OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Task 3: 5 Page Project Presentation

Reilly Ertman

Student ID: 3283484

16th January 2024

All rights and privileges of this work belong to the OTH Regensburg. I consent to handing over this intellectual property at submission.

v1 January 2024

Table of Contents

List of Figures

1 Brief Project Description

The first task is to extract HTML content from IMDB to create and populate a database. More information on the algorithms used to extract data can be found later on this document.
The page in question is http://www.imdb.com/list/ls053501318/ The process of fulfilling this project is as follows:

- Step 1: “py setup.py” will call “webscraper.py”, which scrapes IMDB pages to create “Master_data_base.db”.
- Step 2: Once we have the database, we run our real-time terminal program, which will listen for user-input and manipulate the data in a meaningful way: “py runtime_program.py”. The second task is to display the saved values in a meaningful way with smart SQL enquiries. This task should run in a while(true) run-time loop, where the user can execute as many enquiries/commands as they like. The program will run in a shell terminal. For example, if the user inputs “2,” a function is called to display all awards won by an actor, or if a user inputs ‘5,’ a function will be called to output biographical information about the actor. Also the terminal program should be user-friendly and well-formatted.

2 Main Issues in Project

Without a doubt the biggest problem was with task 1: webscraping IMDB’s page. The data to extract is located in a certain section of the HTML page. I needed to find the correct tag and extract the information. This was a tiresome process.

Also, IMDB often changes the ID for the html tags, because they want to make webscrapping their pages difficult. Changing the html tags constantly means webscrappers constantly have to update their scripts.

3 Modules, Data Structures, Tools used in Project

Modules
We will require a few python packages for the program to work:

- 1. Import requests
- 2. From bs4 import BeautifulSoup This package allows us to grab the HTML code embedded on web page for web scraping: example functions: requests.get; beautifulsoup(text, type_of_html_parsing); html_page.find_all
- 3. From random import randint
- 4. From time import sleep between each web scrape, we need to wait a random amount of time so that the website does not think we are a script and reject our https request.
- 5. Device header Even when our web scraping application waits to access webpages, IMDB still receives https requests coming from a script. IMDB will therefore reject our https request. We must disguise our requests, as if they were coming from a device. For example, an Ipad using Mozilla: example: HEADERS = ’User-Agent’: ’Mozilla/5.0 (iPad; CPU OS 12_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile/15E148’
- 6. Import sqlite3 This is our API to send/receive SQL enquiries between python code and DB.

Data Structures
The data structures necessary for this project are limited. All structures are created at run-time and destroyed at program termination. Only the database data structure will be kept in cold storage. The general principle of data scraping is, put the HTML page in a container and break up the page into smaller and smaller pieces, until you extract the necessary data. Then use the same set of instructions to loop through the rest of the HTML content. To

tackle this task, I will need a front end and a back end. The front end will listen for correct input. When data is to be saved or a valid sql inquiry wishes to output data, the backend is called to handle the request.

Tools

- 1. Windows Powershell on Windows 10 will be the Interface/GUI to help facilitate user inputs
- 2. Python 3.12.0 to compile code
- 3. Program SQLite 3.35.5 to offer GUI and create actual database file
- 4. SQL is used to manage enquiries.
- 5. Stable internet connection to grab HTML content

4 Flowchart of Project Design

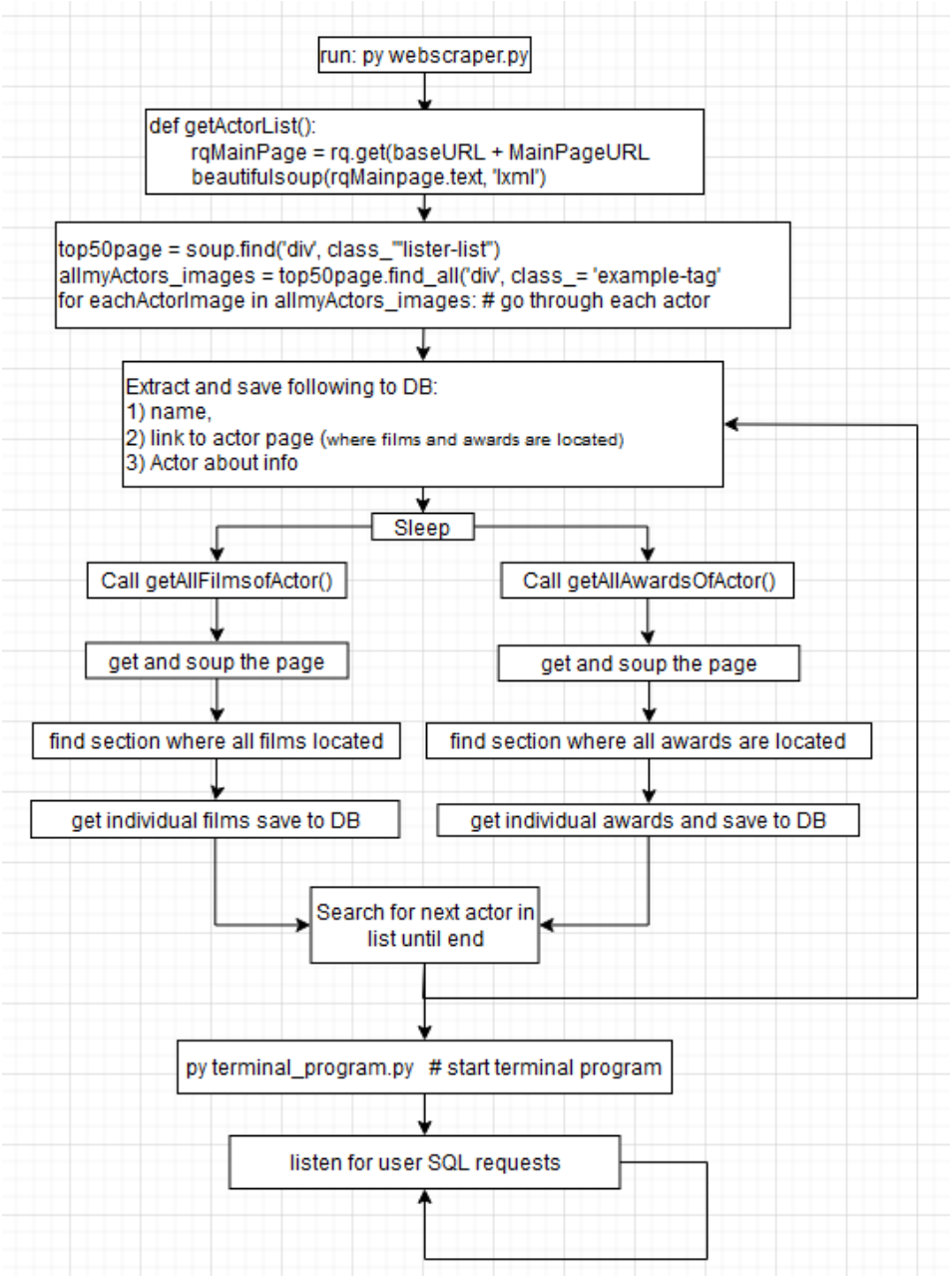

Figure A. Project Flowchart

5 Functions used in project

Def getActorList():

This function visits the URL http://www.imdb.com/list/ls053501318, scrubs it and saves the (1) actors name, (2) actor's biographical info (3) actor's biography URL and (4) actor's award page URL to the database.

It also calls the functions def getAllfimsofActor() and def getAllAwardsofActor()

def getAllFilmsofActor(link_to_actor):

The function sleeps 2 seconds to simulate human behavior. If too many http requests are sent within milliseconds, IMDB page will think we are a script and fail our requests.

This function takes the actor's page URL as an argument.

It scrubs the actor page URL for all films the actor has been in and saves the following to the database: (1) the actor's name, (2) each film title, (3) each film year, (4) the genre of each film and (5) the rating of each film.

def getAllAwardsOfActor(link_to_actorsawards):

The function sleeps 2 seconds to simulate human behavior. If too many http requests are sent within milliseconds, IMDB page will think we are a script and fail our requests.

This function takes the actor's award page URL as an argument.

It scrubs the award page URL for all awards the actor has received and saves the following to the database: (1) award year, (2) award name, (3) award description, (4) actor's name.

6 Screenshot of Graphical Interface

In this example screenshot, I show the list of all available actors. I now want to see all the movies that Orlando Bloom has acted in.

I type and enter 2 to do something else with the data.
I type and enter 3 to see all time movie names and years.
I type and enter 38 to see Orlando Bloom's movie names and years.

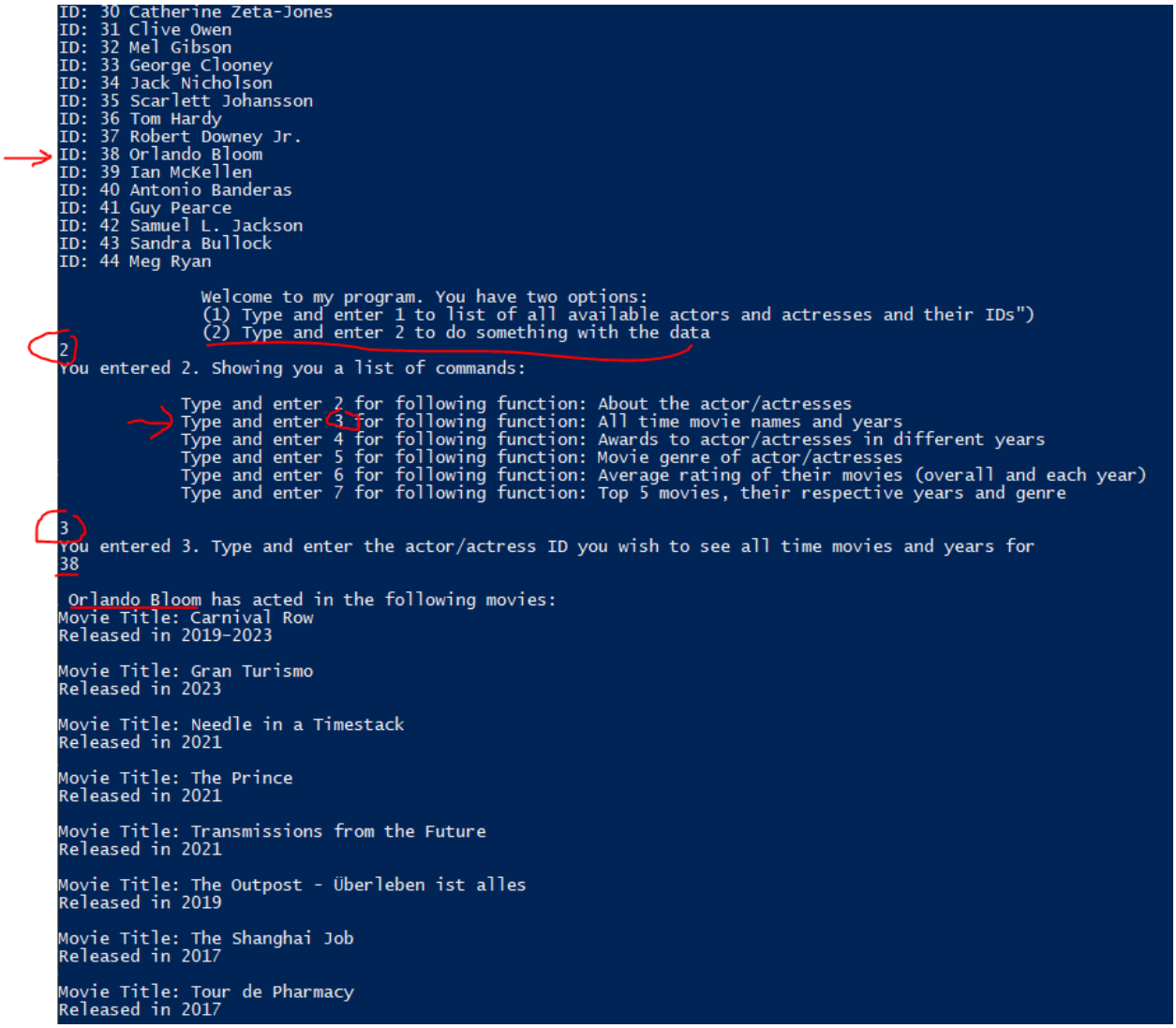

Figure B. Command Line Interface