README - Faddy Sunna, CS236, Fall 2019
Weather station data analysis with PYSPARK

How to run:

On a system with Spark and Pyspark installed along with the correct spark/pyspark environment variables.
In terminal:

1. cd to the location of WeatherStations.py

2. python WeatherStations.py --locations ./pathto/folder --recordings ./pathto/folder --output ./pathto/folder

OR

python WeatherStations.py --l ./pathto/folder --r ./pathto/folder --o ./pathto/folder

OR

sudo ./spark-submit --py-files WeatherStations.py WeatherStations.py --l ./pathto/folder --r ./pathto/folder --o ./pathto/folder

The output is much cleaner with the python command over the spark-submit.

Process:

Importing data in`to spark dataframes & selecting/filtering/grouping/joining:

Dataframe # 1 (recordings) 11.67 million rows, 3 columns

I. **Text files:**
   A. *Import:* I was unable to import the text files into a spark dataframe with properly separated columns due to the variable spacing delimiters. Pandas is capable of separating the columns with multiple delimiters, but it was too time consuming to later convert a Pandas dataframe to a Spark dataframe. I found the best solution to import all the 2006-2009 text files into one spark dataframe as a single column.
   B. *SELECT columns:* Then I select only the columns STN, YEARMODA, and TEMP based off the position of the characters, and also apply the header names. This may be unrealistic if the data changes , but for this dataset it will work. I then filter out all the rows that have the repeating headers.

C. *DATES:* To make it easier to analyze and categorize the data, I converted the yyyymmdd format date to string months. I did this by selecting only the mm digits of that column, then using regex_replace to replace with its string equivalent.This change to the YEARMODA column is applied to all of the years, and removes the year and day since they are not needed.

II. **Dataframe # 2 (stalocs) CSV file: 23.7 thousand rows and 2 columns.**

A. *Import:* Importing from a .csv file is much easier. Setting header = 'true' takes care of the columns.

B. *Filter:* Next I select only the 2 columns USAF and STATE. Then I filtered out States that are not in the list(statelist) of U.S. States and territorys. **This dataframe has 23.7 thousand rows and 2 columns.**

III. **Dataframe # 3 (join): 4.27 million row**

Joined df of recordings and locations. With STN == USAF.

IV. **Dataframe # 4 (final) 636 rows, 4 columns**

State, Month, Avg temp for that state and month for all the years, and the count of data points From the join df, groupBy STATE and YEARMODA, and calculate the average temp for each distinct month and state.

V. **Dataframe # 5 (finaly) 636 rows, 6 columns.**

State, Month, Avg temp, max avg temp, min avg temp, count.
Using the windowspec function, max avg and min avg are found through a groupBy on state and month.

VI. **Dataframe # 6 (finalymax) 53 rows**

From finaly, States and months are selected where the temp is equal to the max average temp.

VII. **Dataframe # 7 (finalymin)**

From finaly, States and months are selected where the temp is equal to the min average temp.

Output Data:
The data is saved as a single .csv file, using coalesce to combine the partitions of each dataframe. It is saved in the folder specified on launch with --o.

File 1: Dataframe # 4 (final) is saved as avgTemp_state.csv
File 2: Dataframe # 6 (finalymax) is saved as Max_avgTemp.csv
File 3: Dataframe # 7 (finalymin) is saved as Min_avgTemp.csv

TASK 1 :Group stations by state
Easily done with the groupBy function on the locations dataframe, grouping USAF and STATE.
This is Dataframe # 2 (stalocs).

TASK 2:For each state, find avg temp for each month
Joined the recordings df and the locations df with USAF = STN.
Grouped the joined df by State and Month, and calculated the average temp for each one, using the agg(avg()) function.
Then rounded the average temp values to three decimal places.
This is Dataframe # 4 (final).

TASK 3: Find months with min & max temp for each state
Using the Window function, I create a windowspec and partionBy State, and orderBy month. This is applied to Dataframe # 4 (final).
Using pyspark functions and my defines windowspec, I define temp_diff as the temperature difference between the max average temp and the min average temp for a state.
Created Dataframe # 5 (finaly), which includes the State, month, temp difference, max, min , and average temp.
Created Dataframe # 6 (finalymax), which SELECTS States, and Months only where the average temp is the MAX average temp.
Created Dataframe # 7 (finalymax), which SELECTS States, and Months only where the average temp is the MIN average temp.

TOTAL RUNTIME: 59.36 seconds. (50% of this time was used for writing the three .csv files)
HARDWARE: AMD Ryzen 2600X 6 core cpu