CIS 41B - Lab 3: web scraping and data storage with requests, beautifulsoup, sqlite3; review string methods and regex

Write an application that lets the user search for data on 2021 movies.

The application has 2 parts: lab3back.py (the backend to get data) and lab3front.py (the frontend with the GUI)

The 2 parts *do not* work with each other (no importing of data or methods from one file to the other).

Instead: - the lab3back.py will produce a JSON file and an SQL database file

- the lab3front.py will read from the SQL database to display data to the user

lab3back.py description

This module has 2 distinct parts: A and B. After you finish writing part A, you comment it out and write part B.

Here's a diagram: webpage === Part A ===> JSON file === Part B ===> SQL Database

Part A

The Rotten Tomatoes movie review website has a page on what they consider to be the most anticipated movies for 2021: <https://editorial.rottentomatoes.com/article/most-anticipated-movies-of-2021/>

The movies are listed in order of months, and include both movies that have been released as well as movies that will be released in the upcoming months.

From the URL above, use requests and beautifulsoup to extract the following information for each movie:

* name of the movie (store as a string)
* url of the web page for the movie (store as a string)
* director name (if there are multiple names, store all names as one string)
* actor names (store one name per string)
* month in 2021 that the movie was/will be released (store as a string)

From the extracted data, create a JSON file to store the data:

* Data for one movie should be grouped together. This means don't have a list of all movie names, then list of all urls, then list of all directors...
* The actor names should be stored individually. This means don't have one string which is a comma separated list of all the actor names. If there are 5 actors, then there should be 5 strings.
* Due to the point above, it's helpful to store in the JSON file the largest count of actor names for one movie.

It's up to you how to organize the data in the JSON file, the goal is to make it easy for you to get the data back out.  
For example, storing all the data of all the movies in a flat 1D list is not a good idea.

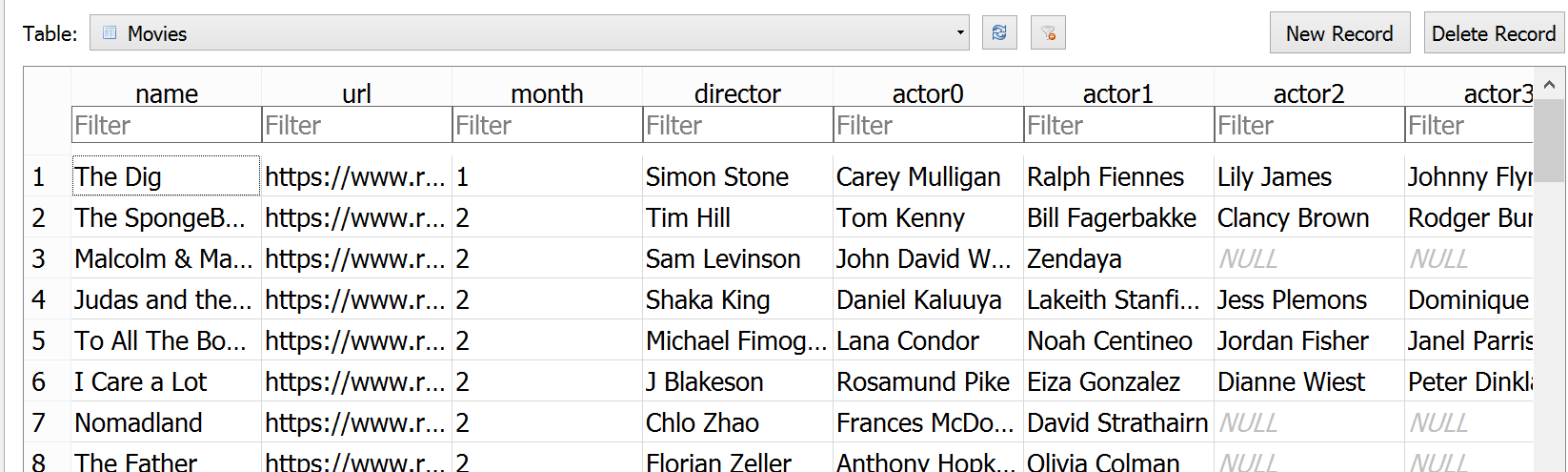
Part B

After you've created the JSON file, you don't need to spend time fetching data from the website any more. Instead you can use the saved data in the file.

1. Store data from the JSON file into an SQL database. The database has 2 tables:

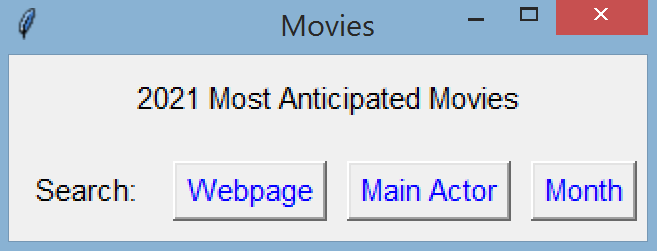
* a table to store all the data for each movie, one movie per row of the table
* a table of all unique months. The table of movie data will have foreign keys into the month table.

1. The movie table has the following columns: name, url, month, director, and one or more actor columns, depending on how many actors are listed for a movie.  
   On the next page is a partial view of a sample movie data table from the DB Browser tool.  
   The column names and the order of the columns are your choices. You don't have to follow the sample table.



Note that the month column contains the foreign key into the months table.

lab3front.py description

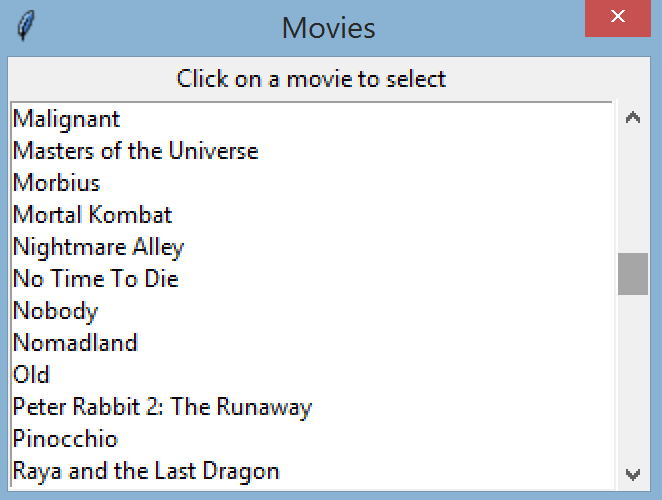
1. Write a GUI with 3 window classes: main window, dialog window, display window.
2. The main window has 3 buttons and text to explain the application.   
   Feel free to use your own wording for the buttons and text. Here's a sample main window: 

The user can:

* See the webpage of a movie
* See the director and all actors for a movie
* See all movies released in a chosen month

1. When the user clicks on the Webpage button:

* A dialog window with a label, listbox, and scrollbar shows up.
* The label tells the user to click on a movie to select it.
* The listbox has a height of 12 and has a scrollbar on the right.
* The movie names in the listbox are in alphabetical order.
* When the dialog window appears, all other windows are de-activated and the user cannot go to the other windows and click on buttons to activate them.
* The user must either click on a choice in the lisbox or click X to close the window.
* If the user clicks X, then the dialog window closes and the user gets back to the main window.
* If the user clicks on a movie, then the dialog window closes and the user movie choice is saved.
* The main window uses the saved choice to open up the web page for the movie in a browser window.  
  To open up a web page in a browser, in your code:  
   import webbrowser  
   webbrowser.open(URL)



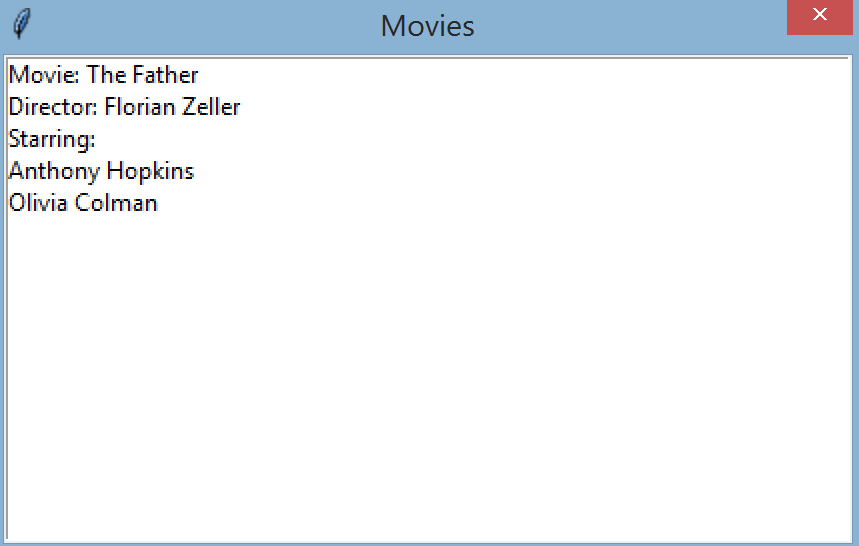
Movies after scrolling to middle of list

1. When the user clicks on Main Actor button on the main window:

* The same dialog window of step 3 is created but with 2 changes: the label and the content of the listbox.
* The label should tell the user to select an actor (not a movie as in step 3).
* The actor names in the listbox are the *first* actor name listed for a movie. The names in the listbox are in alphabetical order.
* All other windows are de-activated while the dialog window is shown.
* The user must either on a name in the listbox or click X to close the window.
* If the user clicks X, then the dialog window closes and the user gets back to the main window.
* If the user clicks on a name, then the dialog window closes and the user choice is saved.
* The main window uses the saved choice to search the database for a movie with the chosen actor name as the first listed name.
* Then the main window creates a display window to display the movie name, director, and all actors.

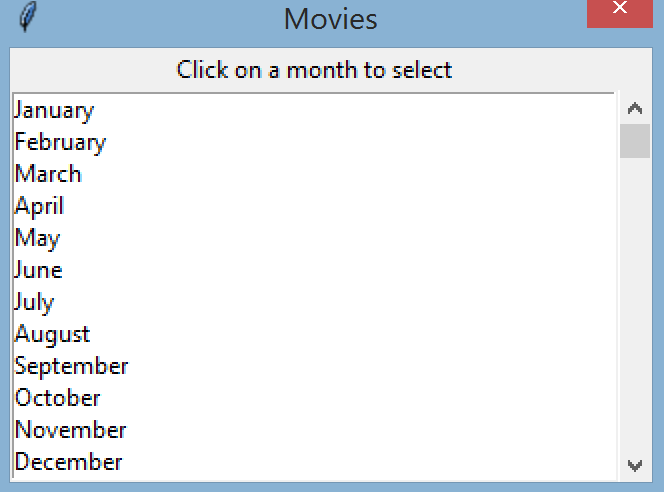
1. The display window has:

* A listbox that contains the data for one movie.
* The data displayed are: movie name, director, and all actors
* The actor names are listed in the same order as shown on the website. (The main actors get offended if they don't get top billing)
* Each type of data is displayed with an explanation, as shown in the sample window, but feel free to use your own wording.



Result from the user choosing Anthony Hopkins for Main Actor

1. When the user chooses the Month button at the main window:



* The same dialog window as step 3 shows up with 2 changes:

the label and the listbox data.

* The label should ask the user to click on a month to select it.
* The listbox data contains the 12 months of the year, in   
  calendar order.
* The 12 months of the year must come from the database  
  month table. Do not hard code the strings.
* Just as with steps 3 and 4, the user choice is saved.
* The main window uses the saved choice to query the database for all movies with the chosen month and display them in a display window (described in step 5). Each movie is one line of the display window.

Some tips on handling data:

* lab3front.py should get data from the database file. Do not go to the website or import anything from lab3back.py or use the JSON file.
* Don't extract all movie data from the database to store in memory. Only query and fetch what you need.
* lab3front.py should connect to the database and keep the connection open for all transactions. Close the database when the GUI is closed.
* The main window object should do all database queries and pass data to other objects and methods if needed.
* For Main Actor, the name Benedict Cumberbatch will show up twice in the list of actors. This is fine. The website has a duplicate movie, but the 2 descriptions for the movie are slightly different. It's most likely a mistake on the website and you don't have to correct their mistake.

Documentation reminders:

* Your name and a short description of the code at the top of each source file.
* A docstring for each public method

When done, turn in 4 files: lab3front.py, lab3back.py, the db file, and the json file.