Important

1. Due Date: **Thursday, March 30th**
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]
5. Don't forget to include the required collaboration statement (outlined on the syllabus). Failing to include the Collaboration Statement will result in no credit.
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. Comment or delete all your function calls. When your code is run, all it should do is build 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)
8. **Read the entire specifications document before starting this assignment.**
9. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%.**

Introduction

APIs, or Application Program Interfaces, are vital tools for businesses in all industries. Most APIs work in really 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 these work and you will be able to tackle any problem. A unique aspect of an API is that it involves sending web requests. In this Lab, 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, March 30th** 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.

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 you email address, click in 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 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 – Lab 2
    - **Application URL:** www.gatech.edu
    - **Application Summary:** Educational Purposes. Intro to APIs.
6. The key will be generated immediately after this form is submitted. Save your API 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 > Class 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 as from the moment I'm 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

getMovieTitle

Function name: `getMovieTitle`
Parameters: an int representing the ID of a movie
Returns: a string representing the original title of the movie

Description: Write a function that takes in the ID of a movie, makes an API call, and returns the <str> original title of the movie related to the ID passed in.

Examples:

```
>>> movie = getMovieTitle(135397)
>>> print(movie)
Jurassic World

>>> movie = getMovieTitle(157336)
>>> print(movie)
Interstellar

>>> movie = getMovieTitle(000000)
>>> print(movie)
None
```

getTopTenMovies

Function name: `getTopTenMovies`
Parameters: **No parameters**
Returns: a list containing the original titles of the top ten rated movies

Description: Write a function that makes an API call, and returns the <str> original title of the top ten rated movies.

Examples:

```
>>> topMovies = getTopTenMovies()
>>> print(topMovies)
['The Shawshank Redemption', '君の名は。', 'The Night Of', 'The Godfather', 'Pride and Prejudice', 'Whiplash', '切腹', 'Band of Brothers', 'Piper', 'Schindler's List']
```

getTopMoviesInRange

Function name: `getTopMoviesInRange`
Parameters: an int representing the startYear (inclusive) of the range, an int representing the endYear (exclusive) of the range
Returns: a list containing the original titles of the top three rated movies that were released between the two years passed in.

Description: Write a function that takes in a two ints representing the start (inclusive) and the end (exclusive) of a period of time, and returns the top three rated movies that were released within the period of time specified by the parameters. To get the top rated movies, make an API call similar to the one you did for `getTopTenMovies`. Then, you should go through the movies, and append the first three movies that have a `releasedDate` between the specified time period.

Notes:

- Since the movie API has tons of movies in their Database, they have an optional parameter called **page** for their GetTopRated endpoint. If you make the API call without specifying a value for the page parameter, it will be 1 by default, giving you the first couple of top rated movies.
- The movies are split into different "pages" (basically, into different JSONs) because having all the movies in a single endpoint would return a really heavy JSON object.
- For this function, the problem you could face is that probably the top three movies that were released in a specific time range won't be in the first page.
- Hence, after you iterate through all the movies in the json returned by the endpoint with page 1, if you haven't found the top three movies, you will have to make an API call to get the top rated movies in page 2, and iterate through those movies.
- It is also probable that the top three movies in the specified time period aren't in the second page, so you will have to make an API call to get the movies in the next page, and so on until you find them.
- The iteration and the API calls to the next pages are given to you in the base code. Consequently, you just have to write the code to iterate through the movies for a single endpoint and append the movies that fall within the time period.
- We have included a 'sleeper' to your implementation. The `time.sleep(0.3)` function call is just waiting 0.3 seconds. The reason for this is that sometimes APIs have a max number of calls you can do per second, so we just wait a little bit to avoid this.

Examples:

```
>>> output = getTopMoviesInRange(2014, 2017)
>>> print(output)
 ['君の名は。', 'The Night Of', 'Whiplash']
```

```
>>> output = getTopMoviesInRange(2010, 2013)
>>> print(output)
['Intouchables', 'Paperman', 'Inception']
```

getMoviesByGenre

Function name: `getMoviesByGenre`
Parameters: a list of movie Ids <ints>, and a genre <str>
Returns: a list containing the original title <str> of the movies passed in that belong to this genre

Description: Write a function that takes in a list of movie Ids and a genre. Make API calls to get information of each movie in the list by using its Id, and append the original title of the movie if it belongs to the genre passed in.

Examples:

```
>>> output = getMoviesByGenre([263115, 341174, 127380, 135397],
"Action")
>>> print(output)
 ['Logan', 'Jurassic World']

>>> output = getMoviesByGenre([15206, 271110, 14564, 284052,
118340], "Horror")
>>> print(output)
 ['La terza madre', 'Rings']

>>> output = getMoviesByGenre([15206, 271110, 14564, 284052,
118340], "Comedy")
>>> print(output)
[]
```

mapProductionCompaniesToMovies

Function name: `mapProductionCompaniesToMovies`
Parameters: a list of movie Ids <ints>, and a list of production companies <strings>
Returns: a dictionary {keys are strings: values are lists of strings}

Description: Write a function that takes in a list of movie Ids and a list of production companies. Each movie has a list companies that were involved in their production. Your task is to create a dictionary were the keys are the production companies in the list passed in, and the values are lists containing the original titles of the movies that were produced by the corresponding company.

Examples:

```
>>> output = mapProductionCompaniesToMovies([263115, 341174, 127380,
135397],  ["Marvel Entertainment", "Universal Pictures", "Pixar
Animation Studios", "DC Entertainment"])
>>> print(output)
{'Marvel Entertainment': ['Logan'], 'DC Entertainment': [], 'Pixar
Animation Studios': ['Finding Dory'], 'Universal Pictures': ['Fifty
Shades Darker']}

>>> output = mapProductionCompaniesToMovies([293660, 263115, 259316,
76341, 246655, 209112, 302946], ["Marvel Entertainment","Warner
Bros.", "Universal Pictures", "Twentieth Century Fox Film
Corporation", "The Donners' Company"])
>>> print(output)
{'Twentieth Century Fox Film Corporation': ['Deadpool', 'Logan', 'X-
Men: Apocalypse'], 'Warner Bros.': ['Fantastic Beasts and Where to
Find Them', 'Batman v Superman: Dawn of Justice', 'The Accountant'],
'Marvel Entertainment': ['Deadpool', 'Logan', 'X-Men: Apocalypse'],
'Universal Pictures': [], "The Donners' Company": ['Deadpool']}
```

Grading Rubric

```
- getMovieTitle:                  10 points
- getTopTenMovies:                20 points
- getTopMoviesInRange:            25 points
- getMoviesByGenre:               20 points
- mapProductionCompaniesToMovies: 25 points
_____________________________________________
- Total                           100/100 points
```

Provided

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

1. `Lab02.py`
   This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.