Plotting Hurricane Data from National Hurricane Center with Python

Nnamdi Y Hill and Jared R Byer

Table of Contents

Project Goals - pg.1
Background - pg.1
Installation and Usage - pg.1
Code structure - pg.3
Functionalities and testing - pg. 6
Results - pg. 8
Discussion - pg. 10

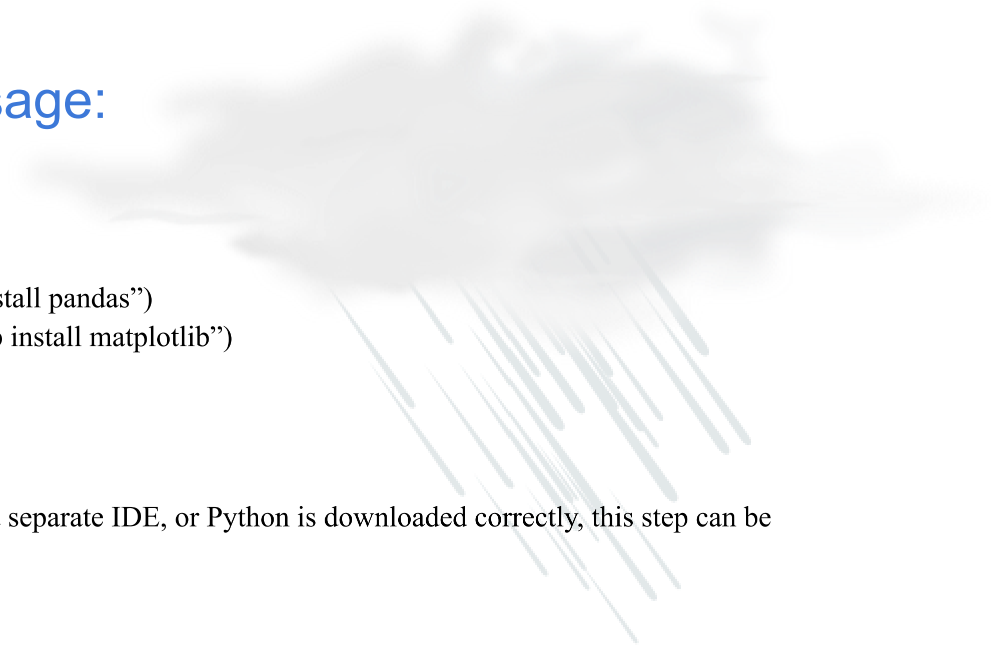
Project Goals:

- Process hurricane data from National Hurricane Center (NHC) using the Atlantic Hurricane Database
- Utilize dynamic plotting for data from CSV file of Atlantic Hurricane Database
- Create user-friendly interface that displays a graph of max wind speed of hurricanes from 1851-present
- Finds average pressure of hurricanes that were processed within the dataframe
- Allow a user-friendly way to choose a custom date range to process hurricane data

Background:

Our project aims to process the data available each year from the NHC into a digestible format for the user. Utilizing the matplotlib.pyplot module, we were able to convert the Atlantic hurricane database (HURDAT2) .txt file into an easy to understand graph that displays the wind speed of each hurricane. The terminal of the program also displays the maximum wind speed of the hurricanes within the timeframe processed, the average pressure of each hurricane, and the count of storms processed thus far. The user also has the ability to process data from a certain range between 1851-2023, and can save the completed graph or stop processing at any time.

Installation and usage:



Requirements:

Python 3.12 or higher
Pandas module (use “pip install pandas”)
Matplotlib module (use “pip install matplotlib”)

Step 1: Download Python

*Note: if you prefer to use a separate IDE, or Python is downloaded correctly, this step can be skipped.

A. Download Python 3.13 or higher here: https://www.python.org/downloads/release/python-3130/ Scroll down to "files" and select the correct version for your operating system. For 64-bit Windows, click "Windows installer (64-bit)".
B. Launch the downloaded setup tool, check both "Use admin privileges" and "Add python.exe to PATH" and click "Install Now" (on macOS, this may differ slightly.)
C. Open Command Prompt or Terminal for macOS (do not open Python).

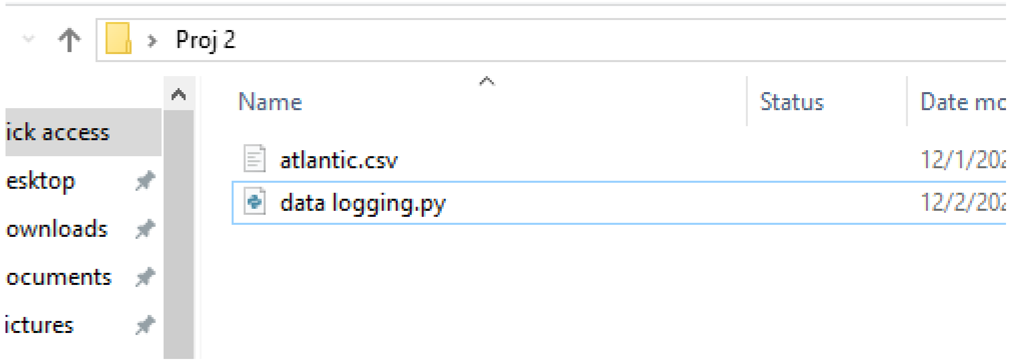
Step 2: Install required modules

A. Once download of Python is complete and terminal/cmd is open, type the following commands to install the required modules, one at a time: (these commands will also work on other IDEs)
   - pip install pandas
   - pip install matplotlib

   *Note: if these commands do not work, make sure Python was added to PATH, and pip was installed along with Python.
B. Close Command Prompt / Terminal.

Step 3: Download required files at https://github.com/deecan14/My-projects/tree/main/Cs%20project4

A. Click "atlantic.csv" and then click "download raw file" on the right-hand side of the screen.
   - Note: atlantic.csv is a slightly modified version of hurdat2.txt, available at https://www.nhc.noaa.gov/data/#hurdat, and is the most recent version as of 12/4/2024.
B. Click "data logging - v3.py" and then click "download raw file" on the right-hand side of the screen
C. Add both files to empty folder, such as in the example shown below:



Name	Status	Date mc
atlantic.csv		12/1/202
data logging.py		12/2/202

Step 4: Run the program

A. Right-click "data logging - v3.py" and select "Open with…", then click Python.
B. If Python and required modules are installed correctly, the program should run.
   - If Python briefly flashes and closes, there is an error. Please see the note below.

C. Follow instructions in the terminal. When typing date range, press enter for all years (1851-2023) or use the format “XXXX-YYYY”, where XXXX and YYYY are years between 1851 and 2023, with XXXX being less than YYYY.

D. Legacy mode is designed as a back-up method of plotting data if the default method is causing issues. To plot using legacy mode, type “2” when prompted by the terminal.
   - Legacy mode is significantly slower at plotting then the default method. It is recommended to use the default method.

E. Optionally, briefly press Ctrl+C at any time to stop the program early once plotting begins.
   - If the program is not stopping, hold Ctrl+C.

F. Finally, the graph may be saved, and the program can be closed by pressing enter in the terminal.

*Please note: if the terminal briefly opens and closes when trying to open the “data logging - v3.py” file with Python, an error has occurred. Verify that the modules were downloaded correctly, the version of Python being used is in the computer’s PATH, and the atlantic.csv file is in the correct location. Refer to the installation and usage section. Optionally, the user may first open a terminal in the file location of the “data logging - v3.py” file, and typing ‘python “data logging - v3.py”’ or ‘python3 “data logging.py”’ to troubleshoot errors.

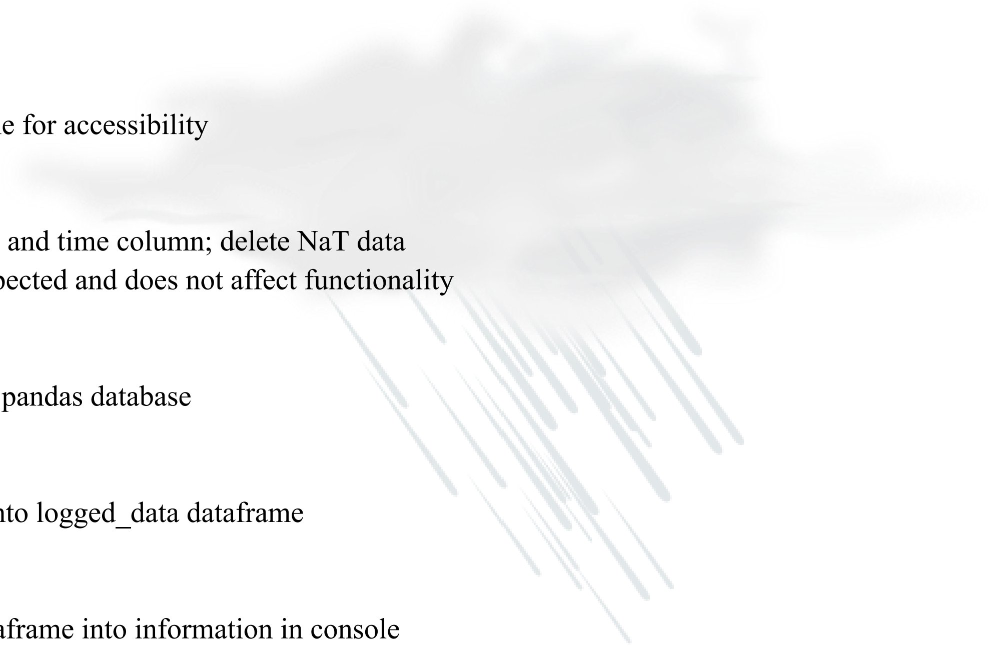
Code structure:

Lines 1 - 5:
Import modules

Lines 7-10:
Find data path of the atlantic.csv file location on user’s computer and read data into pandas dataframe

Line 12:
Print data columns of atlantic.csv

Lines 14-23:
Rename columns of pandas dataframe for accessibility



Lines 27-29:
Create timestamp columns from date and time column; delete NaT data
- Raises warning, which is expected and does not affect functionality

Lines 32-33:
Verifies that required columns are in pandas database

Lines 36-38:
Sorts data from csv and stores data into logged_data dataframe

Lines 41-54:
Function that converts data from dataframe into information in console

Lines 57-77:
Processes wind speed from previous function into data on a graph
- If Legacy=True, plots data using legacy mode (lines 59-71)
- If Legacy=False, plots data using default mode (lines 73-75)

Lines 80-83:
Gets a custom range for years to process (if chosen). If range is empty, run line 83, use default range of 1851-2023.

Lines 84-96:
Loop runs to verify correct syntax of date range chosen by user, using try and except blocks. If incorrect, run lines 94-96, repeating the loop. Otherwise, break from the loop.

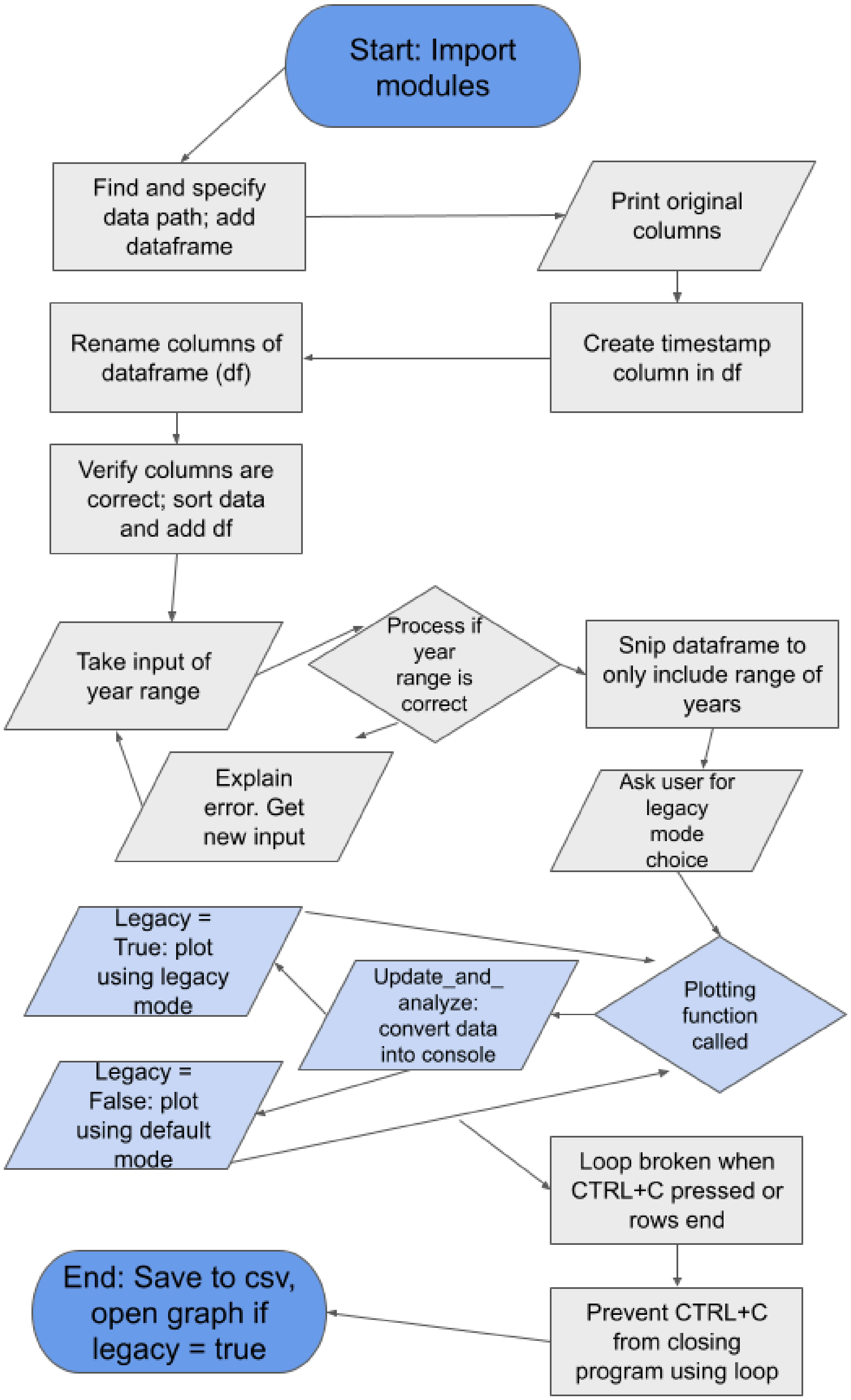
Lines 98-106:
Add a new column in the dataframe for the year by snipping the timestamp column, then get indexes of the start and ending years using the new column. Finally, update dataframe to only include new data.

Lines 108-114:
Option to use legacy or default mode. Legacy is slower but less processor intensive.
- If user input is 2, save option to use legacy mode (line 112)
- Otherwise, save option to use default mode (line 114)

Lines 116-151:
Begins processing data into plot. Utilizes try and except to catch cancellation of program by user pressing CTRL+C.
- If using legacy mode, loop lines 119 - 127
- Otherwise, loop lines 137 - 142

Legacy mode, rather than the default of keeping the graph open and plotting each point, creates a new graph with each new row and then closes the graph. To prevent frequent flashing, legacy mode is purposely slowed using time.sleep and plt.pause. To increase speed, reduce line 127 and line 69 to a decimal lower than 0.5.
- If CTRL+C is pressed, run line 146, stopping loop.

Line 157-170:
Utilizes a while loop and try+except blocks to prevent holding CTRL+C from cancelling the program. Once CTRL+C is let go, data is saved to csv, and the user can close the program.

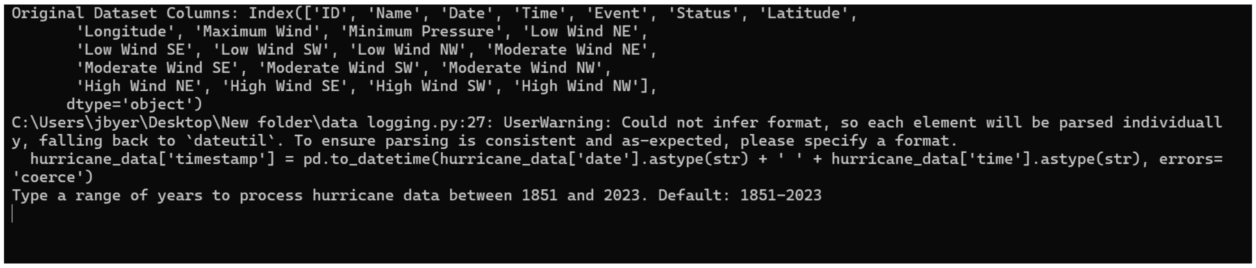

Start: Import modules
Find and specify data path; add dataframe
Print original columns
Create timestamp column in df
Rename columns of dataframe (df)
Verify columns are correct; sort data and add df
Take input of year range
Process if year range is correct
Snip dataframe to only include range of years
Explain error. Get new input
Ask user for legacy mode choice
Legacy = True: plot using legacy mode
Update_and_analyze: convert data into console
Plotting function called
Legacy = False: plot using default mode
Loop broken when CTRL+C pressed or rows end
End: Save to csv, open graph if legacy = true
Prevent CTRL+C from closing program using loop

Functionalities and testing:

Our project has several functionalities, including:

- Utilize data available from the National Hurricane Center to create a plot and various information
    - https://www.nhc.noaa.gov/data/#hurdat
- Plot wind speed data based on hurricanes from a specified range of years by the user]
- Get average pressure of each hurricane and maximum wind speed (printed in terminal)
- Ability to cancel plotting early by pressing CTRL+C
- Save plotting data into CSV format

When first running the program, the terminal will produce a result similar to the figure below:



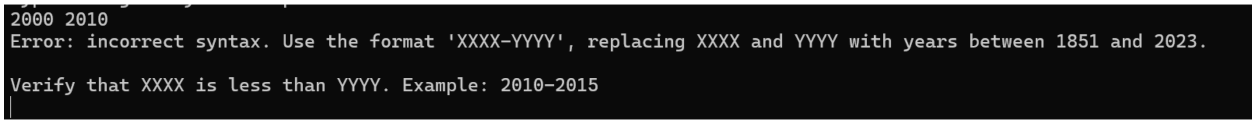
```
Original Dataset Columns: Index(['ID', 'Name', 'Date', 'Time', 'Event', 'Status', 'Latitude',
       'Longitude', 'Maximum Wind', 'Minimum Pressure', 'Low Wind NE',
       'Low Wind SE', 'Low Wind SW', 'Low Wind NW', 'Moderate Wind NE',
       'Moderate Wind SE', 'Moderate Wind SW', 'Moderate Wind NW',
       'High Wind NE', 'High Wind SE', 'High Wind SW', 'High Wind NW'],
      dtype='object')
C:\Users\jbyer\Desktop\New folder\data logging.py:27: UserWarning: Could not infer format, so each element will be parsed individuall
y, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  hurricane_data['timestamp'] = pd.to_datetime(hurricane_data['date'].astype(str) + ' ' + hurricane_data['time'].astype(str), errors=
'coerce')
Type a range of years to process hurricane data between 1851 and 2023. Default: 1851-2023
```

Output:

- The first paragraph printed before the datapath shows the output of line 12, printing the original dataset columns.
- The next paragraph shows a warning, due to no date format specified in the code. This is expected.
- Finally, the program asks the user to type a range of years to process hurricane data between 1851 and 2023.

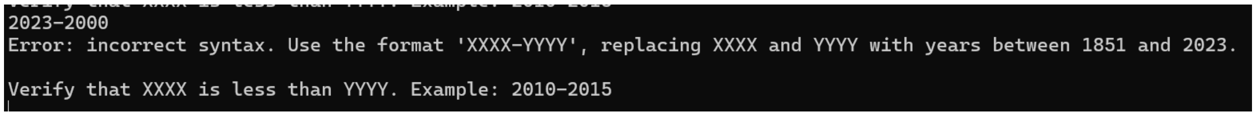
When typing a range of years, a number of things can occur, and the code catches most errors to prevent the program from terminating:

- Incorrect syntax (no hyphen)



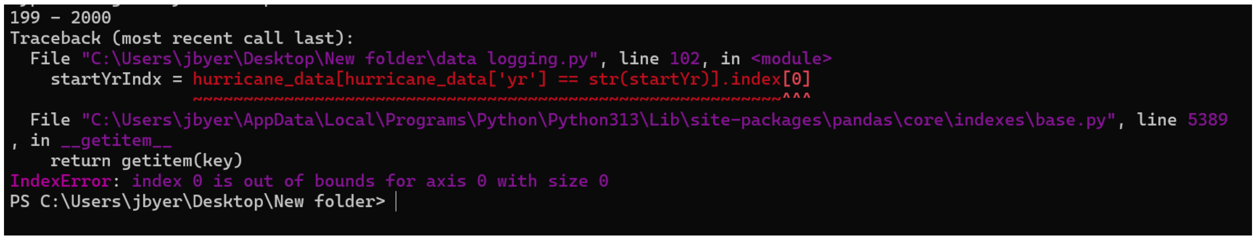
```
2000 2010
Error: incorrect syntax. Use the format 'XXXX-YYYY', replacing XXXX and YYYY with years between 1851 and 2023.

Verify that XXXX is less than YYYY. Example: 2010-2015
```

- Incorrect syntax (XXXX > YYYY)



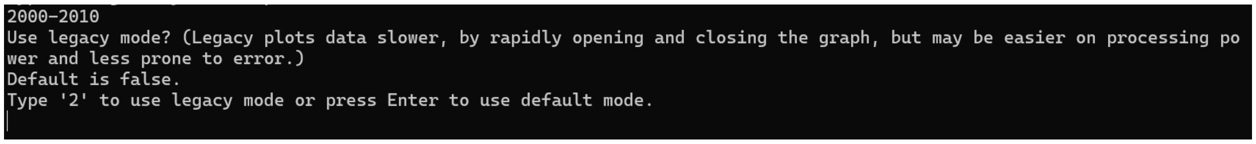
```
2023-2000
Error: incorrect syntax. Use the format 'XXXX-YYYY', replacing XXXX and YYYY with years between 1851 and 2023.

Verify that XXXX is less than YYYY. Example: 2010-2015
```

- Error is thrown if numbers are not in range of 1851-2023



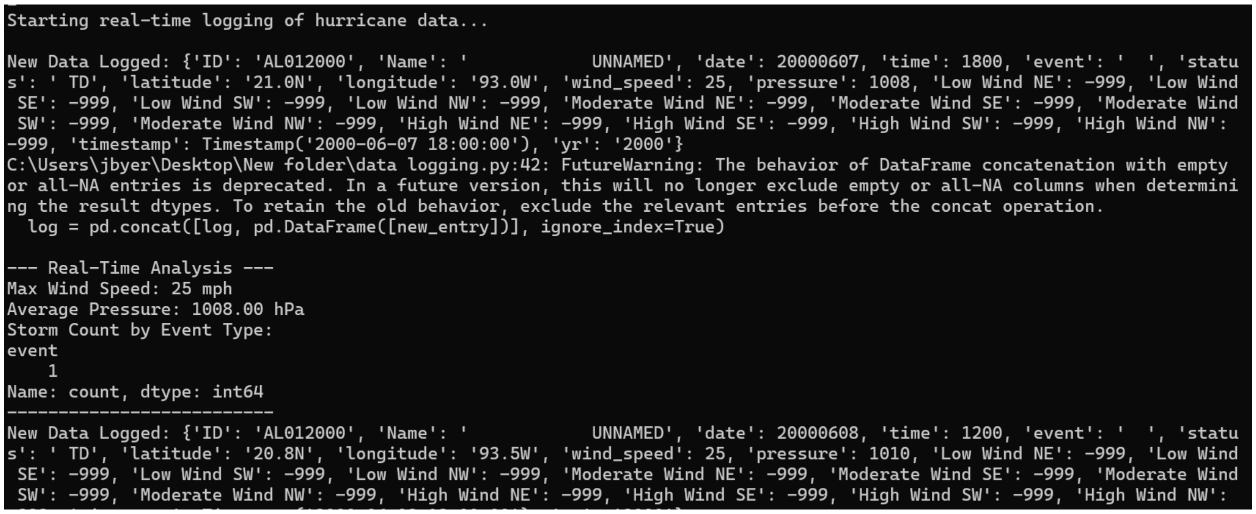
```
199 - 2000
Traceback (most recent call last):
  File "C:\Users\jbyer\Desktop\New folder\data logging.py", line 102, in <module>
    startYrIndx = hurricane_data[hurricane_data['yr'] == str(startYr)].index[0]
                  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^
  File "C:\Users\jbyer\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\indexes\base.py", line 5389
, in __getitem__
    return getitem(key)
IndexError: index 0 is out of bounds for axis 0 with size 0
PS C:\Users\jbyer\Desktop\New folder>
```

After inputting a correct date range, the option for legacy processing is available, but not recommended.



```
2000-2010
Use legacy mode? (Legacy plots data slower, by rapidly opening and closing the graph, but may be easier on processing po
wer and less prone to error.)
Default is false.
Type '2' to use legacy mode or press Enter to use default mode.
```

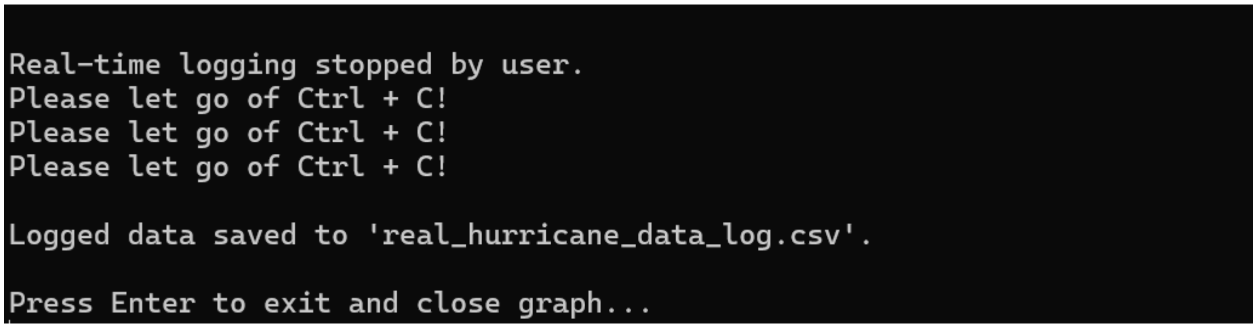
Finally, processing will begin. Here is an example of the terminal during processing:



```
Starting real-time logging of hurricane data...

New Data Logged: {'ID': 'AL012000', 'Name': '            UNNAMED', 'date': 20000607, 'time': 1800, 'event': '  ', 'statu
s': ' TD', 'latitude': '21.0N', 'longitude': '93.0W', 'wind_speed': 25, 'pressure': 1008, 'Low Wind NE': -999, 'Low Wind
 SE': -999, 'Low Wind SW': -999, 'Low Wind NW': -999, 'Moderate Wind NE': -999, 'Moderate Wind SE': -999, 'Moderate Wind
 SW': -999, 'Moderate Wind NW': -999, 'High Wind NE': -999, 'High Wind SE': -999, 'High Wind SW': -999, 'High Wind NW':
-999, 'timestamp': Timestamp('2000-06-07 18:00:00'), 'yr': '2000'}
C:\Users\jbyer\Desktop\New folder\data logging.py:42: FutureWarning: The behavior of DataFrame concatenation with empty
or all-NA entries is deprecated. In a future version, this will no longer exclude empty or all-NA columns when determini
ng the result dtypes. To retain the old behavior, exclude the relevant entries before the concat operation.
  log = pd.concat([log, pd.DataFrame([new_entry])], ignore_index=True)

--- Real-Time Analysis ---
Max Wind Speed: 25 mph
Average Pressure: 1008.00 hPa
Storm Count by Event Type:
event
    1
Name: count, dtype: int64
--------------------------
New Data Logged: {'ID': 'AL012000', 'Name': '            UNNAMED', 'date': 20000608, 'time': 1200, 'event': '  ', 'statu
s': ' TD', 'latitude': '20.8N', 'longitude': '93.5W', 'wind_speed': 25, 'pressure': 1010, 'Low Wind NE': -999, 'Low Wind
 SE': -999, 'Low Wind SW': -999, 'Low Wind NW': -999, 'Moderate Wind NE': -999, 'Moderate Wind SE': -999, 'Moderate Wind
 SW': -999, 'Moderate Wind NW': -999, 'High Wind NE': -999, 'High Wind SE': -999, 'High Wind SW': -999, 'High Wind NW':
```

With each new data logged by the program, the sections “Real-Time Analysis” and “New Data Logged …” will print, informing the user on various information available, including max wind speed and average pressure.

An example of terminal output when holding CTRL+C to stop processing early:



```
Real-time logging stopped by user.
Please let go of Ctrl + C!
Please let go of Ctrl + C!
Please let go of Ctrl + C!

Logged data saved to 'real_hurricane_data_log.csv'.

Press Enter to exit and close graph...
```

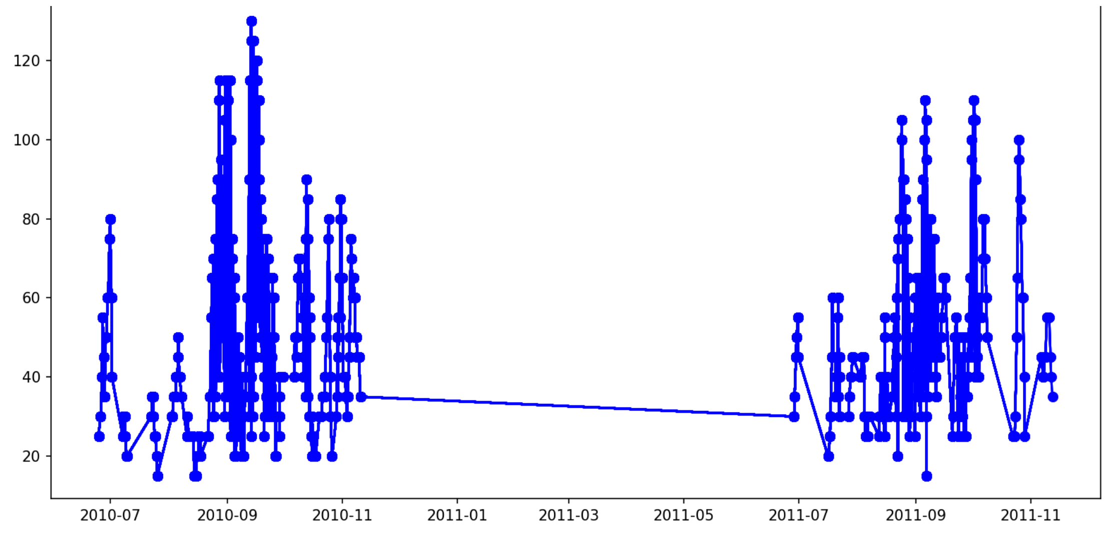
- “Please let go of Ctrl + C!” repeating is due to the user holding CTRL + C, which will repeat until Ctrl + C is no longer being pressed.

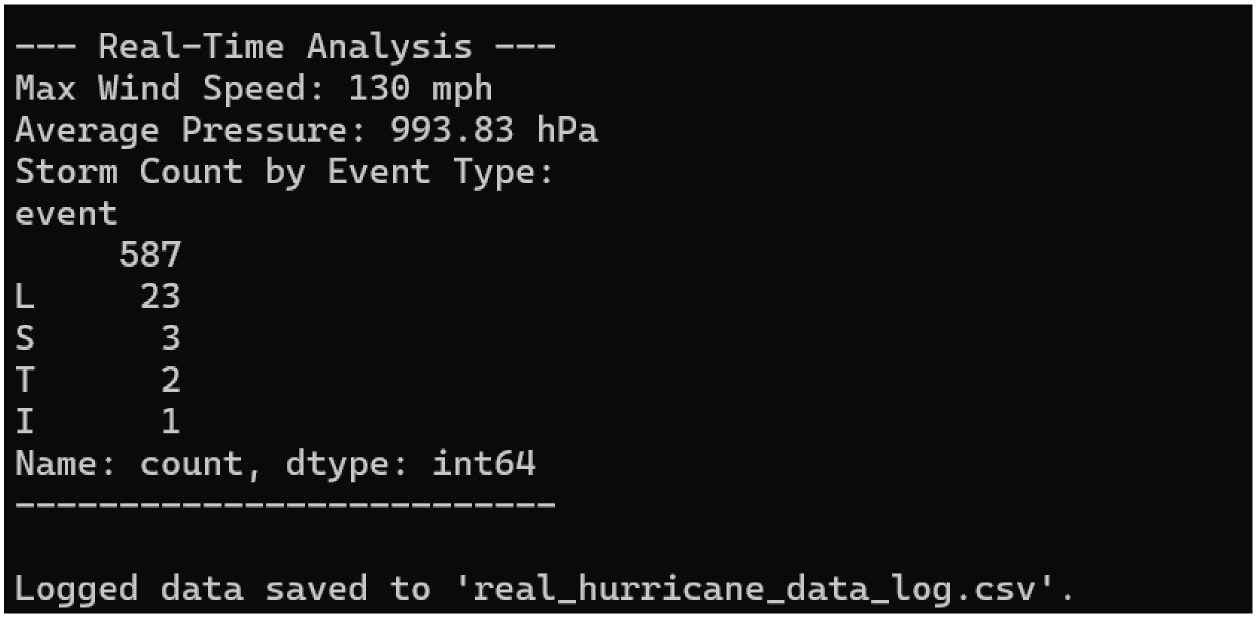
Finally, logged data is saved to ‘real_hurricane_data_log.csv’.

The user can now press enter to close the graph and exit the program.

Results:

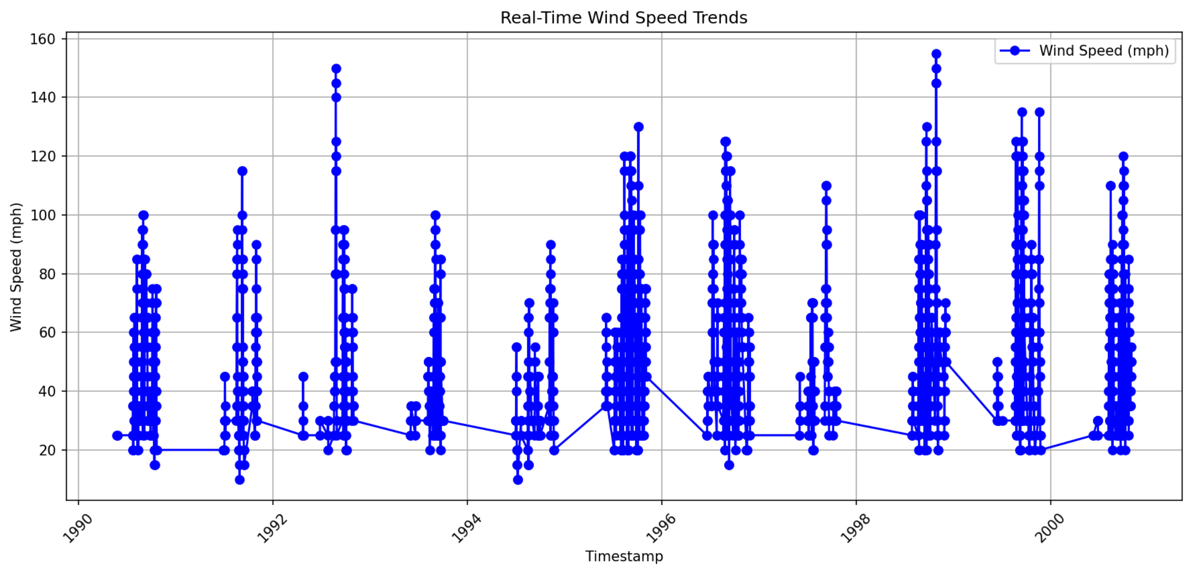
Below is an example of plot and terminal output from hurricanes 2010-2011.

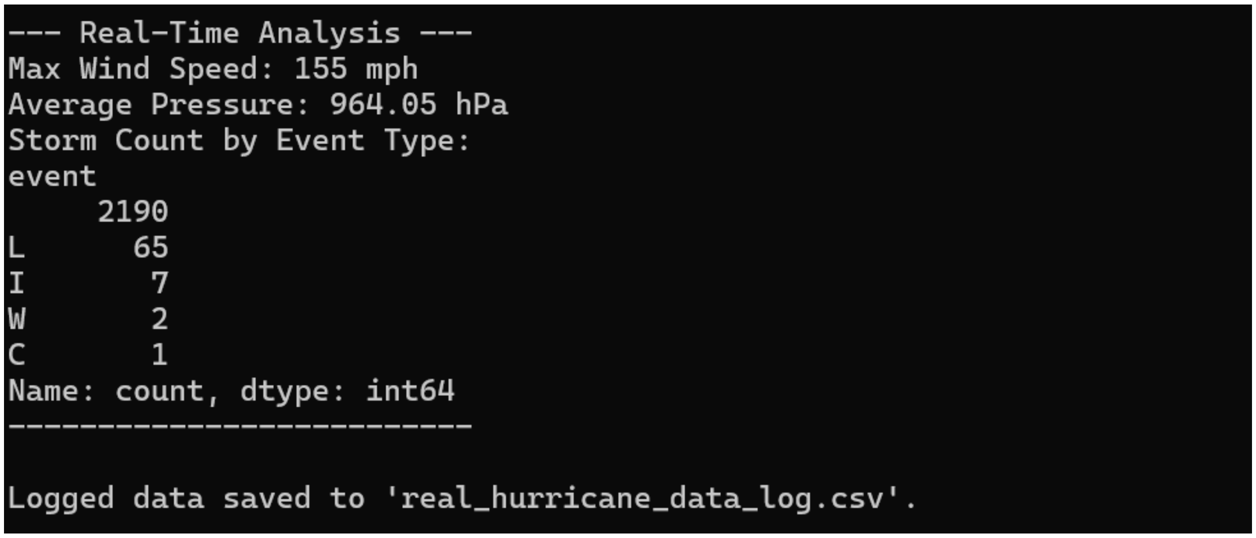




```
--- Real-Time Analysis ---
Max Wind Speed: 130 mph
Average Pressure: 993.83 hPa
Storm Count by Event Type:
event
     587
L     23
S      3
T      2
I      1
Name: count, dtype: int64
--------------------------

Logged data saved to 'real_hurricane_data_log.csv'.
```

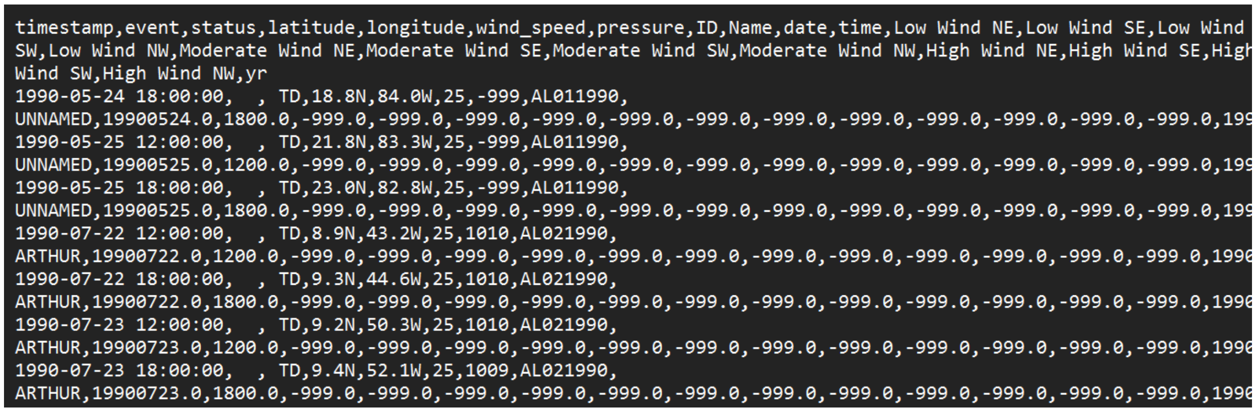
Another example, this time using data from 1990-2000:





```
--- Real-Time Analysis ---
Max Wind Speed: 155 mph
Average Pressure: 964.05 hPa
Storm Count by Event Type:
event
     2190
L      65
I       7
W       2
C       1
Name: count, dtype: int64
--------------------------

Logged data saved to 'real_hurricane_data_log.csv'.
```

Finally, an example of saved data in real_hurricane_data_log.csv is shown below:



```
timestamp,event,status,latitude,longitude,wind_speed,pressure,ID,Name,date,time,Low Wind NE,Low Wind SE,Low Wind
SW,Low Wind NW,Moderate Wind NE,Moderate Wind SE,Moderate Wind SW,Moderate Wind NW,High Wind NE,High Wind SE,High
Wind SW,High Wind NW,yr
1990-05-24 18:00:00,  , TD,18.8N,84.0W,25,-999,AL011990,
UNNAMED,19900524.0,1800.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,199
1990-05-25 12:00:00,  , TD,21.8N,83.3W,25,-999,AL011990,
UNNAMED,19900525.0,1200.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,199
1990-05-25 18:00:00,  , TD,23.0N,82.8W,25,-999,AL011990,
UNNAMED,19900525.0,1800.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,199
1990-07-22 12:00:00,  , TD,8.9N,43.2W,25,1010,AL021990,
ARTHUR,19900722.0,1200.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,1990
1990-07-22 18:00:00,  , TD,9.3N,44.6W,25,1010,AL021990,
ARTHUR,19900722.0,1800.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,1990
1990-07-23 12:00:00,  , TD,9.2N,50.3W,25,1010,AL021990,
ARTHUR,19900723.0,1200.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,1990
1990-07-23 18:00:00,  , TD,9.4N,52.1W,25,1009,AL021990,
ARTHUR,19900723.0,1800.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,-999.0,1990
```

Implementing matplotlib, pandas, and python, the following project goals were successfully achieved:

- Process hurricane data from National Hurricane Center (NHC) by utilizing pandas to convert the CSV file into a pandas dataframe (lines 7-40)
- Utilize dynamic plotting for data from CSV file of Atlantic Hurricane Database through matplotlib (lines 116-151)
- Create user-friendly interface that displays a graph of max wind speed of hurricanes from 1851-present using matplotlib (lines 116-151)
- Finds average pressure of hurricanes that were processed within the dataframe (lines 41-54)
- Allow a user-friendly way to choose a custom date range to process hurricane data (lines 80-106)

Discussion:

Limitations and issues

There have been several issues throughout the process of creating the code, including:

- How to improve performance of matplotlib?
- How to convert data from a .csv file into pandas?
- How to utilize pandas to iterate through rows and plot data using matplotlib?
- How to specify a custom year range for the user to choose in a simple way?
- How to properly install Python from the Python website, add it to system PATH, and install the correct modules outside of an IDE like PyCharm so it is more accessible to the general user?

Through solving issues such as these, we have learned a large amount of new information relating to Python, matplotlib, and pandas, which will be useful in our Computer Science careers.

Unfortunately, there are some limitations in the program:

- Hurdat2.txt was modified into atlantic.csv, thus, it will need to be modified again for the 2024 hurricane season and beyond.
- real_hurricane_data_log.csv is not able to be graphed again by the program. However, the graph processed from the program can be saved separately.
- Matplotlib may take a while to process data over a large timeframe, for example from years 1900-2000.

Conclusion

We have applied subjects learned throughout the course when creating this project, as well as topics that have been learned outside the class. For instance, we used lists, iteration, databases, strings, and try and except clauses to complete the project. We also implemented modules that may have not been discussed as thoroughly within the class, such as matplotlib and the os and time module. Combining these subjects has been a great learning experience, and now that this project has been completed to the best of our ability we feel that it has been a great contribution to our Python and Computer Science learning experience. More specifically, figuring out how to use and implement matplotlib, researching how to effectively use a pandas database, and figuring out how to allow the user to specify which years to choose from have been challenging tasks but rewarding. Debugging the code has also been challenging, especially relating to the improvement in the performance of matplotlib and the pandas dataframe. Overall, this project has been a great learning experience!

Background image: https://commons.wikimedia.org/wiki/File:Weather_icon_-_heavy_rain.svg