Assignment 2: Northern Hemisphere Cyclones

IS597PR. Last modified 8/30/22

The NOAA National Hurricane Center has several databases available. We're going to work with a historical one called "HURDAT2". From http://www.nhc.noaa.gov/data/#hurdat we can download two data files (Pacific and Atlantic) in a non-standard CSV/fixed-width format. The strangeness is due to the mixture of two interrelated line formats, the lack of column headers, and many missing data values (-99 & -999). There are accompanying PDFs on that same webpage that describe the data format in fine detail. Download both HURDAT2 data files and the descriptive PDFs. Read through the PDFs and peruse the data files to get familiar with the custom data structure.

RESTRICTIONS

Because there are many relative novice Python programmers in this class and they'll need to review and understand your code, we don't want you to jump far ahead by using complex features and libraries we haven't discussed yet or that aren't in the assigned readings from class sessions 1-3. Just use the fundamentals for now, only standard library modules.

This means even if you have used Pandas or NumPy modules before, **do not** use them on this assignment, it would be very confusing to Python novices. These are not as much help here as you'd expect anyway, but we will study and use them soon.

CORE GOALS and Specifications

OVERALL TIPS: You'll need to use strings, lists, and dictionaries (or the similar collections.Counter or collections.DefaultDict class instead) and simple text file I/O for this program.

For now, focus on writing a Python program to do these things:

1. Remember the DRY principle (Don't Repeat Yourself). Use looping and functions instead of duplicating lines of code to do identical or similar work again.
2. [Skill: Memory-efficient programming]
   Write your code so that it does not read in or retain the entire data set in memory at once. [We discussed a similar example in class.] The data files are already sorted chronologically, so you can take advantage of that. An incremental algorithm here is more memory efficient and theoretically unlimited in how much data it can process. Many students want to learn about "Big Data" computing – one of the important techniques for that is incremental processing. This means your program will need to release all the detail rows for each storm from memory before it proceeds to load the next storm.
3. As it processes each file, it should compute and print out the following data for each storm:
   a. Storm system ID. Also show its name if it had one.
   b. Did it have any landfall(s) WHILE at Category 1 or higher intensity? (see below for Categories)
   c. The highest ***Maximum sustained wind (in knots)*** value.
   d. The duration of time the storm was tracked.
4. After all the per-storm output above (for BOTH files), your program should output these aggregate annual counts (for both files combined). Use either the string .format() method or an f-string to arrange these as right-justified, fixed-width columns, sorted chronologically, by year. Do not omit years that have no storms at all -- show the zeroes. The columns to show are: "Year"; "Storms" - *Total* number of storms tracked per year; "Cat1" – the number of storms (not rows) that *peaked* at Category 1; "Cat2" = the number that *peaked* in Category 2; "Cat3"; "Cat4"; "Cat5". Refer to

https://www.nhc.noaa.gov/aboutsshws.php for the strength ranges. These yearly totals are combined from both oceans/files.
Example:. The output should begin with:

```
Year Storms Cat1 Cat2 Cat3 Cat4 Cat5
1851      6    2    0    1    0    0
```

5. Go back through your program to improve anything you have unnecessarily hard-coded. This is part of high-quality coding practices. For example, if the year 1851 or 2021 is in your program code as a looping boundary, it shouldn't be. Make the loops work no matter what years are in the data files! Similarly, if your program has a complete filesystem path to the datafiles, it will not work on anyone else's computer. Only use relative path names. Python's file I/O functions default to looking in the same directory where the executing program is located.

SUBMISSION REQUIREMENTS:

1. Your final submitted solution to this assignment must be a Python version 3.8 or higher .py script file. (Not an IPython session or Jupyter Notebook.)
2. Please have only one student from your group submit the completed work. You must include comments at the top of your program file that list every student's full name & NetID, and a brief summary of how you worked together (who wrote which parts, etc.).
3. INCLUDE a text file showing the complete output from your program having processed both files together! If the program prints to the console, then you may copy-and-paste from the console window into an editor to create that file OR have your program print directly to a single text file.

STRETCH GOALS:

These are **optional**, to encourage you to try and learn more beyond the *Core Goals*. We encourage everyone to try implementing **any or all** of these enhancements, but a solution is considered "complete" just by achieving the *Core Goals* above. See the Syllabus on grading for more.

1. [Skill: Writing Functions] In all later assignments, writing good functions will be a "Core" requirement. But on this early assignment, you are just strongly encouraged to try. Doing a good job with functions here so they follow proper rules for scope, parameters, and return values gets credit as a "Stretch".
2. [Skill: Debugging] Use the PyCharm Debugger to help you verify that your program successfully does memory-efficient "incremental file processing". To do this, make it pause the execution by setting a breakpoint at a line after the program has *finished* reading the input files. Then carefully inspect all the list or dictionary type variables you created to make sure that they don't contain things like all the Storm IDs, or the storm Names, and definitely not individual detail rows of data. How confident are you that it's working right? Take screenshots to show evidence of using the debugger in this way. Let us see where in the code execution is paused and the relevant variables expanded if necessary.
3. [Skill: Basic Analytical Calculations] The Tropical Zone is the belt of Earth between the Tropic of Cancer (23.436 degrees North) and the Tropic of Capricorn (23.436 degrees South). Calculate these statistics considering all storms in HURDAT2:
    a. What percentage of all storms ORIGINATE within the Tropics?
    b. What percentage of all storms STAYED ENTIRELY within the Tropics?
    c. What percentage of the TIME are storms located within the Tropics? In your comments, clearly describe how you are interpreting this concept. For example, what will you do with storms having only one detail row?