CSE 231 Spring 2018

Programming Project: Dictionaries
U.S Pollution Data Analysis

Edit 3/28: the data descriptions have been reordered to match the order in the csv data file, specifically the AQI value is the last value for each chemical.

Edit 2, 3/28: instructor's solution is now multiplying values in parts per million by 1000. The test case results are fixed in this document and also on Mimir.

This project is worth 50 points (5% of your grade). You will use dictionaries (and lists). **It must be completed and turned in before 11:59 on Monday, April 2, 2018.**

Assignment Overview:

Pollution plays a role in the lives of all, and standards of living, health, and environmental consciousness are a concern of many. Thus, pollution data analysis has been a key interest to those including the likes of policy makers, scientists, and activists. This analysis over a period of time allows for a better understanding of trends and pollution effects, in turn influencing policy making, manufacturers, and investment decisions amongst others. Within this project you are given a dataset documented by the U.S EPA [1] containing records of pollution within the U.S for every day from 2000 until 2016. Four of the major pollutants -Nitrogen Dioxide, Sulphur Dioxide, Carbon Monoxide and Ozone- are documented. Given this data, we will create a program to preform a simple analysis of the data. We will determine the trends in pollutants, times, and locations. The details of the project can be found below.

Dataset:

Download *pollution_tiny.csv* and *pollution_small.csv* to access the dataset. The data is organized as follows [2]:

- **Record ID**: a unique record identifier
- **State Code**: The code allocated by the US EPA to each state
- **County Code**: The code of counties in a specific state allocated by the US EPA
- **Site Num**: The site number in a specific county allocated by the US EPA
- **Address**: Address of the monitoring site
- **State**: State of the monitoring site
- **County**: County of the monitoring site
- **City**: City of the monitoring site
- **Date Local**: Date of monitoring

The four pollutants (NO2, O3, SO2 and CO) each have 5 specific columns. Given NO2:

- **NO2 Units** : The unit type measured for NO2

- **NO2 Mean** : The arithmetic mean of concentration of NO2 within a given day
- **NO2 1st Max Value** : The maximum value obtained for NO2 concentration in a given day
- **NO2 1st Max Hour** : The hour when the maximum NO2 concentration was recorded in a given day
- **NO2 AQI** : The calculated air quality index of NO2 within a given day

Limitations: It is important to note that this dataset does not provide data for every year consistently for a given state. This greatly impacts the accuracy of our results, however, the implementation of our project provides good practice in using dictionaries over a large dataset.

Functions:

The following are the functions that you are required to implement:

`open_file():` This function takes no parameters. It's purpose is to prompt for a file name and attempt to open said file. If unable to do so, you should re-prompt. Return the file pointer upon successful opening of the file.

`read_file( fp ):` This function takes the file pointer received from open_file() as a parameter and returns a dictionary which contains all data. Use csv reader to read the file—see notes below. Please note that you must use dictionaries for this project. You may organize your data as follows, where the states are keys and all other data is stored within a list. Because there are multiple records for any given state, you will have multiple record values, represented by r1-r3 in the example below, which are lists in their own right. In the example below we can show Michigan as having 3 records, the contents of the records are displayed by title, as we can see through r1. Note that we do not record the first four values in a line, i.e. do not include Record ID, State Code, County Code, Site Num, or Units in the values you return.

```
{Michigan: [ r1, r2, r3 ] }
r1 = [city, date, no2mean, o3mean, so2mean, comean]
```

Ignore two types of records

(i) for any pollutant whose AQI value is blank—that entire record is considered to be invalid.

(ii) Ignore duplicate records: records that have the same city and date. Keep the first such record in the file and discard any subsequent duplicates. Assume that duplicate records will be in adjacent lines in the file—otherwise your program will run too long and timeout in Mimir. See notes below.

Convert all mean values to floats. Do not round any values.

Important: all mean values should be in parts per *billion* so if the units are parts per *million* you need to multiply the value by 1000.

`total_years( D, state ):` This function takes in the dictionary returned from the read_file() function and a state name that you should prompt the user to input in the main function. It calculates the total pollution for each of the 4

pollutants for the given state over the course of 16 years. You should sum the pollution mean values for every day of the year (e.g. NO2 mean) in order to calculate the total average pollution for each year. This function should return a list containing the average total pollution per year for each pollutant, as well as the maximum and minimum pollution values. The list is ordered by year. We will use the list for plotting, so structure it as follows (where zeros represent the values of NO2, O3, SO2, and CO respectively, and we have 16 lists to represent the totals for each year):

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0,
0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0,
0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0,
0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

You need to find the maximum and minimum total pollution values over all types of pollutants, which will result in one max value and one min value.
In summary, you return a tuple with three items: (list, float, float), specifically (one list (containing 16 lists of totals), one max value, one min value).

`cities(` `D, state, year` **`):`** This function takes in the dictionary returned from the read_file() function, state name, and a year (int) that you should prompt the user to input in the main function. That is, the parameters are a dictionary, a string and an int. Within this function you will be finding the total average pollution per city for a state and year. Given the specified state, you should extract the state's cities as well as the corresponding pollution data for each city (mean pollution data is to be summed – similar to what you did for the previous function, total_years). You should return a dictionary containing the contents of the city and its corresponding pollution data. Your dictionary must be structured as follows: {City: [NO2,O3,SO2,CO] } That is, the key is a city and the value is a list of summed pollution data for the city.

`months(` `D, state, year` **`):`** This function takes in the dictionary returned from the read_file() function, state name (string), and a year (int) that you should prompt the user to input in main. Within this function we want to find the top 5 months with the greatest total pollution for each pollutant. You must return 4 sorted lists where each list has the top 5 months with the most pollution for each pollutant, largest first. The order of the lists is NO2, O3, SO2, CO. For sorting you may find itemgetter useful as you did in Project 7, and remember to sort largest to smallest. An easy way to do this function is to sort and then slice off the top five.

`display(totals_list,maxval,minval,D_cities,top_months)`This function displays the values calculated in the functions described above.

`plot_years( list,minval,maxval ):` This function is provided for you and will plot the total average concentrations for each pollutant for a given state

over 16 years. You should pass the list, minvalue and maxvalue returned from the total_years() function as parameters.

`main()`: Within this function you should call open_file() and read_file() to set up your data. Then, you should continuously carry out the following actions until the user enters quit/Quit.

- Prompt for a state and year in that order ('quit' to quit)
- Calculate total_years()
- Find top cities by cities()
- Find top months with months()
- Display results
- Prompt (yes/no) to generate a graph with plot_years()

Be sure to check whether the state name is valid, i.e. it is in the dictionary. If it does not exist within our dictionary, output an error message and allow them to continually enter a new state until they have entered a valid state.

The user must be able to enter 'quit' at either the state or the year prompt. Furthermore, if 'quit' is entered as a state, the program must stop without unnecessarily prompting for a year.

Notes:

- Use the csv reader. Using the csv reader each line read from the file will be a *list* of items. That is, the split() has already been done. We need to use the csv reader because the data file is messy so simply reading strings and splitting them on commas doesn't work.
  - `import csv  # place this at the top of your file`
  - `reader = csv.reader(fp)  # using the file pointer fp from open_file()`
  - `header = next(reader,None) # how to skip a line`
  - `for line_list in reader:  # how you loop through the file`
- How to check for duplicate cities and dates in read_file() assuming that they only occur in adjacent lines in the file. Create two variables to hold the previous values of city and date. Let's name them previous_city and previous_date.
  - `previous_city, previous_date = "","" # init before loop`
  - `if #some Boolean using previous_city and previous_date`
  - `# after using previous_city and previous_date set them to current city and date`
  - `previous_city = current_city`
  - `previous_date = current_date`

Examples & Test Cases:

Function Test 1: read_file()

```
import csv
fp = open("pollution_tiny.csv")
```

```
D_student = read_file(fp)
D_instructor = {'Michigan': [['Detroit', '3/31/2000', 34.708333, 2.0,
4.916667, 900.0], ['Detroit', '4/1/2000', 37.666667, 19.75, 9.333333, 325.0],
['Detroit', '4/2/2000', 25.833333, 25.666999999999998, 3.958333, 275.0]],
'Maine': [['Presque Isle', '1/1/2006', 2.808696, 30.667, 2.943478, 200.0],
['Presque Isle', '1/2/2006', 3.556522, 25.375, 1.713043, 200.0]]}
```

Function Test 2: total_years()

```
D = {'Michigan': [['Detroit', '3/31/2000', 34.708333, 0.002, 4.916667, 0.9],
['Detroit', '4/1/2000', 37.666667, 0.01975, 9.333333, 0.325], ['Detroit',
'4/2/2000', 25.833333, 0.025667, 3.958333, 0.275]], 'Maine': [['Presque
Isle', '1/1/2006', 2.808696, 0.030667, 2.943478, 0.2], ['Presque Isle',
'1/2/2006', 3.556522, 0.025375, 1.713043, 0.2]]}

T_student = total_years(D,'Michigan')

T_instructor= ([[98.208333, 0.047417, 18.208333, 1.5], [0, 0, 0, 0], [0, 0,
0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0],
[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0,
0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]], 98.208333, 0)
```

Function Test 3: total_years() reading from file

```
fp = open("pollution_small.csv")
D = read_file(fp)
T_student = total_years(D,'Michigan')

T_instructor = ([[4082.771702, 4206.028000000003, 975.7437869999992,
50522.54300000002], [3348.641588000001, 4125.926, 796.8429829999999,
58442.695999999996], [6414.392806000004, 10074.466999999995, 1209.747425,
130065.49799999999], [6058.463629, 9388.806999999993, 1041.455613,
138250.002], [4996.965389000002, 8887.057000000008, 824.6742960000004,
122338.31200000002], [5108.685554999998, 9692.338000000005, 885.188937,
106651.459], [4408.862753, 9565.044000000009, 675.2983520000001,
111601.37800000004], [8.125, 21.0, 2.333333, 229.167], [0, 0, 0, 0], [0, 0,
0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0],
[0, 0, 0, 0], [0, 0, 0, 0]], 138250.002, 0)
```

Function Test 4: cities()

```
D = {'Michigan': [['Detroit', '3/31/2000', 34.708333, 0.002, 4.916667, 0.9],
['Detroit', '4/1/2000', 37.666667, 0.01975, 9.333333, 0.325], ['Detroit',
'4/2/2000', 25.833333, 0.025667, 3.958333, 0.275]], 'Maine': [['Presque
Isle', '1/1/2006', 2.808696, 0.030667, 2.943478, 0.2], ['Presque Isle',
'1/2/2006', 3.556522, 0.025375, 1.713043, 0.2]]}

D_student = cities(D,'Michigan',2000)

D_instructor= {'Detroit': [98.208333, 0.047417, 18.208333, 1.5]}
```

Function Test 5: months()

```
fp = open("pollution_small.csv")
D = read_file(fp)
T_student = months(D,'Michigan',2005)

T_instructor = ([1007.4459379999998, 910.981578, 898.3775449999999,
886.9802479999998, 867.4222229999997], [2099.1289999999995,
1862.3470000000002, 1733.1389999999997, 1471.5560000000005, 1360.181],
[208.13328399999997, 179.59141600000007, 160.52126900000005,
142.36984600000002, 110.60325999999998], [21283.334999999992, 19275.002,
19261.305, 18566.390000000003, 17852.926])
```

Test 1

```
Input a file name: pollution_small.csv

Enter a state ('quit' to quit): Michigan

Enter a year ('quit' to quit): 2006

Max and Min pollution

    Minval    Maxval
      0.00 138250.00

Pollution totals by year

Year       NO2       O3      SO2       CO
2001   4082.77  4206.03   975.74 50522.54
2002   3348.64  4125.93   796.84 58442.70
2003   6414.39 10074.47  1209.75 130065.50
2004   6058.46  9388.81  1041.46 138250.00
2005   4996.97  8887.06   824.67 122338.31
2006   5108.69  9692.34   885.19 106651.46
2007   4408.86  9565.04   675.30 111601.38
2008      8.12    21.00     2.33   229.17

Pollution by city

City                NO2       O3      SO2       CO
Grand Rapids     2154.55  5455.61   144.00 64253.27
Detroit          2254.31  4109.43   531.30 47348.11

Top Months

     NO2       O3      SO2       CO
  851.79  2163.11   186.99 20743.06
  776.95  1857.75   135.16 19833.33
  763.79  1780.62   101.77 19662.50
  723.59  1400.58    95.73 19175.00
  712.19  1342.45    92.24 16266.67

Do you want to plot (yes/no)? no
```

```
Enter a state ('quit' to quit): quit
```

Test 2 (Error check 1)

```
Input a file name: pollution_tiny.csv

Enter a state ('quit' to quit): xxx
Invalid state.

Enter a state ('quit' to quit): Vermont
Invalid state.

Enter a state ('quit' to quit): Michigan

Enter a year ('quit' to quit): quit
```

Test 3 (Error check 2)

```
Input a file name: pollution_tiny.csv
Enter a state ('quit' to quit): quit
```

Test 4 (Error check 3)

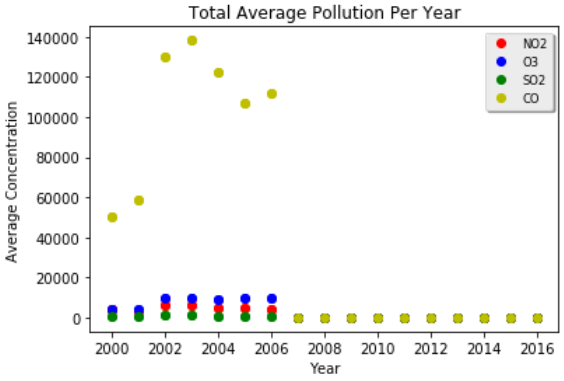
```
Input a file name: pollution_tiny.csv

Enter a state ('quit' to quit): Michigan

Enter a year ('quit' to quit): 2000

Max and Min pollution

    Minval    Maxval
      0.00   1500.00

Pollution totals by year

Year       NO2       O3      SO2       CO
2001     98.21    47.42    18.21  1500.00

Pollution by city

City                NO2       O3      SO2       CO
Detroit           98.21    47.42    18.21  1500.00

Top Months

     NO2       O3      SO2       CO
   63.50    45.42    13.29   900.00
   34.71     2.00     4.92   600.00
    0.00     0.00     0.00     0.00
    0.00     0.00     0.00     0.00
```

```
    0.00     0.00     0.00     0.00

Do you want to plot (yes/no)? no

Enter a state ('quit' to quit): Idaho
Invalid state.

Enter a state ('quit' to quit): Maine

Enter a year ('quit' to quit): quit
```

Test 5: blind test

Test 6 (plot: not on Mimir)

```
Input a file name: pollution_small.csv

Enter a state ('quit' to quit): Michigan

Enter a year ('quit' to quit): 2007

Max and Min pollution

    Minval    Maxval
      0.00 138250.00

Pollution totals by year

Year       NO2       O3      SO2       CO
2001   4082.77  4206.03   975.74 50522.54
2002   3348.64  4125.93   796.84 58442.70
2003   6414.39 10074.47  1209.75 130065.50
2004   6058.46  9388.81  1041.46 138250.00
2005   4996.97  8887.06   824.67 122338.31
2006   5108.69  9692.34   885.19 106651.46
2007   4408.86  9565.04   675.30 111601.38
2008      8.12    21.00     2.33   229.17

Pollution by city

City                 NO2       O3      SO2       CO
Grand Rapids        8.12    21.00     2.33   229.17

Top Months

     NO2       O3      SO2       CO
    8.12    21.00     2.33   229.17
    0.00     0.00     0.00     0.00
    0.00     0.00     0.00     0.00
    0.00     0.00     0.00     0.00
    0.00     0.00     0.00     0.00

Do you want to plot (yes/no)? yes
```




```
Enter a state ('quit' to quit): quit
```

Grading Rubric

General Requirements:

(5 pts) Coding Standard 1-9

(descriptive comments, function headers, etc...)

Implementation:

(2 pts) open_file function (no Mimir test)

(8 pts) read_file function

(4 pts) total_years function

(4 pts) cities function

(4 pts) months function

(7 pts) Pass Test1

(2 pts) Pass Test2

(2 pts) Pass Test3

(2 pts) Pass Test4

(5 pts) Pass Test5 (Blind Test)

(5 pts) Pass Test6 (Plotting – No Mimir test)

References:

[1] https://www.epa.gov/
[2] https://www.kaggle.com/sogun3/uspollution