Homework 8 - API

CS 1301 - Intro to Computing - Fall 2021

Important

- Due Date: **Tuesday, November 2nd, 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 for you to enhance your understanding of APIs. The homework will consist of 5 functions for you to implement. You have been given the HW08.py skeleton file to fill out. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin.

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 Jakob Johnson (jjohnson473@gatech.edu) & Nelson Alexander (nalexander34@gatech.edu)

Helpful Information

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

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

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

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

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.

Average Country Population

Function Name: averagePopulation()
Parameters: regionalBloc (`str`)
Returns: average population (`float`)
Description: For a geography class, you are interested in finding the average population of countries in different regional blocs. Given the name of a regional bloc (`str`), calculate the average population of the countries in that regional bloc (`float`). Round you answer to two decimal places.
Note: You can assume that the regional bloc will always be valid.

```
>>> averagePopulation("EU")
14457962.26
```

```
>>> averagePopulation("EFTA")
3605233.25
```

Common Countries

Function Name: commonCountries()
Parameters: langTup1 (`tuple`), langTup2 (`tuple`)
Returns: list of countries (`list`)
Description: You and your friend are looking for a country to visit for Thanksgiving break. You want to look for a country where you both can speak the native language. Given two language tuples (`tuple`), find the list of names of the countires that speak both languages (`list`). The language tuples will be of the form `(language code, language name)` . Sort your answer in alphabetical order.
Note: You can assume that the languages will always be valid.

```
>>> commonCountries(("es", "Spanish"),("pt", "Portuguese"))
['Equatorial Guinea']
```

```
>>> commonCountries(("en", "English"),("es", "Spanish"))
['Belize', 'Guam', 'Puerto Rico']
```

Unique Regions

Function Name: uniqueRegions()
Parameters: countryList (`list`)
Returns: True or False (`bool`) or Error Message (`str`)
Description: Given a list of country codes (`list`), determine whether the countries are located in different regions (`bool`). If all the countries are located in the same region, return `False` . Otherwise, return `True` .
Note: If any country code is invalid, return the string `"Invalid country code!"` .

```
>>> uniqueRegions(["kor", "usa", "de"])
True
```

```
>>> uniqueRegions(["usa", "invalid", "ger"])
'Invalid country code!'
```

Organize Capitals

Function Name: organizeCapitals()
Parameters: capitalList (`list`)
Returns: Dictionary mapping regions to a list of countries (`dict`)
Description: Given a list of countries' capital cities, construct a dictionary that maps a region to the countries of the capitals located in that region.
Note: If a capital is invalid, ignore it.

```
>>> organizeCapitals(["Beijing", "Washington", "Jakarta"])
{'Asia': ['China', 'Indonesia'], 'Americas': ['United States of America']}
```

```
>>> organizeCapitals(["Rome", "Tokyo", "Invalid"])
{'Europe': ['Italy'], 'Asia': ['Japan']}
```

Visitable Countries

Function Name: visitableCountries()
Parameters: countryCodeList (`list`)
Returns: list of country names (`list`)
Description: You're planning a road trip and want to figure out what countries you can visit. You are given a possible path `countryCodeList (list)`, which is a list of country codes, and you want to visit the countries in the order they appear in the list. However, you can only travel from one country to the next if the adjacent country in the `countryCodeList` are borders of each other. Given a possible path `countryCodeList`, return a list of the **names** of the countries that can actaully be visited. If none of the countries border each other, you can only visit the first country in the give list.

Note: You may assume all countries are valid and that at least 2 countries are given.

```
>>> visitableCountries(["ZWE", "BWA", "NAM"])
['Zimabwe','Botswana','Namibia']

#Botswana borders Zimbabwe and Namibia borders Botsawana
#So you can travel to all three countries.
```

```
>>> visitableCountries(["NGA", "BEN", "COL"])
['Nigeria','Benin']

#Benin borders Nigeria but Colombia doesn't border Benin
#So you can only travel to Nigeria and Benin
```

Grading Rubric

Function	Points
averagePopulation()	20
commonCountries()	20
uniqueRegions()	20
organizeCapitals()	20
visitableCountries()	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 "Re-submit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW08.py` on Canvas.