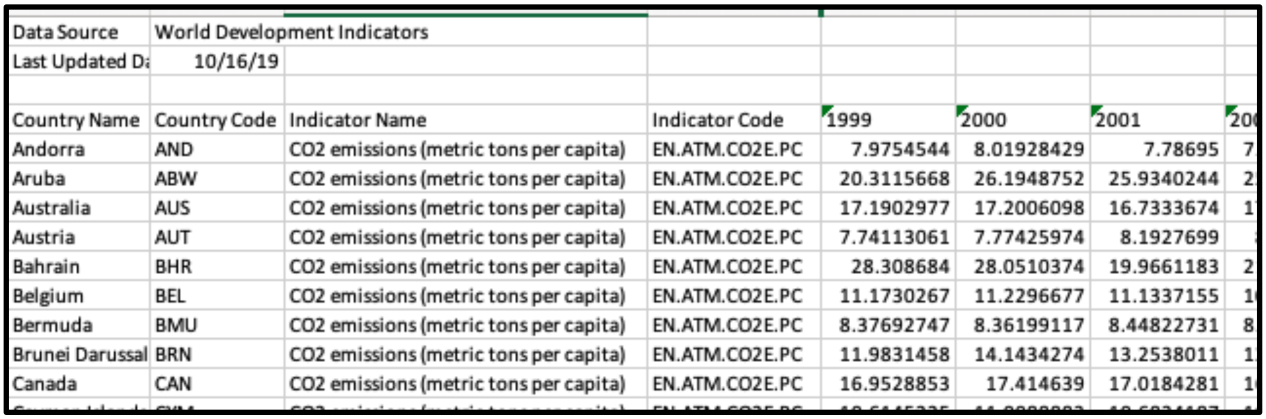
Homework 8 Climate Data and Caching

In this assignment, you will get the data using the World Bank API and extract certain information from that data about the climate, namely CO2 emissions. As a first step, you may look at the image below as a sample data from the World Bank API**.** In this assignment, you will get a part of the data searched by country and year using the API.



Data Source	World Development Indicators						
Last Updated Da	10/16/19						
Country Name	Country Code	Indicator Name	Indicator Code	1999	2000	2001	20
Andorra	AND	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	7.9754544	8.01928429	7.78695	7
Aruba	ABW	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	20.3115668	26.1948752	25.9340244	2
Australia	AUS	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	17.1902977	17.2006098	16.7333674	1
Austria	AUT	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	7.74113061	7.77425974	8.1927699	
Bahrain	BHR	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	28.308684	28.0510374	19.9661183	2
Belgium	BEL	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	11.1730267	11.2296677	11.1337155	1
Bermuda	BMU	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	8.37692747	8.36199117	8.44822731	8
Brunei Darussal	BRN	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	11.9831458	14.1434274	13.2538011	1
Canada	CAN	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	16.9528853	17.414639	17.0184281	1

You will write code to search for data using country codes and cache any results from each search in a dictionary and a JSON formatted file (cache file). We have given you the function **read_cache** for reading the dictionary from the cache file.

You will complete the following functions in the file **H8.py.** The code will read from the **cache_climate.json** file and write to it.

1. def write_cache(cache_file, cache_dict):
This function encodes the cache dictionary (cache_dict) into JSON format and writes the contents in the cache file (cache_file) to save the search results.
Note: see cache_example.py on Canvas (Week 10 "XML-JSON-LectureFiles-2.zip") for example code

2. def get_data_with_caching(country_code, year, per_page = 50):
This function uses the passed country_code and year to generate a request_url and then checks if this url is in the dictionary returned by read_cache. If the request url exists as a key in the dictionary, it should print 'Using cache for ' followed by the country_code and return the results for that request_url. If the request_url does not exist in the dictionary, the function should print "Fetching for " followed by the country_code and make a call to the World Bank API to get and return the CO2 emission data list searched by country (or countries) and year. The documentation of the API is "https://datahelpdesk.worldbank.org/knowledgebase/articles/889392-about-the-indicators-api-documentation". If there were results, it should add them to a dictionary (the key is the request_url, and the value is the results) and write out the dictionary to a file using the function

write_cache. If there was an exception during the search, it should print "Exception" and return None.

3. def country_dict():
This function returns a dictionary of all countries' information in 2014. The key of the dictionary will be a country name (e.g. United States, Canada) and the value will be a three letter country code (e.g. USA, CAN). Call get_data_with_caching and analyze the returned list to create the dictionary.
HINT1: API request with the country code "all" returns the data of all countries.
HINT2: Adjust the third parameter of get_data_with_caching (per_page) to get all countries' data (You can find the number of the country data from the API data).

4. reduced_percent(country_name, before, after):
This function receives three parameters: the name of a country and a before and after year. Call country_dict and convert the country_name to a country_code. Then, call get_data_with_caching and analyze the returned list to obtain the CO2 emission data for the two different years and return the percentage difference from the before year to the after year. The return value should be rounded to the first decimal place. For example, if the emission in 2000 and 2014 is 16 and 13 respectively, the percentage difference is calculated by *(16-13)/16*100 = 7.7%*.

Example Output:

```
This should use the cache
Using cache for BRA
This should fetch new data
Fetching for ARB
This should use the cache
Using cache for ARB
Getting the country dictionary
Fetching for all
The country code for 'United States' is: USA
----CO2 Reduction----
Using cache for all
Fetching for USA
Fetching for USA
The US reduced 15.8% of CO2 emission in 2014 compared to 2005
----[EXTRA] Top ten countries----
The top ten CO2 emission countries in 2014:
Using cache for all
Fetching for BHR;BRN;KWT;LUX;NCL;QAT;SAU;TTO;ARE;USA
No.1: Qatar (43.9)
No.2: Trinidad and Tobago (34.0)
No.3: Kuwait (25.8)
No.4: Bahrain (23.5)
No.5: United Arab Emirates (22.9)
No.6: Brunei Darussalam (22.2)
No.7: Saudi Arabia (19.4)
No.8: Luxembourg (17.4)
No.9: United States (16.5)
No.10: New Caledonia (16.0)
------------
test_country_dictionary (__main__.TestHomework8) ... Using cache for all
ok
test_get_data_with_caching (__main__.TestHomework8) ... Using cache for
BRA
Using cache for ARB
ok
test_reduced_percent (__main__.TestHomework8) ... Using cache for all
Fetching for CAN
Fetching for CAN
Using cache for all
Using cache for USA
Using cache for USA
Using cache for all
Fetching for CAN
Using cache for CAN
ok
test_top_ranking_EXTRA_ (__main__.TestHomework8) ... Using cache for all
Using cache for BHR;BRN;KWT;LUX;NCL;QAT;SAU;TTO;ARE;USA
ok
test_write_cache (__main__.TestHomework8) ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.685s

OK
```

Important!
You should cache all of the data from this exercise in a JSON file, and *include the cache file along with your source code in your github repository.* Remember to submit a link to your github repository on Canvas. **You have a starter cache file "cache_climate_orig.json" that you should copy and rename to "cache_climate.json" before each run of your code.**

Grading Rubric:
test_write_cache(5 points)

- 5 points for writing the JSON object correctly to the cache file.

test_get_data_with_caching(35 points)

- 5 points for correctly getting existing data from the cache

- 5 points for getting new data using the request_url from the API
- 5 points for adding the new data to the dictionary
- 5 points for writing out the changed dictionary to the cache
- 5 points for returning the correct type
- 5 points for returning the correct value
- 5 points for printing "Exception" if there was an exception and returning "None"

test_country_dictionary(10 points)

- 5 points for returning the correct type
- 5 points for returning the correct value

test_reduced_percent(10 points)

- 5 points for returning the correct type
- 5 points for returning the correct value

Extra Credit (6 points) top_ ranking:

This function returns the top ten CO2 emission countries in 2014.
The list of top ten countries in 2014 is provided as the list 'top_countries'.
Return a list of a tuple (country name, CO2 emission value) sorted by the value.
The value should be rounded to the first decimal place.
HINT: The API returns several countries data in a single request if you use
a country code separated by semicolons, such as "USA;CAN;BRA".

Submission:

On Github -

- JSON file containing your cached results - **cache_climate.json**
- Source code – **H8.py**

On Canvas

- Submit URL of your Github repo