CS150 Spring 2018

Project : Climate Analysis
Due: May 18, 2018 at 4:00PM

1 Introduction

Decorah has one of the longer climate data histories for recording stations in the state of Iowa. Twenty-four hour interval maximum temperature, minimum temperature, liquid equivalent precipitation accumulation and snowfall accumulations readings have been recorded since 1893. Missing data any data record is designated by the integer "-99." Temperatures are recorded in whole number degrees Fahrenheit. Precipitation values are recorded in hundredths of an inch. That is, a reading of "123" translates to 1.23 inches. Snow fall is recorded as tenths of an inch, so that a reading of "123" translates to 12.3 inches.

The skeleton Python programs **temperature_analysis.py** and **precip_analysis.py** are provided as a means to begin developing your code to accomplish the tasks outlined below. The completed function `read_data(c_var)`, included in both, is there to facilitate the proper reading of the various data files.

2 Tasks

1. Write Python code that includes the following functions:

   (a) `clc_ave(list_in)` - The function receives a list as input and determines the average value of the list, excluding the missing data days. The function returns the average value.

   (b) `julian_2_mn_day(j)` - The function receives the Julian day number (1 - 365) and returns `int` objects for the corresponding month and day number.

   (c) `daily_ave(t_max, t_min)` - The function returns an array of float values for the daily average temperature (the average of the daily maximum and daily minimum temperatures).

   (d) `annual_ave_t(a_in)` - The input parameter is an array of daily temperatures (max, min, or daily average) and the function returns the annual average temperature - a list of float objects, one each for a given year.

   (e) `monthly_ave_t(a_in, m_no, yr_s, yr_e)` - The input parameters are an array of temperatures, the month number for which the average is to be calculated, the year for which the process starts, and the year for which the process ends. The function returns a list of float values for the monthly average for each year in the year interval specified as `[yr_s, yr_e]`.

   (f) `rank_list(l_in, yr_b)` - The parameters are a `l_in`, list of either integer or float objects ordered chronologically with `yr_b` specifying the first year. The function returns a list whose elements are sublists with two items, the temperature value and the corresponding year. The sublists are ordered by the temperature values in decreasing order. If two temperature value are equal, they are ordered by year, in decreasing order. An example of a returned object is

   ```
   [ [72, 1963], [71, 2001], [69, 1899], ..., [56, 1979], [56, 1938], ...  ]
   ```

   (g) `plot_temp_vs_day(t_list)` - The function plots the temperatures (on the vertical axis) in `t_list` (365 values) versus the Julian day number on the horizontal. The input parameter is a list of temperature sequences. For example,

   `t_list = [t_year, t_ave]` where `t_year` are the average daily temperatures for a given year and `t_ave` is a list of the corresponding daily average temperature.

   (h) `n_day_average(a_in, n)` - This functions calculates the average temperature over an contiguous n-day interval. The parameter `a_in` is an array of temperatures (number of rows = number of years, number of columns = 365). The function returns a same size array with entries corresponding to the average temperature of the n-day interval that concludes on the day corresponding the location in the array. For example, if n = 3 and the returned array is named `a_out`, then

   ```
   a_in = 10  20  30  40  50 ...
   ```

results in

```
a_out = -99 -99  20  30  40 ...
```

The -99 entries signify that a 3-day average cannot be calculated for the first two locations because the days do not have a contiguous 3-day interval ending on that day. The -99 entry should be used for any other interval that does not have complete interval data.

I suggest you explore some of the array manipulation routines available to the `numpy` module. Here is a url you may want to use for this purpose. `http://docs.scipy.org/doc/numpy-1.10.0/reference/`

(i) `n_day_precip_total(a_in, n)` - This function is similar to the `n_day_average(a_in, n)` function, except that the precipitation is summed over n-day interval. The input array `a_in` is the year-row and day-column array of precipitation, and n is the length of the interval. The function returns the n-day interval precipitation total, in hundredths of an inch, in the year-day cell corresponding to the last day of the n-day interval.

(j) `clc_hyetograph(precip_a)` - The input parameter is the array of daily precipitation data for each year. The function calculates the hyetograph for each year and the average value for each day using the yearly hyetographs. The function returns an array with rows corresponding to years and columns corresponding to days. The last row holds the average value for the given day.

(k) `plot_hyetograph(p_list)` - The function plots the hyetograph for each item in `p_list`

2. Design and implement a scheme to deal with missing data when determining the annual average temperature. (See 3 below.) How accurate is the average calculated when using your scheme? How does the accuracy depend on the number of missing data points?

3. Calculate the annual average temperature for each year. Rank the years from greatest to least, and write the results to a text file named `ann_ave_temp_his.txt`. Use the following format, where `m` indicates missing data.

Chronilogical		Ranked		
Year	Ave	No.	Year	Ave
1893	m	001	2012	51.60
1894	m	002	1998	50.62
1895	44.78	003	1931	50.56
.		.		
.		.		
.		.		

4. Repeat 3 for each of the twelve months. Name the output files following the convention `m01_ave_temp_his.txt` for January and so on.

5. Determine the monthly average maximum and minimum temperature for each day of each month. Write the results to a file (one file for each month) using the format shown below. Use the file nomenclature `daily_ave_temps_mn.txt` where `mn` = 01, 02, ..., 12.

Daily Average Maximum and Minimum Temperatures for January

Day	Average Max	Average Min
01	26.60	8.77
02	26.12	8.11
03	25.70	7.50
.	.	.
.	.	.
.	.	.

6. Use the function `plot_temp_vs_day(t_list)` to create a plot of the year 2012 daily temperature version the annual average daily temperature.

7. Determine the top ten hottest / coldest 3-day intervals using the function `n_day_average(a_in, n)`. Report the result in month, day and year form (not Julian day and year) using the function `julian_2_mn_day(j)`.

8. Determine the top ten hottest / coldest 7-day intervals using the function `n_day_average(a_in, n)`. Report the result in month, day and year form (not Julian day and year) using the function `julian_2_mn_day(j)`.

9. Calculate the annual precipitation totals for each year, rank them, and write the results, using a format similar to that in 3, to a file named `ann_prec_his.txt`.

10. Determine the top ten wettest 3-day intervals using the function `n_day_precip_total(a_in, n)`. Report the result in month, day and year form (not Julian day and year) using the function `julian_2_mn_day(j)`.

11. Use daily year-to-day precipitation accumulations to create a hyetograph using the Decorah precipitation data for the driest year, the average hyetograph and the wettest year.

3 Assessment

Requirement	Percentage
Completing Tasks 1(a) - 1(g)	60%
Completing Tasks 1(a) - 1(h)	70%
Completing Tasks 1(a) - 1(i)	80%
Completing Tasks 1(a) - 1(k)	90%
Completing Tasks 1 - 11	100%
Completing Tasks 1 - 11 by 05/07	110%