Homework 8 - APIs

CS 1301 - Intro to Computing

Important

- Due Date: **Tuesday, October 27th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this homework is for you to enhance your understanding of APIs and making requests. The homework will consist of 5 functions for you to implement. You have been given `HW08.py` skeleton file to fill out. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Fareeda Kasim (kasimfareeda@gatech.edu), Grace Saad (gsaad6@gatech.edu), & Josh Tabb (jtabb6@gatech.edu)

Helpful Information to Know

Important Note: API data can sometimes change over time. Don't be alarmed if the outputs you're seeing in the Shell are different than the test cases in this PDF. For debugging, you may have to rely on manually looking through the API data and checking if your outputs match the requirements of the function.

API Documentation: This homework uses two APIs across its problems. In order to use an API properly, looking through the documentation is essential. Here is the documentation for the two APIs in this homework:

- **Harry Potter**: https://www.potterapi.com/#introduction
- **Star Wars**: https://swapi.dev/documentation

API Keys: In order to prevent people from spamming/abusing their services, some APIs require you to register for keys so that you can use them. You will need to register for an API key for the Harry Potter API, the API used for the first two problems. Register for a key at this link (it's free): https://www.potterapi.com/login/#signup.

Once you create an account, click your email address on the right menu, and your key is the long string beside the key icon. Add this key in each of your requests to the Harry Potter API. You do not need a key for the Star Wars API.

To actually use your key, you'll need to attach it to the end of your request URL. For example, let's say your key is the string below:

```
apiKey = "$2a$10$GtizJjARVxLrtGV48PWen.qjzANT3QFwRUSH/byVQvFUJC4utULv."
```

Now, we can add our key paramater to our API request. Let's say we want to access the `spells` endpoint. We can add our key as a parameter to our URL, shown below:

```
baseUrl = "https://www.potterapi.com/v1/"
url = baseUrl + "spells" + "?key=" + apiKey
r = requests.get(url)
```

Meet New People

Function Name: meetNewPeople()
Parameters: house (`str`)
Returns: list of people (`list`)
Description: You're new to Hogwarts School of Witchcraft and Wizardry and you want to get to know all the people in your house. Using the Harry Potter API, write a function that lets you look up all the people that live in a given house. You're particularly interested in being friends with only the **pure-blood**, **non-death eaters**. Return a list of all the names of people who you can be friends with. You will always be given a valid house.

```
>>> meetNewPeople("Slytherin")
['Phineas Nigellus Black', 'Theodore Nott', 'Salazar Slytherin',
'Andromeda Tonks']
```

```
>>> meetNewPeople("Hufflepuff")
['Cedric Diggory', 'Ernest Macmillan']
```

Matching Students

Function Name: matchingStudents()
Parameters: character name (`str`)
Returns: list of students (`list`)
Description: You want to see how many students are similar to a given character. Using the Harry Potter API, write a function that returns a list of all characters that are **students** and have the same **blood status** and **house** as the given character. The given character should not be included in this list. The given character will always be a real character, and will have a house and blood status.

```
>>> matchingStudents("Albus Dumbledore")
['Seamus Finnigan', 'Harry Potter', 'Dean Thomas']
```

```
>>> matchingStudents("Amelia Bones")
['Justin Finch-Fletchley', 'Zacharias Smith']
```

Similar Characters

Function Name: similarCharacters()
Parameters: movieID1 (`str`), movieID2 (`str`)
Returns: number of people (`int`)
Description: You just found out that the Star Wars API has info on 6 Star Wars movies, and you've decided to binge watch all of them this weekend. Given two movie IDs, you want to find out how many characters appear in both movies. Using the Star Wars API, write a function that takes in two movie IDs, and returns the number of characters that appear in both movies. If any of the movie IDs are invalid, return `0`.

Hint: Look for an API endpoint related to films that also takes an "id". You only need 2 API requests to solve this problem.

```
>>> similarCharacters('1', '2')
9
```

```
>>> similarCharacters('1', '7')
0
```

Space Drifting

Function Name: spaceDrifting()
Parameters: passengers(`int`), max price(`int`)
Returns: list of valid starships (`list`)
Description: You recently heard about some illegal starship racing going down on the intergalactic streets, and you want in. But first, you'll need a starship. Create a function that takes in a minimum number of passengers (inclusive) and maximum price that you're willing to pay (exclusive), return a list of starships that meet our requirements in the form of tuples. The first element of each tuple is the **name** of the starship, and the second element is the **manufacturer** of the starship. If there are no starships that satisfy our requirements, return an empty list.

Note: For simplicity's sake, we will only be looking at the first page of results of starships in the API.

Hint: Some starships will have values of 'n/a' for passengers or some other format of string which prevents us from converting that string to an integer. We will ignore these starships; to do this, you might consider using `Try/Except`.

```
>>> spaceDrifting(4, 350000)
[('Sentinel-class landing craft', 'Sienar Fleet Systems, Cyngus Spaceworks'),
('Millennium Falcon', 'Corellian Engineering Corporation')]
```

```
>>> spaceDrifting(0, 1000000)
[('Sentinel-class landing craft', 'Sienar Fleet Systems, Cyngus Spaceworks'),
('Millennium Falcon', 'Corellian Engineering Corporation'),
('Y-wing', 'Koensayr Manufacturing'), ('X-wing', 'Incom Corporation')]
```

Leia's Perfect Match

Function Name: perfectMatch()
Parameters: list of candidates (`list`)
Returns: list of potential matches (`list`)
Description: After unforunately learning that Luke was her brother in *Return of the Jedi*, Leia has decided to try out some intergalactic speed dating. With so many people in the galaxy, Leia needs some kind of filter so she doesn't end up wasting her time. Leia has decided that any candidate must be at least **180** centimeters tall (inclusive), must be a **male**, and **CAN-NOT** be Darth Vader or Luke Skywalker (they are both automatically disqualified). Write a function that takes in a list of candidates, and returns a list of potential matches based on Leia's requirements. This list should be in **alphabetical order**. If either the height or gender are unavailable, then the candidate should be disqualified. If none of the candidates are a good match for Leia, then return an empty list.

Note: For simplicity's sake, we will only be looking at the first page of results of people in the API.

```
>>> perfectMatch(["C-3PO", "Darth Vader", "Biggs Darklighter"])
["Biggs Darklighter"]
```

```
>>> perfectMatch(["Luke Skywalker", "Leia Organa"])
[]
```

Grading Rubric

Function	Points
meetNewPeople()	20
matchingStudents()	20
similarCharacters()	20
spaceDrifting()	20
perfectMatch()	20
Total	**100**

Provided

The `HW08.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW08.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the "Re-submit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW08.py` on Canvas.