Database Access in Python

The EasyLearn Academy

9662512857

What is database access / database connectivity in python

- when we connect our python application with any RDBMS package like mysql, mssql, oracle, to execute any sql queries like insert, update, delete, select query it is called database connectivity.
- Python can connect with any available DBMS & RDBMS package available in market.
- All you need to do is to get driver for particular (r)dbms package, install it, import in your python code to start using it.
- In our examples we will connect python with mysql RDMBS.
- to do this exercise, mysql RDBMS package & mysql driver must be installed in your PC laptop.

Install MySQL Driver

- Python needs a MySQL driver to access the MySQL database.
- In this course we will use the driver "MySQL Connector".
- We recommend that you use PIP to install "MySQL Connector".
- It is possible that PIP is already installed in your Python environment.
- To check open command line to the location of PIP, and type the following:

pip install mysql-connector

What is pip?

- Python supports third-party libraries and frameworks that you can install to prevent having to reinvent the wheel with every new project.
- You can find these on a central repository called PyPI (Python Package Index).
- To download, install, and manage these packages , many Python developers rely on a special tool called PIP for Python (or Python PIP).
- PIP is a recursive acronym that stands for “PIP Installs Packages” or “Preferred Installer Program”.
- It’s a command-line utility that allows you to install, reinstall, or uninstall PyPI packages with a simple and straightforward command: **pip**.

Some useful commands of pip

- Once PIP is ready, you can start installing packages from PyPI:
- **pip install package-name**

- To install a specific version of a package instead of the latest version:
- **pip install package-name==1.0.0**

- To see details about an installed package:
- **pip show package-name**

- To list all installed packages:
- **pip list**

- To list all outdated packages:
- **pip list --outdated**

- To upgrade an outdated package:
- **pip install –upgrade package-name**

  OR
- **pip install --upgrade --force-reinstall package-name**

- To uninstall package:
- **pip uninstall package-name**

Create Connection

```
import mysql.connector
database = connector.connect(host="localhost",user="root",passwd="",da
    tabase="py9",port=3308)
print ("Connection Established")
#If the database does not exist, you will get an error.
```

Insert record into table

- #create cursor
- mycursor = database.cursor()

- #build query
- sql = "INSERT INTO **category** (title,photo,status) VALUES (%s,%s,%s)"
- #create list whose size must be same no of placeholder in sql
- values = ["Books 2","photo.jpg",1];
- #execute query
- mycursor.execute(sql,values)
- #It is required to make the changes, otherwise no changes are made to the table.
- database.commit()
- print(mycursor.rowcount, "record inserted.")

Delete record

```
mycursor = database.cursor()
sql = "DELETE FROM category WHERE id=%s"
values = [1];
mycursor.execute(sql,values)
database.commit()
print(mycursor.rowcount, "record(s)
  deleted")
```

Update record

- mycursor = database.cursor()
- sql = “update category set title = %s where id=%s’"
- values = [“Rich dad poor dad”,2];
- mycursor.execute(sql,values)
- database.commit()

select

```
mycursor = database.cursor(dictionary=True)
sql = “SELECT * FROM category”
mycursor.execute(sql)
single_row= mycursor.fetchone() #fetch one row as list
print(single_row) # print single record
#print all record
table = mycursor.fetchall() #fetch all rows as 2d list
for row in table:
  msg = "id = " + row[‘id’] + " title = " + row[‘title’]
  + " photo =" + row[‘photo’];
  #concatenate string and variables
  print(msg) #print variable
```