Hands-on lab: Webscraping

Estimated Effort: 30 mins

Web scraping is used for extraction of relevant data from web pages. If you require some data from a web page in a public domain, web scraping makes the process of data extraction quite convenient. The use of web scraping, however, requires some basic knowledge of the structure of `HTML` pages. In this lab, you will learn the process of analyzing the `HTML` code of a web page and how to extract the required information from it using web scraping in Python.

Objectives

By the end of this lab, you will be able to:

- Use the `requests` and `BeautifulSoup` libraries to extract the contents of a web page

- Analyze the `HTML` code of a webpage to find the relevant information

- Extract the relevant information and save it in the required form

Scenario

Consider that you have been hired by a Multiplex management organization to extract the information of the top 50 movies with the best average rating from the web link shared below.

```
1. 1
```

```
1. https://web.archive.org/web/20230902185655/https://en.everybodywiki.com/100_Most_Highly-Ranked_Films
```

Copied!

The information required is `Average Rank`, `Film`, and `Year`.
You are required to write a Python script `webscraping_movies.py` that extracts the information and saves it to a `CSV` file `top_50_films.csv`. You are also required to save the same information to a database `Movies.db` under the table name `Top_50`.

Initial steps

You require the following libraries for this lab.

1. `pandas` library for data storage and manipulation.

2. `BeautifulSoup` library for interpreting the `HTML` document.

3. `requests` library to communicate with the web page.

4. `sqlite3` for creating the database instance.

While `requests` and `sqlite3` come bundled with `Python3`, you need to install `pandas` and `BeautifulSoup` libraries to the IDE.

For this, run the following commands in a terminal window.

```
1. 1
2. 2
```

```
1. python3.11 -m pip install pandas
2. python3.11 -m pip install bs4
```

Copied! Executed!

Now, create a new file by the name of `webscraping_movies.py` in the path `/home/project/`.

You will write all of your code in this file.

Code setup

To create a Python script, call the relevant libraries and the initializations as a first step.

Importing Libraries

Import the following four libraries by adding lines of code noted below to your `webscraping_movies.py`file.

```
1. 1
2. 2
3. 3
4. 4
```

```
1. import requests
2. import sqlite3
3. import pandas as pd
4. from bs4 import BeautifulSoup
```

Copied!

Initialization of known entities

You must declare a few entities at the beginning. For example, you know the required `URL`, the `CSV` name for saving the record, the database name, and the table name for storing the record. You also know the entities to be saved. Additionally, since you require only the top 50 results, you will require a loop counter initialized to 0.

You may initialize all these by using the following code in `webscraping_movies.py`:

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
```

```
1. url = 'https://web.archive.org/web/20230902185655/https://en.everybodywiki.com/100_Most_Highly-Ranked_Films'
2. db_name = 'Movies.db'
3. table_name = 'Top_50'
4. csv_path = '/home/project/top_50_films.csv'
5. df = pd.DataFrame(columns=["Average Rank","Film","Year"])
6. count = 0
```

Copied!

Loading the webpage for Webscraping

To access the required information from the web page, you first need to load the entire web page as an `HTML` document in `python` using the `requests.get().text` function and then parse the text in the `HTML` format using `BeautifulSoup` to enable extraction of relevant information.

Add the following code to `webscraping_movies.py`:

```
1. 1
2. 2
```

```
1. html_page = requests.get(url).text
2. data = BeautifulSoup(html_page, 'html.parser')
```

Copied!

Save your file using `Ctrl+S`.

Analyzing the HTML code for relevant information

Open the web page in a browser and locate the required table by scrolling down to it. Right-click the table and click `Inspect` at the bottom of the menu, as shown in the image below.

This opens the `HTML` code for the page and takes you directly to the point where the definition of the table begins. To check, take your mouse pointer to the `tbody` tag in the `HTML` code and see that the table is highlighted in the page section.

Notice that all rows under this table are mentioned as `tr` objects under the table. Clicking one of them would show that the data in each row is further saved as a `td` object, as seen in the image above. You require the information under the first three headers of this stored data.

It is also important to note that this is the first table on the page. You must identify the required table when extracting information.

Scraping of required information

You now need to write the loop to extract the appropriate information from the web page. The rows of the table needed can be accessed using the `find_all()` function with the `BeautifulSoup` object using the statements below.

```
1. 1
2. 2
```

```
1. tables = data.find_all('tbody')
2. rows = tables[0].find_all('tr')
```

[Copied!]

Here, the variable `tables` gets the body of all the tables in the web page and the variable `rows` gets all the rows of the first table.

You can now iterate over the rows to find the required data. Use the code shown below to extract the information.

```
 1. 1
 2. 2
 3. 3
 4. 4
 5. 5
 6. 6
 7. 7
 8. 8
 9. 9
10. 10
11. 11
12. 12
```

```
 1. for row in rows:
 2.     if count<50:
 3.         col = row.find_all('td')
 4.         if len(col)!=0:
 5.             data_dict = {"Average Rank": col[0].contents[0],
 6.                          "Film": col[1].contents[0],
 7.                          "Year": col[2].contents[0]}
 8.             df1 = pd.DataFrame(data_dict, index=[0])
 9.             df = pd.concat([df,df1], ignore_index=True)
10.             count+=1
11.     else:
12.         break
```

[Copied!]

The code functions as follows.

1. Iterate over the contents of the variable `rows`.
2. Check for the loop counter to restrict to 50 entries.
3. Extract all the `td` data objects in the row and save them to `col`.
4. Check if the length of `col` is 0, that is, if there is no data in a current row. This is important since, many timesm there are merged rows that are not apparent in the web page appearance.
5. Create a dictionary `data_dict` with the keys same as the columns of the dataframe created for recording the output earlier and corresponding values from the first three headers of data.

6. Convert the dictionary to a dataframe and concatenate it with the existing one. This way, the data keeps getting appended to the dataframe with every iteration of the loop.
7. Increment the loop counter.
8. Once the counter hits 50, stop iterating over rows and break the loop.

Print the contents of the dataframe using the following:

1. 1

```
1. print(df)
```

Copied!

Save the file using Ctrl+S.

Storing the data

After the dataframe has been created, you can save it to a CSV file using the following command:

1. 1

```
1. df.to_csv(csv_path)
```

Copied!

Remember that you defined the variable `csv_path` earlier.

To store the required data in a database, you first need to initialize a connection to the database, save the dataframe as a table, and then close the connection. This can be done using the following code:

1. 1
2. 2
3. 3

```
1. conn = sqlite3.connect(db_name)
2. df.to_sql(table_name, conn, if_exists='replace', index=False)
3. conn.close()
```

Copied!

Save the file using Ctrl+S.

This database can now be used to retrieve the relevant information using SQL queries. You will learn how to do that later in the course.

Execute the code

Execute the code using the statement below in a terminal window. Make sure the current directory is `/home/python/`.
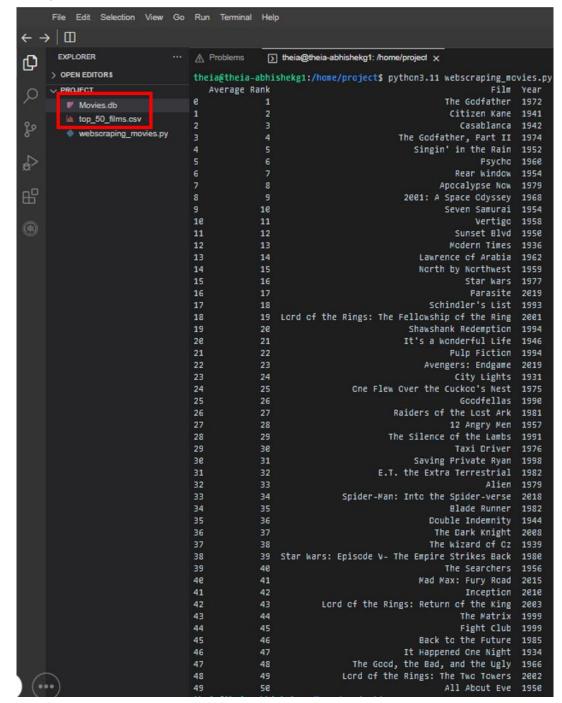
1. 1

```
1. python3.11 webscraping_movies.py
```

Copied!   Executed!

You can expect the output to look as shown below.
Note that the CSV and the DB files are also created under the project folder.

```
File   Edit   Selection   View   Go   Run   Terminal   Help

EXPLORER                    ...        ⚠ Problems         ▣ theia@theia-abhishekg1: /home/project  ✕
> OPEN EDITORS
∨ PROJECT                         theia@theia-abhishekg1:/home/project$ python3.11 webscraping_movies.py
   🏴 Movies.db                         Average Rank                                    Film   Year
   📊 top_50_films.csv            0          1                             The Godfather   1972
   🔹 webscraping_movies.py       1          2                             Citizen Kane   1941
                                  2          3                               Casablanca   1942
                                  3          4                       The Godfather, Part II   1974
                                  4          5                         Singin' in the Rain   1952
                                  5          6                                   Psycho   1960
                                  6          7                              Rear Window   1954
                                  7          8                            Apocalypse Now   1979
                                  8          9                     2001: A Space Odyssey   1968
                                  9         10                            Seven Samurai   1954
                                 10         11                                  Vertigo   1958
                                 11         12                              Sunset Blvd   1950
                                 12         13                             Modern Times   1936
                                 13         14                        Lawrence of Arabia   1962
                                 14         15                        North by Northwest   1959
                                 15         16                                Star Wars   1977
                                 16         17                                 Parasite   2019
                                 17         18                          Schindler's List   1993
                                 18         19   Lord of the Rings: The Fellowship of the Ring   2001
                                 19         20                       Shawshank Redemption   1994
                                 20         21                       It's a Wonderful Life   1946
                                 21         22                             Pulp Fiction   1994
                                 22         23                         Avengers: Endgame   2019
                                 23         24                              City Lights   1931
                                 24         25               One Flew Over the Cuckoo's Nest   1975
                                 25         26                               Goodfellas   1990
                                 26         27                     Raiders of the Lost Ark   1981
                                 27         28                            12 Angry Men   1957
                                 28         29                  The Silence of the Lambs   1991
                                 29         30                              Taxi Driver   1976
                                 30         31                       Saving Private Ryan   1998
                                 31         32                   E.T. the Extra Terrestrial   1982
                                 32         33                                    Alien   1979
                                 33         34               Spider-Man: Into the Spider-verse   2018
                                 34         35                              Blade Runner   1982
                                 35         36                          Double Indemnity   1944
                                 36         37                          The Dark Knight   2008
                                 37         38                           The Wizard of Oz   1939
                                 38         39   Star Wars: Episode V- The Empire Strikes Back   1980
                                 39         40                            The Searchers   1956
                                 40         41                       Mad Max: Fury Road   2015
                                 41         42                               Inception   2010
                                 42         43          Lord of the Rings: Return of the King   2003
                                 43         44                              The Matrix   1999
                                 44         45                              Fight Club   1999
                                 45         46                       Back to the Future   1985
                                 46         47                   It Happened One Night   1934
                                 47         48              The Good, the Bad, and the Ugly   1966
                                 48         49          Lord of the Rings: The Two Towers   2002
                                 49         50                            All About Eve   1950
```

Important Note:

> To maintain consistency of the lab structure, the web page you access is routed through an archive database. Often, in case the archive server is busy, the users may encounter delayed execution and/or an error such as:
> `requests.exceptions.ConnectionError: HTTPSConnectionPool(host='web.archive.org', port=443): Max retries exceeded with url.`
> In such a situation, try executing the code again. In case the problem persists, you can change the URL to the live version, such as:
> `https://en.everybodywiki.com/100_Most_Highly-Ranked_Films`

Lab solution

In case you are not able to get the required output from the code or are facing some errors, the final file for `webscraping_movies.py` is shared below. Please note that this is for your help, and we encourage you to first try to resolve the errors on your own.

▼ webscraping_movies.py

```
 1.  1
 2.  2
 3.  3
 4.  4
 5.  5
 6.  6
 7.  7
 8.  8
 9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
```

```
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
```

```
 1. import requests
 2. import sqlite3
 3. import pandas as pd
 4. from bs4 import BeautifulSoup
 5.
 6. url = 'https://web.archive.org/web/20230902185655/https://en.everybodywiki.com/100_Most_Highly-Ranked_Films'
 7. db_name = 'Movies.db'
 8. table_name = 'Top_50'
 9. csv_path = 'top_50_films.csv'
10. df = pd.DataFrame(columns=["Average Rank","Film","Year"])
11. count = 0
12.
13. html_page = requests.get(url).text
14. data = BeautifulSoup(html_page, 'html.parser')
15.
16. tables = data.find_all('tbody')
17. rows = tables[0].find_all('tr')
18.
19. for row in rows:
20.     if count < 50 :
21.         col = row.find_all('td')
22.         if len(col)!=0:
23.             data_dict = {"Average Rank": int(col[0].contents[0]),
24.                          "Film": str(col[1].contents[0]),
25.                          "Year": int(col[2].contents[0])}
26.             df1 = pd.DataFrame(data_dict, index=[0])
27.             df = pd.concat([df,df1], ignore_index=True)
28.             count+=1
29.     else:
30.         break
31.
32. print(df)
33.
34. df.to_csv(csv_path)
35.
36. conn = sqlite3.connect(db_name)
37. df.to_sql(table_name, conn, if_exists='replace', index=False)
38. conn.close()
```

Copied!

Practice problems

Try the following practice problems to test your understanding of the lab. Please note that the solutions for the following are not shared. You are encouraged to use the discussion forums in case you need help.

1. Modify the code to extract `Film`, `Year`, and `Rotten Tomatoes' Top 100` headers.

2. Restrict the results to only the top 25 entries.

3. Filter the output to print only the films released in the 2000s (year 2000 included).

Conclusion

In this lab, you learned how to:

1. Analyze HTML pages for relevant data position.

2. Use `BeautifulSoup`, `requests` and `pandas` libraries for web scraping the required information.

3. Store the extracted data as a CSV file and SQL Database.

Author(s)

[Abhishek Gagneja](#)

Changelog

Date	Version	Changed by	Change Description
2023-09-03	0.1	Abhishek Gagneja	Initial version created
2023-09-04	0.2	Anita Narain	ID Review
2023-09-12	0.3	Steve Hord	QA pass with edits