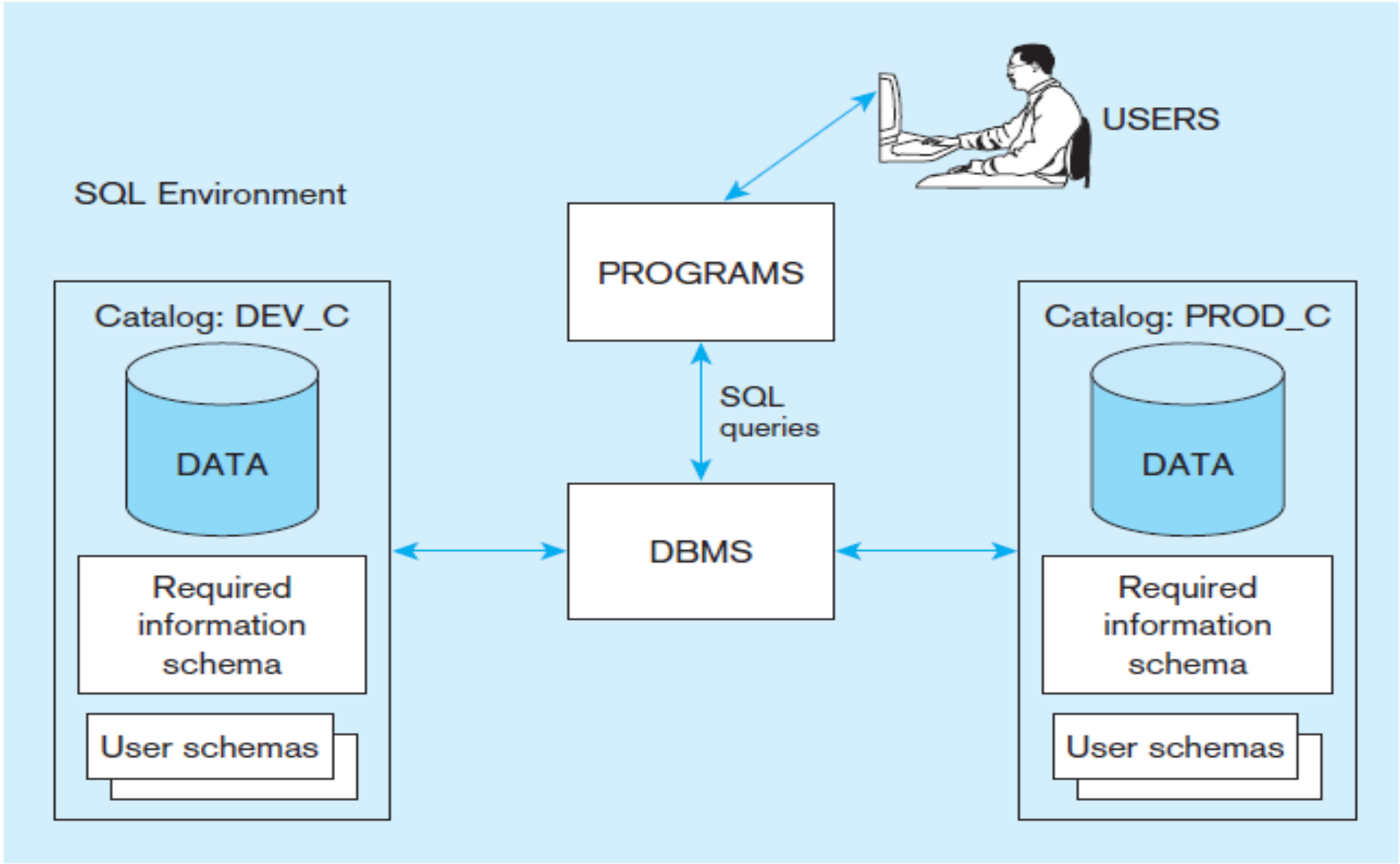
sqlite

- SQLite is a software library that provides a relational database management system(RDBMS). It was designed by D. Richard Hipp on August 2000. The design goals of SQLite were to allow the program to be operated without installing a database management system(DBMS) or requiring a database administrator. The lite in SQLite means light weight in terms of setup, database administration, and required resource

What is MySQL

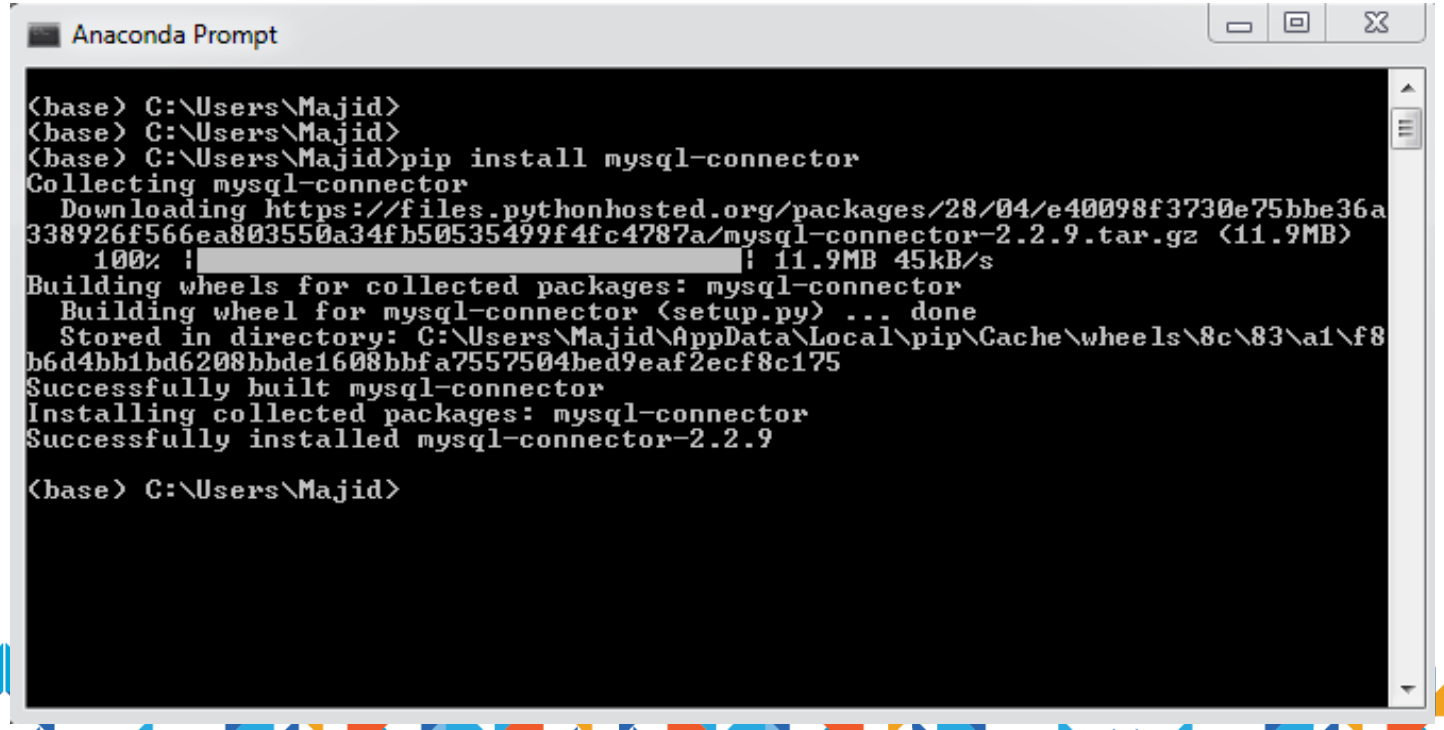
- MySQL is an Open-Source database and one of the best type of RDBMS (Relational Database Management System).
- This DBMS is used to manage databases systems and run different SQL (Structured Query Language) queries into the Databases.

SQL Environment

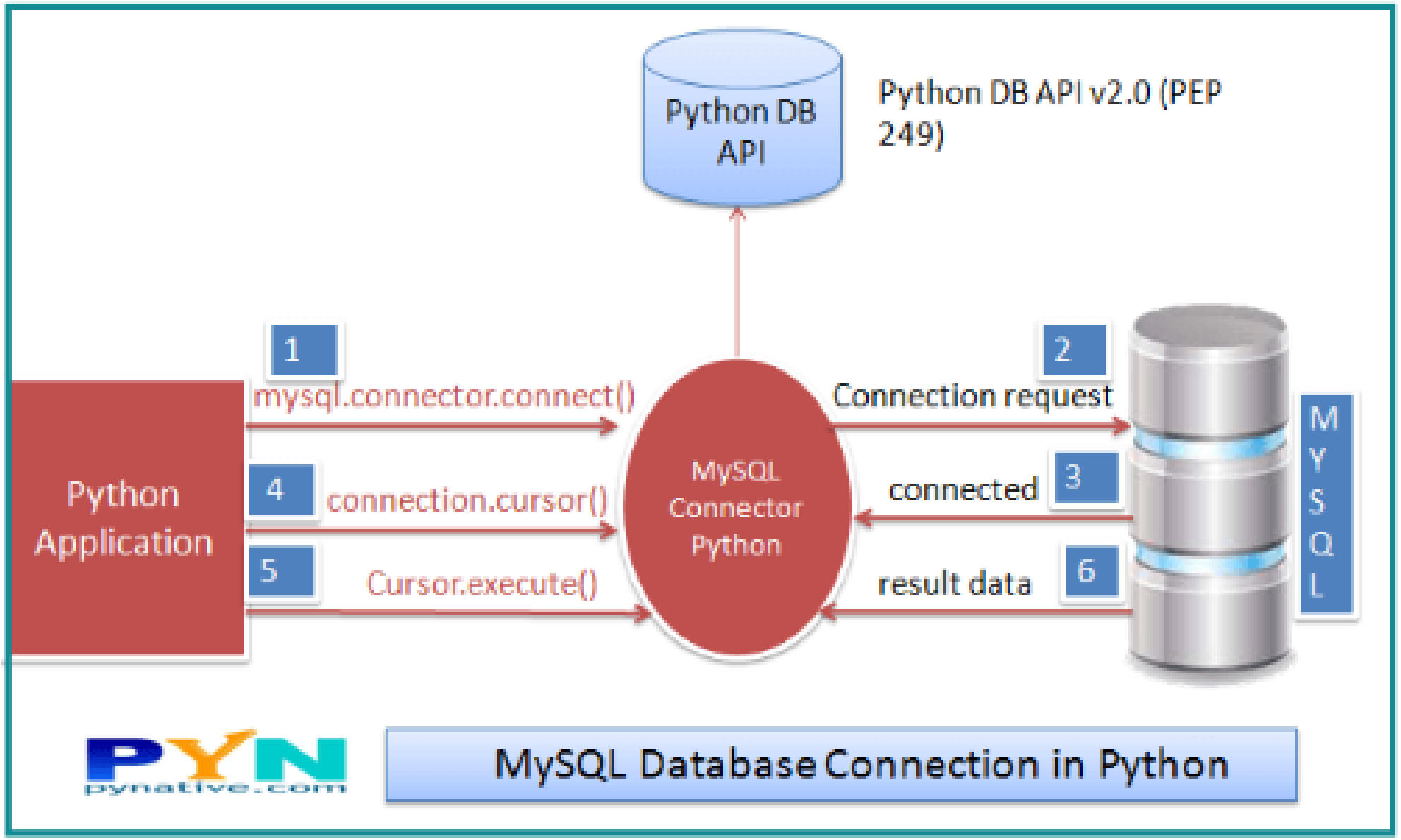


Required information to connect to database

- Install MySQL Connector Python using pip, or download installer, if necessary.
- Find it at: dev.mysql.com: or pip install mysql-connector-python-rf
- pip3 install mysql-connector-python
-



Scenario


Python DB API
Python DB API v2.0 (PEP 249)
1
mysql.connector.connect()
2
Connection request
Python Application
MySQL Connector Python
MYSQL
4
connection.cursor()
connected
3
5
Cursor.execute()
result data
6
PYN
pynative.com
MySQL Database Connection in Python

Explaining steps

- Use the mysql.connector.connect() method of MySQL Connector Python with required parameters to connect MySQL.
- Use the connection object returned by a connect() method to create a cursor object to perform Database Operations.
- The cursor.execute() to execute SQL queries from Python.
- Close the Cursor object using a cursor.close() and MySQL database connection using connection.close() after your work completes.
- Catch Exception if any that may occur during this process.

How to connect the Python with the Database?

- To connect the python program with the database we need the following lines of code.

```
1  import mysql.connector
2
3  db_connect=mysql.connector.connect(host='localhost', user='root', passwd='Afghan.123')
4
```

- First, import the connector module from the mysql package.
- Try to connect with the database, with your predefined credentials.
- Here, host computer is : localhost, username is: ‘root’, passwd=‘MyPassword’
- We also can define the database attribute as: database=‘School’

Handling the Exception

- There are many problems happen during the database connectivity, in turn, it is required to catch different errors.

- from mysql.connector import Error
- try:
  - Code here
- except Error as e:
  - Message here +e

Create the database and see existing databases?

- From SQL we know that if we want to create a table in SQL, we need to provide the specific queries for this.
- Example:
  - CREATE DATABASE ‘DB_NAME’
  - ‘SHOW DATABASES’

```
('information_schema',)
('mypython',)
('mysql',)
('performance_schema',)
('sakila',)
('sys',)
('testing',)
('world',)
```

Output

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword"
)
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE testing")
```

```
mycursor.execute("SHOW DATABASES")
for x in mycursor:
  print(x)
```

Selecting DataBase

- Try connecting to the database “testing":
- `import mysql.connector`

```
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database=“testing"
)
```

Create a Table in MySQL with Python

- Let's create a simple table "student" which has two columns.
- SQL Syntax:
- CREATE TABLE student (id INT, name VARCHAR(255))

```
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
mycursor.execute("CREATE TABLE students (id INT, name VARCHAR(255), address
VARCHAR(255))")
mycursor.execute("SHOW TABLES")

for x in mycursor:
  print(x)
```

Output:

('customers',)
('students',)

ALTER table in MySQL with Python

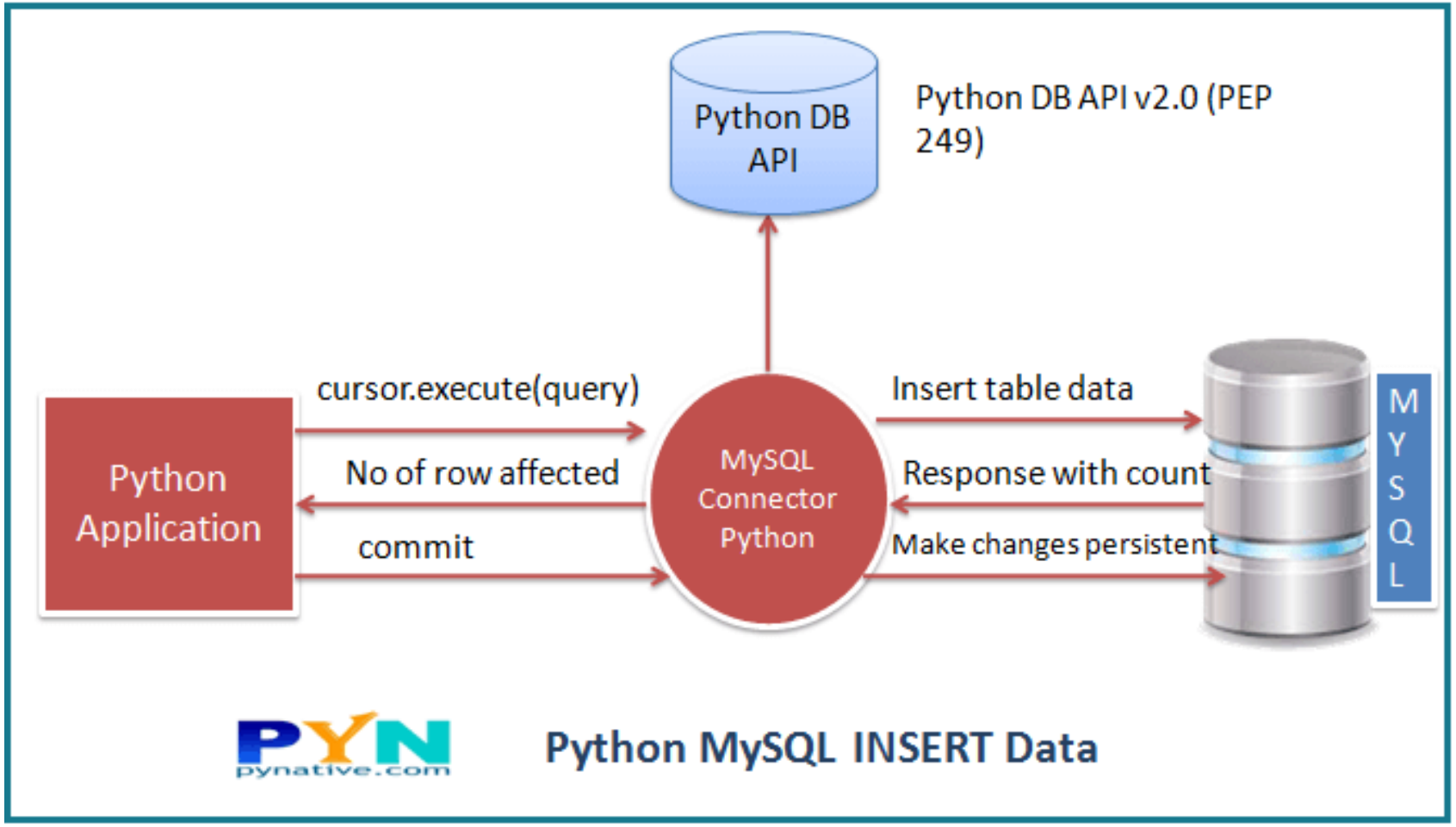
- Alter command is used for modification of Table structure in SQL. Here we will alter **Student** table and add a primary key to the **id** field.

```
import mysql.connector
mydb =
mysql.connector.connect(host="localhost",user="yourusername",passwd="yourpassword",database="m
ydatabase")
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE customers (id INT AUTO_INCREMENT, name VARCHAR(255), address
VARCHAR(255))")
```

```
mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY")
```

Insert Data


Python DB API
Python DB API v2.0 (PEP 249)
Python Application
cursor.execute(query)
No of row affected
commit
MySQL Connector Python
Insert table data
Response with count
Make changes persistent
MYSQL
PYN
pynative.com
Python MySQL INSERT Data

Insert data

- We use insert operation in MySQL Database table which we already created.
- We will insert data into STUDENT table

```
1  import mysql.connector
2
3
4  connection=mysql.connector.connect(host='localhost', user='root', passwd='Afghan.123',
5                                     database='NEWDB2')
6
7  mydb=connection.cursor()
8  query="insert into student values (%s, %s)"
9  data=(11, 'Ahmad')
10 mydb.execute(query, data)
11
12 connection.commit()
13 print(mydb.rowcount, 'Rows Inserted')
14
15
16
```

```
1 Rows Inserted
```

How to use Place Holders

- Place holders are used to get inputs from the python
- We can define a placeholder as “%s”
- Example:

```
22  db_cursor = db_connect.cursor()
23  student_sql_query = "INSERT INTO student(id,name) VALUES(%s, %s)"
24
25  db_cursor.execute(student_sql_query,(20,'Saeed',))
26
27  db_connect.commit()
```

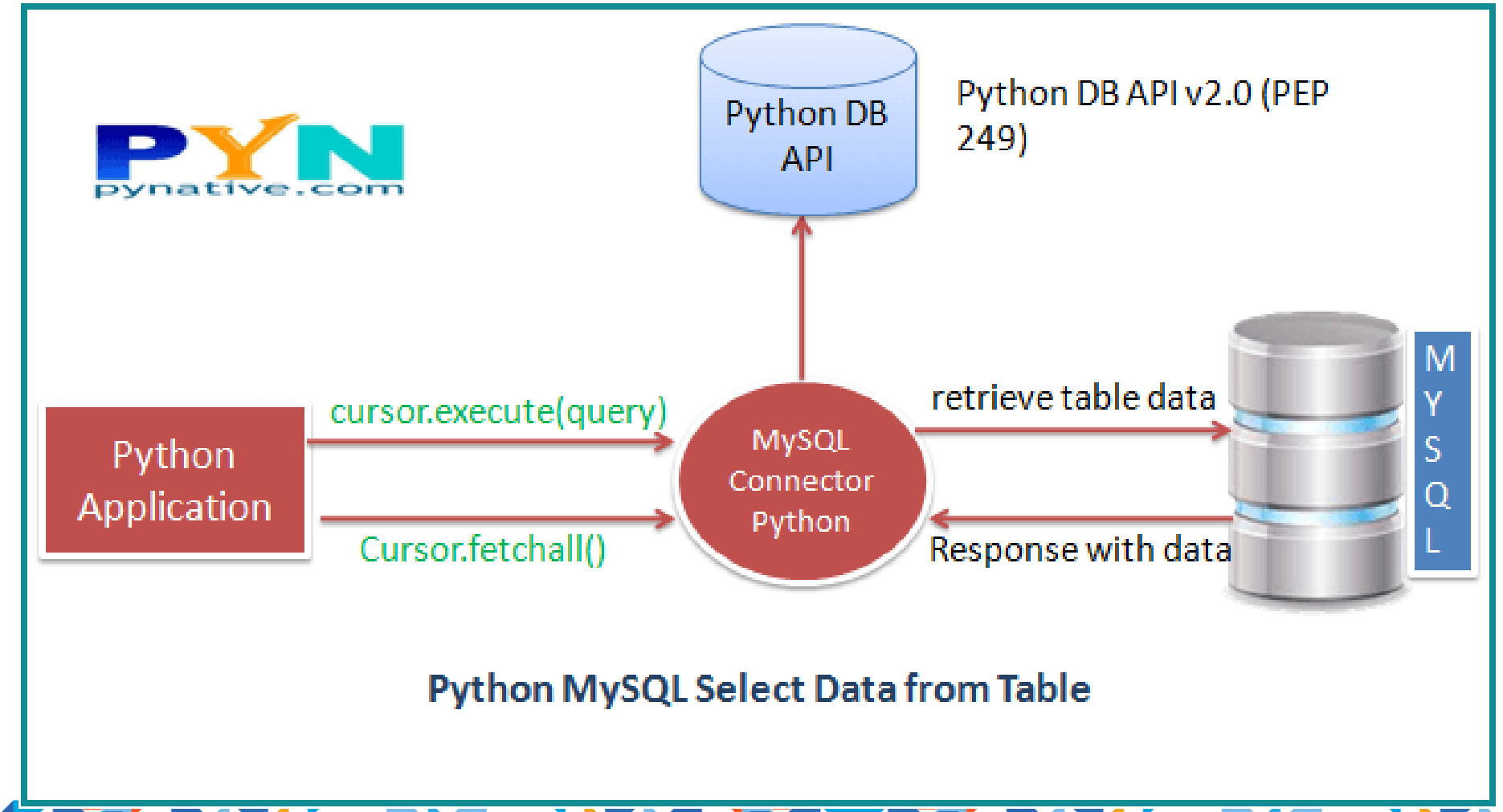
Insert multiple data at once

- Sometimes it is required to insert many rows of information by single query
- In this case we use the executemany () function

```
1  import mysql.connector
2
3
4  connection=mysql.connector.connect(host='localhost', user='root', passwd='Afghan.123',
5                                     database='NEWDB2')
6
7  mydb=connection.cursor()
8  query="insert into student values (%s, %s)"
9
10 mylist=[(12, 'Ahmad'),(13,'Naeem')]
11 mydb.executemany(query, mylist)
12
13 connection.commit()
14 print(mydb.rowcount, 'Rows Inserted')
15
16
17
```

```
2 Rows Inserted
```

Read data from



READ data from DB

- Select Query is used to get the data from the database.

```
36  db_cursor = db_connect.cursor()
37  student_sql_query="SELECT * FROM student"
38
39  db_cursor.execute(student_sql_query)
40  record=db_cursor.fetchall()
41
42  for r in record:
43      print(r)
```

- To get specific data filed, call the ‘r[0]’ ‘r[1]’
- fetchmany()
- fetchone()

```
mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

Output:

```
(1, 'John', 'Highway 21')
(2, 'Peter', 'Lowstreet 27')
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
```

Delete data

- DELETE FROM table WHERE id=98

```
31  db_cursor = db_connect.cursor()
32  student_sql_query = "DELETE FROM STUDENT WHERE id=01"
33
34  db_cursor.execute(student_sql_query)
35
36  db_connect.commit()
37
38  print(db_cursor.rowcount, "Record Deleted")
```

```
mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address =
'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s)
deleted")
```

Update data of DB

```
mycursor = mydb.cursor()
sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) affected")
```

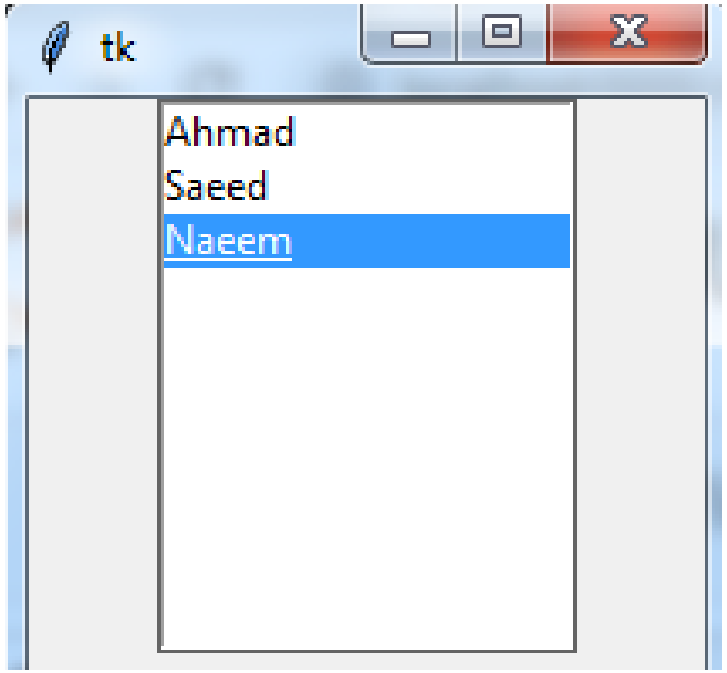
Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

```
1  sql_update_query = "Update Student set name = 'Saeed' where id = 1"
2      cursor.execute(sql_update_query)
3      connection.commit()
4      print("Record Updated successfully ")
```

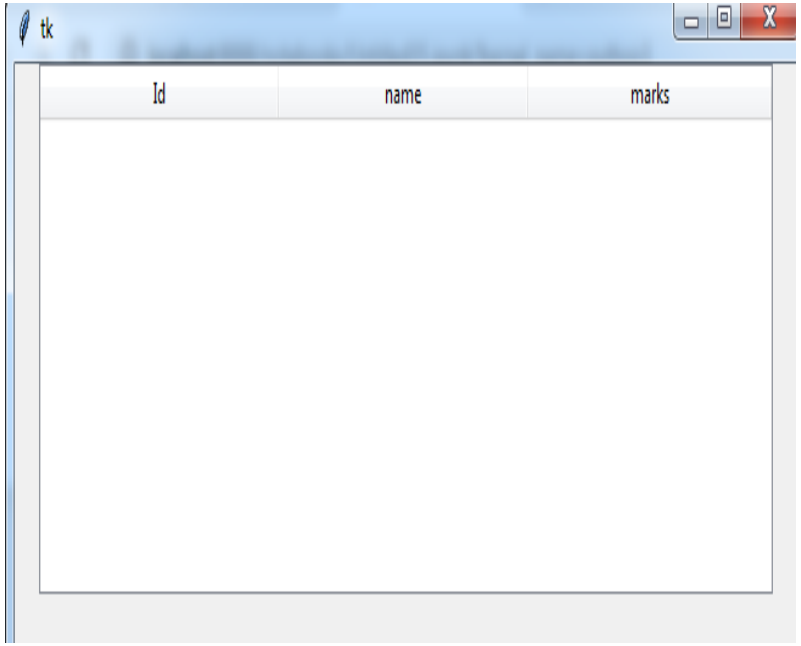
ListBox Widget

```
6  box=Listbox(root)
7  box.insert(1,'Ahmad')
8  box.insert(2,'Saeed')
9  box.insert(3,'Naeem')
10 box.pack()
```



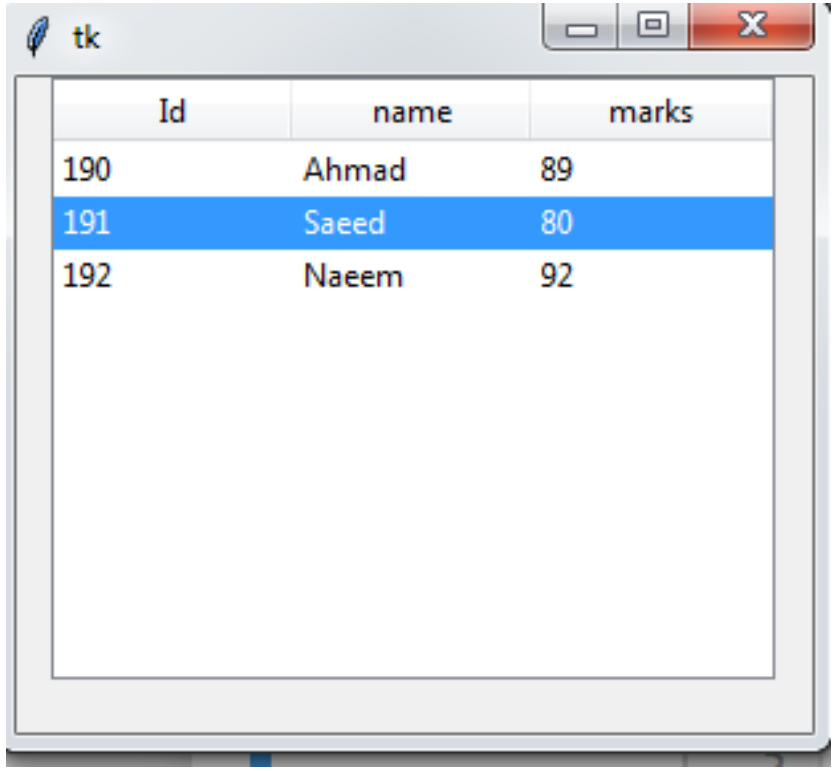
TreeView

```
1  from tkinter import *
2  import tkinter.ttk as ttk
3
4  root=Tk()
5
6  root.geometry("300x400")
7  cols=('Id','name', 'marks')
8  tree=ttk.Treeview(root, columns=cols,show='headings')
9  tree.pack()
10 for col in cols:
11     tree.heading(col, text=col)
12 tree.column("Id",width=90)
13 tree.column ("name", width=90)
14 tree.column("marks", width=90)
```



Insert data into table

```
15
16  tree.insert("", "end", values=("190","Ahmad","89"))
17  tree.insert("", "end", values=("191","Saeed","80"))
18  tree.insert("", "end", values=("192","Naeem","92"))
19
```

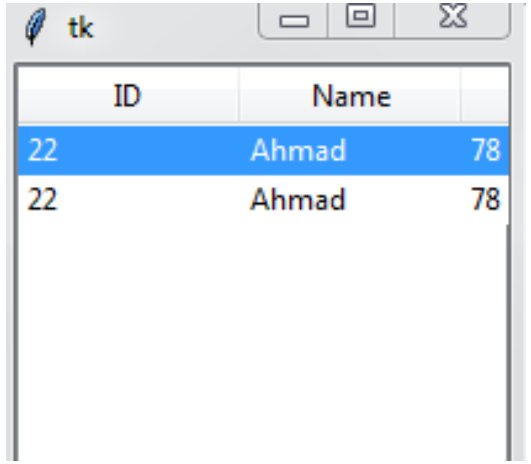


Get the selected items:

```python
from tkinter import *
from tkinter.ttk import *
root=Tk()
def select(a):
    b=tree.focus()
    dic=tree.item(b)
    print(dic["values"])

Col=("ID", "Name","Marks")
tree=Treeview(root, column=Col, show="headings")
tree.pack()
for col in Col:
    tree.heading(col, text=col)
tree.column("ID", width=90)
tree.column("Name", width=90)
tree.column("Marks", width=90)
tree.bind('<Button-1>', select)

tree.insert ("","end", values=("22","Ahmad", "78"))
tree.insert ("","end", values=("22","Ahmad", "78"))
root.geometry("200x300")
root.mainloop()
```




```
[22, 'Ahmad', 78]
```

Images

pip3 install --upgrade Pillow

```
1  from tkinter import *
2  from PIL import ImageTk, Image
3
4  root = Tk()
5  root.geometry("300x300")
6  img = ImageTk.PhotoImage(Image.open("123.jpg"))
7  panel = Label(root, image = img, width=100, height=200)
8  panel.pack()
9  root.mainloop()
```

Resizing

```
1  from tkinter import *
2  from PIL import ImageTk, Image
3
4  root = Tk()
5  root.geometry("300x300")
6  img1=Image.open("123.jpg")
7  resized=img1.resize((200,200))
8  img = ImageTk.PhotoImage(resized)
9  panel = Label(root, image = img, width=200, height=200)
10 panel.pack()
11 root.mainloop()
```

Thank You!