Homework 8 - CSV Files/APIs

CS 1301 - Intro to Computing - Fall 2022

Important

- Due Date: **Tuesday, November 1st, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
  - API Handout (in Canvas Files)
  - Installing Pip and Requests (in Canvas Files)
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this homework is to extend your knowledge of File I/O to be able to work with important and frequently used data exchange formats like CSV and JSON. We'll be pulling data from the web with APIs, analyzing and writing data from and to CSV files, and converting one data exchange format to the other! This homework will require you to implement 5 functions. You have been given the HW08.py skeleton file to aid you with your responses.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not True
```

Written by Chris Liding (zliding3@gatech.edu), Farah Alkadri (falkadri3@gatech.edu), & Priyam Kadakia (pkadakia3@gatech.edu)

PART 1 - CSV Files

Helpful information

For Part 1 (CSV files), the `.csv` file being read from will be formatted as shown below. Be sure to download the provided file from Canvas, and move it into the same folder as your `HW08.py` file.

By default, your computer will likely use Excel on Windows, or Sheets on Mac, to open the `.csv` file, but don't be alarmed: reading values from a `.csv` file is no different than a `.txt` file; the only difference lies in the formatting of the data.

The activities CSV (and other `.csv` files like it) will contain an activity name, its rating (out of 100), the time commitment required (in hours), and the month in which the activity can be performed. Each data entry will be separated by a newline. Below is an example of how the `.csv` file will be formatted. Note that the first row contains the headers for each of the columns, and does not contain any actual data.

Format of the activities CSV:

```
activity,rating,timecommitment,month
activity1,rating1,timecommitment1,month1
activity2,rating2,timecommitment2,month2
...
```

Best Activity

Function Name: fallActivities()
Parameters: fileName(`str`), interestedList(`list`), month(`str`), minimumRatio(`float`)
Returns: greatestTupule(`tuple`)
Description: You and your friends are trying to decide which fall activity you want to do on Saturday. Given the name of a CSV file with the list of activies, ratings, time commitment, and the best month in which to do the activites, return a tuple with the activity in the given month with the greatest rating to time commitment ratio as (activity name, ratio). The ratio must be greater than the minimum ratio provided in the parameters. When trying to find the best fall activity, you shoud only consider the activites that are in `interestedList`.

Note: If there are no activities in the specified month or with a rating to time commitment ratio above the minimum ratio, then you need to return `"No activities this month!"`.

```
>>> filename = "activity.csv"
>>> interestedList = ["hiking", "watch football", "have a picnic",
                      "apple picking", "eat candy corn", "visit haunted house",
                      "eat pie", "bob for apples"]
>>> month = "September"
>>> minimumRatio = 30.0
>>> fallActivities(filename, interestedList, month, minimumRatio)
("have a picnic", 77)
```

```
>>> filename = "activity.csv"
>>> interestedList = ["pumpkin patch", "trick or treating", "jump in leaves",
                      "drink chai latte", "visit haunted house",
                      "make caramel apples"]
>>> month = "October"
>>> minimumRatio = 35.0
>>> fallActivities(filename, interestedList, month, minimumRatio)
('pumpkin patch', 40.0)
```

Available Months

Function Name: funFallFavs()
Parameters: filename (`str`), letter (`str`)
Returns: monthDict (`dict`)
Description: Write a function that takes in the name of the activities CSV file and a single letter and returns a dictionary that contains the months mapped to number of activities during that month that contain the given letter. Your function should be case insensitive, i.e. it should treat "a" the same as "A" and so on.

Note: If there are no activities that contain the letter passed in, return the string `"Letter not found."`

```
>>> filename = "activity.csv"
>>> letter = "b"
>>> funFallFavs(fileName, letter)
{September: 1, November: 2}
```

```
>>> filename = "activity.csv"
>>> letter = "t"
>>> funFallFavs(fileName, letter)
{September: 1 , October: 4 , November: 3}
```

PART 2 - APIs

Helpful information

For this assignment, we will be using the REST countries API (https://restcountries.com/#api-end-points-v2). **Please read through the documentation, as it will be extremely helpful for completing this assignment.**

For all of your requests, make sure to use version 2 of the REST countries API (V2), not version 3 (V3.1).

If you make a valid request with the URL: https://restcountries.com/v2/alpha/usa, you will receive the following JSON formatted response:

```
{
        "name": "United States of America",
        "topLevelDomain": [".us"],
        "alpha2Code": "US",
        "alpha3Code": "USA",
        "callingCodes": ["1"],
        "capital": "Washington, D.C.",
        "altSpellings": ["US", "USA", "United States of America"],
                .
                .
                .
}
```

If you make a request with an invalid URL, you will receive the following response:

```
{
  "status": 400,
  "message": "Bad Request"
}
```

Please read the description of each problem carefully to ensure you're correctly handling errors.

Neighbors

Function Name: shareBorder()
Parameters: country1 (`str`), country2 (`str`)
Returns: areNeighbors (`bool`)
Description: How can we check if two countries are neighbors? Write a function that uses the rest-countries API to check whether two countries passed in as parameters are neighbors or not. Your function should return `True` if the two countries share a border, and `False` if they don't.

Note: A country does not share a border with itself.

```
>>> shareBorder("United States of America", "Canada")
True
```

```
>>> shareBorder("China", "Japan")
False
```

Currency Popularity

Function Name: currencyRatio()
Parameters: continent (`str`), currency (`str`)
Returns: ratio (`float`)
Description: You're planning to go on vacation, but you only have one type of currency. Given the name of a continent and a currency, return the ratio of the number of countries in which your chosen currency is official to the total number of countries in the continent **rounded to 2 decimal places**. You can assume that both the continent and currency will be valid.

Note: The currency passed in will be the currency's common name, **not** its currency code.

```
>>> currencyRatio("Europe","Euro")
0.49
```

```
>>> currencyRatio("Americas","United States dollar")
0.16
```

Country Information

Function Name: countriesInfo()
Parameters: countriesList (`list`)
Returns: None (`NoneType`)
Description: You want to create a CSV file that contains information about the specific countries you are given. Write a function that takes in countriesList, a list containing names of countries (`str`). Your function should create a CSV file called `countries.csv` that follows the below format:

```
Country,Capital,Population,Languages,Currencies
Country1,Capital1,Population1,Languages1,Currencies1
Country2,Capital2,Population2,Languages2,Currencies2
...
```

For the currencies and languages only include the name of the currrency and language, and if a country uses more than one currency or langauge write all the currencies and languages that it uses seperated by a `"-"`. If the country's name is invalid, don't add it to your file. You should always create the CSV file even if it only contains the header. There should not be any additional `"\n"` at the end of the file.
Note: The country name written on the file should be the name found in the list given.
Hint: Use the following URL format: https://restcountries.com/v2/name/{name}

```
>>> countriesInfo(['Canada','Cuba'])
```

Output: `countries.csv`

```
Country,Capital,Population,Languages,Currrencies
Canada,Ottawa,38005238,English-French,Canadian dollar
Cuba,Havana,11326616,Spanish,Cuban convertible peso-Cuban peso
```

```
>>> countriesInfo(['Syria','France','Spain'])
```

Output: `countries.csv`

```
Country,Capital,Population,Languages,Currrencies
Syria,Damascus,17500657,Arabic,Syrian pound
France,Paris,67391582,French,Euro
Spain,Madrid,47351567,Spanish,Euro
```

Grading Rubric

Function	Points
fallActivities()	20
funFallFavs()	20
shareBorder()	20
currencyRatio()	20
countriesInfo()	20
Total	**100**

Provided

The `HW08.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW08.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the “Resubmit” button at the lower right-hand corner of Gradescope. You do not need to submit your `HW08.py` on Canvas.