Homework 8 - APIs

CS 1301 - Intro to Computing - Fall 2024

Important

- Due Date: **Thursday, November 7th, 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 Ed Discussion
  - 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 to extend your knowledge of frequently used data exchange formats such as JSON. We'll be pulling data from the web with APIs. This homework will require you to implement 5 functions. You have been given the HW08.py skeleton file to aid you with your responses.

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: Zoe Daniel, Jane Crowley, Sila Keha & Priya Annapureddy

Helpful Information to Know

We will be using the Taylor Swift API. The base url is https://taylor-swift-api.sarbo.workers.dev/, with several endpoint options (songs, lyrics, albums) that are described in detail in the documentation.

If you make a valid request with the URL: https://taylor-swift-api.sarbo.workers.dev/albums, you will receive the following JSON formatted response:

```
[
    {
        "album_id": 1,
        "title": "1989",
        "artist_id": 1,
        "release_date": "2014-10-27"
    },
    {
        "album_id": 2,
        "title": "Taylor Swift",
        "artist_id": 1,
        "release_date": "2006-10-24"
    },
    {
        "album_id": 3,
        "title": "Fearless",
        "artist_id": 1,
        "release_date": "2021-04-09"
    },
        ...
]
```

If you make a request with an invalid URL, you will receive the following response:

```
404 Not Found
```

or if the URL is valid but no page exists there:

```
{
    "error": "Album with the albumID of 13 does not exist"
}
```

Taylor Swift endpoints correspond to albums, songs, lyrics, etc., searchable by querying, with optional URL extensions for specific content retrieval or actions, supporting filtering resources by ID for requests like:

```
https://taylor-swift-api.sarbo.workers.dev/lyrics/3
```

to retrieve lyric information about the song `Out of the Woods` with the song ID of `3`.

You will have to use this form of querying to retrieve specific information for questions. We recommend you familiarize yourself with the API before beginning.

Note: Your code may take a minute to run while it retreives all the data!

Song Capsule

Function Name: findReleaseDate()
Parameters: song (`str`)
Returns: releaseDate (`str`)
Description: You hear a Taylor Swift song on the radio, but you don't remember when it came out.

Given the title of song, **return** a string containing its release date. If the song is not found, **return** the string `"Still writing our song..."`

Hint: To get started, locate the album a song belongs to using the information found at the `base url + "songs"` endpoint. Then, use that album to find release date information using the same base url and the `"albums"` endpoint. The base url is "https://taylor-swift-api.sarbo.workers.dev/".

```
>>> findReleaseDate("Sparks Fly")
"2010-10-25"
```

```
>>> findReleaseDate("Red Wine Supernova")
"Still writing our song..."
```

Pass the Aux

Function Name: longestSong()
Parameters: albumID (`int`)
Returns: longest (`str`)
Description: Your friend just passed you the aux cord in their car. You're only allowed to play one Taylor Swift song, so you want to make it last as long as possible.

Given an album ID, find how long it would take to sing each song by finding each track's word count and assuming that it takes 1.5 seconds to sing each word in a song. For every song that will take over 10 minutes to sing, **print** the string `"{song} would be a good choice..."`. Finally, **return** information about the longest song on the album in terms of how long it will take to sing it as a string in the format `"{longestSong}, {minutes} mins and {seconds} secs"`, with minutes and seconds rounded to two decimal places. If the given album ID is invalid, **return** the string `"Let's listen to something else."`.

Note: Each word in the song's lyrics (separated by a *space*) should be counted toward its word count. To separate each word by *only* spaces, be sure to use `.split(" ")`.

```
>>> longestSong(7)
Miss Americana & The Heartbreak Prince would be a good choice...
Paper Rings would be a good choice...
Daylight would be a good choice...
Miss Americana & The Heartbreak Prince, 11.0 mins and 21.0 secs
```

```
>>> longestSong(10)
Would've, Could've, Should've would be a good choice...
Would've, Could've, Should've, 10.0 mins and 36.0 secs
```

Lyric Matching

Function Name: lyricFinder()
Parameters: lyricList (`list`), album (`str`)
Returns: songList (`str`)
Description: You have a Taylor song on the tip of your tongue, but you just can't remember it! Luckily, you can recall a handful of lyrics from the song and which album it's on.

Write a function that takes in two parameters: a list of lyrics from a song and an album title. **Return** a list of tuples that contains the name of each song on the album that includes at least one of the lyrics from lyricList the number of lyrics from lyricList that are included in the song in the format `[(lyricCount, song), ...]`. Do not include songs with counts of 0. The list should be sorted in descending order by the lyric count. If there are ties in the count, break them using the reverse alphabetical order of the two song names. However, if the given album does not exist, simply return the string `That's not a Taylor album!`.

Note: Checking the lyrics should be case-insensitive.

```
>>> lyricFinder(["perfectly fine", "in the rain"], "Fearless")
[(2, 'The Way I Loved You'), (1, 'Mr. Perfectly Fine'), (1, 'Hey Stephen')]
```

```
>>> lyricFinder(["your eyes", "reputation", "I know"], "Reputation")
[(3, 'End Game'), (3, 'Delicate'), (1, 'So It Goes…'),
(1, 'Call It What You Want'), (1, '...Ready for It?')]
```

Spotify Sweep

Function Name: playlistOrganizer()
Parameters: albumList (`list`), leastFavoriteSongs (`list`)
Returns: playlistDict (`dict`)
Description: Since you've been listening to Taylor Swift nonstop on Spotify, you figure it's time to reorganize your playlists.

Given a list of album titles, return a dictionary mapping the number of liked songs on each album to a list of albums that contain the same amount of songs. The number of liked songs can be calculated by subtracting the number of least favorite songs from the song count of the album.

```
>>> albumList = ["1989", "Red", "Evermore"]
>>> leastFavoriteSongs = ["happiness", "Blank Space", "right where you left me",
                          "evermore", "coney island", "gold rush", "ivy"]
>>> playlistOrganizer(albumList, leastFavoriteSongs)
{11: ['1989', 'Evermore'], 29: ['Red']}
```

```
>>> albumList = ["The Tortured Poets Department", "Lover"]
>>> leastFavoriteSongs = ["Daylight"]
>>> playlistOrganizer(albumList, leastFavoriteSongs)
{17: ['Lover'], 31: ['The Tortured Poets Department']}
```

We Have Taylor Swift at Home...

Function Name: favAlbum()
Parameters: album (`str`)
Returns: None (`NoneType`)
Description: You're bummed you weren't able to get Eras Tour tickets for you and your best friend, so you decide to have a little concert at home with all your favorite songs.

Given a Taylor Swift album, write to a CSV file called `album.csv` that contains details about each song on the album. You should include the song ID, name, and number of times the song title is mentioned in its lyrics, as formatted below.

Output: `album.csv`

```
Song ID, Song Name, # Song Title Mentions
Song ID 1, Song Name 1, # Song Title Mentions 1
Song ID 2, Song Name 2, # Song Title Mentions 2
```

Note: Remove all commas from the titles of the songs (before counting the number of song title mentions) and make sure there isn't a newline character present at the end of your file to ensure proper formatting of the file.

Test cases are continued on the next two pages.

```
>>> favAlbum("Folklore")
```

Output of `album.csv`

```
Song ID, Song Name, # Song Title Mentions
116, exile, 3
117, mirrorball, 4
118, seven, 1
119, august, 6
120, mad woman, 7
121, the last great american dynasty, 2
122, cardigan, 3
123, hoax, 2
124, betty, 5
125, My Tears Ricochet, 3
126, peace, 5
127, the 1, 0
128, invisible string, 3
129, this is me trying, 6
130, epiphany, 2
131, illicit affairs, 2
169, the lakes, 3
```

```
>>> favAlbum("Fearless")
```

Output of `album.csv`

```
Song ID, Song Name, # Song Title Mentions
27, Breathe, 7
28, Love Story, 4
29, Tell Me Why, 4
30, The Best Day, 4
31, Hey Stephen, 5
32, You Belong with Me, 11
33, Fifteen, 9
34, Change, 3
35, Fearless, 9
36, Forever & Always, 0
37, The Way I Loved You, 8
38, White Horse, 4
39, You're Not Sorry, 0
170, We Were Happy, 9
171, Mr. Perfectly Fine, 0
172, You All Over Me, 5
173, Don't You, 0
174, That's When, 0
175, Bye Bye Baby, 9
```

Grading Rubric

Function	Points
findReleaseDate()	20
longestSong()	20
lyricFinder()	20
playlistOrganizer()	20
favAlbum()	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 "Resubmit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW08.py` on Canvas.