Homework 6: APIs, JSON, and Caching

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

Click here to sign up for an API key. We strongly recommend reading the API documentation before getting started. It's not long!

Strongly Recommended:

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

1. https://jsonformatter.org/
2. https://jsoneditoronline.org/

Tasks:

def get_json_content(filename):

- This function reads a cached JSON file (filename) and returns a dictionary with JSON data
  - If the request is unsuccessful, return an empty dictionary

def save_cache(dict, filename):

- This function converts the cache dictionary into JSON format and writes the JSON to the cache file (filename) to save the search results

def search_movie(movie):

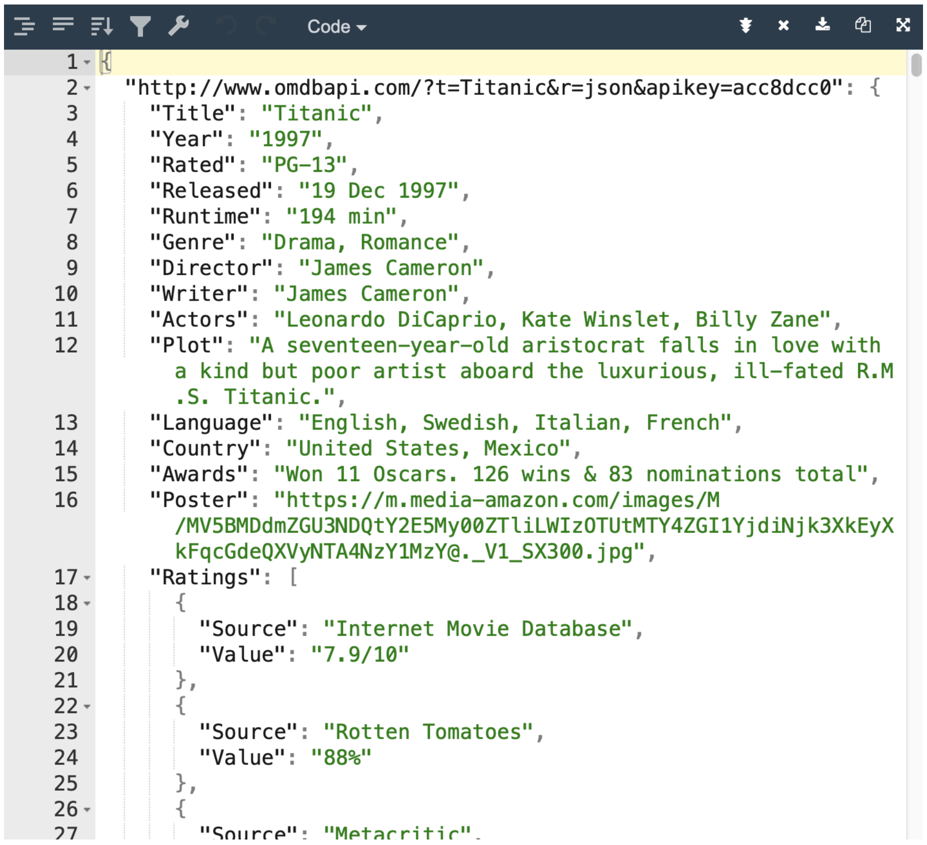
- This function takes in a movie name and makes a request to the OMDB API
  - If the request is successful, return a tuple with the data you got back from the API and the URL you requested

- If a request is unsuccessful, return None
- Hint: Make sure you set the response type to JSON when you make the request

def update_cache(movies, cache_file):

- This function goes through a list of movies (predefined for you in the main function) and gets IMDB data for each of them
  - Save data from successful requests to the cache
    - In your cache, the keys should be the URL you requested and the value should be a dictionary with the response data
    - If a request does not return any data, you should not add anything to the cache
    - If a movie is already in the cache, you should not add a duplicate entry
- This function returns a string that says the percentage of the films in movies that were successfully added to the cache
  - e.g. if you have 20 films in movies and can get the data for 19 of them, it should return "Cached data for 95% of movies"
  - If your code works correctly, this function should return "Cached data for 100% of movies" on the first call and "Cached data for 0% of movies" on all subsequent calls (because that data is already in the cache, there is no need to make an API request again)

Sample of a correctly formatted cache. Note that the full dictionary output is cut off since it's quite lengthy.



```
{
  "http://www.omdbapi.com/?t=Titanic&r=json&apikey=acc8dcc0": {
    "Title": "Titanic",
    "Year": "1997",
    "Rated": "PG-13",
    "Released": "19 Dec 1997",
    "Runtime": "194 min",
    "Genre": "Drama, Romance",
    "Director": "James Cameron",
    "Writer": "James Cameron",
    "Actors": "Leonardo DiCaprio, Kate Winslet, Billy Zane",
    "Plot": "A seventeen-year-old aristocrat falls in love with a kind but poor artist aboard the luxurious, ill-fated R.M.S. Titanic.",
    "Language": "English, Swedish, Italian, French",
    "Country": "United States, Mexico",
    "Awards": "Won 11 Oscars. 126 wins & 83 nominations total",
    "Poster": "https://m.media-amazon.com/images/M/MV5BMDdmZGU3NDQtY2E5My00ZTliLWIzOTUtMTY4ZGI1YjdiNjk3XkEyXkFqcGdeQXVyNTA4NzY1MzY@._V1_SX300.jpg",
    "Ratings": [
      {
        "Source": "Internet Movie Database",
        "Value": "7.9/10"
      },
      {
        "Source": "Rotten Tomatoes",
        "Value": "88%"
      },
      {
        "Source": "Metacritic",
```

def get_highest_box_office_movie_by_genre(genre_name, cache_file):

- For a given genre, this function gets the movie with the highest box office total
    - Gets data from the cache
        - Return the movie title and box office total for the most successful film as a tuple
        - You do not need to do anything to handle potential ties
        - Hint: The box office totals may not be formatted as a number (e.g. you may get values like "$1,000,000")
    - If there are no movies in our cache that belongs to the genre, return "No films found for [genre_name]"
- Example:
    - get_highest_box_office_movie_by_genre("Action", 'cache.json') should return ('The Avengers', '$1,518,812,988')

def recommend_similar_movies(movie_title, cache_file):

- This function locates a target movie in the cache by title and recommends others that share at least one genre.
    - First, read the cache data from cache_file (using get_json_content).
        - If there is no data, return "No movie data found in cache."
    - Find the movie matching movie_title.
    - If not found, return "'[movie_title]' is not in the cache."
        - If no Genre field, return "No genre information available for '[movie_title]'."
    - Gather the target movie's genres by splitting its Genre string and converting to lowercase.
    - Iterate through other movies in the cache:
        - Skip the target movie itself.
        - Check if there is any overlap in genres.
        - If so, add that movie's Title to a list of recommendations.
    - If there are no recommendations, return "No recommendations found based on '[movie_title]'."
    - Otherwise, return a list of recommended movie titles.
- ● For example, recommend_similar_movies('Titanic', 'cache.json') should return:
    - ['Little Women', 'Top Gun: Maverick', 'La La Land', 'Whiplash', '12 Years a Slave', 'Life of Pi', 'The Help', 'Killers of the Flower Moon', 'Oppenheimer', 'The Lord of the Rings: The Fellowship of the Ring', 'Braveheart', 'Clueless', '10 Things I Hate About You', 'Gone with the Wind', 'Casablanca', 'Parasite']

Extra Credit (6 Points):

2 points: get_api_key(filename):

- It is best practice to never copy and paste your API key into your code in plain text. This is especially true in professional environments, where you may be working with APIs that require you to pay for each request.

- Save your API key to a .txt file called api_key.txt. Then, read that file and use its contents to reference your API key using get_api_key.
    - This file should be in the same directory as your HW6_starter.py file
    - You will not receive points if you don't name your file correctly or if it is in the wrong directory
- You will not receive points if your API key is visible in your code in plaintext, even if you implement this function
    - Having a variable that stores the value of your API key is okay, but there should not be a line like api_key = ##### where your key is visible in plain text

4 points: get_movie_rating(title, cache_file):

- This function gets the Rotten Tomatoes rating for a given film
    - If there is no Rotten Tomatoes rating, return "No rating found"
    - Otherwise, return the rating given
- Example:
    - get_rotten_tomatoes_rating('Titanic', 'cache.json') should return "88%"

Rubric:

Note: Do not modify any of the test cases we have provided. Deleting or modifying the test cases will result in points being deducted.

- get_json_content (5 points)
- save_cache (5 points)
- search_movie (5 points)
- update_cache (15 points)
- get_highest_box_office_movie_by_genre (15 points)
- recommend_similar_movies (15 points)

Extra credit:

- get_api_key (2 points)
- get_movie_rating (4 points)