Assignment 2: Hurricane/tropical storm tracking analytics

IS590PR. Last modified 9/13/17

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 sets of data (Pacific and Atlantic) in a strange CSV format. The strangeness is due to the intermixture of two interrelated line formats, the lack of column headers, and a ton of N/A data values (-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.

PHASE A:

This assignment will have a later phase we'll discuss next week, and you won't submit the work until both phases are completed. This will give you time to work on the fundamentals for a week before we add some more complexity next week.

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

1. As it reads those data files, it should compute and print out the following data for each storm system:
    a. Storm system name
    b. Date range recorded for the storm
    c. The highest ***Maximum sustained wind (in knots)*** and when it occurred (date & time). Notice you can't rely on finding a ***Record Identifier*** of value "W" for this, because not all storm records have these. So, you must find the maximum numerically.
    d. How many times it had a "landfall"
2. After the storm-specific output above, your program should output these SUMMARY statistics, sorted chronologically, by year. Omit years that have no storms:
    a. Total number of storms tracked per year.
    b. Total number of hurricanes tracked per year (many storms didn't reach hurricane category).

PHASE B:

As we'll discuss in class, certain calculations using lat/long coordinates are a lot more complex than you'd think at first from looking at a map. So instead of trying to derive formulas ourselves using spherical trigonometry, we'll take a look at information available here:

- http://www.nhc.noaa.gov/data/
- http://www.movable-type.co.uk/scripts/latlong.html
- https://pypi.python.org/pypi/PyGeodesy/17.8.31

Enhance your program so that it also calculates the following things:

1. For each storm, calculate the maximum and mean (average) speed in knots that the storm center moved (calc from lat+longs and time lag). That is NOT wind-speed! This requires a special function such as provided in the PyGeodesy library or described at the blogpost above.
2. Also, when computing the distance, use an accumulator and report the TOTAL distance each storm was tracked. [as a simplification for all these distances & speeds, assume piecewise continuous linear movements between each recorded sample.]

3. OPTIONAL CHALLENGE: Also calculate compass heading (in degrees) of the storm's movement, between each sample, then for each storm, calculate the greatest directional change per unit time. WHY? Based on storm maps I've seen, I *hypothesize* that hitting landfall often causes a storm to turn more, so this usually occurs after landfall. To find out, compute the percentage of storms where that occurred after the first landfall. If there was no landfall at all for the storm, don't count it either way.

REQUIREMENTS & RESTRICTIONS:

Because there are many novice Python programmers in this class, and we want to develop the concepts as the semester proceeds, I don't want you to unnecessarily jump far ahead by using unnecessary features and libraries. Just use the fundamentals for now. That way the less-experienced students in your group won't get lost with unfamiliar concepts.

1. Your final submitted solution to this assignment must be a Python 3 .py script. (Not an IPython session or Jupyter Notebook.)
2. Write functions where sensible in your code so that you don't produce one monolithic program that has no re-usable components. If you didn't do that during phase 1, now's a great chance to practice refactoring your code.
3. Write proper docstrings, as shown during Week 3 of class.
4. Create a solution that does not read in the entire large data files into memory at once. The data is already sorted on disk, so you can take advantage of that. An incremental algorithm here is more memory efficient and theoretically unlimited in how much data it can handle.
5. Even if you have used the Pandas or NumPy libraries before, don't use them on this assignment, it would only give a small benefit but complicate things for the Python novices. We will study them soon.
6. Even though an OO solution may be appealing here, try implementing this without writing any new object-oriented classes. We will also discuss OO class design in Python soon.

GROUP WORK:

For this assignment, work in a group comprising 2 or 3 students. If you haven't found partners during class, you can post in the class weekly forum to find people. You'll need to work with other students on some of the later assignments, so just introduce yourself and make acquaintances.

Please have ONLY one student from each group submit your group's work. You must have comments at the top of your program file that list every student's full name contributing to your group. As I grade these, I will manually copy the score and all feedback I provide into all of the students' gradebooks.