Contact Notebook Application

Key Features and Functionality

1. Contact Class

- **Purpose**: Represents a contact.
- **Attributes**:
  - `name`: Name of the contact.
  - `phone`: Phone number.
  - `email`: Email address.
  - `address`: Physical address.
- **Methods**:
  - `to_dict()`: Converts the contact instance to a dictionary for JSON serialization.
  - `from_dict(data)`: Creates a `Contact` instance from a dictionary.

---

2. ContactBook Class

- **Purpose**: Manages the storage and manipulation of contacts.
- **Key Functions**:
  - **`load_contacts`**: Reads contact data from a JSON file.
  - **`save_contacts`**: Saves contacts to a JSON file.
  - **`add_contact(contact)`**: Adds a new contact to the list and saves changes.
  - **`delete_contact(name)`**: Deletes a contact by matching the name (case-insensitive).

---

3. ContactBookApp Class

- **Purpose**: Handles the GUI and user interactions using Tkinter.
- **Key Features**:
  - **Modern Styles**:
    - Uses the `ttk.Style` to define custom styles for buttons, labels, and the treeview widget.
    - Rounded buttons with hover effects.
  - **Tabbed Interface**:
    - Organized into three tabs:
      - **Home**: Introduction and welcome text.
      - **Add Contact**: A form for adding new contacts.

        - **View Contacts**: Displays the contact list in a table format with an option to delete selected contacts.
    - **Header Creator**:
        - A utility function to create styled headers for each tab.
    - **Input Validation**:
        - Checks for required fields (name, phone, email).
        - Validates phone numbers as 10-digit numeric values using regular expressions.
        - Validates email format using a regex pattern.
    - **Treeview for Contact List**:
        - Displays the list of contacts in a table with columns for **Name**, **Phone**, **Email**, and **Address**.
        - Alternating row colors for better readability.
    - **Persistent Data**:
        - Contacts are stored in a `contacts.json` file for data persistence across application restarts.

Key Points

1. Design and Usability

- **Tabbed Layout**: Separates functionality into logical sections, improving the user experience.
- **Modern Aesthetic**: Uses rounded buttons, alternating table row colors, and clear fonts for a polished appearance.

2. Functional Features

- **Add Contact Tab**:
    - Provides input fields for contact details.
    - Performs input validation before saving.
- **View Contacts Tab**:
    - Displays a table of all saved contacts.
    - Allows users to delete selected contacts from the list.
- **Real-time Updates**:
    - Contacts are added to or removed from the table dynamically without restarting the app.

3. Validation

- **Phone Number**: Ensures it contains only 10 digits.
- **Email**: Enforces a valid email structure (e.g., `example@example.com`).

4. Persistent Data

- Uses a JSON file (`contacts.json`) to store contacts, enabling the application to retain data even after it is closed.

How It Works

1. **Starting the Application**:
    - The app initializes the `ContactBook` to load existing contacts from the `contacts.json` file.
    - The GUI is created with three tabs.
2. **Adding a Contact**:
    - Enter contact details in the "Add Contact" tab and click **Add Contact**.
    - The details are validated, saved to the `contacts.json` file, and displayed in the contact list.
3. **Viewing and Deleting Contacts**:
    - The "View Contacts" tab displays all saved contacts in a table.
    - Select a contact and click **Delete Selected** to remove it from the list and the JSON file.
4. **Error Handling**:
    - Displays alerts for invalid input (e.g., missing fields, incorrect phone number or email format).
    - Alerts when trying to delete a contact without selection.

Key Features at a Glance

- **Object-Oriented Design**: Modular and reusable code.
- **Persistent Data Storage**: Stores contacts in a JSON file for long-term use.
- **Dynamic Contact List**: Automatically updates the view after adding or deleting contacts.
- **Custom Styling**: Provides a clean, modern interface using `ttk.Style`.
- **Input Validation**: Ensures correct data is entered for each contact.
- **User Notifications**: Uses `messagebox` for warnings, errors, and success alerts.