**Project: Single-File Contact Book Application**

**Objective:** Your task is to build a complete, interactive contact book application that runs in the terminal. The application will be contained in a **single Python file** and will store contact data in a list of dictionaries in memory.

This project will test your knowledge of: **Data Types, Conditionals, Loops, Functions, Exception Handling, and Regular Expressions.**

**Step 1: The Main Data Structure**

At the top of your Python script (main.py), create a list that will store all your contact dictionaries.

contacts = []

**Step 2: Define Your Helper Functions**

In the same file, define all the functions that will manage your contact data.

**1. Create Helper Functions:**

* **validate\_phone(phone)**:
  + This function takes a phone string as an argument.
  + Use the **re module (Regular Expressions)** to validate the phone number. The number is valid only if it contains **exactly 10 digits**.
  + It should **return** True if the phone number is valid, and False otherwise.
* **add\_contact(contacts\_list, name, phone, )**:
  + Takes the main contacts list and the details for a new contact.
  + It should first call validate\_phone() to check the phone number.
  + If the phone number is valid, it should create a new dictionary (e.g., {'name': name, 'phone': phone, ' ': }), append it to the contacts\_list, and print a success message.
  + If the phone number is invalid, it should print an error message.
* **view\_all\_contacts(contacts\_list)**:
  + Takes the main contact list.
  + If the list is empty, it should print "Your contact book is empty."
  + Otherwise, it should loop through the list and print the details of each contact in a user-friendly format.
* **Contacts(contacts\_list, name)**:
  + Takes the main contact list and a name to search for.
  + It should loop through the list and **return** the contact dictionary if a matching name is found.
  + If no contact with that name is found, it should return None.
* **update\_contact(contacts\_list, name, new\_phone, new\_ )**:
  + Takes the list, the name of the contact to update, and the new details.
  + It should first find the contact dictionary.
  + If found, and if the new phone number is valid (use validate\_phone()), it should update the phone and for that contact and print a success message.
  + If the contact is not found, print an error message.
* **delete\_contact(contacts\_list, name)**:
  + Takes the list and the name of the contact to delete.
  + It should find the contact dictionary and remove it from the list.
  + Print a success message if the contact is deleted, and an error message if it's not found.

**Step 3: Write the Main Application Loop**

Below your function definitions in the same main.py file, write the logic for the user interface.

1. **The Main Loop:**
   * Create a while True: loop that continuously runs the application.
   * Inside the loop, print a menu of options for the user:
   * --- Contact Book Menu ---
   * 1. Add a new contact
   * 2. View all contacts
   * 3. Search for a contact
   * 4. Update a contact
   * 5. Delete a contact
   * 6. Exit
   * Prompt the user for their choice.
2. **Handle User Choices:**
   * Use an if/elif/else block to handle the user's choice.
   * For each choice, call the appropriate helper function you defined earlier, passing the contacts list as an argument.
   * **Example for "Add Contact":**
     + Ask the user for a name, phone, and .
     + Call the add\_contact(contacts, name, phone, ) function.
   * **For "Exit":** Simply break the loop to end the program.
   * **Exception Handling:** Wrap the user input for the menu choice in a try...except block to handle ValueError if they enter text instead of a number.

**Final Product:**

When you run your main.py file, the application should allow you to perform all the menu actions. Since the contacts are stored in a list, the data will be reset each time you restart the script.