**1. Write a Python program to create a GUI application using Tkinter that controls the text font size in a label using a slider. Additionally, include a dropdown to change the font colour and a button to randomly change the window's background colour.**

import tkinter as tk

from tkinter import ttk

import random

class FontSizeApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Font Size Controller")

        # Default settings

        self.default\_font\_size = 16

        self.default\_font\_color = "black"

        self.default\_bg\_color = "white"

        # Create a label to display the text

        self.label = tk.Label(self.root, text="Welcome to Yenepoya", font=("Arial", self.default\_font\_size), fg=self.default\_font\_color)

        self.label.pack(pady=20)

        # Create a slider to control font size

        self.font\_size\_slider = tk.Scale(self.root, from\_=10, to=100, orient="horizontal",  label="Font Size", command=self.update\_font\_size)

        self.font\_size\_slider.set(self.default\_font\_size)

        self.font\_size\_slider.pack(pady=10)

        # Dropdown to change text color

        self.color\_label = tk.Label(self.root, text="Choose Font Color:")

        self.color\_label.pack(pady=5)

        self.color\_var = tk.StringVar(value="black")

        self.color\_dropdown = ttk.Combobox(self.root, textvariable=self.color\_var, values=["black", "red", "blue", "green", "purple"], state="readonly")

        self.color\_dropdown.pack(pady=5)

        self.color\_dropdown.bind("<<ComboboxSelected>>", self.update\_color)

        # Button to change background color randomly

        self.bg\_button = tk.Button(self.root, text="Change Background Color", command=self.change\_bg\_color)

        self.bg\_button.pack(pady=10)

    def update\_font\_size(self, size):

        # Update the font size of the label

        self.label.config(font=("Arial", int(size)))

    def update\_color(self, event):

        # Update the font color of the label

        self.label.config(fg=self.color\_var.get())

    def change\_bg\_color(self):

        # Change the background color to a random one

        random\_color = random.choice(["lightblue", "lightgreen", "lightyellow", "lightpink", "lightgray"])

        self.root.config(bg=random\_color)

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    app = FontSizeApp(root)

    root.geometry("400x300")

    root.mainloop()

**2. Write a python script that performs the following operations on the SQLite database:**

* 1. **Creates a student’s table.**
  2. **Inserts two records into the table.**
  3. **Retrieves and displays the records.**
  4. **Updates a student's grade.**
  5. **Deletes a student record.**
  6. **Closes the database connection.**

import sqlite3

**# Step 1: Connect to SQLite database (or create it if it doesn't exist)**

conn = sqlite3.connect('students.db')

cursor = conn.cursor()

**# Step 2: Create the students table**

cursor.execute('''

CREATE TABLE IF NOT EXISTS students (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

grade INTEGER NOT NULL)''')

**# Step 3: Insert two records into the students table**

cursor.execute("INSERT INTO students (name, grade) VALUES ('Alice', 90)")

cursor.execute("INSERT INTO students (name, grade) VALUES ('Bob', 85)")

# Commit the changes

conn.commit()

**# Step 4: Retrieve and display all records**

cursor.execute("SELECT \* FROM students")

print("Students before update:")

rows = cursor.fetchall()

for row in rows:

print(row)

**# Step 5: Update the grade of a student (e.g., update Bob's grade to 95)**

cursor.execute("UPDATE students SET grade = 95 WHERE name = 'Bob'")

# Commit the changes

conn.commit()

**# Step 6: Retrieve and display the updated records**

cursor.execute("SELECT \* FROM students")

print("\nStudents after update:")

rows = cursor.fetchall()

for row in rows:

print(row)

**# Step 7: Delete a student record (e.g., delete Alice)**

cursor.execute("DELETE FROM students WHERE name = 'Alice'")

# Commit the changes

conn.commit()

**# Step 8: Retrieve and display the records after deletion**

cursor.execute("SELECT \* FROM students")

print("\nStudents after deletion:")

rows = cursor.fetchall()

for row in rows:

print(row)

**# Step 9: Close the database connection**

conn.close()

**3.** **Write a Python program to perform the following tasks related to employee data using Pandas:**

1. **Create a DataFrame containing the following details of employees:**
   * **Name, Age, Salary, Department ( IT, HR, Finance).**
2. **Save the DataFrame to a CSV file named employees.csv.**
3. **Read the DataFrame from the employees.csv file.**
4. **Perform the following operations on the DataFrame:**
   * **Indexing: Select and display the Age column.**
   * **Selection: Select and display the Name and Department columns for all employees.**
   * **Filtering:**
     + **Display all employees who are older than 30.**
     + **Display all employees who work in the "IT" department.**

**CODE**:

import pandas as pd

**# Step 1: Create a DataFrame**

data = {

    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

    'Age': [23, 35, 45, 27, 32],

    'Salary': [60000, 52000, 72000, 48000, 91000],

    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR']

}

df = pd.DataFrame(data)

**# Step 2: Write the DataFrame to a CSV file**

df.to\_csv('employees.csv', index=False)

print("DataFrame saved to 'employees.csv'.")

**# Step 3: Read the DataFrame from the CSV file**

df\_read = pd.read\_csv('employees.csv')

print("\nDataFrame read from 'employees.csv':")

print(df\_read)

**# Step 4: Data Indexing, Selection, and Filtering**

# Indexing: Select a specific column (Age)

print("\nIndexing: Select 'Age' column:")

print(df['Age'])

# Selection: Select multiple columns (Name and Department)

print("\nSelection: Select 'Name' and 'Department' columns:")

print(df[['Name', 'Department']])

# Row selection by index (select row at index 2)

print("\nSelect row at index 2:")

print(df.iloc[2])

# Filtering: Select rows where Age is greater than 30

print("\nFiltering: Rows where Age > 30:")

print(df[df['Age'] > 30])

# Filtering: Select rows where Department is 'IT'

print("\nFiltering: Rows where Department is 'IT':")

print(df[df['Department'] == 'IT'])