CMPS 6100 Lab 02

In this lab, we will perform some data analysis. The National Centers for Environmental Information within NOAA publishes large datasets of climate and weather data. You'll analyze temperature data for New Orleans, specifically from a weather station located in Audubon Park across from Tulane's campus. You will also analyze a dataset of your choice.

Some prompts will require you to edit `main.py` and others will require answers will go in `answers.md`.

Unlike Lab 01, this lab does not have automated tests. Your grade for this lab will be based on your implementations and materials provided in `answers.md`.

Refer back to the README.md for instruction on git, how to test your code, and how to submit properly to get all the points you've earned.

The Data Set

An example dataset is provided in the file nola-temps.csv. This is a comma-separated-values file whose format makes it easy to read in, even easier since python has a built-in module for reading from csv files: csv module.

The format of the file is pretty simple. The first line in the file is a header which contains labels for each column in the file. All following lines contain weather data for an individual day.

Each line has six fields. The first two state the weather station and its name. The next is the date in ISO format. After than is the precipitation measured in inches, followed by the high and low temperature for that day measured in Farenheit.

Not every line has complete data. Any of the precipitation, high, or low fields may be missing.

This data was downloaded from the NCEI's Climate Data Search Tool. You can also download the same data for locations around the country from this portal, for example, from your home town. To download a file containing the same fields as provided for this lab, search for datasets of the type Daily Summaries. When you find a location you like, add it to your "cart" and "check out". For file format, choose "Custom GHCN-Daily CSV, select your date range, and then select the data you want included. You'll want to choose standard"Precipitation (PRCP)" under the Precipitation options, and "Maximum temperature (TMAX)" and "Minimum Temperature (TMIN)" under the Air Temperature options.

1. Examine `nola-temps.csv` and get familiar with its format.
2. Download the same data for a location of your choice. In this lab, you will generate plots for the NOLA dataset, and then for the dataset of your choice. What is the location for your chosen dataset?

The Lab

Your goal in this lab is to read in general datasets formatted as stated above and plot the trends in average temperates over years and decades.

3. Skim through main.py to get a sense of its structure. You have several stubbed out functions that you will implement to read in and analyze the data. Each funtion is documented to explain its purpose, parameters, and required return.

   Towards the bottom you will also find implemented functions to print and plot the data once you've implemented the rest.

   Finally, example usage code is given to illustrate how to use all of the above to actually plot and print the data.

4. Start by implementing `read_in_dataset`. This function takes in the filename of a file formatted asRead in the dataset. You will have to decide on how you choose to structure the data. The one requirement is that you use a dictionary whose keys are individual years in the dataset. Beyond that, the structure is us to you. Example usage: `python daily_summaries`

`= read_in_dataset("nola-temps.csv")` `single_year_data = daily_summaries[2021]` `list_of_years_in_dataset = list(daily_summaries.keys())` You will need to be careful about missing data here. I suggest for ease of analysis later that you fill in any missing precipitation data as `0.0`, and omit any lines which are missing either temperature data point.

After implementing this function, I recommend informally testing it to make sure that it is structured as you expect.

5. Once you have decided on the structure of your data, read it in, and tested that it is structured as expected, implement `calculate_month_averages()`. This function calculates and returns the average daily precipiation, high, and low for a specific month. It returns this imformation as a single tuple containing the three values. If the specified month is missing from the dataset, return `(math.nan, math.nan, math.nan)`.
6. Implement `calculate_year_averages()`. This is much the same as the month averages function, but everything is averaged over a whole year. You can use `calculate_month_averages()` to implement `calculate_year_averages()`.
7. Implement `calculate_decade_averages()`. Again, similar to above, just a different scope of time
8. With the calculation functions implemented, now implement `get_yearly_averages()`. This function returns a list of the averages for each year. The return is structured as a list of tuples. Every tuple contains its year, and the three averages: precipitation, high, and low. By default, there should be a tuple for every year in the dataset. If a list of years is passed into this function, then only return the data for those years.
9. Implement `get_decadal_averages()`. This is the same as `get_yearly_averages()` except now on the decade scale.

   With this, you've finished the number crunching and major implementation part of this lab! Now let's examine the data.

10. Plot the yearly averages for all years in the provided nola-temps dataset. Add images of your plots to your repository and embed them in your `answers.md`

    You can embed `image.png` stored in the same directory as `answers.md` by adding the line:

    `<img src = "image.png" width = "100%">`

    to `answers.md`

11. Looking at your yearly plot, you will probably notice a few years that are outliers, whose averages are way too low to be correct. There are two in particular that stand out. Which years are they and why are they outliers? What could you do to guard against this so that you could reliably identify trends?
12. Outside of the outliers, the yearly plot is pretty messy. Now plot the data, but averaged out over the decades instead of individual years. You can call on the `plot_temperatures()` to do this. You can also implement a helper function to do so as well. A function `get_decades_in_dataset()` is provided to you which returns a list of all full decades included in the dataset. You may use it if it is useful to you.
13. Now generate yearly and decadal plots, but for the location of your choice. Add your dataset and images of your plots to your repository, and embed the plots in answers.md.

Wrapping Up

Don't forget to add all necessary files to your repository, including your chosen dataset, and the images for all plots you have generated. Ensure that it is all pushed to GitHub.

Once you've pushed your completed lab to GitHub, don't forget to submit your link to Canvas.

Epilogue

We've calculated and plotted averages for temperatures across over a century. We didn't even examine the precipitation data, even though we read and processed it. There is so much more that we could do. What about examining trends over individual months? For example, what are the trends for the winters across the dataset? We could, for example, look at the average lows for February of every year. Or we could examine the highs for June of every year.

This all isn't even to mention all of the other data that would could download from NOAA.

With your skills as programmers, insights from the world's data are at your fingertips.