Weather Analysis System Documentation

Macé Ulysse, Morvan Arthur

December 17, 2023

Contents

1 Introduction

This documentation offers an extensive overview of a Python-based weather analysis system designed for managing weather data and providing a graphical user interface (GUI) for data analysis. The system consists of two primary files: `Days.py` and `gui.py`. These files handle data management and visualization, respectively, creating a comprehensive weather analysis tool.

2 Overview of Files

2.1 `Days.py`

The `Days.py` file serves as the backbone of the weather analysis system, containing two essential classes: `Days` and `Day`. The `Days` class manages weather data for multiple cities and provides various methods for data analysis and retrieval. On the other hand, the `Day` class encapsulates individual weather data points with specific attributes.

2.1.1 Functionality of `Days` Class

The `Days` class offers a broad range of methods to handle weather data efficiently:

- `getCityNames`: Retrieves a set of city names available in the weather dataset.
- `addDay`: Adds a new weather data point to the dataset.
- `getAvgTemperature`: Calculates and returns the average temperature for a specified city.
- `getTotalPrecipitation`: Computes and returns the total precipitation for a given city.
- `plotTemperature`: Generates plots illustrating temperature trends over time for a selected city.
- `correlationMatrix`: Calculates and visualizes correlation matrices for weather attributes.

2.1.2 Role of `Day` Class

The `Day` class represents individual weather data points, each encapsulating attributes such as temperature, precipitation, wind speed, etc. It provides methods to retrieve these attributes based on specific dates and locations.

2.2 `gui.py`

The `gui.py` file complements the functionality of the `Days.py` file by offering a user-friendly graphical interface (GUI) using the PySimpleGUI library. It acts as an intermediary between users and the weather data management functionalities provided by the `Days` class.

2.2.1 Functionalities in `gui.py`

The graphical user interface (`gui.py`) empowers users to interact with the weather analysis system seamlessly. Users can:

- Query specific weather information for cities and dates of interest.
- Generate visual plots illustrating temperature trends over time for selected cities.
- Calculate statistical measures like average temperature and total precipitation for chosen locations.
- Visualize correlation matrices to understand the relationship between various weather attributes.

3 Classes and Functions

3.1 `Days` Class

3.1.1 Exploration of `Days` Class

The `Days` class within `Days.py` encapsulates the core functionalities for weather data management and analysis. Each method within this class contributes to the comprehensive analysis of weather attributes.

For instance, the `getCityNames` method retrieves a list of available city names from the weather dataset. This method plays a crucial role in enabling users to select cities of interest for further analysis.

The `addDay` method facilitates the addition of new weather data points to the dataset. This functionality is pivotal in updating the dataset with the latest weather information for various locations.

3.1.2 Usage of Statistical Methods

Methods like `getAvgTemperature` and `getTotalPrecipitation` offer statistical insights into weather patterns. The `getAvgTemperature` method calculates the average temperature for a specified city, aiding users in understanding the typical temperature ranges for different locations.

Similarly, the `getTotalPrecipitation` method computes the total precipitation for a given city, providing insights into rainfall patterns over time.

3.1.3 Visualization Methods

Methods such as `plotTemperature` and `correlationMatrix` focus on data visualization. The `plotTemperature` method generates graphical plots illustrating temperature trends over time for selected cities. These plots facilitate visual analysis and comparison of temperature variations across different locations.

The `correlationMatrix` method calculates correlation matrices for weather attributes, offering users a visual representation of the relationships between temperature, precipitation, wind speed, and other weather factors.

3.2 `Day` Class

The `Day` class is responsible for representing individual weather data points. Each instance of this class encapsulates weather attributes for a specific date and location. Methods within this class allow users to retrieve specific weather attributes, such as temperature, precipitation, and wind speed, for a given date and city.

4 Usage Instructions

The graphical user interface (`gui.py`) provides an intuitive platform for users to interact with the weather analysis functionalities offered by the system. By following simple steps within the GUI, users can access various weather statistics, generate visualizations, and derive meaningful insights from the weather data.

4.0.1 Querying Weather Information

Users can enter the names of cities and specific dates of interest within the GUI to retrieve detailed weather information. For instance, by entering a city name and a date, users can view temperature, precipitation, and wind speed data for that particular day.

4.0.2 Generating Visual Plots

The GUI enables users to generate visual plots illustrating temperature trends over time for selected cities. These plots aid in the visual comparison of temperature variations across different locations and over specific periods.

4.0.3 Statistical Analysis

Users can calculate statistical measures such as average temperature and total precipitation for chosen cities. This functionality provides insights into typical weather patterns and rainfall distribution for specific locations.

5 Conclusion

In conclusion, the weather analysis system outlined in this documentation presents a robust platform for managing, analyzing, and visualizing weather data effectively. By leveraging the capabilities of the `Days.py` file for data management and the `gui.py` file for user interaction, the system offers an accessible and comprehensive approach to weather analysis.

Our script for comparing two variables is flexible for the names of the cities entered. If a city is not in the list of cities, as `hello` in our example, the function will print this error and not include it in the graph. But it will give a different color to each of the other given cities.

If the names of the variables to compare is not write, it will raise an exception to the user: `raise Exception(f"Error: variable1 or variable2 is not a valid variable")`.

```
a.compare_variables('maxTemp', 'precipitation', 'Paris', 'tokyo', "hello")
```

✓ 0.0s

```
Error: hello is not a valid city name
```

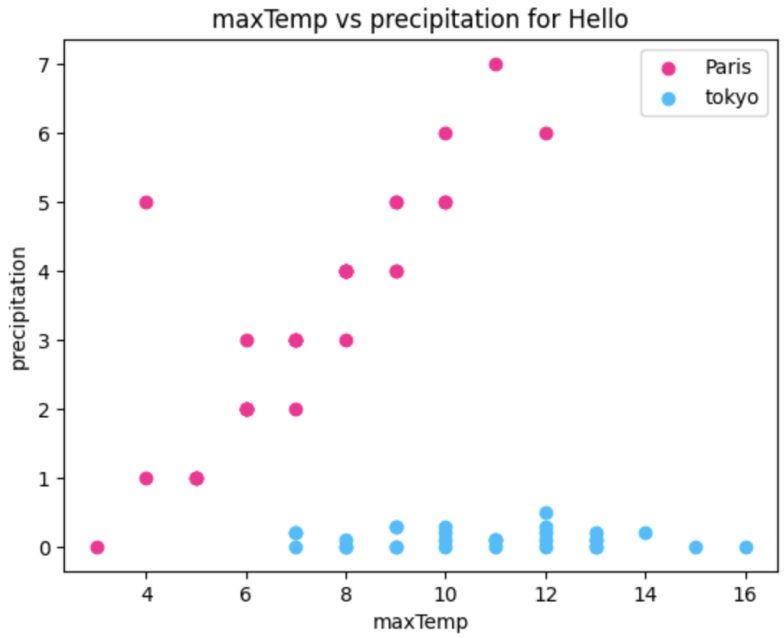

Figure 1: