Unit 5 Lab: Finalizing the Movie App

Overview

This is the lab we've been working toward. Here, you'll add an API call to actually get the Rotten Tomatoes rating, prompt a user for whether they want to search for a movie or view a rating, and read in an API key from a file.

We'll be using the OMDb API.

It's a long lab, but it's going to be worth it. Buckle up — here we go!

Deliverables

You're going to continue building this locally from the last lab. You'll write all of your code in the same `movie_app.py` file.

Run the file from the command line to check your work.

Additionally, you will have a file, `omdb-api-key.txt`, with your *own* OMDb API key.

Requirements

1. Your app will need to do the following:

    - Handle user input to determine the mode in which to run
    - Leverage the OMDB database to retrieve information
    - Provide ratings for a given movie
    - Provide seach results for a given movie query
    - Handle exceptions appropriately

Directions

Start with the `movie_app.py` you last edited in Lab 3.

Pair up with a partner!

API Setup

1. First, you'll need to sign up for a free API key
   - Go to this URL.
   - Put in your email address, name, and for the "Use" box just put "Educational Purposes".
   - Check your email- you should have an email with a unique API key. Make sure you click the link to activate the key!!
   - Test out your API key by putting this URL in your browser: `https://www.omdbapi.com/?apikey=[your key]&t=clerks`.
   - It returns quite a lot, but can you identify which pieces of information we'll need?
2. Let's make the key available to your program. Create a file called `secret.py` in the same folder as your `movie_app.py`.
   - In this file, create a variable called `AUTH_KEY`, and set it equal to the API key string.
   - You'll need to grant `movie_app.py` access to this variable. To do so, at the top, write `import secret`. You can now refer to the AUTH_KEY as `secret.AUTH_KEY`. Print this value to verify that it's working.
   - You could also accomplish this by storing the file as a string in a `secret.txt` file, and reading it in with the `open()` function. Either way works.
   - **Pro Tip:** When you put this exercise into your Github account, DO NOT include the file with your auth key!!
3. Next step is to create a class representing an API Client. Clients are reusable sets of code that handle interactions with APIs.
   - In `movie_app.py`, create a new class called `OMDBClient`.
   - Check out the OMDB Documentation. Which parameters will we need? Remember, we want to provide two main functions: the ability to search for movies, and the ability to retrieve movie ratings. Build 2 URL strings, one for each function, and test them out in your browser. *Don't forget to include your Auth Key!*
   - Create class variables for each of the URL strings. You'll want to template the parameters. When you use these URLs, use the `.format()` function to replace the templates with passed in values.
   - Create a function called `get_movie` that consumes a movie title as a string, and returns a `Movie` object built from the result of calling the API. *You may want to edit the Movie class' `__init__` function to use the keys present in the JSON object returned from OMDB.*
   - Create a function called `search_movies` that consumes a movie title as a string, and returns the 'Search' results from the API call.
   - You may notice some code repetition in the preceding two functions. Try refactoring it out into a function called `call_api`.

Movie Object

We'll need to give the `Movie` object a little love to reflect our changes.

1. As mentioned in the last section, you might want to edit the `__init__` function to set the title and ratings values using the correct keys for the JSON values returned by the API.

2. How are Ratings returned from OMDB? They're coming back as a list of dictionaries. Let's store this entire object in the `self.ratings` variable. Our `get_rating` function can now choose between all of these different sources.

3. Give `get_rating` a parameter for source. *We'll want to provide a default value*.

4. Refactor `get_rating` to provide the rating that corresponds to the provided source. Don't forget to handle the case where no rating is found for the given source!

Wiring up the API

Let's edit our search and ratings functions to leverage the `OMDBClient` class.

1. The `print_search_results` Function
    - First, we'll need to create an instance of the `OMDBClient` in this function.
    - Use the `OMDBClient` function to retrieve the search results for the passed in `movie_query`. Loop over the results and print them out.
2. The `print_movie_rating` Function
    - Create an instance of the `OMDBClient` in this function.
    - Call the API, and load the response into a `Movie` object.
    - Print the title and rating from the `Movie` object.
3. The `print_all_ratings` Function
    - This shouldn't need to be changed, but test it out to make sure!

User Input

Our application requires user input to work. We can accomplish everything we need with the `input()` function.

1. User input to set the mode.

    - You should have a function that will ask the user which mode they want the application to run in (e.g. `search` or `ratings`).
    - This function will need to be called in the proper place, and should only allow the user to set the mode to a valid value.

2. User input to define the movie query.

    - You'll need a second prompt that will get a movie query. This will be needed both for searching and retrieving results.

Exception Handling

What happens if you search for the movie "asdffdadf"? Let's fix this!

1. Create your own OMDBException class that inherits from the base `Exception` class.

2. Add in `raise OMDBException(<error_message>)` calls wherever necesary.

3. Wrap every function call to the methods that may throw an `OMDBException` in a `try`... `catch` block. If the error is incurred, print a message to the user.

Bonus #1: Source Selection

Add support for the user to pick which source the rating should come from.

Bonus #2: Multiple Movies

Allow the user to input multiple movies for which to recieve ratings.

YOU DID IT!!!!!!