➢ Contact Book Project Report

Project Title: Contact Book Management

Course /Subject: Python Programming

Student Name: SINGH SAURAV DURGA PRASAD

Roll Number: 25BCE11074

Submitted To: M K Jayanthi

Date: 24 November 2025

1. Introduction

This project presents a simple command-line contact management application written in Python. It allows users to create, view, search, and delete contact entries conveniently, demonstrating fundamental programming concepts and basic data management in a user-friendly format.

2. Problem Statement

Digital contact management tools are often overly complex for simple needs. This project addresses the problem by providing an easy-to-use CLI solution for managing contacts, ideal for beginners and situations where a lightweight tool is preferred.

3. Functional Requirements

- Add new contacts with a name and phone number
- View all saved contacts
- Search contacts by name
- Delete a contact by name
- Simple, user-driven navigation via a menu

4. Non-functional Requirements

- Quick execution and response times

- Robust input validation and user feedback
- Maintainable, readable source code
- No external dependencies; pure Python standard library
- Portable and easily executable on any system with Python installed

5. System Architecture

- User Interface: Command-line menu and prompts
- Logic: Structured as individual functions for each feature
- Storage: In-memory storage using Python's data structures
- Persistence: None in the current version (session-based only)

6. Design Diagrams

Use Case Diagram

- The user can add, view, search, and delete contacts via a menu interface.

Workflow Diagram

- The user's input is directed to respective functions based on menu selection, and results are shown before returning to the menu.

Sequence Diagram

- User selects an option → Application prompts for details → User provides input → Application processes and displays results.

Class/Component Diagram

- Main module contains functions responsible for adding, viewing, searching, and deleting contacts; all interact with a shared in-memory data structure.

7. Design Decisions & Rationale

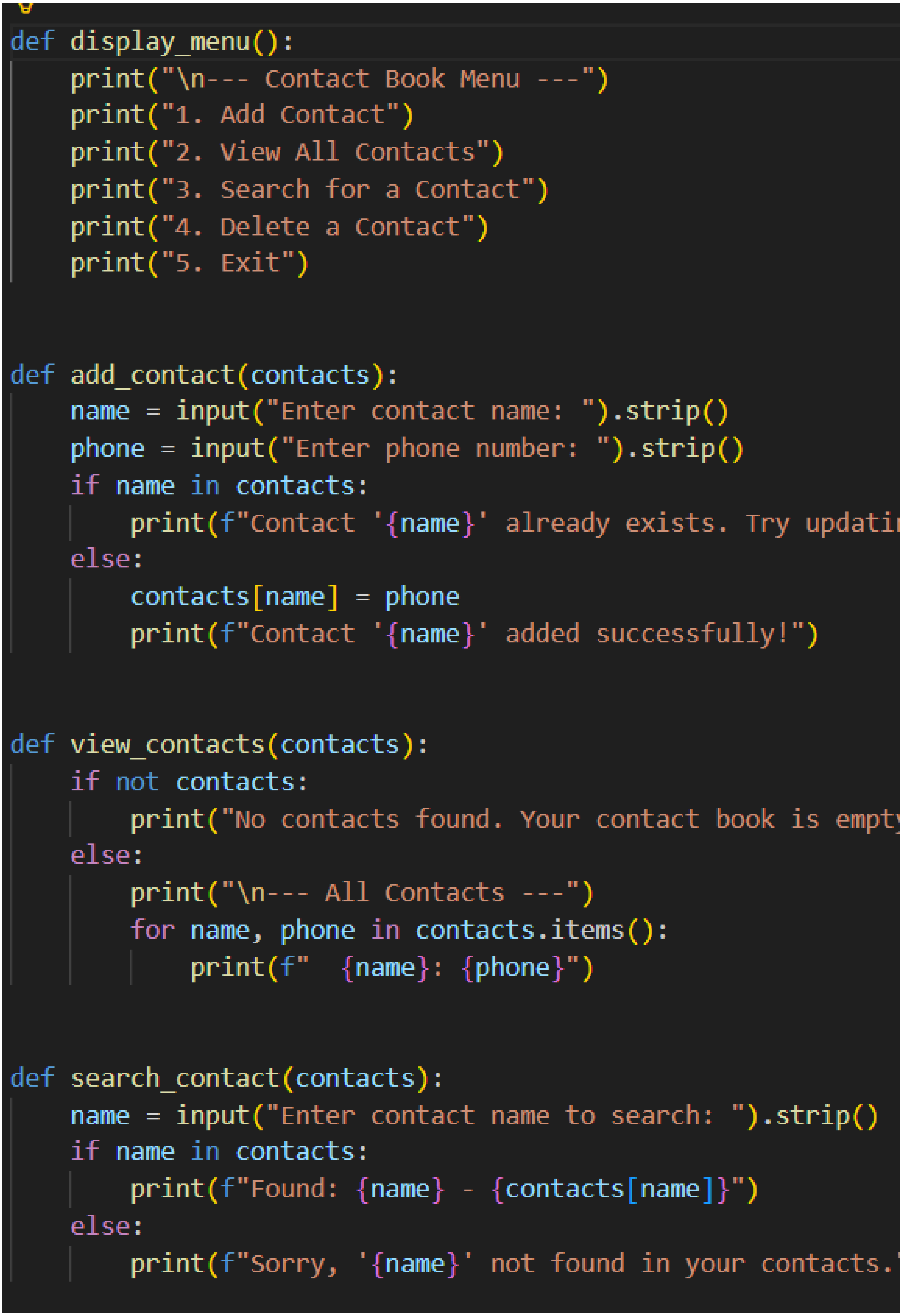
- Chose dictionary data structure for efficient retrieval and update of contacts.
- Adopted a menu-driven approach for simplicity and clarity.
- Kept the code modular by separating core functionalities into distinct functions.

- Refrained from using file/database storage to maintain project scope and focus on logic.

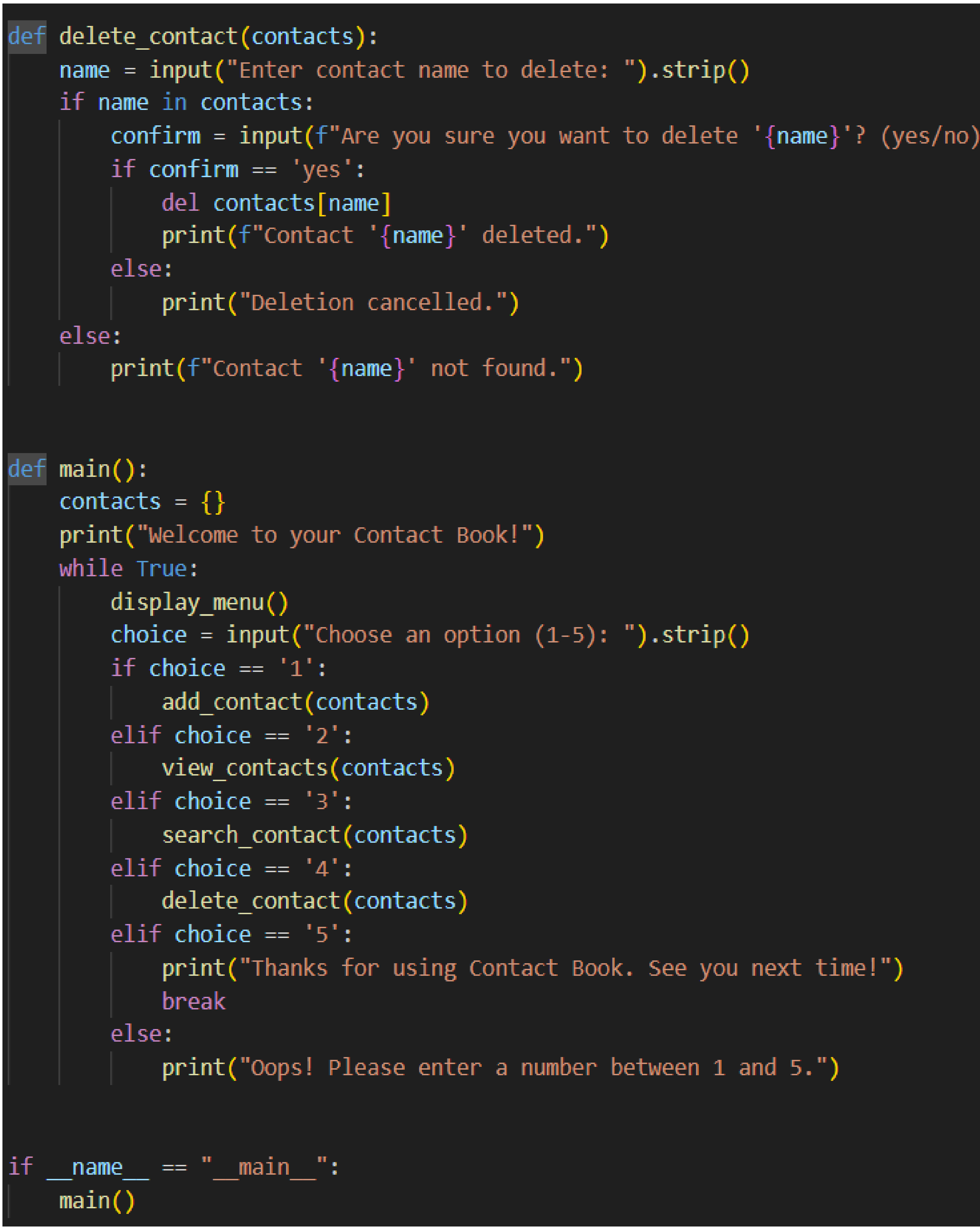
8.Implementation Details

- All program logic is concentrated in a single Python file.
- Navigation is handled through a menu system, looping until the user exits.
- Data is temporarily stored in a dictionary within the session.

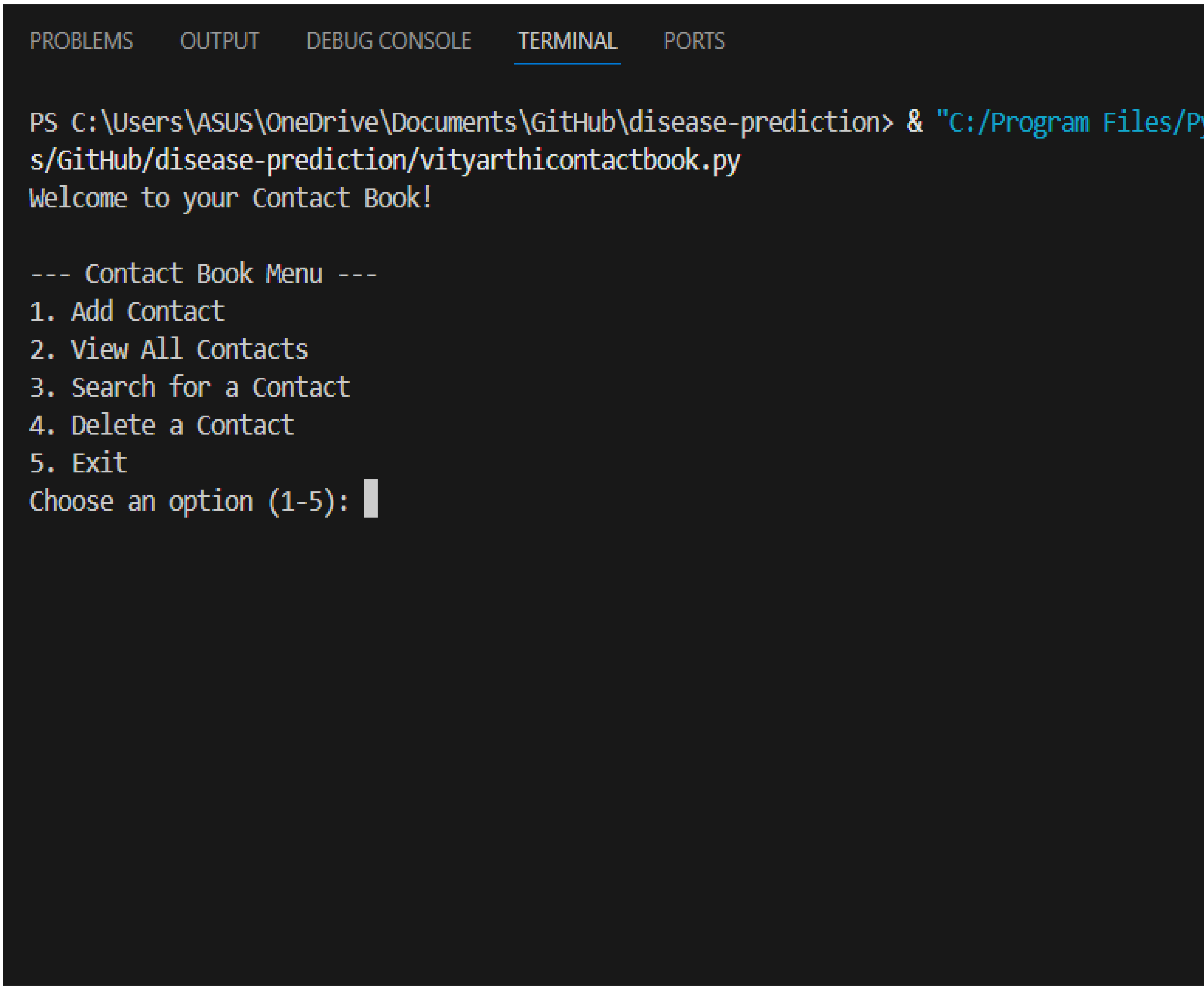
9.Screenshots / Results



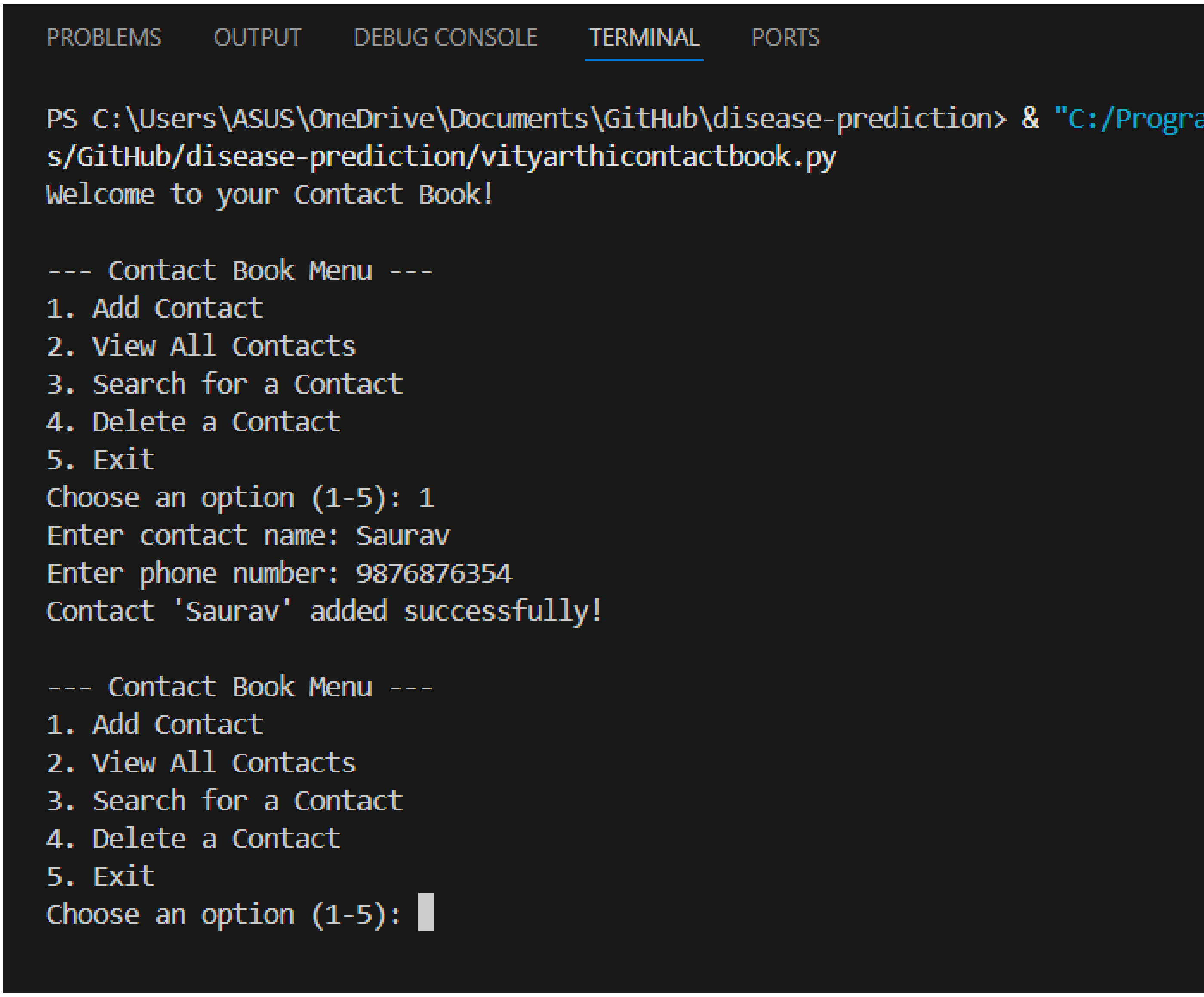
```
def display_menu():
    print("\n--- Contact Book Menu ---")
    print("1. Add Contact")
    print("2. View All Contacts")
    print("3. Search for a Contact")
    print("4. Delete a Contact")
    print("5. Exit")


def add_contact(contacts):
    name = input("Enter contact name: ").strip()
    phone = input("Enter phone number: ").strip()
    if name in contacts:
        print(f"Contact '{name}' already exists. Try updati
    else:
        contacts[name] = phone
        print(f"Contact '{name}' added successfully!")


def view_contacts(contacts):
    if not contacts:
        print("No contacts found. Your contact book is empty
    else:
        print("\n--- All Contacts ---")
        for name, phone in contacts.items():
            print(f"  {name}: {phone}")


def search_contact(contacts):
    name = input("Enter contact name to search: ").strip()
    if name in contacts:
        print(f"Found: {name} - {contacts[name]}")
    else:
        print(f"Sorry, '{name}' not found in your contacts.'
```



```python
def delete_contact(contacts):
    name = input("Enter contact name to delete: ").strip()
    if name in contacts:
        confirm = input(f"Are you sure you want to delete '{name}'? (yes/no)
        if confirm == 'yes':
            del contacts[name]
            print(f"Contact '{name}' deleted.")
        else:
            print("Deletion cancelled.")
    else:
        print(f"Contact '{name}' not found.")


def main():
    contacts = {}
    print("Welcome to your Contact Book!")
    while True:
        display_menu()
        choice = input("Choose an option (1-5): ").strip()
        if choice == '1':
            add_contact(contacts)
        elif choice == '2':
            view_contacts(contacts)
        elif choice == '3':
            search_contact(contacts)
        elif choice == '4':
            delete_contact(contacts)
        elif choice == '5':
            print("Thanks for using Contact Book. See you next time!")
            break
        else:
            print("Oops! Please enter a number between 1 and 5.")


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



```
PS C:\Users\ASUS\OneDrive\Documents\GitHub\disease-prediction> & "C:/Program Files/Py
s/GitHub/disease-prediction/vityarthicontactbook.py
Welcome to your Contact Book!

--- Contact Book Menu ---
1. Add Contact
2. View All Contacts
3. Search for a Contact
4. Delete a Contact
5. Exit
Choose an option (1-5):
```



```
PS C:\Users\ASUS\OneDrive\Documents\GitHub\disease-prediction> & "C:/Progra
s/GitHub/disease-prediction/vityarthicontactbook.py
Welcome to your Contact Book!

--- Contact Book Menu ---
1. Add Contact
2. View All Contacts
3. Search for a Contact
4. Delete a Contact
5. Exit
Choose an option (1-5): 1
Enter contact name: Saurav
Enter phone number: 9876876354
Contact 'Saurav' added successfully!

--- Contact Book Menu ---
1. Add Contact
2. View All Contacts
3. Search for a Contact
4. Delete a Contact
5. Exit
Choose an option (1-5):
```

10. Testing Approach

- Performed manual testing by executing all menu options and entering valid/invalid data.
- Tested edge cases such as duplicate entries, blank inputs, and deleting non-existent contacts.

---

11. Challenges Faced

- Ensuring contact uniqueness and preventing duplicate entries.
- Implementing user confirmation dialogs for deletion.
- Designing for intuitive user interaction in a command-line setting.

---

12. Learnings & Key Takeaways

- Reinforced fundamental data structures in Python.
- Gained practical experience designing modular, menu-driven CLI programs.
- Appreciated the importance of input validation and user-centric design.

13. Future Enhancements

- Add persistent storage via files (JSON/CSV) or databases.
- Enable editing/updating contacts.
- Expand contact fields (e.g., email, address).
- Introduce input validation for contact information.
- Potentially develop a graphical interface.