Ganpat University | Department of Computer Science

॥ विद्यया समाजोत्कर्षः ॥

M.Sc. (CA & IT) Semester – I

Subject Code: P21A3PY Subject Name: PYTHON PROGRAMMING

Practical List: July-Dec 2025

1 Write a console program that generates a student's marksheet

2 Write a program that identifies the random student from the list to participate in event

3 Write a program to find out the average height of the person available on the list

4 Write a program to generate random password from the lists comprising capital letters, small letters, numbers (0-9) and special symbols respectively

5 A command-line calculator that performs basic arithmetic operations: addition, subtraction, multiplication, and division

6 A simple console application to manage a to-do list, allowing users to add, view, and delete tasks

Implementation Guidelines

- **Libraries**: None required.
- **Steps**:
  1. Use a list to store tasks.
  2. Implement functions to add, view, and delete tasks.
  3. Use a loop to keep the program running until the user decides to exit.
- **Example Code**:

```
def todo_list():
    tasks = []
    while True:
        action = input("Enter 'add', 'view', 'delete', or 'exit': ")
        if action == 'add':
            task = input("Enter a task: ")
            tasks.append(task)
        elif action == 'view':
            print("Tasks:", tasks)
        elif action == 'delete':
            task = input("Enter a task to delete: ")
            tasks.remove(task) if task in tasks else print("Task not found.")
        elif action == 'exit':
            break
        else:
            print("Invalid input.")

todo_list()
```

7 Write a code for console application to store and manage contacts, including names and phone numbers

Implementation Guidelines

- **Libraries**: None required.
- **Steps:**
  1. Use a dictionary to store contact names and phone numbers.
  2. Implement functions to add, view, and delete contacts.
  3. Use looping for continuous interaction.
- **Example Code:**

```
def contact_book():
  contacts = {}
  while True:
    action = input("Enter 'add', 'view', 'delete', or 'exit': ")
    if action == 'add':
      name = input("Enter contact name: ")
      phone = input("Enter phone number: ")
      contacts[name] = phone
    elif action == 'view':
      print("Contacts:", contacts)
    elif action == 'delete':
      name = input("Enter contact name to delete: ")
      contacts.pop(name, "Contact not found.")
    elif action == 'exit':
      break
    else:
      print("Invalid input.")
contact_book()
```


8 A program that reads a text file and analyzes its content, providing word count, line count, and character count

Implementation Guidelines

- **Libraries**: None required.
- **Steps**:
  1. Open and read the text file.
  2. Calculate the number of lines, words, and characters.
  3. Display the results.
- **Example Code**:

```
def text_file_analyzer(file_path):
  with open(file_path, 'r') as file:
    content = file.read()
    lines = content.splitlines()
    words = content.split()
    print(f"Lines: {len(lines)}, Words: {len(words)}, Characters: {len(content)}")

text_file_analyzer('example.txt')
```

9 An application that allows users to track their expenses by categorizing them and displaying a summary

Implementation Guidelines

- **Libraries**: None required.
- **Steps**:
    1. Use a list to store expenses as dictionaries with category, amount, and description.
    2. Implement functions to add and view expenses.
    3. Display the total expenses by category.
- **Example Code**:

```
def expense_tracker():
  expenses = []
  while True:
    action = input("Enter 'add', 'view', or 'exit': ")
    if action == 'add':
      category = input("Enter category: ")
      amount = float(input("Enter amount: "))
      description = input("Enter description: ")
      expenses.append({'category': category, 'amount': amount, 'description':
description})
    elif action == 'view':
      total = sum(exp['amount'] for exp in expenses)
      print(f"Total Expenses: {total}")
      for exp in expenses:
        print(exp)
    elif action == 'exit':
      break
    else:
      print("Invalid input.")

expense_tracker()
```


10 A simple GUI application that allows users to open and view images.

Implementation Guidelines

- **Libraries**: tkinter, PIL
- **Steps**:
    1. Create a GUI using tkinter.
    2. Implement a file dialog to open images.
    3. Display the selected image in the application.

Example Code:

```
from tkinter import Tk, Button, Label
from tkinter.filedialog import askopenfilename
from PIL import Image, ImageTk

def image_viewer():
  root = Tk()
  root.title("Image Viewer")

  def open_image():
    file_path = askopenfilename()
```

```
    img = Image.open(file_path)
    img = img.resize((250, 250), Image.ANTIALIAS)
    img_tk = ImageTk.PhotoImage(img)
    label.config(image=img_tk)
    label.image = img_tk

  button = Button(root, text="Open Image", command=open_image)
  button.pack()
  label = Label(root)
  label.pack()
  root.mainloop()

image_viewer()
```

11 An application that analyzes the sentiment of a given text as positive, negative, or neutral using a simple sentiment analysis library.

Implementation Guidelines

- **Libraries**: textblob
- **Steps**:
  1. Install textblob using pip.
  2. Prompt the user to input a sentence.
  3. Analyze the sentiment and display the result.
- **Example Code**:

```
from textblob import TextBlob

def sentiment_analysis():
  text = input("Enter a sentence: ")
  analysis = TextBlob(text)
  print("Sentiment:", "Positive" if analysis.sentiment.polarity > 0 else "Negative"
if analysis.sentiment.polarity < 0 else "Neutral")

sentiment_analysis()
```

12 Write a code to develop GUI application that allows teachers to input student marks for various subjects, analyzes performance, and generates reports.

Implementation Guidelines

- Libraries: Tkinter, Pandas, Matplotlib
- Features:
- Input student information and marks through a form.
- Use Pandas to calculate averages, grades, and visualize performance trends using charts.
- Generate and export reports as CSV or Excel files.
- Example Code

```
import tkinter as tk
from tkinter import messagebox
import pandas as pd
import matplotlib.pyplot as plt

class PerformanceAnalyzer:
  def __init__(self, root):
```

```
        self.root = root
        self.root.title("Student Performance Analyzer")
        self.students = []

        # Input fields
        tk.Label(root, text="Student Name:").grid(row=0, column=0)
        self.name_entry = tk.Entry(root)
        self.name_entry.grid(row=0, column=1)

        tk.Label(root, text="Marks:").grid(row=1, column=0)
        self.marks_entry = tk.Entry(root)
        self.marks_entry.grid(row=1, column=1)

        tk.Button(root, text="Add Student", command=self.add_student).grid(row=2,
columnspan=2)
        tk.Button(root, text="Generate Report",
command=self.generate_report).grid(row=3, columnspan=2)

    def add_student(self):
        name = self.name_entry.get()
        marks = self.marks_entry.get()
        if name and marks.isnumeric():
            self.students.append((name, int(marks)))
            self.name_entry.delete(0, tk.END)
            self.marks_entry.delete(0, tk.END)
            messagebox.showinfo("Success", f"Added {name} with {marks} marks!")
        else:
            messagebox.showerror("Error", "Invalid input!")

    def generate_report(self):
        if not self.students:
            messagebox.showwarning("Warning", "No students to report!")
            return

        df = pd.DataFrame(self.students, columns=["Name", "Marks"])
        average = df['Marks'].mean()
        df['Grade'] = pd.cut(df['Marks'], bins=[0, 50, 60, 70, 80, 90, 100],
                             labels=['F', 'D', 'C', 'B', 'A', 'A+'], right=False)

        # Generate report
        report = f"Average Marks: {average:.2f}\n\n{df}\n"
        messagebox.showinfo("Report", report)

        # Plotting
        df.plot(kind='bar', x='Name', y='Marks', legend=False)
        plt.title('Student Marks')
        plt.ylabel('Marks')
        plt.show()
```

```
if __name__ == "__main__":
    root = tk.Tk()
    app = PerformanceAnalyzer(root)
    root.mainloop()
```

13 An expense tracking application that helps users manage their finances and visualize spending habits.

Implementation Guidelines

- **Libraries**: Tkinter, Pandas, Matplotlib
- **Features**:
  - Input expenses through a GUI and categorize them.
  - Use Pandas to store and analyze data.
  - Visualize spending patterns with charts.
- **Example Code**

```
import tkinter as tk
from tkinter import messagebox
import pandas as pd
import matplotlib.pyplot as plt

class ExpenseTracker:
    def __init__(self, root):
        self.root = root
        self.root.title("Expense Tracker")
        self.expenses = pd.DataFrame(columns=["Category", "Amount"])

        # Input fields
        tk.Label(root, text="Category:").grid(row=0, column=0)
        self.category_entry = tk.Entry(root)
        self.category_entry.grid(row=0, column=1)

        tk.Label(root, text="Amount:").grid(row=1, column=0)
        self.amount_entry = tk.Entry(root)
        self.amount_entry.grid(row=1, column=1)

        tk.Button(root, text="Add Expense",
command=self.add_expense).grid(row=2, columnspan=2)
        tk.Button(root, text="Show Report", command=self.show_report).grid(row=3,
columnspan=2)

    def add_expense(self):
        category = self.category_entry.get()
        amount = self.amount_entry.get()
        if category and amount.isnumeric():
            self.expenses = self.expenses.append({"Category": category, "Amount":
float(amount)}, ignore_index=True)
            self.category_entry.delete(0, tk.END)
            self.amount_entry.delete(0, tk.END)
            messagebox.showinfo("Success", f"Added {amount} to {category}!")
```

```
            else:
                messagebox.showerror("Error", "Invalid input!")

        def show_report(self):
            if self.expenses.empty:
                messagebox.showwarning("Warning", "No expenses to report!")
                return

            report = self.expenses.groupby('Category').sum()
            report.plot(kind='bar')
            plt.title('Expenses by Category')
            plt.ylabel('Total Amount')
            plt.show()

    if __name__ == "__main__":
        root = tk.Tk()
        app = ExpenseTracker(root)
        root.mainloop()
```