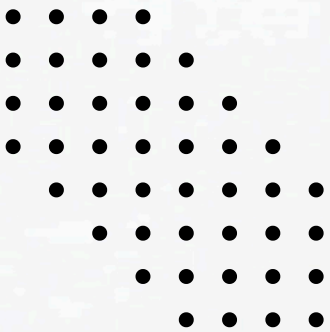
TOPSTechnologies

SQLite3 and PyMySQL (Database Connectors)

Presented for :

TOPs Technologies

Presented by :
Sumit B Yadav



Que 1

Databases play a crucial role in storing, managing, and retrieving data efficiently. Two popular ways to connect and interact with databases in Python are SQLite3 and PyMySQL.

1. SQLite3

SQLite is a lightweight, self-contained database engine that requires no server configuration. Python provides built-in support for SQLite via the sqlite3 module.

Advantages of SQLite

- No separate server process required.
- Simple to set up and use.
- Useful for small to medium-scale applications.
- Ideal for development, testing, and prototyping.

Basic syntax:

```
import sqlite3

# Create a connection to the database (or create it if it doesn't exist)
conn = sqlite3.connect("example.db")

# Create a cursor object to interact with the database
cursor = conn.cursor()

# Execute SQL commands
cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT, age INTEGER)")
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Alice", 25))

# Commit and close the connection
conn.commit()
conn.close()
```

2. PyMySQL

PyMySQL is a MySQL database adapter for Python that allows communication with a MySQL database.

Advantages of PyMySQL

- Supports remote database connections.
- Scalable and suitable for production use.
- Works with MySQL and MariaDB.

Installing PyMySQL

If you haven't installed PyMySQL, use the following command:

```
pip install pymysql
```

Connecting to MySQL with PyMySQL

To connect to a MySQL database:

```
import pymysql

# Establish the connection
conn = pymysql.connect(
    host="localhost",
    user="root",
    password="yourpassword",
    database="testdb"
)

cursor = conn.cursor()

# Create a table
cursor.execute("CREATE TABLE IF NOT EXISTS users (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), age INT)")

# Insert data
cursor.execute("INSERT INTO users (name, age) VALUES (%s, %s)", ("Bob", 30))

conn.commit()
conn.close()
```

Que 2

1. Executing SQL Queries using SQLite3

SQLite3 is a lightweight database that comes pre-installed with Python.

1.1 Connecting to SQLite3:

```
import sqlite3

# Establish a connection and create a database file (if not exists)
conn = sqlite3.connect("example.db")

# Create a cursor object to execute SQL queries
cursor = conn.cursor()
```

1.2 Creating a Table:

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS users (
id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL,
age INTEGER NOT NULL
)
""")
conn.commit()  # Save changes
```

1.3 Inserting Data:

```
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Alice", 25))
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Bob", 30))
conn.commit()  # Save changes
```

1.4 Retrieving Data:

```
cursor.execute("SELECT * FROM users")
rows = cursor.fetchall()

for row in rows:
    print(row)
```

1.5 Updating Records:

```
cursor.execute("UPDATE users SET age = ? WHERE name = ?", (28, "Alice"))
conn.commit()
```

1.6 Deleting Records:

```
cursor.execute("DELETE FROM users WHERE name = ?", ("Bob",))
conn.commit()
```

2. Executing SQL Queries using PyMySQL

PyMySQL allows connecting to MySQL databases and executing SQL commands.

2.1 Connecting to MySQL:

```
import pymysql

# Establish connection to MySQL database
conn = pymysql.connect(
host="localhost",
user="root",
password="yourpassword",
database="testdb"
)

cursor = conn.cursor()
```

2.2 Creating a Table:

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS users (
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
age INT NOT NULL
)
""")
conn.commit()
```

2.3 Inserting Data:

```
cursor.execute("INSERT INTO users (name, age) VALUES (%s, %s)", ("Charlie", 27))
cursor.execute("INSERT INTO users (name, age) VALUES (%s, %s)", ("Diana", 22))
conn.commit()
```

2.4 Retrieving Data:

```
cursor.execute("SELECT * FROM users")
rows = cursor.fetchall()

for row in rows:
print(row)
```