Python Project Description: Contact List Management System

Overview

In this project, you will develop a simple contact list management system in Python using object-oriented programming (OOP) concepts. The system maintains a list of contacts, where each contact has a name and a phone number. The program will allow the user to:

- **Add or Change a Contact:** Insert a new contact or update the phone number of an existing contact.
- **Look Up a Contact:** Retrieve the phone number associated with a given contact name.
- **Remove a Contact:** Delete a contact from the list.
- **Save the Directory:** Write the current contact list back to a file.
- **Exit the Application:** Save all changes and exit.

The contact list is initially loaded from a text file (`phonedata.txt`), which contains name–number pairs. When the user makes changes, they are saved back to the same file.

Project Requirements

- **Person Class:**
  - **Attributes:** `name` and `number`
  - **Methods:**
    - `__init__(name, number=None)` : Create a new person. If a number is not provided, it is set to an empty string.
    - `get_name()` , `get_number()` : Return the name and number.
    - `set_number(new_number)` : Update the contact's number.
    - `__str__()` : Return a string in the format `"name number"` .
    - `__eq__(other)` : Compare two Person objects.
- **ContactList Class:**
  - **Attributes:**
    - `contacts` : A Python list that holds `Person` objects.
    - `file_name` : The filename from which data is loaded and to which it is saved.
  - **Methods:**
    - `load_data(file_name)` : Read the file and populate the contact list.

        - `get_contacts()` : Return a string representation of all contacts.
        - `add(name, number)` : Add a new contact.
        - `find_person(name)` : Return the index of a contact (or `-1` if not found).
        - `remove_person(name)` : Remove a contact and return the removed contact's number (or `None` if not found).
        - `add_or_change_contact(name, number)` : Update an existing contact or add a new one.
        - `save()` : Write all contacts to the file.

- **CLUser Class (Command-Line User Interface):**
    - Presents a menu with options to add/change, look up, remove contacts, save the directory, and exit.
    - Each command is processed by a corresponding method.
    - The application loops until the user chooses to exit.

How It Works

1. **Loading Data:**
   When the program starts, the `ContactList` class is instantiated with the filename `phonedata.txt`. The `load_data` method reads each line from the file, splits it into a name and number, and creates a `Person` object that is added to the list.

2. **User Interface:**
   The `CLUser` class displays a menu of options. The user can:
    - **Add/Change Contact:** Input a name and phone number. If the contact exists, the number is updated; if not, a new contact is created.
    - **Look Up Person:** Enter a name to view the corresponding phone number.
    - **Remove Person:** Delete a contact from the list.
    - **Save Directory:** Save all current contacts to `phonedata.txt`.
    - **Exit:** Save any changes and exit the application.

3. **Saving Data:**
   The `save` method writes each contact back to the file, ensuring that changes persist between program runs.

4. **Program Execution:**
   The `main()` function creates the `ContactList` and `CLUser` objects, then begins processing

user commands in a loop until the user chooses to exit.

You can test the program using the following sample data in `phonedata.txt` :

```
Murthy 905121221
Deepak 121211444
Nivas 1212-12122
Abhijith 122-1215
```

Run the Python script, and the console-based menu will prompt you for input. Use the menu options to add, look up, or remove contacts, and save changes as needed.

Happy coding!