Important

1. Due Date: **Thursday November 2rd**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk (Schedule posted on class website.)
    - Email TA's or use Piazza Forums Notes:
    - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
    - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
2. Don't forget to include the required collaboration statement (outlined on the syllabus). **Failing to include the Collaboration Statement will result in no credit**.
3. Do not wait until the last minute to do this assignment in case you run into problems.
4. Comment or delete all your function calls. When your code is run, all it should do is run without any errors. Having function calls or extraneous code outside the scope of functions will result in a lower grade. (import statements and global variables are okay to be outside the scope of a function)
5. **Read the entire specifications document before starting this assignment.**
6. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%.**

Introduction

APIs, or **A**pplication **P**rogram **I**nterfaces, are vital tools for businesses in all industries. Most APIs work in similar ways, so if you learn how to use one API, chances are that the next time you encounter one you will be already familiar with how it works! Using an API involves sending web requests. In this homework, you will use an API that contains tons of information about movies! You will implement 5 functions that answer different questions. The details about this functions can be found in this document or in the docstrings of the provided script. You have until Thursday, November 2nd to complete this assignment. Don't forget to include your name and your collaboration statement. Re-download your submission from T-Square after you submit it to make sure that your code runs successfully.

Authors: John Wood and Caroline Kish

Part 1. Start using the API

The website we are going to use for this assignment is https://www.themoviedb.org/. To use this website's API, you will need to get an API key. To do so, you will have to do a couple of things.

1. Get an account from https://www.themoviedb.org/. Go to this link and create an account: https://www.themoviedb.org/account/signup
2. Verify your account. The website should have sent you an Email Verification so just verify your email address by clicking on the link provided in the email.
3. Login to your account. Enter your credentials (username and password)

4. Request an API key. You can apply for an API key by clicking the "API" link from the left hand sidebar within your **account SETTINGS** page. Now, Click on “Developer”, and read and accept the terms. You should be in a page where the URL is similar to this:
   https://www.themoviedb.org/account/{username}/request-api?type=developer
5. Fill out the form:
   Type of Use: Education
   Application Name: CS1301 - HW08
   Application URL: www.gatech.edu
   Application Summary: Educational Purposes. Intro to APIs.

6. Thekeywillbegeneratedimmediatelyafterthisformissubmitted.SaveyourAPI Key (v3 auth) somewhere, you’ll need this key to make the API calls. Note: you won’t need the other key (v4 auth) for this lab.

You are all set. Now, let's go to the API Developer section in the following link: https://developers.themoviedb.org/3/getting-started. Here, you will find all the API endpoints you can use from this website. In the left column, click on the header called Movies. Under this Movies section, https://developers.themoviedb.org/3/movies, you will find all the API endpoints you will need to complete this assignment.

Tips / Hints

- Read the handout provided for you about APIs. You can find this under Resources > Lecture Notes > requests and API handout
- Do the tutorial in the PDF and check your solutions against intro_api.py
- When you find the endpoint that you want to use read the documentation about it. For example if you want to use the Get Popular endpoint, go to that documentation: https://developers.themoviedb.org/3/movies/get-popular- movies, read the ‘Definition’ and click the ‘Try it out’, put your api_key and click ‘ SEND

REQUEST ‘. Analyze the JSON you get as a response. Do this for all the other endpoints.

- Under every function you can see a set of ‘Examples’ on how your function should work. This might not be exactly the same as what you get in your functions because from the moment I am writing this to the moment you test your code the ranking of the top movies might have changed. We will grade your functions dynamically; this means that we will test your answers against the answers generated by the solution key at that moment in time.

Part 2. Functions

Function name: get_movie_recommendations
Parameters: movie_ids (list of ints)
Returns: recommended movie titles (list of strings)
Description: Write a function that takes in a list of movie IDs. Find the movies corresponding to the IDs. If the popularity of the movie is greater than 20, add the movie title (string) to a list. Return the list of titles.

Note: Your function should be able to handle invalid movie ids.

Test Cases:

```
>>> test_1 = get_movie_recommendations([346364, 17473, 603])
>>> print(test_1)
['It', 'The Matrix']

>>> test_2 = get_movie_recommendations([672, 314, 12142, 325245,
333333333])
>>> print(test_2)
['Harry Potter and the Chamber of Secrets']
```

Function name: get_upcoming
Parameters: None
Returns: the next 10 upcoming movies titles (list of strings)
Description: Write a function that uses an API call and returns a list of the next ten upcoming movie titles (strings) that come first alphabetically. Only consider the upcoming movies that are listed on page 1.

Hint: First, find the movie titles from page 1 and add them to a list. Sort this list. Then take the first ten movies titles from this list.

Test Case:

```
>>> test = get_upcoming()
>>> print(test)
['A Bad Moms Christmas', 'American Assassin', 'Atomic Blonde',
'Battle of the Sexes', 'Flatliners', 'Happy Death Day',
'Jigsaw', 'Justice League', 'My Little Pony: The Movie', 'Only
the Brave']
```

Function name: get_cast_members
Parameters: movie_id (int), job_title (string), section (string)
Returns: names of people who worked the job specified by the job title for the given movie (list of strings)
Description: Write a function that takes in a movie_id, a job_title, and a section (either “cast” or “crew”). Find the names of every person working on the movie who works in the specified section and has the specified job title. Assume everything passed in is valid. Add these names to a list. Return the list at the end of the function.

Note: Every job_title listed as “Actor” will be considered as working for the section "cast" and every person working for the section "cast" will be considered under the job_title "Actor". The only job_title that works in the “cast” section is “Actor”. All other job types will be found under the “crew” section.

Hint: The credits endpoint will be useful for solving this function.

Test Cases:

```
>>> test_1 = get_cast_members(1493, "Actor", "cast")
>>> print(test_1)
['Sandra Bullock', 'Benjamin Bratt', 'Michael Caine', 'Candice
Bergen', 'William Shatner', 'Ernie Hudson', 'John DiResta',
'Heather Burns', 'Melissa De Sousa', 'Steve Monroe', 'Deirdre
Quinn', 'Wendy Raquel Robinson', 'Asia De Marcos', 'Ken Thomas',
'Gabriel Folse', 'Leeanne Locken', 'John Cann']

>>> test_2 = get_cast_members(157336, "Director", "crew")
>>> print(test_2)
['Christopher Nolan']

>>> test_3 = get_cast_members(315635, "Writer", "crew")
>>> print(test_3)
```

```
['John Francis Daley', 'Jonathan M. Goldstein', 'Erik Sommers',
'Chris McKenna', 'Christopher D. Ford', 'Jon Watts']
```

Function name: map_movies_to_language
Parameters: movie_ids (list of ints), languages (list of strings)
Returns: a dictionary mapping languages to the movies titles available in that language
Description: Write a function that takes in a list of movie ids and a list of languages. Each string in the language list passed in will be a key in a new dictionary. For each key (language), the corresponding value should be a list of the movie titles that are offered in that language. Each movie title should be in the values list for a given language only once (even if there are different dialects / versions available for the language).

Note: Your function should be able to handle invalid movie ids.

Test Cases:

```
>>> test_1 = map_movies_to_languages([387, 17473, 2493, 1648,
210577], ["Serbian", "German", "Persian"])
>>> print(test_1)
{'Serbian': ['The Princess Bride', "Bill & Ted's Excellent
Adventure", 'Gone Girl'], 'German': ['Das Boot', 'The Room',
'The Princess Bride', "Bill & Ted's Excellent Adventure", 'Gone
Girl'], 'Persian': ['Gone Girl']}

>>> test_2 = map_movies_to_languages([207, 36968, 11451, 9820],
["Mandarin", "Bulgarian", "Swedish", "English"])
>>> print(test_2)
{'Mandarin': ['Dead Poets Society', 'The Parent Trap'],
'Bulgarian': ['Dead Poets Society', 'Life-Size', 'Herbie Fully
Loaded'], 'Swedish': ['Dead Poets Society', 'Herbie Fully
Loaded', 'The Parent Trap'], 'English': ['Dead Poets Society',
'Life-Size', 'Herbie Fully Loaded', 'The Parent Trap']}

>>> test_3 = map_movies_to_languages([150689, 321612, 109445,
10674], ["Greek", "French", "Tajiki"])
>>> print(test_3)
{'Greek': ['Cinderella', 'Beauty and the Beast', 'Frozen',
'Mulan'], 'French': ['Cinderella', 'Beauty and the Beast',
'Frozen', 'Mulan'], 'Tajiki': []}
```

Function name: get_genre_movies

Parameters: movie_ids (list of ints), genre (string), start_year (int), end_year (int)
Returns: movie titles (list of strings)
Description: Write a function that takes in a list of movie ids, a genre, and a range of years. Find all the movie titles that are of the genre passed in and whose release date falls between the start year and end year passed in. Return a list of the movie titles that meet this criteria.

Note: Your function should be able to handle invalid movie ids and genres. Years will always be valid.

Test Cases:

```
>>> test_1 = get_genre_movies([672, 597], "Fantasy", 2000, 2006)
>>> print(test_1)
['Harry Potter and the Chamber of Secrets']

>>> test_2 = get_genre_movies([207, 36968, 11451, 1493, 9820],
"Comedy", 1998, 2004)
>>> print(test_2)
['Life-Size', 'Miss Congeniality']
```

Grading Rubric

```
get_movie_recommendations: 10 pts
get_upcoming: 20 pts
get_cast_members: 20 pts
map_movies_to_language: 25 pts
get_genre_movies: 25 points
```

Provided

The following file(s) have been provided to you.

1. HW8.py
   This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. HW8.py

CS1301 - HOMEWORK 8: Intro To APIs

If this file does not run (if it encounters an error while trying to run), you will get no credit on the assignment.