Homework 7: APIs, JSON, and Caching

In this assignment, you will get data on movies using the OMDB API. You will also store the data in a cache file so that you can retrieve the data from the cache instead of requesting data from the API repeatedly.

We have provided the following in the starter code:

1. **read_cache(CACHE_FNAME)**: This function reads JSON from the cache file and returns a dictionary from the cache data. If the file doesn't exist, it returns an empty dictionary.
2. **main()** function
3. Test cases to test the functions you will write

Before you proceed to the tasks:

You will need an API key for this HW. You can generate your key here:
http://www.omdbapi.com/ apikey.aspx

Assign your API key to the variable API_KEY on line 13!

Strongly Recommended

Choose an online JSON viewer. We recommend printing the API data/cache data and pasting it in the viewer to examine the structure of the data. Here are few of the many available options for JSON viewers:

1. https://jsonformatter.org/
2. https://jsonformatter-online.com/
3. https://jsonlint.com/

Tasks - You will write the following functions.

def write_cache(CACHE_FNAME, CACHE_DICT):

This function encodes the cache dictionary (***CACHE_DICT***) into JSON format and writes the JSON to the cache file (***CACHE_FNAME***) to save the search results.

def create_request_url(title):

This function prepares and returns the request url for the API call.

The documentation of the API parameters is at http://www.omdbapi.com/

Make sure you provide the following parameters besides the title when preparing the request url:

1. type: one of movie, series, episode

2. plot: set to short
3. r: set to json

Example of a request URL for movie title The Dark Knight:
http://www.omdbapi.com/?t=The Dark Knight&apikey=xxxxxx&type=movie&plot=short&r=json

The API key has been blurred out since one shouldn't share API keys publicly

def get_data_with_caching(title, CACHE_FNAME):

This function uses the passed movie title to first generate a ***request_url*** (using the **create_request_url** function). It then checks if this URL is in the dictionary returned by the function **read_cache**. If the ***request_url*** exists as a key in the dictionary, it should print `"Using cache for <title>"` and return the results for that ***request_url***.

If the ***request_url*** does not exist in the dictionary, the function should print `"Fetching data for <title>"` and make a call to the OMDB API to get the movie data.

If data is found for the movie, it should add it to a dictionary (the key is the ***request_url***, and the value is the results) and write out the dictionary to a file using **write_cache**.

In certain cases, the OMDB API may return a response for the ***request_url*** but it may not contain any data for the movie: `{"Response":"False","Error":"Movie not found!"}`

DO NOT WRITE THIS DATA TO THE CACHE FILE!
Print "Movie Not Found!" and return None

If there was an exception during the search (for reasons such as no network connection, etc), it should print out `"Exception"` and return None.

def top_movies_rated(rated, CACHE_FNAME):

This function returns the top ten movies on the basis of the genre specified. For example, if the “Rated”= “PG”, the function will return top ten movies in a list ranked by their imdbRating

Example Output

NOTE: Your example output ***may*** look different from this. It will depend on two things: (1) ***whether you have commented out the unit tests*** - the test cases use a different list of movies and their results will also get stored in the cache file
(2) ***whether you delete the cache file*** - then you lose all the data stored in the cache file and you may see print statements which say "Fetching data from…"

```
Fetching data for The Terminator
Fetching data for Monsters, Inc.
Fetching data for Inside Out
Fetching data for V for Vendetta
Fetching data for My Neighbor Totoro
Fetching data for Coco
Fetching data for WALL·E
Fetching data for Aladdin
Fetching data for Brave
Fetching data for Cinderella
Fetching data for The Little Mermaid
Fetching data for Up
Fetching data for Frozen
Fetching data for Moana
Fetching data for Princess and the Frog
Fetching data for Snow White and the Seven Drawfs
Movie not found!
Fetching data for Toy Story
Fetching data for Toy Story 2
Fetching data for Toy Story 3
Fetching data for Tangled
Fetching data for Mulan
Fetching data for Sleeping Beauty
Fetching data for The Sword in the Stone
____________________
Fetching data for Inception
Using cache for Inception
____________________
Fetching data for Parasite
Using cache for Parasite
____________________
Fetching data for Ladybird
Using cache for Ladybird
____________________
Top movies rated PG
['Coco', 'Up', 'Inside Out', 'Tangled', 'Moana', 'Frozen', 'Brave', 'Cinderella']
____________________
Top movies rated PG-13
['Inception']
____________________
EXTRA CREDIT!
Movie list with BoxOffice cost more than $747,000
[('Moana', '757044'), ('WALL·E', '808164'), ('Tangled', '821936'), ('Toy Story 2', '852179'), ('Inside Out', '921711'), ('Frozen', '953009')]
____________________
Movie list with BoxOffice cost more than $80,000
[('Cinderella', '151353'), ('Toy Story', '225679'), ('My Neighbor Totoro', '250213'), ('Brave', '283207'), ('Aladdin', '350219'), ('Parasite', '369749'), ('The Termina
tor', '371200'), ('The Princess and the Frog', '400899'), ('Coco', '460015'), ('V for Vendetta', '511035'), ('The Little Mermaid', '543479'), ('Inception', '576195'),
('Sleeping Beauty', '600000'), ('Mulan', '620254'), ('Monsters, Inc.', '642256'), ('Moana', '757044'), ('WALL·E', '808164'), ('Tangled', '821936'), ('Toy Story 2', '85
2179'), ('Inside Out', '921711'), ('Frozen', '953009')]
____________________
test_create_request_url (__main__.TestHomework7) ... ok
test_get_data_with_caching (__main__.TestHomework7) ... Using cache for The Terminator
Using cache for Monsters, Inc.
Using cache for Inside Out
Using cache for V for Vendetta
Using cache for My Neighbor Totoro
Using cache for Coco
Using cache for WALL·E
Using cache for Aladdin
Using cache for Brave
Using cache for Cinderella
Using cache for The Little Mermaid
Using cache for Up
Using cache for Frozen
Using cache for Moana
Using cache for Princess and the Frog
Fetching data for Snow White and the Seven Drawfs
Movie not found!
Using cache for Toy Story
Using cache for Toy Story 2
Using cache for Toy Story 3
Using cache for Tangled
Using cache for Mulan
Using cache for Sleeping Beauty
Using cache for The Sword in the Stone
ok
test_movie_list (__main__.TestHomework7) ... ok
test_top_movie_rated (__main__.TestHomework7) ... ok
test_write_cache (__main__.TestHomework7) ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.128s

OK
PS C:\Users\Uche\Desktop\GSI>
```

Grading Rubric

def test_write_cache - 5 points

- 5 points for writing the JSON data correctly to the cache file

def test_create_request_url - 5 points

- 2 points for including the API key in the request URL
- 2 points for including the movie title in the request URL
- 1 point for including the remaining 3 parameters - plot, type, r

def test_get_data_with_caching - 35 points

- 5 points for correctly getting existing data from the cache file
- 5 points for getting new data using the request_url from the API
- 5 points for checking if the data was found for the movie title provided
- 5 points for adding data to the cache dictionary and cache file only for movies that exist
- 5 points for writing out the changed cache dictionary to the cache file
- 5 points for returning the correct result & type
- 5 points for printing "Exception" if there was an exception and returning "None"

def test_top_movies_rated - 15 points

- 3 points for returning a list
- 3 points for sorting the items in the list on the basis of imdbRating
- 3 points for returning no more than the number of movies required
- 3 points for returning the right values in the list
- 3 points for using the imdbRating as the rating measure

Extra Credit - 6 points

def movie_list(cost, CACHE_FNAME):
"""
The function calls read_cache() to get the movie data stored in the cache file. It analyzes the dictionary returned by read_cache() to return a list of all the movies that have received more than the passed in BoxOffice cost in an ascending sorted list of tuples """
def movie_list - 6 points

- 3 points for returning the correct number of items in an ascending sorted list of tuples
- 3 points for the right movies in the list of tuples