Use the decorahclimate.txt dataset.

This is a comma delimited data set for climate data for Decorah Iowa from 1893 to 2021.

source: https://www.ncdc.noaa.gov/cdo-web/search

(NCDC = National Climate Data Center; NOAA = National Oceanographic & Atmospheric Administration. The National Weather Service (NWS) is part of NOAA (which is pronounced like 'Noah'.)

The data has this format:

"STATION","NAME","DATE","PRCP","TMAX","TMIN"

STATION is the official weather service station identifier. For the main Decorah site, this is USC00132110.

NAME is the official name of the monitoring station: Decorah.

DATE: in the form of "YYYY-MM-DD" so 2000-04-30 would be April 30 on 2000. The earliest data are from 1893. The 'month code' will be the two digit code for a month: January would be '01'.

PRCP: the precipitation in a 24 hour period, in inches. Sometimes this value is missing, in which case PRCP =""

TMAX: the maximum temperature in a 24 hour period, in Fahrenheit (F). Sometimes this value is missing, in which case TMAX = "".

TMIN: the minimum temperature in a 24 hour period, in Fahrenheit (F). Sometimes this value is missing, in which case TMIN = "".

Download climate_analysis.py and make sure that this python script is in the same folder as decorahclimate.txt.

This project will make extensive use of matplotlib. See the directions on Katie on how to install matplotlib.

1. Write a function, read_climate_data(filename), which will open a file, read it, and put the climate data for each day into a list of lists, clist. The list should have the following format.

   [['1893-03-01', '0.00', '26', '4'], ['1893-03-02', '0.00', '20', '9'], . . .]

2. Write a function, monthly_ppt(clist, year), which will calculate the total monthly precipitation in one year, and will return a dictionary, month_ppt, which will contain the string code for each month (eg '01', '03', etc) as keys and the total precipitation for that month as the value. Run this for 1894, then print the dictionary. The first half of your output should look like this.

{'01': 1.14000, '02': 0.35, '03': 2.93, '04': 3.64, '06': 2.639 . . . }

Note that May (month '05') is missing from the data for 1894.

3. Write a function, average_monthly_ppt(clist, year1, year2) which will calculate the total precipitation for each month in each year, starting at the start year1 and ending at the end of year2. This function will then calculate the average precipitation in each month for that period, and will return a list of months and average precipitation (that is, a list of lists). For example, average_montly_ppt(clist, 1950, 1970) would result in a list with twelve entries, each a list with the month code and average precipitation. eg [[“01”, 3.31], [“02”, 1.39], etc.]. Be sure that your function is able to deal with missing months appropriately!

average_monthly_ppt(clist, 1894,1900) should return the following results (with two decimals shown here – yours will have more decimals):

[['01', 1.11], ['02', 0.81], ['03', 1.87], ['04', 3.08], ['06', 4.31], ['07', 2.04], ['08', 2.08], ['09', 3.16], ['10', 1.84], ['11', 1.57], ['12', 1.23], ['05', 3.18]]

Here, May (‘05’) is last because I used a dictionary, and the keys in a dictionary are not sorted. May was not present in 1894, and so was only added when the data for 1895 were processed.

4. Write a function, average_monthly_maxmin(clist, year1, year2) which will calculate the average minimum temperature and average maximum temperature for each month in each year from year1 to year 2. **(Think carefully about how you will deal with the different number of days in each month, and with missing data**). Then, the function will calculate the average of the average monthly minimum for all the years, for each of the twelve months. Finally, the function will calculate the average of the average monthly maximum for all years, for each of the twelve months. Your function should return a list of lists, with each entry including the month code, the average minimum for that month for the time period, and the average maximum for the month for the time period.

   average_montly_maxmin(clist, 1894, 1900) should produce the following output.

   [['01', 24.86635944700461, 6.880184331797234], ['02', 25.703526768797705, 6.025503688065265] . . .

5. Write a function, extreme_ppt(clist, year1, year2), which will calculate the number of days with **more than 1.5 inches of precipitation** in each year from year 1 to year 2. The function will return a list of lists, each containing the year and the number of days with extreme rainfall. (eg. [[1950, 3], [1951, 0], etc.]

   extreme_ppt(clist, 1894, 1900) should produce the following output:

   [[1894, 1], [1895, 1], [1896, 4], [1897, 2], [1898, 0], [1899, 2], [1900, 3]]

6. Write a function, decade_july_jan_minmax(clist), that will calculate the average maximum July and the average minimum January temperature in each decade, starting with the 1890s and ending with 2019. (We will define decades as starting with a year ending in 0, ending with the last day of the year ending in 9.) The function should return a list of lists, with each element

consisting of the starting year of the decade, the mean July maximum temperature for that decade, and the mean minimum January temperature for that decade. **Be careful how you handle the data for the 1890s.** Your results should begin with:

[[1890, 85.01, 6.01], [1900, 83.94, 7.742], . . .

7. Write a function, decade_extreme_average(clist), which will calculate the average number of days with extreme precipitation (more than 1.5 inches in a day) per year in each decade, starting with 1890 and ending with 2019. The function should return a list of lists, with each element including the first year of the decade and the number mean number of extreme precipitation events per year in that decade.

   [[1890, 1.43], [1900, 3.1],. . .

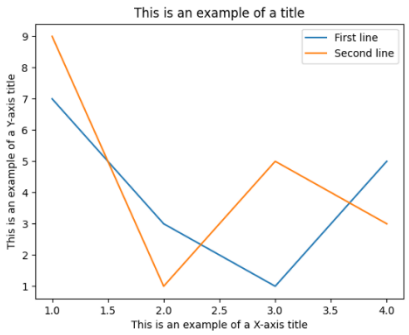
8. Write a function, century_ppt(clist), which will calculate the mean precipitation per month for 1920-1969, and 1970-2019. The function should return a list of lists, with each element including the starting year of the time period, the month, and the mean precipitation.

[[1920, '01', 0.97], [1920, '02', 0.95], . .

9. Write two functions that will graph the output from #7 and #8. The first, plot_temps(data) will take the output from decade_july_jan_minmax and will use matplotlib.pyplot. Create a line graph showing the average January minimum and average July maximum by decade.

Example of a line plot in matplotlib.

```
import matplotlib.pyplot as plt
x_list =[1,2,3, 4]
y_list1 = [7,3,1,5]
y_list2 = [9,1,5,3]
fig = plt.figure()
ax = plt.axes()
ax.set_title("This is an example of a
   title")
ax.set_xlabel("This is an example of a
   X-axis title")
ax.set_ylabel("This is an example of a Y-axis title")
ax.plot(x_list,y_list1, label = "First line")
ax.plot(x_list,y_list2, label = "Second line")
plt.legend()  # add legend to plot.  Must have labels previously.
plt.show()
```



The second function graph_ppt(ppt_list), to make a line graph of the precipitation data from century_ppt(clist). One line on the graph will be for 1920-1969, with other for 1970-2019.

Project requirements and grading.

1. You must include the academic honesty statement at the top of your code: the work submitted must be your own, and you must be able to explain all parts of it.
2. 10 points for each successfully implemented function (= 90 possible)
3. 10 points for project check-ins on Apr 26, Apr 30, and May 5 (3 pts each: 10 pts possible)

4. 10 points extra credit for functions that print the data questions 6 & 7 in a cleanly formatted way, and short document that includes images from 8 & 9, these tables, and your interpretation of the changes in climate, if any, that you observe in these data. (More thoughtful, careful answers will receive more credit here: 10 points is the maximum awarded, fewer will be awarded for superficial thinking.)