Module 1 Assignment

Weather Data Analysis Tool

Project Description: Create a Python application that can fetch, process, and display weather data for different cities. The application should allow users to input a city and retrieve its current weather data, as well as historical weather data for the last seven days. The user should be able to perform simple statistical analysis on the historical data and save the results to a file.

Key Features:

1. **User Input:**
   - Allow users to enter the name of the city for which they want weather data.
   - Validate the user input to ensure it is a string and not empty.
2. **API Integration (Functions and Modules):**
   - Use a weather API to fetch current and historical weather data.
   - Modularize the application by creating separate functions for fetching data, processing data, and other functionalities.
3. **Math Operations:**
   - Implement functions to calculate the average, median, and mode of the historical temperature data.
4. **String Manipulation:**
   - Format the output to the user in a readable way, including proper units for temperature and other weather-related data.
5. **Iterables (Sequences, Dictionaries, Sets):**
   - Store the weather data in appropriate data structures (like dictionaries for daily data, lists for historical data).
6. **Virtual Environments and Packages:**
   - Set up a virtual environment for the project and document the required third-party packages.
7. **Flow Control:**
   - Use if/else statements and loops to control the flow of the application based on the user input and data retrieved from the API.
8. **Exception Handling:**
   - Implement try/except blocks to handle potential errors, such as network issues or invalid user input.
9. **Date and Time:**
   - Use the `datetime` module to work with dates when fetching historical weather data.
10. **File Processing:**
    - Save the fetched data and statistical analysis results to a file (e.g., CSV or JSON format).

11. **Code Style and Linting:**
    - Follow PEP8 guidelines and use a tool like PyLint to ensure the code is well-formatted and adheres to Python's best practices.

Assessment Criteria:

- Code Quality: Readability, use of functions, adherence to PEP8.
- Error Handling: Application must handle errors gracefully without crashing.
- Correctness: The application should provide correct statistical analysis.
- Documentation: Code should be well-commented, and usage instructions should be provided.
- Packaging: Proper use of virtual environments and third-party libraries.