Unit 6: Database and GUI Programming

6.1 Opening and Closing Database Connection with SQLite

SQLite is a lightweight, embedded database that comes pre-installed with Python. It is widely used for small to medium-sized applications, especially for local storage.

- Use sqlite3.connect('filename.db') to open or create a database.
- Use .cursor() to create a cursor object.
- Always commit changes with .commit() and close connection with .close() to avoid data loss.

```
import sqlite3
conn = sqlite3.connect("school.db")
cursor = conn.cursor()
conn.commit()
conn.close()
```

6.2 Creating and Deleting Tables

- Use SQL CREATE TABLE to make new tables.
- Use DROP TABLE to delete them.

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY,
    name TEXT,
    age INTEGER
)
""")

cursor.execute("DROP TABLE IF EXISTS students")
```

6.3 Adding Data to a Table

- Insert data using INSERT INTO.

```
cursor.execute("INSERT INTO students (name, age) VALUES (?, ?)", ("Alice", 20))
conn.commit()
```

6.4 CRUD Operations

CRUD stands for Create, Read, Update, and Delete—four basic functions of persistent storage in databases.

Create

- Add new records into a table using INSERT INTO.

```
cursor.execute("INSERT INTO students (name, age) VALUES (?, ?)", ("Bob", 22))
```

Read

- Retrieve data using the SELECT statement.

```
cursor.execute("SELECT * FROM students")
rows = cursor.fetchall()
for row in rows:
    print(row)
```

Update

- Modify existing data using the UPDATE statement.

```
cursor.execute("UPDATE students SET age = ? WHERE name = ?", (23, "Bob"))
```

Delete

- Remove records using the DELETE statement.

```
cursor.execute("DELETE FROM students WHERE name = ?", ("Bob",))
```

Don't forget to use conn.commit() after each operation to save changes.

6.5 Using the tkinter Module

Tkinter is the **standard GUI (Graphical User Interface) library** for Python. It provides tools to create windows, buttons, labels, text boxes, and other GUI elements in desktop applications.

- Comes pre-installed with Python.
- Based on the **Tk GUI toolkit**, which is written in C.
- Widely used for simple to moderately complex GUI applications.

Example:

```
from tkinter import *
root = Tk()
root.title("My App")
root.geometry("300x200")
Label(root, text="Hello, Tkinter!").pack()
root.mainloop()
```

6.6 Working with Widgets

Displaying Text

```
Label(root, text="Hello").pack()
Button(root, text="Click").pack()
```

Info Dialog Boxes

```
from tkinter import messagebox
messagebox.showinfo("Info", "This is a message")
```

Getting Input with Entry

```
entry = Entry(root)
entry.pack()
```

Using Labels as Output Fields

```
output = Label(root, text="")
output.pack()
output.config(text="Updated Text")
```

Radio and Check Buttons

```
Radiobutton(root, text="Option 1", value=1, variable=var).pack()
Checkbutton(root, text="Check me").pack()
```

6.7 Organizing Widgets with Frames

- Use Frame() to organize layout.

```
frame = Frame(root)
frame.pack()
Label(frame, text="Inside Frame").pack()
```

6.8 Drawing Shapes with Canvas Widget

- Use Canvas() to draw shapes.

```
canvas = Canvas(root, width=200, height=100)
canvas.pack()
canvas.create_rectangle(50, 20, 150, 80, fill="blue")
canvas.create_oval(50, 20, 150, 80, fill="green")
```