250-CS Data Structures & Algorithms
Phone Book Management

Areeba Ahmed & Zoya Ahsan Mehmood

January 2024

Contents

1 Introduction

The functional phone book application is designed to manage contact information efficiently. The script implementing this project employs a procedural programming approach, using modular functions to perform key operations such as adding, displaying, modifying, searching, and deleting contacts. Utilizing CSV files for data storage ensures persistence across sessions, and the script offers a command-line interface for user interaction. The code is both a practical tool for organizing contacts and an educational resource for learning Python, file handling, and basic data structures.

2 Objectives

1. **Contact Addition:** Users can add new contacts by providing necessary details such as first name, last name, mobile number, and optional tags.
2. **Directory Display:** The functionality can display the entire contact directory, providing users with an overview of all saved contacts.
3. **Tag-Based Display:** Users can view contacts filtered by specific tags, facilitating organized categorization and retrieval.
4. **Contact Modification:** The option to modify existing contact details, including first name, last name, mobile number, and tags is available.
5. **Contact Deletion:** Users are enabled to delete contacts based on either the mobile number or a combination of first and last names.
6. **Search Functionality:** A search feature has been implemented allowing users to find contacts by name or phone number.
7. **Clear Console Screen:** A clear screen function is developed to enhance the user interface and maintain a clean display.
8. **Data Persistence:** CSV files are utilized for storing contact data, and ensuring that the information persists between different program executions.
9. **Display Sorting Options:** Users are allowed to choose between ascending and descending order while displaying contacts.

10. **User-Friendly CLI:** A user-friendly command-line interface for interaction with the application is provided.
11. **Informative Display:** For better user understanding contact information is displayed in a formatted and informative table.
12. **Error Handling:** Implementations manage unexpected situations and provide informative messages to users.
13. **Directory and Tag Information:** Functions are offered to retrieve information about the total number of contacts in the directory and the count of contacts with a specific tag.
14. **Modular Design:** The code is designed in a modular way, with separate functions for different operations, enhancing readability and maintainability.
15. **Educational Tool:** Serves as an educational resource for learning Python programming concepts, file handling, and basic data structures.

3 Methodology

The methodology used in the phone book application script follows a procedural programming paradigm. Procedural programming is a programming paradigm based on the concept of procedures, or routines, that operate on data. In this case, the script consists of a collection of functions that perform specific tasks, and the main program (the **main()** function) orchestrates these functions to achieve the desired functionality.

3.1 Characteristics of the Methodology

- **Modular Design:** The script is modular, with functions dedicated to specific tasks such as adding contacts, displaying information, searching, modifying, and deleting contacts.
- **Step-by-Step Execution:** The program follows a step-by-step execution flow where user input is processed, and corresponding functions are called based on the user's choices.
- **Function Decomposition:** Each function is designed to perform a specific operation, contributing to the overall functionality of the application. For example, functions handle contact addition, display, modification, and deletion.
- **Procedural Abstraction:** Procedural abstraction is utilized to encapsulate the details of specific tasks within functions, promoting code reusability and maintainability.
- **Linear Flow:** The program executes linearly, responding to user inputs and calling functions accordingly.

4 Classes

4.1 Contact Class

Attributes:

- **fname:** First name of the contact.
- **lname:** Last name of the contact.
- **contact no:** Mobile number of the contact.
- **tag:** Tag associated with the contact.

5 Functions

5.1 clear_screen()

Purpose: Clears the console screen.

Implementation: Utilizes the os module to execute system commands for clearing the screen based on the operating system.

5.2 get_contact_details():

Purpose: Takes user input to create a new Contact object.

Returns: A Contact object with user-provided details.

5.3 add_contact():

Purpose: Adds a new contact to the phone book.

Implementation: Obtains contact details using `get_contact_details()` and appends them to a CSV file (`"PhoneBook.csv"`).

5.4 directory_info():

Purpose: Retrieves the total number of contacts in the phone book.

Returns: The count of contacts by reading the CSV file.

5.5 tag_info(tag):

Purpose: Retrieves the count of contacts with a specific tag.

Parameters:

- `tag`: The tag to search for.

Returns: The count of contacts with the specified tag by reading the CSV file.

5.6 display_contacts(lines):

Purpose: Displays contacts in a formatted table.

Parameters:

- `lines`: List of lines from the CSV file.

Implementation: Parses CSV lines and prints contact details in a tabular format.

5.7 display():

Purpose: Displays the phone book contents with sorting options.

Implementation: Reads the CSV file, allows the user to choose the display mode (ascending or descending), and calls `display_contacts()`.

5.8 display_by_tag():

Purpose: Displays contacts filtered by a specific tag.

Implementation: Reads the CSV file and filters contacts based on the provided tag before calling `display_contacts()`.

5.9 search_contact():

Purpose: Searches for a contact by name or phone number.

Implementation: Reads the CSV file and displays contacts matching the search criteria.

5.10 delete_contact():

Purpose:

Deletes contacts based on user input (mobile number or name).

Implementation: Reads the CSV file, creates a new file (`"helper.csv"`) with an updated contact list, and renames it to replace the original.

5.11 modify_contact():

Purpose: Modifies contact details based on user input.

Implementation: Reads the CSV file, modifies contact details if found, and updates the file with the modified information.

5.12 main():

Purpose: Main menu for user interaction.

Implementation: A continuous loop offering options to perform various operations on the phone book (add, display, search, etc.).

6 Time Complexity & Efficiency

6.1 Adding a Contact (`add_contact` function)

- Opening and writing to a file: $O(1)$ operation.
- Adding a contact to the file: $O(1)$.

6.2 Displaying Contacts (`display_contacts` function)

- Reading lines from a file: $O(N)$, where N is the number of contacts.
- Enumerating over the lines: $O(N)$.

6.3 Deleting a Contact (`delete_contact` function)

- Reading lines from a file: $O(N)$.
- Filtering lines based on the deletion option: $O(N)$.
- Writing the modified lines back to the file: $O(N)$.

6.4 Modifying a Contact (`modify_contact` function)

- Reading lines from a file: $O(N)$.
- Modifying the contact in the list: $O(N)$.
- Writing the modified lines back to the file: $O(N)$.

6.5 Searching for a Contact (`search_contact` function)

- Reading lines from a file: $O(N)$.
- Filtering lines based on the search criteria: $O(N)$.

6.6 Display by Tag (`display_by_tag` function)

- Reading lines from a file: $O(N)$.
- Filtering lines based on the tag: $O(N)$.

The dominant factor in most functions is reading and writing lines from/to a file, resulting in $O(N)$ complexity, where N is the number of contacts in the file.

6.7 Time Complexity Graph

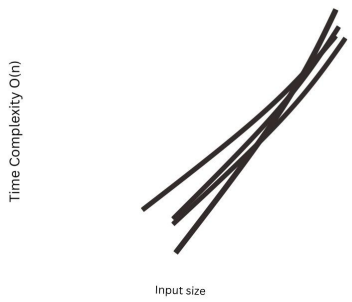

Figure 1: Linear Time Complexity Graph

7 Advantages & Disadvantages

7.1 Advantages:

1. **Organization:** A phone book helps users organize and store contact information in a structured manner, making it easy to locate and manage.
2. **Efficiency:** Users can quickly add, search, modify, and delete contacts, saving time and effort compared to manual methods.
3. **Persistence:** Storing contacts in a file (such as a CSV file) allows for data persistence, meaning contact information is retained across different sessions of the program.

4. **User-Friendly:** With a command-line interface, it may be suitable for users comfortable with text-based interactions.

5. **Educational Value:** The code provides a practical example for learning Python, file handling, and basic data structures.

7.2 Disadvantages:

1. **Limited User Interface:** A command-line interface may not be user-friendly for individuals who prefer graphical interfaces. A graphical user interface (GUI) could enhance usability.

2. **Security:** The current system doesn't include features like user authentication, which could be a concern for securing sensitive contact information.

3. **Scalability:** For a large number of contacts, the system's linear search approach might become inefficient. Implementing more advanced data structures and algorithms could improve scalability.

4. **Error Handling:** The system lacks comprehensive error handling, which could lead to unexpected behavior or crashes if users input invalid data.

5. **Dependency on Terminal Commands:** The system uses terminal commands for clearing the screen, which is not compatible with all operating systems.

6. **Data Integrity:** The use of a simple CSV file for data storage lacks features like data validation and relational integrity that a database system would provide.

7. **Lack of Collaboration Features:** The system doesn't support collaboration or sharing of contact information among multiple users.

8 Software Requirements

To run the phone book management system, you need the following software:

- **Visual Studio Code:** You can insert the codes here; the GUI will pop up on a Python PowerShell. The data saved in the CSV file can also be viewed on VSCode.

9 References

[1] Youtube (2005). YouTube. YouTube. Available at: `https://www.youtube.com`.

[2] GitHub (2023). GitHub. [online] GitHub. Available at: `https://github.com/`.

[3] GeeksforGeeks. (2020). Implementing a Contacts directory in Python. [online] Available at: `https://www.geeksforgeeks.org/implementing-a-contacts-directory-in-python/`.

The source code for the DSA Phonebook Management in Python project can be found on GitHub:

https://github.com/areebaaah/DSA_Phonebbook_Management_in_Python.git