Scrape Movie Details using BeautifulSoup

Table of Contents

Learning Outcome

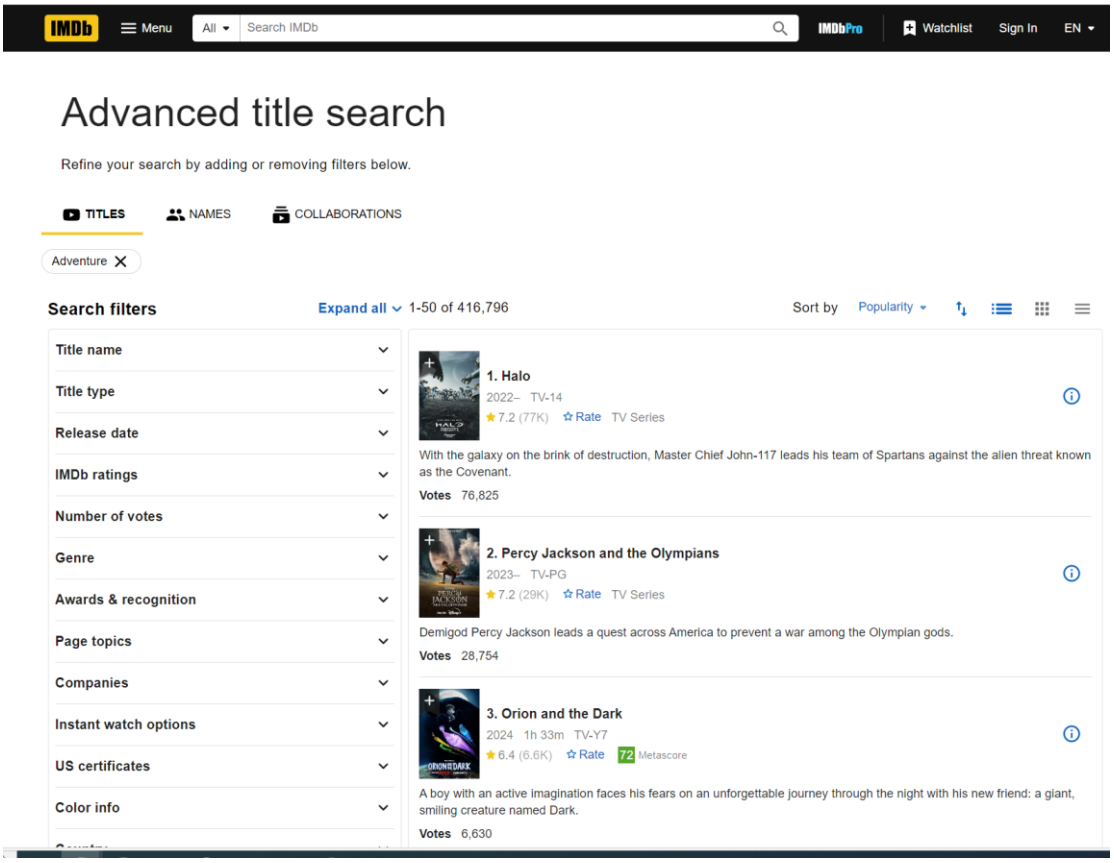
At the end of this lab, you will be able to extract the movie details from IMDB using BeautifulSoup and store the data in a CSV file.

Web Scraping

Web Scraping is the extraction of data from a website, and in this case, the Python library called Beautiful Soup will be used. The scraper loads the HTML code of the page the user wants to collect data from, then the scraper will either extract all the data on the page or the user will go through the process of selecting the specific data they want from the page. That is done by looking at the website's HTML code and selecting the the specific element or tag that the desired information is in.

Data to Scrape

In this practical we will look at how to do web scraping on imdb.com to fetch information about movies with different genres using Python BeautifulSoup and requests. IMDB (Internet Movie Database) website is owned by Amazon, is one of the best platforms for finding information about films, television shows, web series, etc.



The data that we want to extract from it are:

- Movie title
- Star
- Metascore
- Description

To extract all of this data, our scrapper will need to go inside each film's webpage. Now let's start scrapping.

Download **BeautifulSoup Scrape Movie Details Starter.ipynb** and open it in Jupyter Notebook.

Load Libraries

Before we begin, we need to import the libraries that will be used for this practical.

```
# load packages
from bs4 import BeautifulSoup
import requests
import pandas as pd
```

Getting URLs of different pages

The first thing we need to do is to get URLs of different movie genres, for example, the genres include Adventure, Animation, Biography, etc.

```
# URls of different Genre page
genres = ["Adventure","Animation","Biography"]

url_dict = {}
for genre in genres:
    formated_url = f"https://www.imdb.com/search/title/?genres={genre}"
    url_dict[genre] = formated_url

print(url_dict)
```

The code iteratively changes the URL with different genres stored in the genres list so that we will get URLs of different movie genres. The genre and the corresponding URLs are then stored inside a dictionary.

Parsing movie information

Now let's parse the movie information from IMDB.

First we will sends a request to the specified URL and returns a response.

```
url = "https://www.imdb.com/search/title/?genres=Adventure"

# Sending a request to the speciifed URL
result = requests.get(url)
print(result.status_code)
```

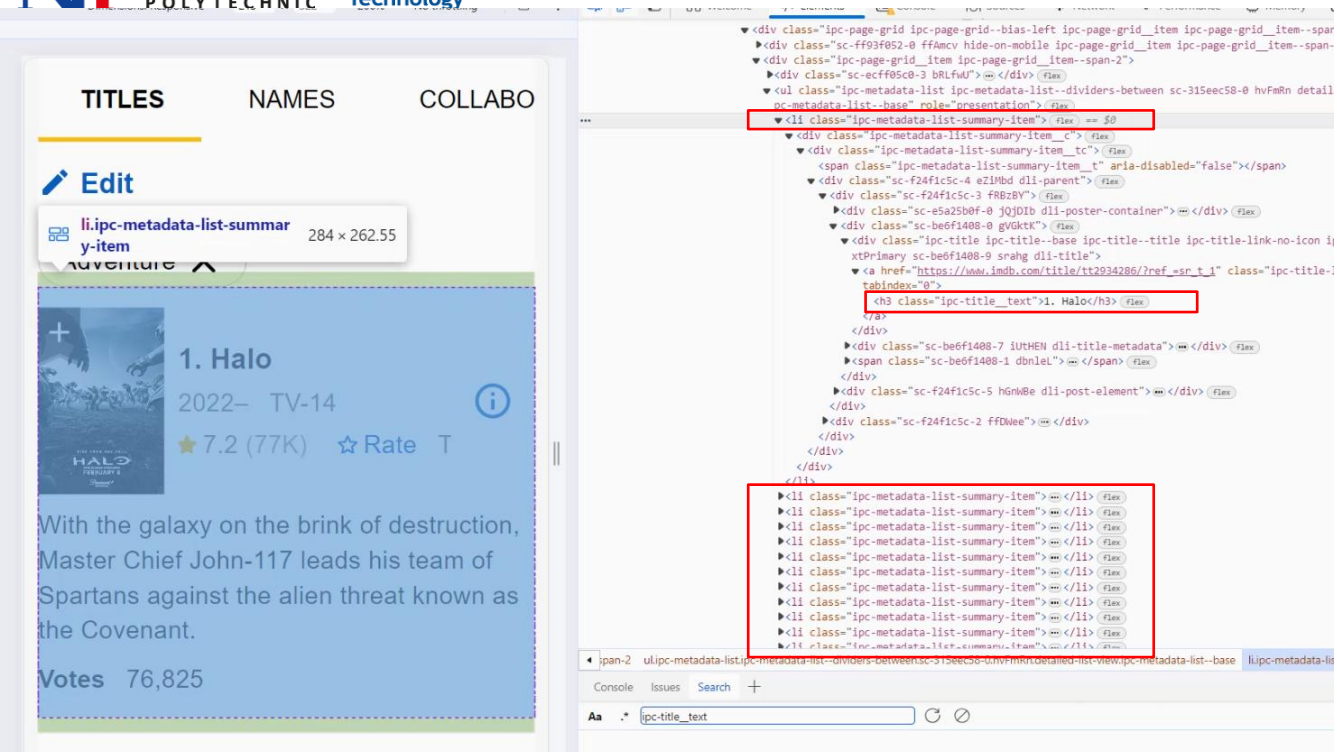
Add the User-Agent header of the GET request if the page rejects GET request.

```
url = "https://www.imdb.com/search/title/?genres=Adventure"

headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) '
                  'AppleWebKit/537.36 (KHTML, like Gecko) '
                  'Chrome/50.0.2661.102 Safari/537.36'
}

# Sending a request to the speciifed URL
result = requests.get(url, headers=headers)
print(result.status_code)
```

This response is then converted to an HTML form using Beautiful Soup.

```
# Converting the response to Beautiful Soup Object
content = BeautifulSoup(result.content, 'html')
```

Next we will iterate through the content to get the list of movies data. We will need to lookat the HTML code to see where we can get all of that information.



All movies titles are in a 'li' element, inside the '.ipc-metadata-list-summary-item' class tag. Inside thatelement, we will need to go to the 'h3' element tag. Inside that tag, you can find the movie title. Now that we know where to find all the codes and we will iterate through the list of movies and store the data in a dictionary.

```
import pandas as pd

# Converting the response to Beautiful Soup Object
content = BeautifulSoup(result.content, 'html')

# Iterating throught the list of movies
movie_list=content.find_all('li',class_='ipc-metadata-list-summary-item')

m_list=[]

for movie in movie_list:
    title=movie.find('h3',class_='ipc-title__text').get_text()


    star=movie.find('span',class_='ipc-rating-star--rating')
    if star is None:
        star=""
    else:
        star=star.get_text()


    metascore=movie.find('span',class_="sc-ae9e80c5-0 gXcoKx metacritic-score-box")
    if metascore is None:
        metascore=""
    else:
```

```
        metascore=metascore.get_text()


    description=movie.find('div',class_='ipc-html-content-inner-div').get_text()


    data={
        "title":title,
        "star":star,
        "metaScore":metascore,
        "description":description
    }

    print(data)
```

Creating a scraping function

Now let's create a function that does the same as above but it can be reused several times for different URLs.

```
def get_movies(url):

    headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) '
                'AppleWebKit/537.36 (KHTML, like Gecko) '
                'Chrome/50.0.2661.102 Safari/537.36'
    }
    result = requests.get(url, headers=headers)

    content = BeautifulSoup(result.content, 'html')

    movie_list=content.find_all('li',class_='ipc-metadata-list-summary-item')

    m_list=[]

    # Iterating throught the list of movies
    for movie in movie_list:
        title=movie.find('h3',class_='ipc-title__text').get_text()


        star=movie.find('span',class_='ipc-rating-star--rating')
        if star is None:
            star=""
        else:
            star=star.get_text()


        metascore=movie.find('span',class_="sc-ae9e80c5-0 gXcoKx metacritic-score-box")
        if metascore is None:
```

```
            metascore=""
        else:
            metascore=metascore.get_text()


        description=movie.find('div',class_='ipc-html-content-inner-div').get_text()


        data={
            "title":title,
            "star":star,
            "metaScore":metascore,
            "description":description
        }

        m_list.append(data)

    return pd.DataFrame(m_list)
```

This function creates a python dictionary that contains all the information we parsed from the web page and then it return a pandas data frame.

```
url = "https://www.imdb.com/search/title/?genres=Adventure"

# Calling the function
get_movies(url)
```

Scraping movies of different genres

The **get_movies()** function we write above can parse details from the IMDB web page of different genre URLs and can save them as a CSV file. So by using this function it is possible to scrape all genres. So let's see how this can be done.

```
df_data = pd.DataFrame()

for genre, url in url_dict.items():
    df_data = pd.concat([df_data, get_movies(url)])

df_data.to_csv('movies.csv')
```

~The End~