Project: Advanced Phonebook Application – Session - 3

Objective:

Build an advanced phonebook application covering various advanced Python concepts.

Tasks Overview:

1. Structural Pattern Matching
2. Working with Modules and Packages
3. High-Level Operations
4. Iterators and Generators
5. File I/O
6. Error Handling and Debugging

Detailed Instructions:

Part 1: Structural Pattern Matching

- **Objective**: Implement pattern matching for user commands and phone number validation.
- **Tasks**:
    1. Use **match-case** statements to handle different user commands (add, search, delete, list).
    2. Use the **re** library to validate phone numbers (e.g., match patterns like (123) 456-7890 or 123-456-7890).

Part 2: Working with Modules and Packages

- **Objective**: Organize the application into modules and packages.
- **Tasks**:
    1. Create a package structure (phonebook package with **commands.py**, **validators.py**, **models.py**).
    2. Use .**env** to load valid patterns for phone numbers.
    3. Use a **requirements.txt** file for any third-party packages used.

Part 3: High-Level Operations

- **Objective**: Demonstrate tuple packing/unpacking, pointers, variable scope.
- **Tasks**:
    1. Implement functions that use tuple packing and unpacking.
    2. Demonstrate variable scope within functions.
    3. Use pointers (references) to update phonebook entries.

Part 4: Iterators and Generators

- **Objective**: Implement list comprehensions, generators, and use the **itertools** module.
- **Tasks**:
    1. Use list comprehensions to filter and transform phonebook entries.
    2. Implement a generator to iterate over phonebook entries.
    3. Use **itertools** to perform advanced operations like grouping.

Part 5: File I/O

- **Objective**: Read and write phonebook entries to JSON files.
- **Tasks**:
    1. Implement functions to read from and write to a JSON file.
    2. Ensure that the phonebook is loaded from the file on startup and saved to the file on exit.

Part 6: Error Handling and Debugging

- **Objective**: Implement error handling and logging.
- **Tasks**:
    1. Use **try** and **except** blocks to handle potential errors.
    2. Implement assertions to validate function inputs.
    3. Use logging to track the application's execution and errors.