CS417 Assignment 2

Due: Friday, September 18th, 2020, before midnight.

Lateness penalties: Sat/Sun/Mon: 5%, Tue: 10%, Wed: 20%, Thu: 50%, Fri: 100%

Getting Started

Create a folder for assignment 2. Then go to `mycourses.unh.edu` and find the assignment. Download these files into your folder:

- `climate.py`
- `03824-weather-history.csv`
- `plot.py`, and
- `03824-plot.csv` .

The data file is for Durham, NH. I'm also including Honolulu and Fairbanks, Alaska.

Skills

This assignment builds your skills in reading and writing text files, dictionaries and lists, statistics, date manipulation, and coping with poorly-specified and missing data.

Your Job

You should create a program that reads a file with historical weather information, and writes a file with climate information. The input file is a `.csv` table, which lists the precipitation (precip), high temperature (temp), and low temp for many days, over possibly more than 100 years. The output file will also be a `.csv` file, which lists 365 calendar days, along with the average precip, average low temp, average high temp, record low temp, and record high temp for each day. Also, the year of the record low and record high are also added.

I am providing some starting code, in `climate.py`. You should modify this file, and submit your work. **Do not add any other files**. The program should be run thus:

```
python climate.py 03824-weather-history.csv 03824-climate.csv
```

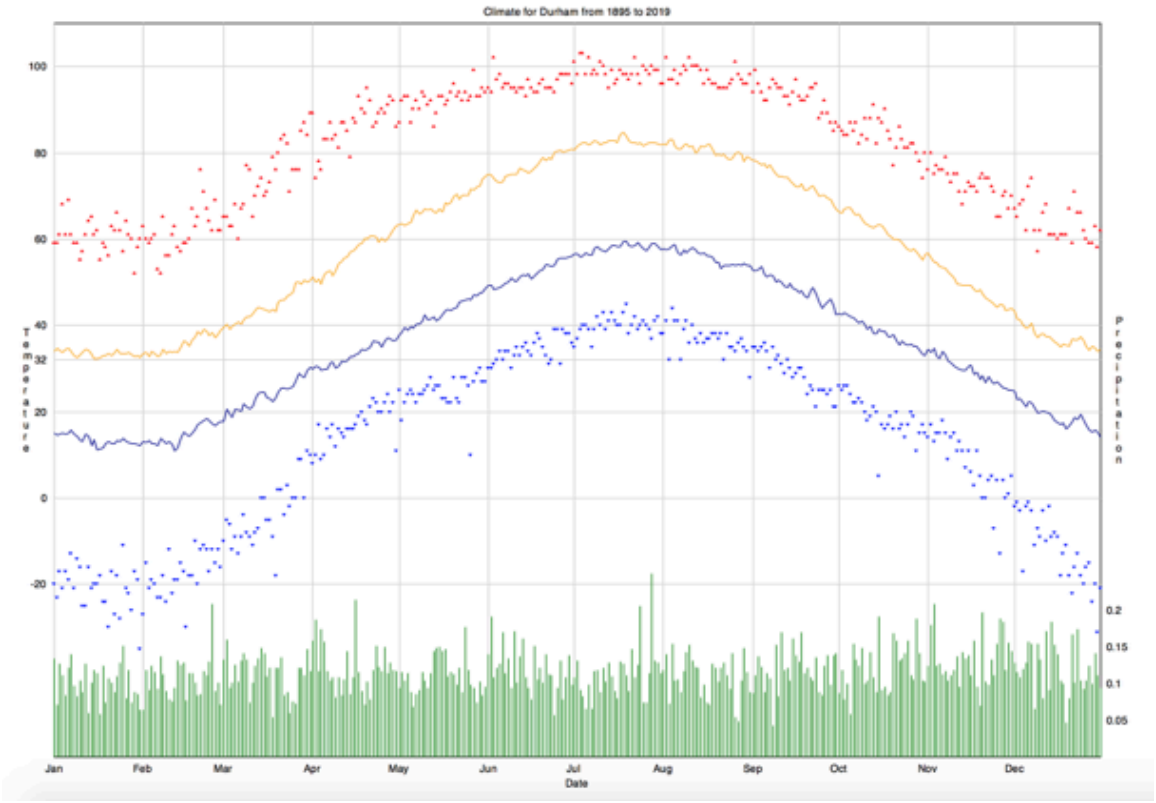
The first argument is the (input) historical data file, and the second is the (output) climate file.

Plotting Your Results

I am providing a plotting program `plot.py`. If your climate output is in `03824-climate.csv`, then running this command:

```
python plot.py 03824-climate.csv 03824-plot.csv
```

should show this plot:



.

Input Data File

The input file is a plain text file. The first line is a descriptive header; you should ignore it. All the other lines may contain four values, separated by commas: date, precipitation in inches, high temperature, and low temperature (both in F).

The format of the date varies. It may be `CCYY-MM-DD` (century, year, month, day, which is the ISO standard format), or `MM/DD/YY` (which has a "Y2K" bug). In the latter case, the century is missing. Thus these two lines may occur in the file (very far apart from each other, of course):

- `1/15/04,0,33,17` (January 15, 1904)
- `1/15/04,0,-5,2` (January 15, 2004)

Even though the date format is ambiguous, the very first date is stored using the un-

ambiguous ISO format, and the days are stored consecutively, so it's possible to keep track of the century as you read each line: whenever the month goes from 12 to 1, a new year begins, and you should check whether the century has changed.

Sometimes the meteorologist on duty did not report all the data. Thus these lines may occur:

- `1895-12-01,,36,15` (Tolerable. use precip=0 instead)
- `3/14/27,,,43` (Bad data, missing low temp)

Do not use bad data lines in your statistics (`continue` the loop, skipping to the next line). You should also *skip data for leap days* (Feb 29th).

Output Data File

Your output file should be a plain text file. The first line should be this header:

```
Day,Avg precip,Avg low,Avg high,Min low,Max high,Min low year,Max high year
```

All the other lines should have eight fields, separated by commas. The first is the date, a string in format `MM/DD`. The seven other fields are numbers.

Go from January to December: The first output line should start with `01/01`, and the last line should begin with `12/31`.

Implementation Details

Your program needs to process many years of data into 365 entries, one for each day of the year. You can do this in two ways. Both involve dictionaries. Remember that a `dict` is a set of (key, value) pairs:

1. Create several `dict`ionaries, one each for avg precip, avg high, avg low, max high, min low, record low year, and record high year. Each key is a `"MM/DD"` date. The corresponding value could be a running sum, or running min or max.

2. Create five dictionaries, for precip, high, and low, and the two record years. Each key is a `"MM/DD"` date. The corresponding value is a `list` of (number, year) tuples.

The `datetime` module includes the `date` class. If you know the `year`, `month`, and `day`, you can create a `date`:

```
the_date = datetime.date(year, month, day)
```

If you have a date, you can get its month and day:

```
    the_month: int = the_date.month
    the_day: int = the_date.day
```

Turning In Your Work

You should *ONLY* turn in one file: `climate.py`. When you are done, go to `mycourses.unh.edu`, find the assignment, click `Submit`, and upload `climate.py`.