Arctic Sea Ice Extent: Pandas and CSV Files
ISTA 131 Hw5, Due 2/22/2024 at 11:59 pm

Introduction. This homework is the first in a three-assignment arc intended to introduce you to dealing with data that is stored on disk in csv format using `pandas`. The data is contained in one file, covering the time period from late 1978, when satellite coverage first went online, through 1/17/2024. The file is from ftp://sidads.colorado.edu/DATASETS/NOAA/G02135/north/daily/data/. It is updated daily, but we will use the csv I downloaded on the 21st of January. You may have noticed that the protocol (the prefix up to the colon) in the URL (web address) is FTP, not the usual HTTP or HTTPS. This is an old school way to transfer data over networks (including the Internet) that you don't see very often anymore.

The goal of this assignment is to load the data into `pandas`, clean it, reformat it and write two new files to disk covering 1979-2023 and 2024, respectively, in csv format. In hw6 and hw7, we will visualize and analyze the data, using it to make a prediction.

Instructions. Create a module named `hw5.py`. Below is the spec for six functions. Implement them and upload your module to the D2L Assignments folder.

Testing. Download `hw5_test.py` and auxiliary testing files and put them in the same folder as your `hw5.py` module. Each of the 6 functions is worth 16.7% of your correctness score. You can examine the test module in a text editor to understand better what your code should do. The test module is part of the spec. The test file we will use to grade your program will be different and may uncover failings in your work not evident upon testing with the provided file. Add any necessary tests to make sure your code works in all cases.

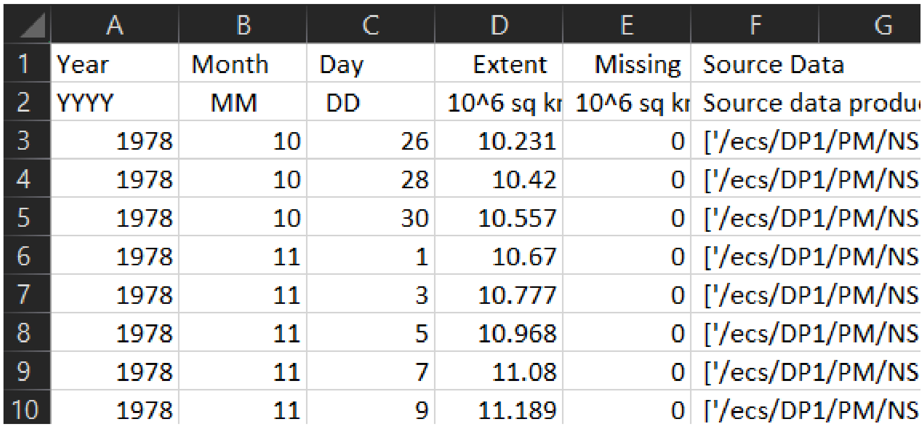
Documentation. Your module must contain a header docstring containing your name, your section leader's name, the date, `ISTA 131 Hw5`, and a brief summary of the module. Each function must contain a docstring. Each docstring should include a description of the function's purpose, the name, type, and purpose of each parameter, and the type and meaning of the function's return value.

Grading. Your module will be graded on correctness, documentation, and coding style. Code should be clear and concise. You will only lose style points if your code is a real mess. Include inline comments to explain tricky lines and summarize sections of code.

Collaboration. Collaboration is allowed. You are responsible for your learning. Depending too much on others will hurt you on the tests. "Helping" others too much harms them in reality. Cite any sources/collaborators in your header docstring. Leaving this out is dishonest.

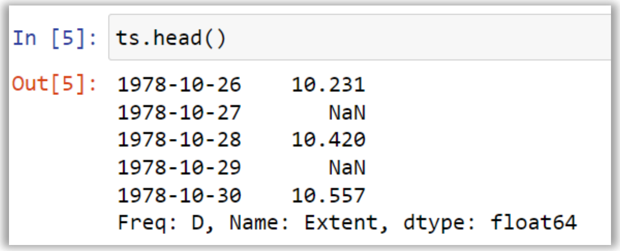
Resources.
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html
https://pandas.pydata.org/pandas-docs/stable/reference/index.html
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.reindex.html
https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

get_data: Use the data in N_seaice_extent_daily_v3.0.csv to create and return a Series object. I suggest using read_csv, https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html. You can do it in two steps: read in a DataFrame, then extract the Series. The csv file looks like this in Excel:



	A	B	C	D	E	F	G
1	Year	Month	Day	Extent	Missing	Source Data	
2	YYYY	MM	DD	10^6 sq kr	10^6 sq kr	Source data produ	
3	1978	10	26	10.231	0	['/ecs/DP1/PM/NS	
4	1978	10	28	10.42	0	['/ecs/DP1/PM/NS	
5	1978	10	30	10.557	0	['/ecs/DP1/PM/NS	
6	1978	11	1	10.67	0	['/ecs/DP1/PM/NS	
7	1978	11	3	10.777	0	['/ecs/DP1/PM/NS	
8	1978	11	5	10.968	0	['/ecs/DP1/PM/NS	
9	1978	11	7	11.08	0	['/ecs/DP1/PM/NS	
10	1978	11	9	11.189	0	['/ecs/DP1/PM/NS	

To get the frame, I suggest the syntax in the Week 6 Thursday Jupyter notebook. There is more than one way to create the Series from the resulting frame. The desired result is imaged below. Notice that the file starts out with data for every other day and I have re-indexed my Series so that it has a position for every day (see the class materials for syntax to do this). A Week 3 notebook may also be helpful.

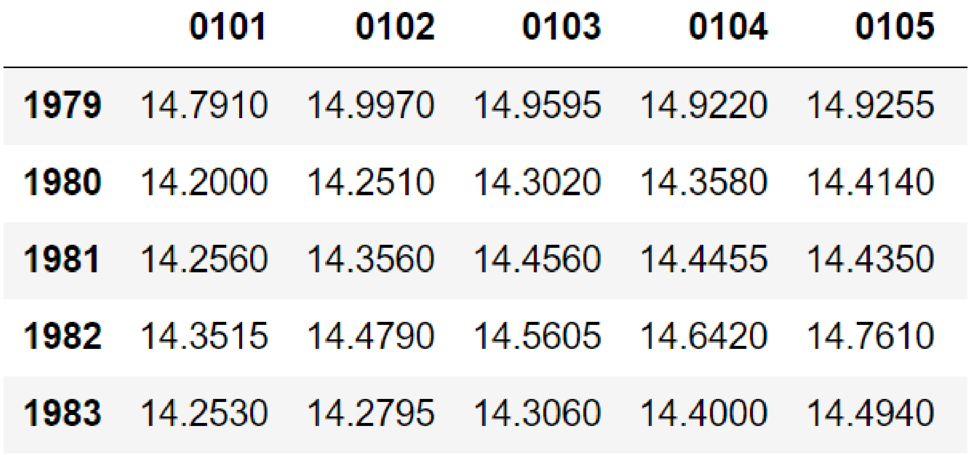


```
In [5]: ts.head()

Out[5]: 1978-10-26    10.231
        1978-10-27       NaN
        1978-10-28    10.420
        1978-10-29       NaN
        1978-10-30    10.557
        Freq: D, Name: Extent, dtype: float64
```

clean_data: This function takes the Series created in get_data and alters it in place by filling in the missing data. For slots that have data for the previous and following days, replace NaN with the mean of those two days. For the extended period of missing data that begins in late 1987 and ends in early 1988, replace NaN with the mean of the previous year and the following year on the same day of the year (hint: 1988 was a leap year). Use sequential for loops (not nested for loops) to accomplish this task – the first loop for the every-other-day missing data, the second for the consecutively missing data. If you do everything inside one loop, when your code reaches the last day of the consecutively missing data, it will have filled in the previous day, fulfilling the every-other-day condition and your code will misfire. A Week 3 notebook will be helpful.

get_column_labels: Generate and return a list of strings that will be used as column labels in a DataFrame that will look like:



	0101	0102	0103	0104	0105
1979	14.7910	14.9970	14.9595	14.9220	14.9255
1980	14.2000	14.2510	14.3020	14.3580	14.4140
1981	14.2560	14.3560	14.4560	14.4455	14.4350
1982	14.3515	14.4790	14.5605	14.6420	14.7610
1983	14.2530	14.2795	14.3060	14.4000	14.4940

Every day of the non-leap year should be represented by an `'mmdd'` string.

`extract_df`: This function takes the cleaned `Series` as its argument and creates and returns a new `DataFrame` (partially pictured above). This `DataFrame` will have the years (`ints`) from 1979 to 2023 as row labels and the strings from `get_column_labels` as column labels. Create an empty `DataFrame` with those labels. Also pass in the argument `np.float64` by keyword `dtype`. Then fill in the data using the appropriate values from the `Series` (hint: nested `for` loops).

`extract_2024`: This function takes the cleaned `Series` as its argument and returns a `Series` containing the data for 2024. This `Series` will have the same format as the cleaned `Series` – `datetime` objects as labels and sea ice extent floats for values.

`main`: Use the above functions to read in the data we want, clean it, and store it to disk in the files `data_79_23.csv` and `data_2024.csv` using the `to_csv` methods for frames and `Series`. When you call `to_csv` on the `Series` object, make sure you pass in this keyword argument: `header=False`.