



Cognizant | Academy

Business Focused. Learner Centric.

Introduction to Python

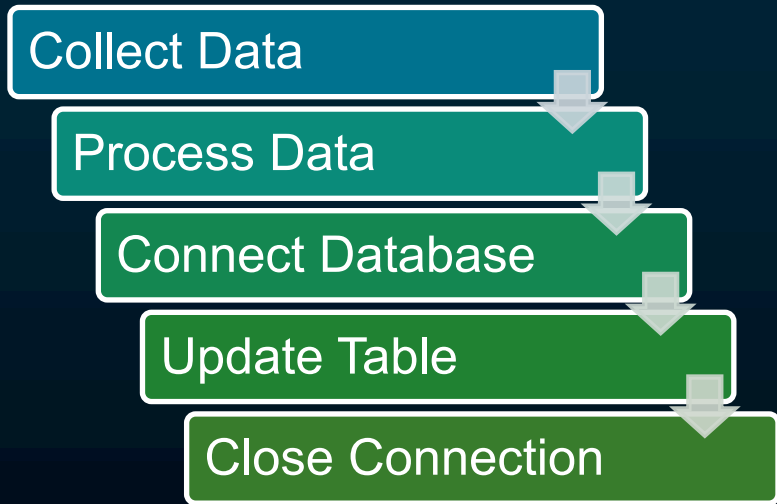
Basics of Python Programming Language

Day 4 | Duration: 2 Hours





The Zen of Python

“Simple is better than complex. Complex is better than complicated.”

- The Zen of Python (by Tim Peters)



Terminal Objectives

-  To learn how to connect to database (MySQL & SQLite) using Python.
-  We will learn to create Database. Compare syntax with other Database packages.
-  To learn how to execute SQL queries to perform database transactions like DDL (CREATE) & DML (SELECT, UPDATE, INSERT, DELETE) etc. in Python.
-  To learn how to write a program in Python to deal with Database.

Why to use Database?

Benefits

Processing queries and object management

Controlling redundancy and data inconsistency

Efficient memory management and indexing

Concurrency control and transaction management

Data security

Integrated data

Topics to Cover

Database Programming using Python

-  Connect to a Database

-  Create Database

-  Create Table

-  Select Record

-  Insert Record

-  Update Record

-  Delete Record

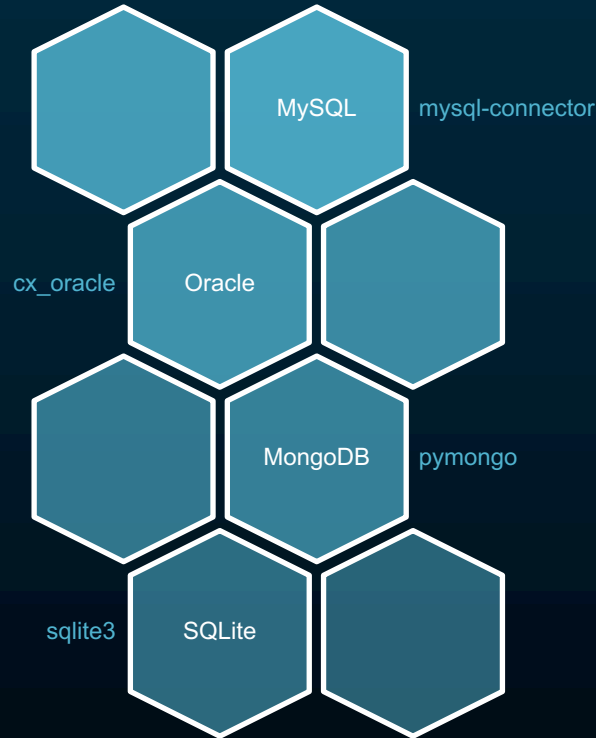
Database Programming using Python

Database Programming using Python

- 🐍 In many programming languages, the developer needs to take care of the open and closed connections of the database, to avoid further exceptions and errors. But in Python, these things are taken care of.
- 🐍 Python supports relational database systems.
- 🐍 Python database APIs are compatible with various databases, so it is very easy to migrate and port database application interfaces.

Database Programming using Python

Python Database API supports a wide range of databases.



Database Programming using Python

DB-API (SQL-API) for Python

- Python **DB-API** is independent of any database engine, which enables you to write Python scripts to access any database engine.
- DB API implementation for *MySQL* is MySQLdb, mysql-connector.
- For *PostgreSQL*, it supports psycopg, PyGresQL & pyPgSQL.
- For *Oracle* `cx_oracle2` and `cx_oracle` are very famous.
- For *MongoDB* its PyMongo.
- Pydb2 is the DB-API implementation for *DB2* & for *SQLite* its `sqlite3`.
- Python's DB-API consists of **connection** objects, **cursor** objects, **standard exceptions** and some other module contents.

Database Programming using Python

- 🐍 **Connection** objects create a connection with the database and these are further used for different transactions. These connection objects are also used as representatives of the database session.
- 🐍 **Cursor** is one of the powerful features of SQL. These are objects that are responsible for submitting various SQL statements to a database server.
- 🐍 **Exception handling** is very easy in the Python DB-API module. We can place warnings and error handling messages in the programs.
- 🐍 Python DB-API has various options to handle this, like *Warning*, *InterfaceError*, *DatabaseError*, *IntegrityError*, *InternalError*, *NotSupportedError*, *ProgrammingError* & *OperationalError*.

Database Programming using Python

| | |
|--------------------------|-----------------------------------------------------------------------|
| <i>Warning</i> | If there is any warning generated during SQL execution |
| <i>DatabaseError</i> | Any transaction failure or connectivity error |
| <i>InterfaceError</i> | Specially for type mismatch, wrong query syntax |
| <i>IntegrityError</i> | If we try to enter duplicate records in the database |
| <i>InternalError</i> | Some unexpected thing like disconnection, access issue |
| <i>NotSupportedError</i> | Some operation that is not supported by the API |
| <i>ProgrammingError</i> | If there are any programming errors like duplicate database creations |
| <i>OperationalError</i> | If there are any operation errors like no databases selected |

Database Programming using Python

- 🐍 **SQLite** is a simple database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM.
- 🐍 It was developed for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.
- 🐍 **SQLITE** is “quite” fast, even though it uses a simple file. it can be used for large databases as well. If you want to use SQLite, you have to import the module *sqlite3*.

Database Programming using Python

Connect to a SQLite Database: `sqlite3.connect(<database>)`

```
>>> import os, sqlite3
>>> dbPath = "D:/Academy"
>>> dbName = "MyDatabase.db"
>>> objConnection = sqlite3.connect(os.path.join(dbPath, dbName))
>>>
```

Connecting to SQLite

```
>>> import mysql.connector
>>> pUser = "admin"
>>> pPass = "P@55w0rD"
>>> pDBName = "MyTestDB"
>>> OBJCONN = mysql.connector.connect(user=pUser, password=pPass,
                                     database=pDBName)
```

Connecting to MySQL

```
>>> import cx_Oracle
>>> pUser = "admin"
>>> pPass = "P@55w0rD"
>>> pHost = '127.0.0.1:/orcl:'
>>> pDBName = "stageDB"
>>> con = cx_Oracle.connect(pUser, pPass, pHost + pDBName)
```

Connecting to Oracle

Database Programming using Python

```
mysqlTest.py
1  import mysql.connector as MySQL
2
3  class DBHandler:
4      def __init__(self, pConnectParam):
5          self.Username = pConnectParam.get('user')
6          self.Password = pConnectParam.get('pass')
7          self.DBName = pConnectParam.get('db')
8          self.Host = pConnectParam.get('host')
9          self.Connect = None
10         self.dbConnect() # Invoke dbConnect() function
11
12     # Connects to the Database and set the connection object
13     def dbConnect(self):
14         try:
15             self.Connect = MySQL.connect(user = self.Username,
16                                         password = self.Password,
17                                         host = self.host,
18                                         database = self.DBName)
19             print('\nConnected to {0} successfully!'.format(self.DBName))
20         except MySQL.Error as e:
21             print('\nMySQLDBError: Failed to connect!\nSee details: ' + str(e))
22         except:
23             print('\nError: Not able to connect due to unknown error!')
```

The `connect()` function creates a connection to the MySQL server and returns a `MySQLConnection` object.

The `__init__()` function of our `DBHandler` class initializes the connection parameters. The `dbConnect()` function connects to the database and set the `self.Connect` attribute with the connection object.

Database Programming using Python

```
25     # Creates table using SQL
26     def createTable(self, pTable):
27         try:
28             vSQLTxt = 'CREATE TABLE IF NOT EXISTS {0} (roll int(5), name varchar(50), marks int(3))'
29             vCmd = vSQLTxt.format(pTable)
30             vCursor = self.Connect.cursor() # Cursor object
31             vCursor.execute(vCmd)
32             vCursor.close()
33             print('\nTable has been created successfully!')
34         except MySQL.Error as e:
35             print('\nMySQLDBError: Failed to create table!\nSee details: ' + str(e))
36         except:
37             print('\nError: Not able to create table due to unknown error!')
38
```

The `createTable()` function creates a table (the table name is sent as argument) in the database using Create Table SQL.

The `cursor()` returns a cursor object (*MySQLCursor* object) which has an `execute()` method to execute SQL commands.

Database Programming using Python

```
39 # Inserts records inside table using SQL
40 def insertRecords(self, pValues, pTable):
41     try:
42         vSQLTxt = "INSERT INTO {0} (roll, name, marks) VALUES ({1}, '{2}', {3})"
43         vCmd = vSQLTxt.format(pTable, pValues[0], pValues[1], pValues[2])
44         vCursor = self.Connect.cursor()
45         vCursor.execute(vCmd)
46         vCursor.close()
47     except MySQL.Error as e:
48         self.Connect.rollback()
49         print('\nMySQLDBError: Failed to insert record!\nSee details: ' + str(e))
50     except:
51         self.Connect.rollback()
52         print('\nError: Not able to insert record due to unknown error!')
53     else:
54         self.Connect.commit()
55         print('\n1 row added successfully!')
56
```

The `insertTable()` function formats an insert query using parameters sent as argument.

Then the `execute()` method to execute SQL commands (an insert query). If there is any error then `rollback()` else `commit()` is called to do a rollback or commit respectively

Database Programming using Python

```
57 # Updates records inside table using SQL
58 def updateRecords(self, pValues, pTable):
59     vRowAffected = 0
60     try:
61         vSQLTxt = "UPDATE {0} SET marks = {1} WHERE roll = {2}"
62         vCmd = vSQLTxt.format(pTable, pValues[0], pValues[1])
63         vCursor = self.Connect.cursor()
64         vRowAffected = vCursor.execute(vCmd) # returns the no. of rows affected
65         vCursor.close()
66     except MySQL.Error as e:
67         self.Connect.rollback()
68         print('\nMySQLDBError: Failed to update record!\nSee details: ' + str(e))
69     except:
70         self.Connect.rollback()
71         print('\nError: Not able to update record due to unknown error!')
72     else:
73         self.Connect.commit()
74         print('\n{0} row(s) updated successfully!'.format(vRowAffected))
75
```

The `updateTable()` function formats an update query using parameters sent as argument.

Then the `execute()` method to execute SQL commands (an update query).

The `execute()` function also returns the number of rows affected.

Database Programming using Python

```
76     # Fetches records from table using SQL
77     def fetchRecords(self, pTable):
78         try:
79             vSQLTxt = 'SELECT * FROM {0}'
80             vCmd = vSQLTxt.format(pTable)
81             vCursor = self.Connect.cursor()
82             vCursor.execute(vCmd)
83             print('Fetched result:-')
84             for vRow in vCursor:
85                 print('\nRoll No: ', vRow[0])
86                 print('Name: ', vRow[1])
87                 print('Marks: ', vRow[2])
88             vCursor.close()
89         except MySQL.Error as e:
90             print('\nMySQLDBError: Failed to fetch record!\nSee details: ' + str(e))
91         except:
92             print('\nError: Not able to fetch record due to unknown error!')
```

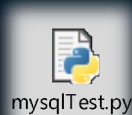
The `fetchRecords()` function formats a select query using parameters sent as argument.

Then the `execute()` method to execute SQL commands (an update query).

Use loops to iterate over cursor object.

Database Programming using Python

The `main()` function has created an object of our `dbHandler` class and invoked the methods using `obj.<method>` notation and passed the required parameters.



```
mysqlTest.py
96 def main():
97     try:
98         vTableName = 'student'
99         vConnectConfig = {
100             'user': 'admin',
101             'pass': 'P@55w0Rd',
102             'db': 'dbSchoolMgt',
103             'host': '127.0.0.1'
104         }
105         objMyDatabase = DBHandler(vConnectConfig)
106         objMyDatabase.createTable(vTableName)
107         while True:
108             vParam = list()
109             vParam.append(input('\nEnter Roll No: '))
110             vParam.append(input('Enter Name: '))
111             vParam.append(input('Enter Marks: '))
112             objMyDatabase.insertRecords(vParam, vTableName)
113             if input('Want to add more records? [y/n]: ').upper() == 'N':
114                 break
115
116         objMyDatabase.fetchRecords(vTableName)
117
118         vParam = list()
119         vParam.append(input('\nEnter Roll No. to update: '))
120         vParam.append(input('Enter New Marks: '))
121
122         objMyDatabase.updateRecords(vParam, vTableName)
123         objMyDatabase.fetchRecords()
124     except Exception as e:
125         print('\nMain.Error: Some error occurred!\nSee details: ', str(e))
126
127     main()
```

Activities

- 1) Write a program to store employee id (AUTO INCREMENT), name, salary, design, project in a database table. During inserting, generate a random password for him/her and store it along with other fields. Display the records in console and also keep functionality to save the displayed result into a file (user choice). Make good formatting to print the data inside the file. Keep functionality to pick records from a formatted file for bulk insert. For delete query remove the record and generate a single log file for all deleted records.



Summary


- 🐍 Python has its own DB-API to support the database programming.
- 🐍 We can use Python to deal with all the major DBMS packages.
- 🐍 We will get separate modules for each DBMS and we can install them as external library and import them in our script.
- 🐍 The *connect()* function helps to connects to the database and returns a connection-object.
- 🐍 The *connection-object.cursor()* returns a cursor which has an *execute()* function to execute SQL queries.

Check on learning

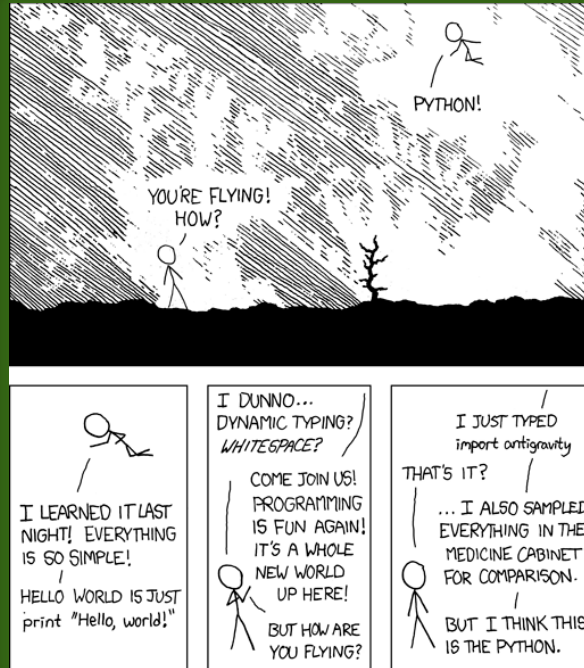
- ✓ We have learnt how to connect to database (MySQL & SQLite) using Python.
- ✓ We also learnt to create Database & Table.
- ✓ We can now execute SQL queries to perform database transactions like SELECT, UPDATE, INSERT, DELETE etc. in Python.
- ✓ We have learnt how to write a complete program in Python to deal with Database.

Links

-  <https://www.sqlite.org/docs.html>
-  <https://dev.mysql.com/doc/connector-python/en/>

Hello, World! This is.. 

It is the extensive library of Python that make programming easier. Agree?



Thank you

Indranil Paul (424241)

Python Developer

Banking & Financial Services | WMC Account