Python Database Connectivity

Introduction

- **Python Database Connectivity** enables Python programs to interact with databases.
- **SQLite** is a lightweight, serverless database included in Python by default.
- In this session, we'll build a **Book Management System** performing:
  - Create
  - Read
  - Update
  - Delete operations

What is SQLite?

- SQLite is a **file-based database**.
- No need for a separate database server.
- Stores data in a **.db** file.
- Comes pre-installed with Python (`sqlite3` module).

✅ Portable
✅ Fast
✅ Ideal for small to medium projects

Importing SQLite in Python

```
import sqlite3
```

- The `sqlite3` module is included with Python.
- Provides APIs to:
- Connect to database
- Execute SQL statements
- Commit and rollback transactions

Basic Database Operations

Operation	Python Command
Connect to database	`sqlite3.connect('books.db')`
Create cursor	`con.cursor()`
Execute query	`cursor.execute("SQL Query")`
Save changes	`con.commit()`
Close connection	`con.close()`

Create Database and Table

Program: `create_table.py`

```
import sqlite3

con = sqlite3.connect('books.db')
cur = con.cursor()

cur.execute('''
CREATE TABLE IF NOT EXISTS book(
    bookid INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    author TEXT NOT NULL
)
''')

print("Table created successfully.")
con.close()
```

Explanation:

- `connect()` → Opens or creates database file.
- `cursor()` → Executes SQL commands.
    - `CREATE TABLE` → Defines Book model.

Insert (CREATE) Operation

Program: `insert_book.py`

```
import sqlite3

con = sqlite3.connect('books.db')
cur = con.cursor()

bookid = int(input("Enter Book ID: "))
name = input("Enter Book Name: ")
author = input("Enter Author: ")

cur.execute("INSERT INTO book VALUES (?, ?, ?)", (bookid, name, author))
con.commit()

print("Book inserted successfully.")
con.close()
```

Explanation:

- `?` → Placeholder for values (prevents SQL injection).
- `commit()` → Saves data permanently.

Read (SELECT) Operation – All Records

Program: `view_books.py`

```
import sqlite3

con = sqlite3.connect('books.db')
cur = con.cursor()

cur.execute("SELECT * FROM book")
records = cur.fetchall()

for row in records:
    print("Book ID:", row[0], "Name:", row[1], "Author:", row[2])

con.close()

Explanation:

fetchall() → Retrieves all rows.

Each row is a tuple.
```

Read by ID (Search Specific Book)

Program: `search_book.py`

```
import sqlite3

con = sqlite3.connect('books.db')
cur = con.cursor()

bookid = int(input("Enter Book ID to Search: "))
cur.execute("SELECT * FROM book WHERE bookid=?", (bookid,))
book = cur.fetchone()

if book:
    print("Book ID:", book[0], "Name:", book[1], "Author:", book[2])
else:
    print("Book not found.")

con.close()
```

Explanation:

- `fetchone()` → Retrieves a single record.

Update Operation

Program: `update_book.py`

```
import sqlite3

con = sqlite3.connect('books.db')
cur = con.cursor()

bookid = int(input("Enter Book ID to Update: "))
new_name = input("Enter New Book Name: ")
new_author = input("Enter New Author: ")

cur.execute("UPDATE book SET name=?, author=? WHERE bookid=?",
            (new_name, new_author, bookid))

if cur.rowcount > 0:
    print("Book updated successfully.")
else:
    print("Book not found.")

con.commit()
con.close()
```

Explanation:

- `UPDATE` modifies existing records.
- `rowcount` helps verify success.

Delete Operation

Program: `delete_book.py`

```
import sqlite3

con = sqlite3.connect('books.db')
cur = con.cursor()

bookid = int(input("Enter Book ID to Delete: "))
cur.execute("DELETE FROM book WHERE bookid=?", (bookid,))

if cur.rowcount > 0:
    print("Book deleted successfully.")
else:
    print("Book not found.")

con.commit()
con.close()
```

Explanation:

- Deletes record based on book ID.

Display All Records After Operations

Program: `display_books.py`

```
import sqlite3

def display_books():
    con = sqlite3.connect('books.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM book")
    books = cur.fetchall()

    print("\nAll Books in Database:")
    for book in books:
        print(book)
    con.close()

display_books()
```

Output Example:

```
(1, 'Python Basics', 'John')
(2, 'Flask Guide', 'Mary')
```