Project Idea: Personal Assistant CLI App

Project Overview

Build a Command-Line Interface (CLI) app that allows users to:

1. Manage tasks (To-Do List).
2. Maintain a personal journal.
3. Perform basic file operations (e.g., reading, writing, deleting files).
4. Perform simple data analysis (e.g., word count in a text file).
5. Provide additional features like a calculator, random quotes, or a weather lookup (mocked).

Features and Requirements

1. Task Manager

- Add, view, delete, or mark tasks as complete.
- Tasks should be stored in a file to persist between sessions.
- Use dictionaries or lists to organize tasks.

2. Personal Journal

- Create journal entries with timestamps.
- View all entries or search for entries by date.
- Save entries to a file for future retrieval.

3. File Operations

- Allow users to:
    - Read the content of a file.
    - Write or append new content to a file.
    - Delete a file after confirmation.
- Handle file-related errors (e.g., file not found).

4. Text Analyzer

- Take a text file as input.
- Provide the following analysis:
    - Word count.
    - Character count (with and without spaces).
    - Most frequently used words.
- Use string manipulation techniques.

5. Calculator

- Provide basic calculator functionality (addition, subtraction, multiplication, division).
- Include input validation and error handling (e.g., divide by zero).

6. Random Quotes Generator

- Store a list of motivational quotes in a file.
- Display a random quote whenever the user requests one.

7. Help Menu

- Provide a user-friendly menu or help section explaining the available commands and features.

Technologies and Concepts to Use

1. **File Handling**
    a. Save and retrieve tasks, journal entries, and quotes.
    b. Analyze text files for the word count feature.
2. **Data Structures**
    a. Lists for storing tasks and quotes.
    b. Dictionaries for organizing tasks and journal entries.
3. **Functions**
    a. Separate logic into different functions (e.g., `add_task()`, `analyze_text()`).
4. **Error Handling**
    a. Handle invalid inputs, file not found errors, etc.
5. **String Manipulation**
    a. For searching entries, analyzing text, and formatting journal entries.
6. **Decorators (Optional)**
    a. Add a time logger for certain functions (e.g., to log when a task is added or journal is updated).

Sample Structure

```
def main():
    while True:
        print("Personal Assistant CLI")
        print("1. Task Manager")
        print("2. Journal")
        print("3. File Operations")
        print("4. Text Analyzer")
        print("5. Calculator")
```

```
        print("6. Random Quotes")
        print("7. Exit")

        choice = input("Choose an option: ")

        if choice == '1':
            task_manager()
        elif choice == '2':
            journal_manager()
        elif choice == '3':
            file_operations()
        elif choice == '4':
            text_analyzer()
        elif choice == '5':
            calculator()
        elif choice == '6':
            random_quote()
        elif choice == '7':
            print("Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

# Define functions for each feature below
def task_manager():
    pass  # Add, view, and delete tasks.

def journal_manager():
    pass  # Create and view journal entries.

def file_operations():
    pass  # Read, write, and delete files.

def text_analyzer():
    pass  # Analyze a text file.

def calculator():
    pass  # Perform basic calculations.

def random_quote():
    pass  # Show a random quote.

if __name__ == "__main__":
    main()
```

Extensions for Extra Challenge

- Add a **search functionality** to the task manager and journal.
- Allow users to set **reminders** (mocked with simple print statements).

- Use JSON or CSV for data storage instead of plain text for a structured approach.