EE2703-Assignment2-EE24B114

Testing Functions

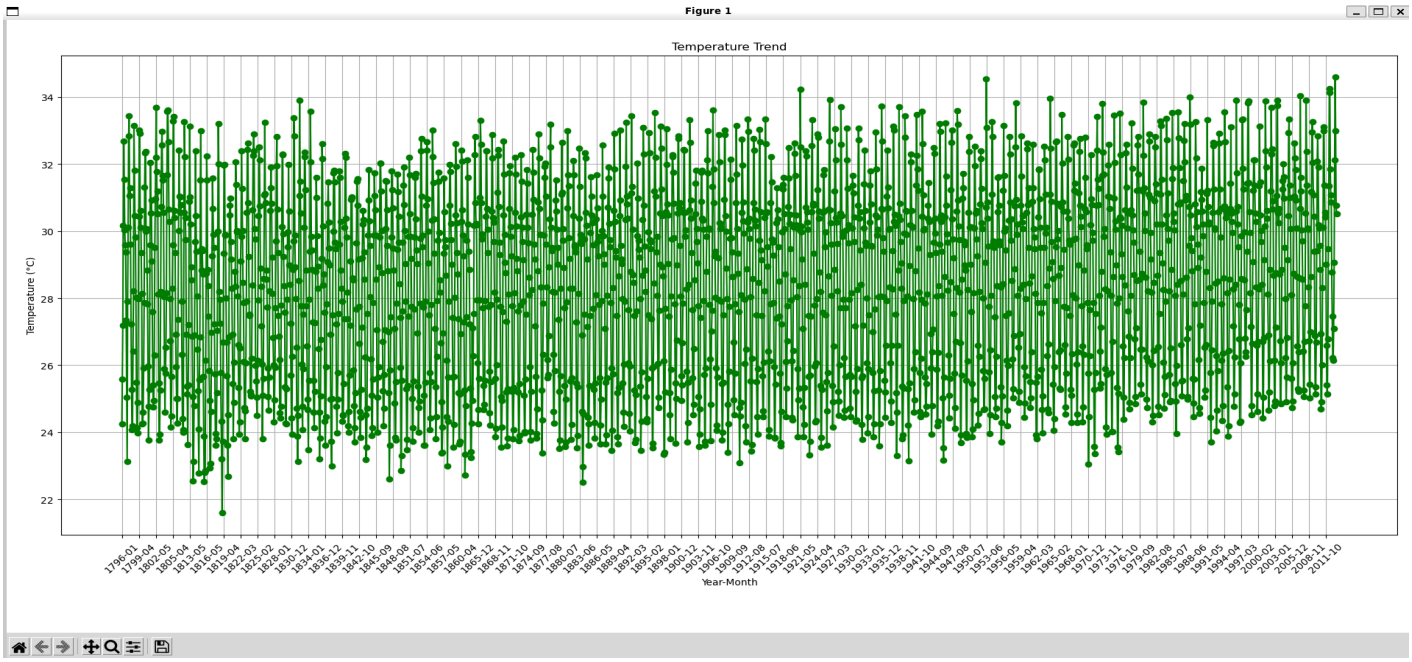
```
1. find_temperature_extremes(filename, city_name)
2. get_seasonal_averages(filename, city_name, season)
3. compare_decades(filename, city_name, decade1, decade2)
4. find_similar_cities(filename, target_city, tolerance=2.0)
5. get_temperature_trends(filename, city_name, window_size=5)
```

CODE Explanation

1. Plot for get_city_temperatures

Take the list of year_month from temperature_data as x-coordinates and list of average temperature as y-coordinates.

Plot the points in circles. Since we have a huge dataset of temperature, we reduce the no:of x-coordinates visible by showing the sliced list of x list by skipping by 35 indexes. Note: The original plotted points are not changed and are plotted for the whole dataset of the city



1. **`find_temperature_extremes`**
   ➜ For each city record, iterate through temperature if `temp>hottest["temperature"]` update hottest, and if `temp < coldest["temperature"]` update coldest.

2. **`get_seasonal_averages`**
   ➜ Check `month`:
   - if `month in (3,4,5)` ➜ add to spring,
   - if `month in (6,7,8)` ➜ add to summer,
   - if `month in (9,10,11)` ➜ add to fall,
   - if `month in (12,1,2)` ➜ add to winter.

     Then compute `average_temperature = sum ÷ count` for the requested season.

3. **`compare_decades`**
   ➜ Filter by decade ranges and calculate averages. Compare:
   - if `decade2_avg > decade1_avg` ➜ warming,
   - if `decade2_avg < decade1_avg` ➜ cooling,
   - if `decade2_avg == decade1_avg` ➜ stable.

4. **`find_similar_cities`**
   ➜ Compute `avg_temp_target_city`. For each city, compute `avg_temp_city` and difference = `abs(avg_temp_city - avg_temp_target_city)`.
   - if `difference ≤ tolerance` ➜ mark as similar.

5. **`get_temperature_trends`**
   ➜ For each year, compute the annual average. For moving average, take a window of surrounding years (from `year - k` to `year + k`, where `k = window_size//2`)and average them.
   - overall slope = `(last_year_avg - first_year_avg) ÷ (last_year - first_year)`.
   - if `next_year_avg > current_year_avg` ➜ warming period,
   - if `next_year_avg < current_year_avg` ➜ cooling period.