C4M Homework: Level 2 Week 4 Files, While Loops, Dictionaries

In the workshop this week, we used the file `january06.txt` which held weather data for measurements taken around January 2006. Using this same file, write full programs that answer questions about this data. Submit all your programs on MarkUs under the W5 assignment.

1. Write a program `total_precipitation.py` that calculates and prints the total precipitation over the period represented by the file. Your program should not need any information from the user since it will work for only this file.

2. Write a program `precipitation_on_day.py` that asks the user for one date (expressed as a Julian day). Your program should find and print the total precipitation for only that date. Notice that each Julian day has a number of precipitation readings which you will need to add together.

   Write two versions of your program. First write your program so that it reads the entire file into a list of lists. Then once you have the entire file, find the measurements you need and use them to calculate the total precipitation for the specified day. Print a message with that value. Submit that version of your program as `precipitation_on_day1.py`.

   Then, change your program so that it **only** reads data from the file until the answer can be calculated. Notice that the dates in the file come in an order. So after you've found the date that you need, reading a new date means you have seen all the relevant measurements and you can stop reading.

   For example, suppose the user entered `'1'` as the day. As soon as you read the line with `2006 2 100 0.608 101.2 2.817 97.7 0 -0.986` you should stop reading the rest of the file.

   Use at least one `while` loop – maybe two depending on your design.

   Test your solution using different dates and submit it as `precipitation_on_day2.py`.

3. Write a program `max_rh.py` that asks the user for two days and prints the maximum relative-humidity measurement for the period between those days (including the days themselves.) Don't assume that the days the user provides are in order. For example she might enter `364, 2` or she might enter `2, 7` or she might enter `7, 2`.

   Take a careful look at the file and notice that the dates actually start on 355, count up to 365 and then start again at 1 and go to 31. Although you are always working with the `january06.txt` file for now, your program should **NOT** take advantage of the specific fact that this file starts from day 355 and continues to day 31. Instead, you should write a solution that would work properly on any of the UTM weather data files. That means that you **can** assume that all the measurements for a given date (like 2 or 355) are together and that the days are recorded in order. But the file (from some other year) might go from 334 to 365 and then from 1 to 2.

   Use a `while` loop to avoid reading values once you have all the data you need to calculate the correct output.

4. Write a function that takes in a filename in the same format as `january06.txt` and returns a dictionary whose keys are dates (stored as strings, e.g. `"2006/10/2"`) and whose values are the temepratures on those dates (as floats).