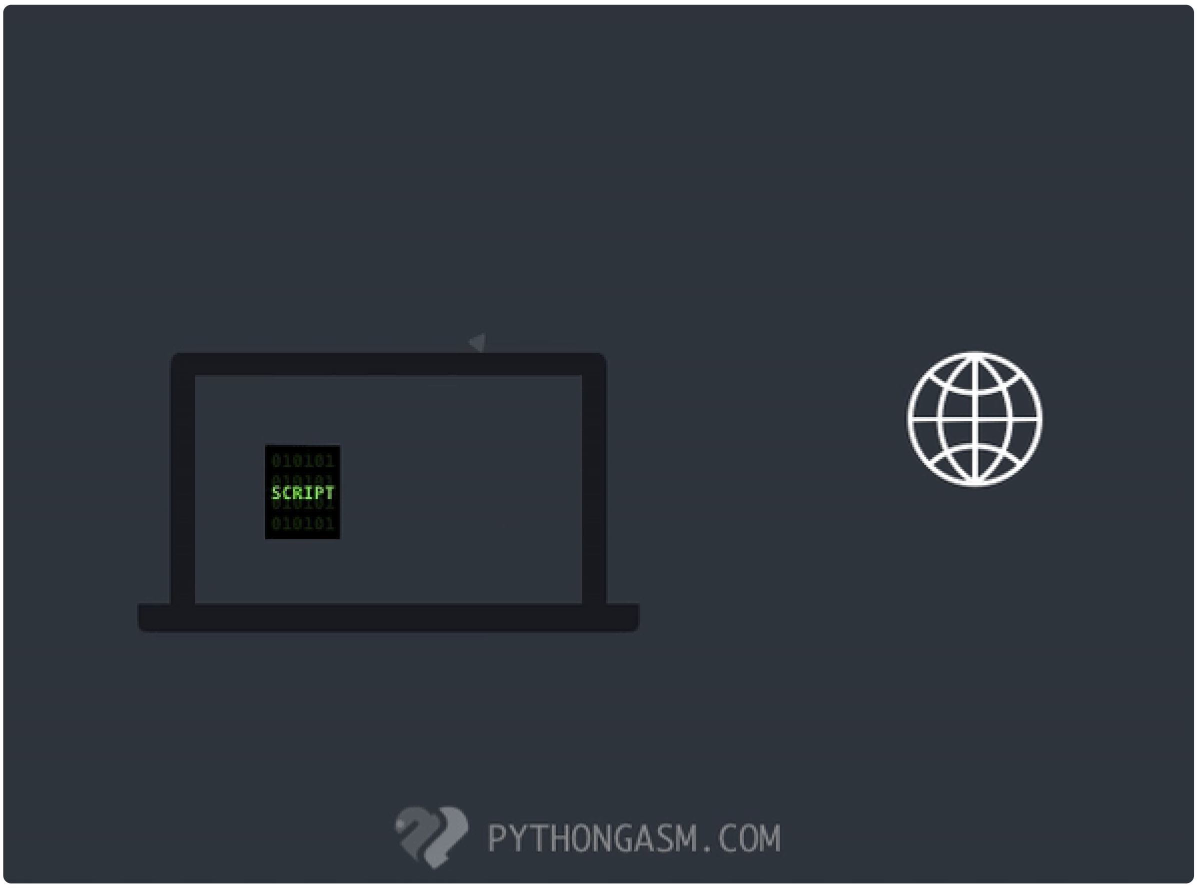
Scraping IMDB using Python & BeautifulSoup

Scraping

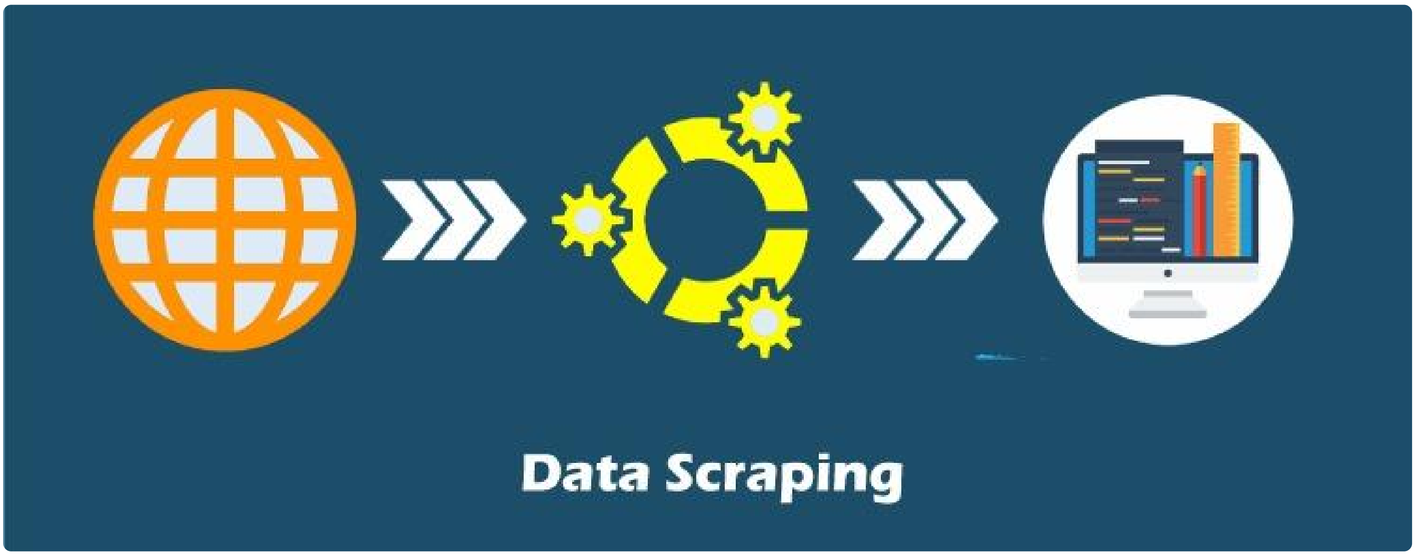


Web scraping is the process of collecting and parsing raw data from the Web. Web scraping is defined as the process of extracting some useful and valuable information from a website. for more information(https://www.datacamp.com/tutorial/web-scraping-using-python)

Github

GitHub is a code hosting platform for collaboration and version control.GitHub lets you (and others) work together on projects.(https://www.w3schools.com/whatis/whatis_github.asp)

Web scraping work



To understand web scraping, it’s important to first understand that web pages are built with text-based mark-up languages – the most common being HTML.

A mark-up language defines the structure of a website’s content. Since there are universal components and tags of mark-up languages, this makes it much easier for web scrapers to pull all the information that it needs. Once the HTML is parsed, the scraper then extracts the necessary data and stores it. Note : Not all websites allow Web Scraping, especially when personal information of the users is involved, so we should always ensure that we do not explore too much, and don't get our hands on information which might belong to someone else. Websites generally have protections at place, and they would block our access to the website if they see us scraping a large amount of data from their website.

To extract data using web scraping with python, you need to follow these basic steps:

```
->Find the URL that you want to scrape
->Inspecting the Page
->Find the data you want to extract
->Write the code
->Run the code and extract the data
->Store the data in the required format
```

About IMDb

IMDb is an online database of information related to films, television programs, home videos, video games, and streaming content online – including cast, production crew and personal biographies, plot summaries, trivia, ratings, and fan and critical reviews.

Almost all of us, at some point in time have looked up for a movie's/show's reviews and ratings on IMDB, to decide if we want to go ahead with watching it or not.

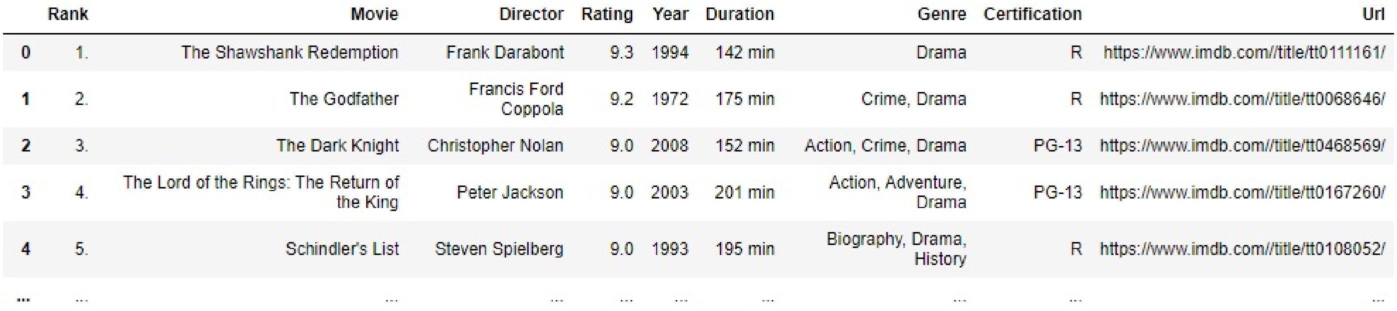
Project Idea

In this project, we will parse through the IMDb's Top rated Movies page to get details about the top rated movies from around the world.

We will retrieve information from the page "Top Rated Movies" using web scraping: a process of extracting information from a website programmatically. Web scraping isn't magic, and yet some readers may grab information on a daily basis. For example, a recent graduate may copy and paste information about companies they applied for into a spreadsheet for job application management.

Project Goal

The project goal is to build a web scraper that withdraws all desirable information and assemble them into a single CSV. The format of the output CSV file is shown below:



	Rank	Movie	Director	Rating	Year	Duration	Genre	Certification	Url
0	1.	The Shawshank Redemption	Frank Darabont	9.3	1994	142 min	Drama	R	https://www.imdb.com//title/tt0111161/
1	2.	The Godfather	Francis Ford Coppola	9.2	1972	175 min	Crime, Drama	R	https://www.imdb.com//title/tt0068646/
2	3.	The Dark Knight	Christopher Nolan	9.0	2008	152 min	Action, Crime, Drama	PG-13	https://www.imdb.com//title/tt0468569/
3	4.	The Lord of the Rings: The Return of the King	Peter Jackson	9.0	2003	201 min	Action, Adventure, Drama	PG-13	https://www.imdb.com//title/tt0167260/
4	5.	Schindler's List	Steven Spielberg	9.0	1993	195 min	Biography, Drama, History	R	https://www.imdb.com//title/tt0108052/
...	...	...	...	...	...	...	...	...	...

Project Steps

```
-> Download the webpage using requests
-> Parse the HTML source code using BeautifulSoup library
-> Extract the desired infromation
-> Building the scraper components
-> Compile the extracted information into Python list and dictionaries
-> Converting the python dictionaries into Pandas DataFrames
-> Write information to the final CSV file
-> Future work and references
```

Packages Used:

```
-> Requests :- For downloading the HTML code from the IMDB URL
-> BeautifulSoup4 :- For parsing and extracting data from the HTML string
-> Pandas :- to gather my data into a dataframe for further processing
```

Lets Begin

Note : We will use the 'Jovian library' and its 'commit()' function throughout the code to save our progress as we move along.

Use the "Run" button to execute the code.

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="first-webscraping-project")
```

```
[jovian] Updating notebook "abhineets17/first-webscraping-project" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/abhineets17/first-webscraping-project
```

```
'https://jovian.ai/abhineets17/first-webscraping-project'
```

Download the webpage using requests

What is requests ??

Requests is a Python HTTP library that allows us to send HTTP requests to servers of websites, instead of using browsers to communicate the web.

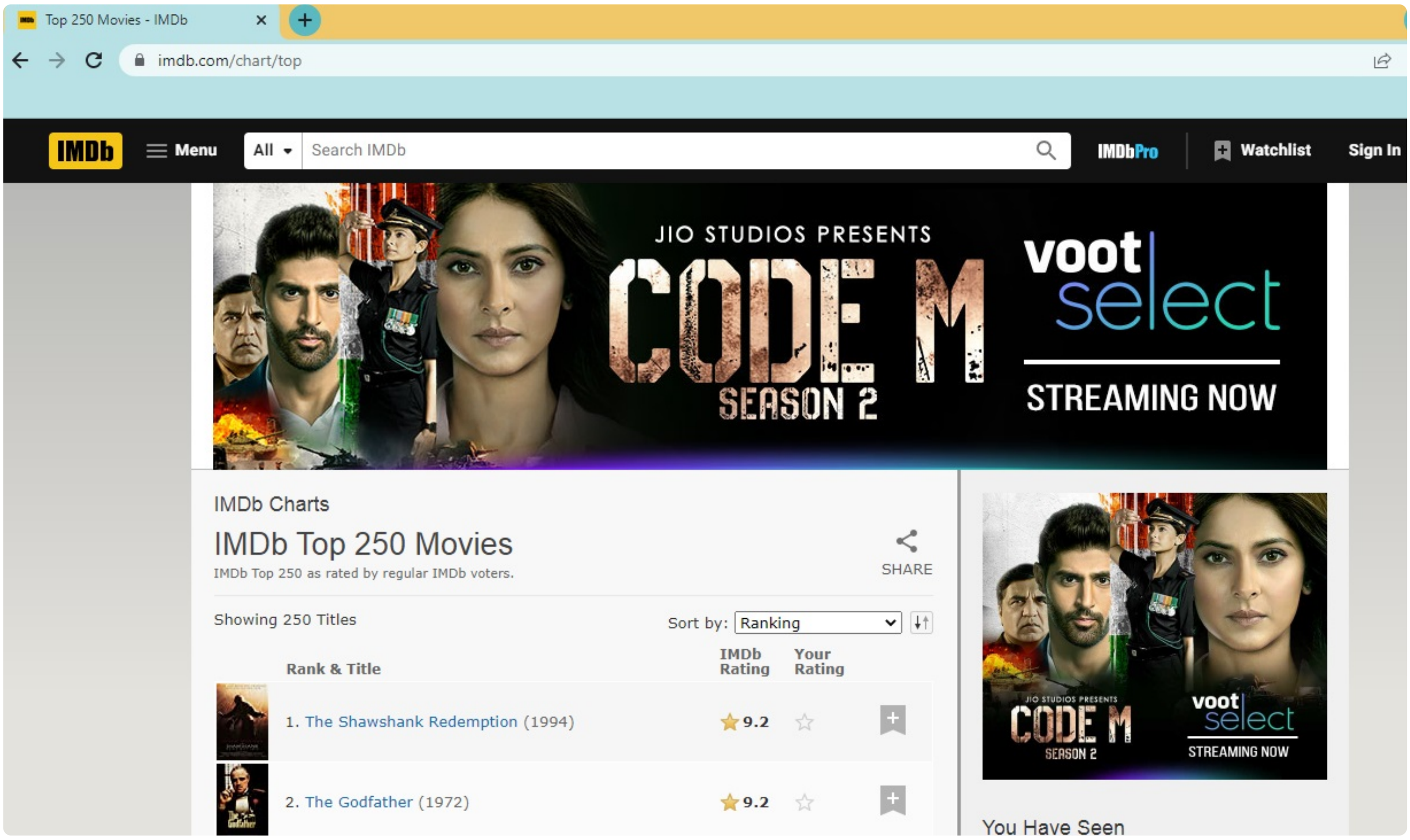
We use pip, a package-management system, to install and manage softwares. Since the platform we selected is Binder, we would have to type a line of code !pip install to install requests. You will see lots codes of !pip when installing other packages.

When we attempt to use some pre-written functions from a certain library, we would use the import statement. e.g. When we would have to type import requests after installation, we are able to use any function from requests library.

```
!pip install requests --quiet --upgrade
import requests
```

requests.get()

In order to download a web page, we use requests.get() to send the HTTP request to the IMDB server and what the function returns is a response object, which is the HTTP response.



IMDb Charts

IMDb Top 250 Movies

IMDb Top 250 as rated by regular IMDb voters.

Showing 250 Titles

Rank & Title	IMDb Rating	Your Rating
1. The Shawshank Redemption (1994)	9.2	
2. The Godfather (1972)	9.2	

```
home_url='https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc'
response = requests.get(home_url)
```

Status code

Now, we have to check if we succesfully send the HTTP request and get a HTTP response back on purpose. This is because we're NOT using browsers, because of which we can't get the feedback directly if we didn't send HTTP requests successfully.

In general, the method to check out if the server sended a HTTP response back is the status code. In requests library, requests.get returns a response object, which containing the page contents and the information about status code indicating if the HTTP request was successful. "Learn more about HTTP status codes here: (https://developer.mozilla.org/en-US/docs/Web/HTTP/Status)"

If the request was successful, response.status_code is set to a value between 200 and 299.

```
# Here we are checking the status code,
# 200 - 299 will mean that the request was sucessfull.
response.status_code
```

```
200
```

response.text

The HTTP response contains HTML that is ready to be displayed in browser. Here we can use "response.text" to retrive the HTML document.

```
page_contents = response.text

# The len() function returns the number of items in an object
# or return length of response object.
len(page_contents)
```

```
433159
```

```
Boom! We have ~4.5Lac characters within the HTML that we just downloaded in a second
```

```
# Its display the first 1000 characters of page contents.
print(page_contents[:1000])
```

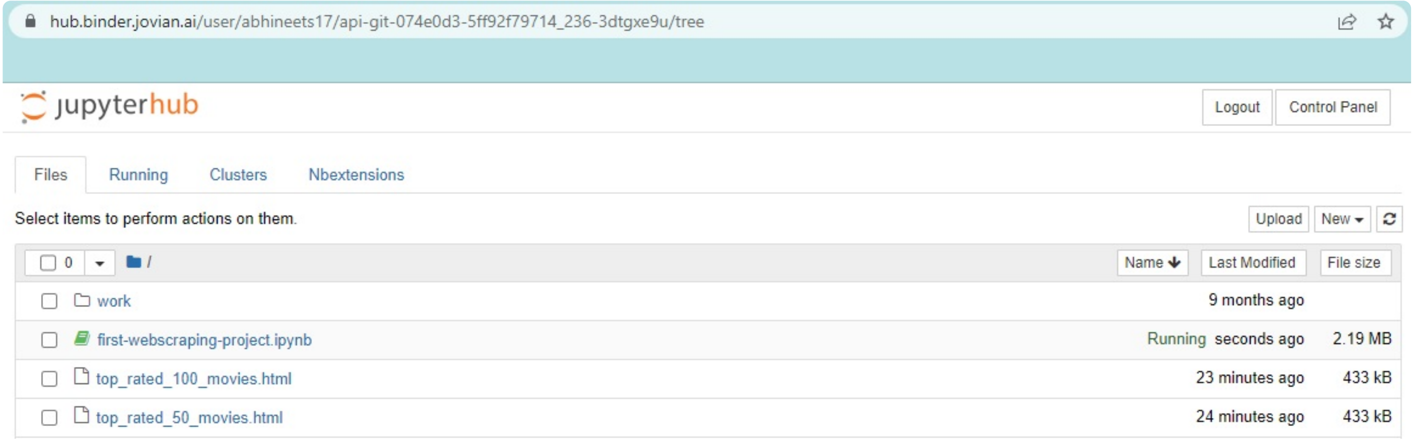
```
<!DOCTYPE html>
<html
    xmlns:og="http://ogp.me/ns#"
    xmlns:fb="http://www.facebook.com/2008/fbml">
    <head>
```

```
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">




        <script type="text/javascript">var IMDbTimer={starttime: new
Date().getTime(),pt:'java'};</script>

<script>
    if (typeof uet == 'function') {
      uet("bb", "LoadTitle", {wb: 1});
    }
</script>
  <script>(function(t){ (t.events = t.events || {})["csm_head_pre_title"] = new
Date().getTime(); })(IMDbTimer);</script>
        <title>IMDb &quot;Top 250&quot;
(Sorted by IMDb Rating Descending) - IMDb</title>
  <script>(function(t){ (t.events = t.events || {})["csm_head_post_title"] = new
Date().getTime(); })(IMDbTimer);</script>
<script>
    if (typeof uet == 'function') {
      uet("be", "LoadTitle", {wb: 1});
    }
</script>
<script>
    if (typeof uex == 'function') {
      uex("ld", "LoadTitle", {wb: 1});
    }
</script>

        <link rel="canonical"
```

```
-> What we see above is the source code of the web page.
-> It is written in a HTML language.
-> It defines and display the content and structure of the web page.
```

```python
# Writing the html page to a file locally.
with open('top_rated_50_movies.html', 'w') as file:
    file.write(page_contents)
```

```
Here, we save the text that we have got into a HTML file with open statement.
Now, a HTML File is created by the name top_rated_movies.html
```



hub.binder.jovian.ai/user/abhineets17/api-git-074e0d3-5ff92f79714_236-3dtgxe9u/tree

Select items to perform actions on them.

□ 0 ▾ /	Name ↓	Last Modified	File size
□ work		9 months ago	
□ first-webscraping-project.ipynb	Running	seconds ago	2.19 MB
□ top_rated_100_movies.html		23 minutes ago	433 kB
□ top_rated_50_movies.html		24 minutes ago	433 kB

```
jovian.commit() # Saving project till now
```

```
[jovian] Updating notebook "abhineets17/first-webscraping-project" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/abhineets17/first-webscraping-project
```

```
'https://jovian.ai/abhineets17/first-webscraping-project'
```

Parse the HTML source code using Beautiful Soup library

Beautiful Soup

Beautiful Soup is a Python package for parsing HTML and XML documents. Beautiful Soup enables us to get data out of sequences of characters. It creates a parse tree for parsed pages that can be used to extract data from HTML. It's a handy tool when it comes to web scraping. You can read more on their documentation site. Learn more about :(https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

To extract information from the HTML source code of a webpage programmatically, we can use the Beautiful Soup library. Let's install the library and import the BeautifulSoup class from the bs4 module.

```
# installing BeautifulSoup
!pip install beautifulsoup4 --quiet --upgrade
from bs4 import BeautifulSoup

# Now 'doc' contains entire html in parsed format.
doc=BeautifulSoup(page_contents, 'html.parser')
```

```
type(doc)
```

```
bs4.BeautifulSoup
```

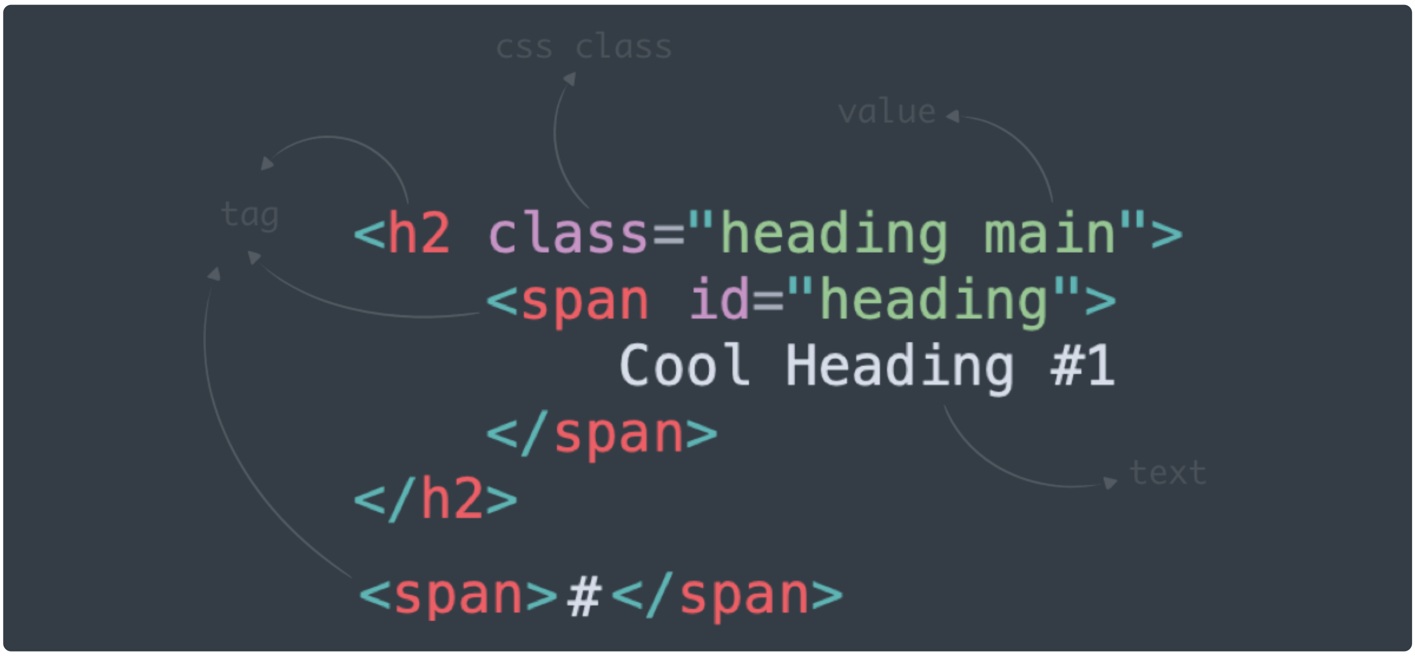
Inspecting the HTML source code of a web page

```
->In BeautifulSoup library, we can specify html.
->Parser to ask Python to read components of the page.
->Instead of reading it as a long string.
```

HTML

Before we dive into how to inspect HTML, we should know the basic knowledge about HTML.

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.



```
<h2 class="heading main">
    <span id="heading">
        Cool Heading #1
    </span>
</h2>

<span>#</span>
```

An HTML tag comprises of three parts:

```
1.Name:(html,head,body,div,etc.)
        Indicates what the tag represents and how a browser should interpret inside i
2.Attributes:(href,target,class,id,etc.)
             Properties of tag used by the browser to customize.
             How a tag is displayed and decide what happens on user interactions.
3.Children: A tag can contain some text or other tags,
           or both between the opening and closing segments.
            e.g.,<div>Some content</div>.
```

Common tags and attributes

Tags in HTML

```
There are around 100 types of HTML tags but on a day to day basis,
around 15 to 20 of them are the most common use, such as:
<div> tag,
<p> tag,
<section> tag,
<img> tag,
<a> tags.
```

Attributes

Each tag supports several attributes. Following are some common attributes used to modify the behavior of tags

```
id
style
class
href (used with <a>)
src (used with <img>)
```

What we can do with **a BeautifulSoup object** is to get **a specifc types of a tag in HTML** by calling the name of a tag, as shown in code cell below.

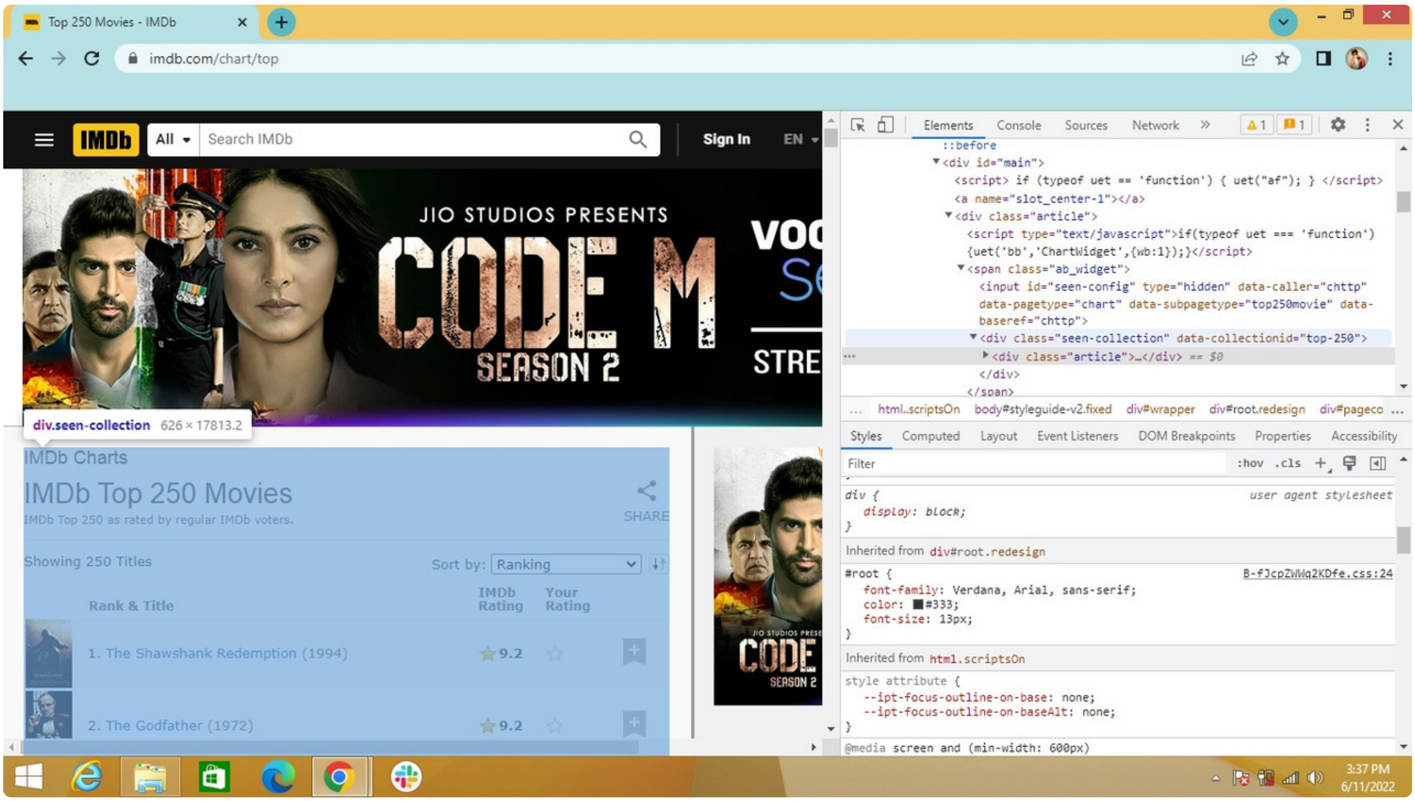
Here, we use the find() function of BeautifulSoup to find the first 'title tag' in the HTML document and display its content.

```
title = doc.find('title')
print(title.text)
```

```
IMDb "Top 250"
(Sorted by IMDb Rating Descending) - IMDb
```

Inspecting HTML in the Browser

To view the source code of any webpage right within your browser, you can right click anywhere on a page and select the "Inspect" option. You access the "Developer Tools" mode, where you can see the source code as a tree. You can expand and collapse various nodes and find the source code for a specific portion of the page.



```
::before
▼<div id="main">
  <script> if (typeof uet == 'function') { uet("af"); } </script>
  <a name="slot_center-1"></a>
 ▼<div class="article">
    <script type="text/javascript">if(typeof uet === 'function')
    {uet('bb','ChartWidget',{wb:1});}</script>
   ▼<span class="ab_widget">
      <input id="seen-config" type="hidden" data-caller="chttp"
      data-pagetype="chart" data-subpagetype="top250movie" data-
      baseref="chttp">
     ▼<div class="seen-collection" data-collectionid="top-250">
...    ▶<div class="article">…</div> == $0
      </div>
    </span>
```

```
... html..scriptsOn  body#styleguide-v2.fixed  div#wrapper  div#root.redesign  div#pageco ...
```

```
div {                                         user agent stylesheet
  display: block;
}
Inherited from div#root.redesign
#root {                                       B-fJcpZWWq2KDfe.css:24
  font-family: Verdana, Arial, sans-serif;
  color: ■#333;
  font-size: 13px;
}
Inherited from html.scriptsOn
style attribute {
  --ipt-focus-outline-on-base: none;
  --ipt-focus-outline-on-baseAlt: none;
}
@media screen and (min-width: 600px)
```

As shown in the photo above, I've cursored over one of the Movie Names to display how the entire content was presented. I found out that each movie-name was present inside the "a tag". Since it does not have any specific class, or other attribute, I will have to check for the desired "a tags" among all the "a tag" present on the page.

Since I've pulled a single page and return to a BeautifulSoup object, we can start to use some function from Beautiful Soup library to withdraw the piece of information we want.

Page1

Movie Name

Now we will use BeautifulSoup to extract the Names and URLs of the top 50 Movies from the HTML Page

```
parent=doc.find_all("h3",class_="lister-item-header")
movi = []
for item in parent:
    movi.append(item.find('a').text)
```

```
movi[:5]        # To fetch first movie name
```

```
['The Shawshank Redemption',
 'The Godfather',
 'The Dark Knight',
 'The Lord of the Rings: The Return of the King',
 "Schindler's List"]
```

Since, the Movie Name is directly wriiten as the text of "a tag", we could directly access the same using the find_all() function of the BeautifulSoup object, i.e. doc here.

But, for the Movie URL we will have to access one of the attributes of the "a tag", i.e. href which contains our desired URL Links.

Movie URL

```
#To fetch movie url.

movie_urls = []
base_urls = 'https://www.imdb.com/'
for item in parent:
    a = item.find('a')
    movie_urls.append(base_urls + item.find('a')['href'])
```

```
movie_urls[:5]        # To fetch first movie url
```

```
['https://www.imdb.com//title/tt0111161/',
 'https://www.imdb.com//title/tt0068646/',
 'https://www.imdb.com//title/tt0468569/',
 'https://www.imdb.com//title/tt0167260/',
 'https://www.imdb.com//title/tt0108052/']
```

Pandas

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

DataFrame

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

```
!pip install pandas --quiet --upgrade        #installing pandas library
import pandas as pd
```

Now we first create a Python Dictionary with the Movie Names and Movie URLs that we have extracted above.

```
movies_dict = {
    'Movie Name' : movi,
    'Movie Url'  : movie_urls
}
movies_dict
```

```
{'Movie Name': ['The Shawshank Redemption',
  'The Godfather',
  'The Dark Knight',
  'The Lord of the Rings: The Return of the King',
  "Schindler's List",
  'The Godfather: Part II',
  '12 Angry Men',
  'Jai Bhim',
  'Pulp Fiction',
  'Inception',
  'The Lord of the Rings: The Two Towers',
  'Fight Club',
  'The Lord of the Rings: The Fellowship of the Ring',
  'Forrest Gump',
  'The Good, the Bad and the Ugly',
  'The Matrix',
  'Goodfellas',
  'Star Wars: Episode V - The Empire Strikes Back',
  "One Flew Over the Cuckoo's Nest",
  'Top Gun: Maverick',
  'Interstellar',
  'City of God',
  'Spirited Away',
  'Saving Private Ryan',
  'The Green Mile',
  'Life Is Beautiful',
  'Se7en',
  'Terminator 2: Judgment Day',
  'The Silence of the Lambs',
  'Star Wars',
  'Hara-Kiri',
  'Seven Samurai',
  "It's a Wonderful Life",
  'Parasite',
  'Whiplash',
  'The Intouchables',
  'The Prestige',
  'The Departed',
  'The Pianist',
  'Gladiator',
  'American History X',
  'The Usual Suspects',
  'Léon: The Professional',
  'The Lion King',
  'Cinema Paradiso',
  'Grave of the Fireflies',
  'Back to the Future',
  'Apocalypse Now',
  'Alien',
  'Once Upon a Time in the West'],
 'Movie Url': ['https://www.imdb.com//title/tt0111161/',
  'https://www.imdb.com//title/tt0068646/',
  'https://www.imdb.com//title/tt0468569/',
  'https://www.imdb.com//title/tt0167260/',
  'https://www.imdb.com//title/tt0108052/',
  'https://www.imdb.com//title/tt0071562/',
  'https://www.imdb.com//title/tt0050083/',
  'https://www.imdb.com//title/tt15097216/',
  'https://www.imdb.com//title/tt0110912/',
  'https://www.imdb.com//title/tt1375666/',
  'https://www.imdb.com//title/tt0167261/',
  'https://www.imdb.com//title/tt0137523/',
  'https://www.imdb.com//title/tt0120737/',
  'https://www.imdb.com//title/tt0109830/',
  'https://www.imdb.com//title/tt0060196/',
  'https://www.imdb.com//title/tt0133093/',
  'https://www.imdb.com//title/tt0099685/',
  'https://www.imdb.com//title/tt0080684/',
  'https://www.imdb.com//title/tt0073486/',
  'https://www.imdb.com//title/tt1745960/',
  'https://www.imdb.com//title/tt0816692/',
  'https://www.imdb.com//title/tt0317248/',
  'https://www.imdb.com//title/tt0245429/',
  'https://www.imdb.com//title/tt0120815/',
  'https://www.imdb.com//title/tt0120689/',
  'https://www.imdb.com//title/tt0118799/',
  'https://www.imdb.com//title/tt0114369/',
  'https://www.imdb.com//title/tt0103064/',
  'https://www.imdb.com//title/tt0102926/',
  'https://www.imdb.com//title/tt0076759/',
  'https://www.imdb.com//title/tt0056058/',
  'https://www.imdb.com//title/tt0047478/',
  'https://www.imdb.com//title/tt0038650/',
  'https://www.imdb.com//title/tt6751668/',
  'https://www.imdb.com//title/tt2582802/',
  'https://www.imdb.com//title/tt1675434/',
  'https://www.imdb.com//title/tt0482571/',
  'https://www.imdb.com//title/tt0407887/',
  'https://www.imdb.com//title/tt0253474/',
  'https://www.imdb.com//title/tt0172495/',
  'https://www.imdb.com//title/tt0120586/',
  'https://www.imdb.com//title/tt0114814/',
  'https://www.imdb.com//title/tt0110413/',
  'https://www.imdb.com//title/tt0110357/',
  'https://www.imdb.com//title/tt0095765/',
  'https://www.imdb.com//title/tt0095327/',
  'https://www.imdb.com//title/tt0088763/',
  'https://www.imdb.com//title/tt0078788/',
  'https://www.imdb.com//title/tt0078748/',
  'https://www.imdb.com//title/tt0064116/']}
```

```
 # We convert the dictionary into pandas DataFrame.
movies_df = pd.DataFrame(movies_dict)    # df -> dataframe
```

```
movies_df
```

	Movie Name	Movie Url
0	The Shawshank Redemption	https://www.imdb.com//title/tt0111161/
1	The Godfather	https://www.imdb.com//title/tt0068646/
2	The Dark Knight	https://www.imdb.com//title/tt0468569/
3	The Lord of the Rings: The Return of the King	https://www.imdb.com//title/tt0167260/
4	Schindler's List	https://www.imdb.com//title/tt0108052/
5	The Godfather: Part II	https://www.imdb.com//title/tt0071562/
6	12 Angry Men	https://www.imdb.com//title/tt0050083/
7	Jai Bhim	https://www.imdb.com//title/tt15097216/
8	Pulp Fiction	https://www.imdb.com//title/tt0110912/
9	Inception	https://www.imdb.com//title/tt1375666/
10	The Lord of the Rings: The Two Towers	https://www.imdb.com//title/tt0167261/
11	Fight Club	https://www.imdb.com//title/tt0137523/
12	The Lord of the Rings: The Fellowship of the Ring	https://www.imdb.com//title/tt0120737/
13	Forrest Gump	https://www.imdb.com//title/tt0109830/
14	The Good, the Bad and the Ugly	https://www.imdb.com//title/tt0060196/
15	The Matrix	https://www.imdb.com//title/tt0133093/
16	Goodfellas	https://www.imdb.com//title/tt0099685/

	Movie Name	Movie Url
17	Star Wars: Episode V - The Empire Strikes Back	https://www.imdb.com//title/tt0080684/
18	One Flew Over the Cuckoo's Nest	https://www.imdb.com//title/tt0073486/
19	Top Gun: Maverick	https://www.imdb.com//title/tt1745960/
20	Interstellar	https://www.imdb.com//title/tt0816692/
21	City of God	https://www.imdb.com//title/tt0317248/
22	Spirited Away	https://www.imdb.com//title/tt0245429/
23	Saving Private Ryan	https://www.imdb.com//title/tt0120815/
24	The Green Mile	https://www.imdb.com//title/tt0120689/
25	Life Is Beautiful	https://www.imdb.com//title/tt0118799/
26	Se7en	https://www.imdb.com//title/tt0114369/
27	Terminator 2: Judgment Day	https://www.imdb.com//title/tt0103064/
28	The Silence of the Lambs	https://www.imdb.com//title/tt0102926/
29	Star Wars	https://www.imdb.com//title/tt0076759/
30	Hara-Kiri	https://www.imdb.com//title/tt0056058/
31	Seven Samurai	https://www.imdb.com//title/tt0047478/
32	It's a Wonderful Life	https://www.imdb.com//title/tt0038650/
33	Parasite	https://www.imdb.com//title/tt6751668/
34	Whiplash	https://www.imdb.com//title/tt2582802/
35	The Intouchables	https://www.imdb.com//title/tt1675434/
36	The Prestige	https://www.imdb.com//title/tt0482571/
37	The Departed	https://www.imdb.com//title/tt0407887/
38	The Pianist	https://www.imdb.com//title/tt0253474/
39	Gladiator	https://www.imdb.com//title/tt0172495/
40	American History X	https://www.imdb.com//title/tt0120586/
41	The Usual Suspects	https://www.imdb.com//title/tt0114814/
42	Léon: The Professional	https://www.imdb.com//title/tt0110413/
43	The Lion King	https://www.imdb.com//title/tt0110357/
44	Cinema Paradiso	https://www.imdb.com//title/tt0095765/
45	Grave of the Fireflies	https://www.imdb.com//title/tt0095327/
46	Back to the Future	https://www.imdb.com//title/tt0088763/
47	Apocalypse Now	https://www.imdb.com//title/tt0078788/
48	Alien	https://www.imdb.com//title/tt0078748/
49	Once Upon a Time in the West	https://www.imdb.com//title/tt0064116/

Now, Let us check the length of the Dataframe that we have created which contains the Movie Names and Movie URLs.

```python
len(movies_df)
```

```
50
```

We can see that the DataFrame consists of 50 items, that is equal to the number of movies that we have on the page Most Rated Movies. Therefore, we can be sure that we have extracted the complete information that we had intended to.

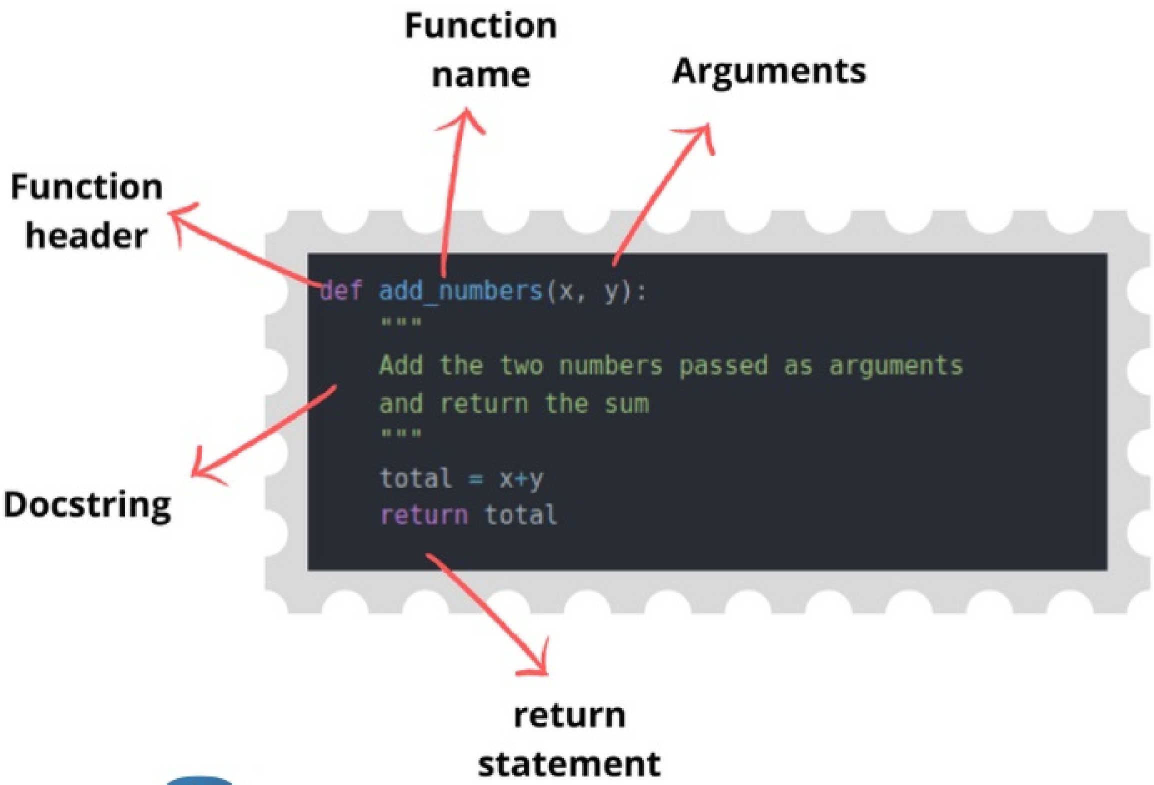
Python def

Python def keyword is used to define a function, it is placed before a function name that is provided by the user to create a user-defined function. In python, a function is a logical unit of code containing a sequence of statements indented under a name given using the “def” keyword. In python def keyword is the most used keyword.

Syntax:

```
def function_name:
    function definition statements...
```

for more information follow link (https://www.geeksforgeeks.org/python-def-keyword/#:~:text=Python def keyword is used,using the “def” keyword.)



```
def add_numbers(x, y):
    """
    Add the two numbers passed as arguments
    and return the sum
    """
    total = x+y
    return total
```

To fetch rating

```
# using def keyword to define a function
#The strip() method Remove spaces at the beginning and at the end of the string

def get_movie_rating(doc):
    rating_selector="inline-block ratings-imdb-rating"
    movie_rating_tags=doc.find_all('div',{'class':rating_selector})
    movie_rating_tagss=[]
    for tag in movie_rating_tags:
        movie_rating_tagss.append(tag.get_text().strip())
    return movie_rating_tagss
```

```
ratings = get_movie_rating(doc)
ratings[:5]
```

```
['9.3', '9.2', '9.0', '9.0', '9.0']
```

To fetch duration

```
def get_movie_duration(doc):

    selection_class="runtime"
    movie_duration_tags=doc.find_all('span',{'class':selection_class})
    movie_duration=[]

    for tag in movie_duration_tags:
#         duration = tag.text[:-4]
        movie_duration.append(tag.get_text().strip())


    return movie_duration
```

```
durations = get_movie_duration(doc)
durations[:5]
# durations[0]
```

```
['142 min', '175 min', '152 min', '201 min', '195 min']
```

To fetch year

```
def get_movie_year(doc):
    year_selector = "lister-item-year text-muted unbold"
    movie_year_tags=doc.find_all('span',{'class':year_selector})
    movie_year_tagss=[]
    for tag in movie_year_tags:
        movie_year_tagss.append(tag.get_text().strip()[1:5])
    return movie_year_tagss
```

```
years = get_movie_year(doc)
years[:5]
# years[0]
```

```
['1994', '1972', '2008', '2003', '1993']
```

To fetch director

```
def get_movie_director(doc):


    movie_director_tags=doc.find_all('p',{'class':""})
    movie_director=[]

    for tag in movie_director_tags:
        directors = tag.find('a').text
        movie_director.append(directors)


    return movie_director
```

```
director = get_movie_director(doc)
director[:5]
```

```
['Frank Darabont',
 'Francis Ford Coppola',
 'Christopher Nolan',
 'Peter Jackson',
 'Steven Spielberg']
```

To fetch genre

```
def get_movie_genre(doc):
    genre_selector="genre"
    movie_genre_tags=doc.find_all('span',{'class':genre_selector})
    movie_genre_tagss=[]
    for tag in movie_genre_tags:
        movie_genre_tagss.append(tag.get_text().strip())
    return movie_genre_tagss
```

```
genre = get_movie_genre(doc)
genre[:5]
# genre[0]
```

```
['Drama',
 'Crime, Drama',
 'Action, Crime, Drama',
 'Action, Adventure, Drama',
 'Biography, Drama, History']
```

To fetch certification

```
def get_movie_certificate(doc):
    certificate_selector="certificate"
    movie_certificate_tags=doc.find_all('span',{'class':certificate_selector})
    movie_certificate_tagss=[]
    for tag in movie_certificate_tags:
        movie_certificate_tagss.append(tag.get_text().strip())
    return movie_certificate_tagss
```

```
certificate = get_movie_certificate(doc)
certificate[:5]
# certificate[0]
```

```
['R', 'R', 'PG-13', 'PG-13', 'R']
```

To fetch rank

```
def get_movie_rank(doc):
    rank_selector="lister-item-index unbold text-primary"
    movie_rank_tags=doc.find_all('span',{'class':rank_selector})
    movie_rank_tagss=[]
    for tag in movie_rank_tags:
        movie_rank_tagss.append(tag.get_text())
    return movie_rank_tagss
```

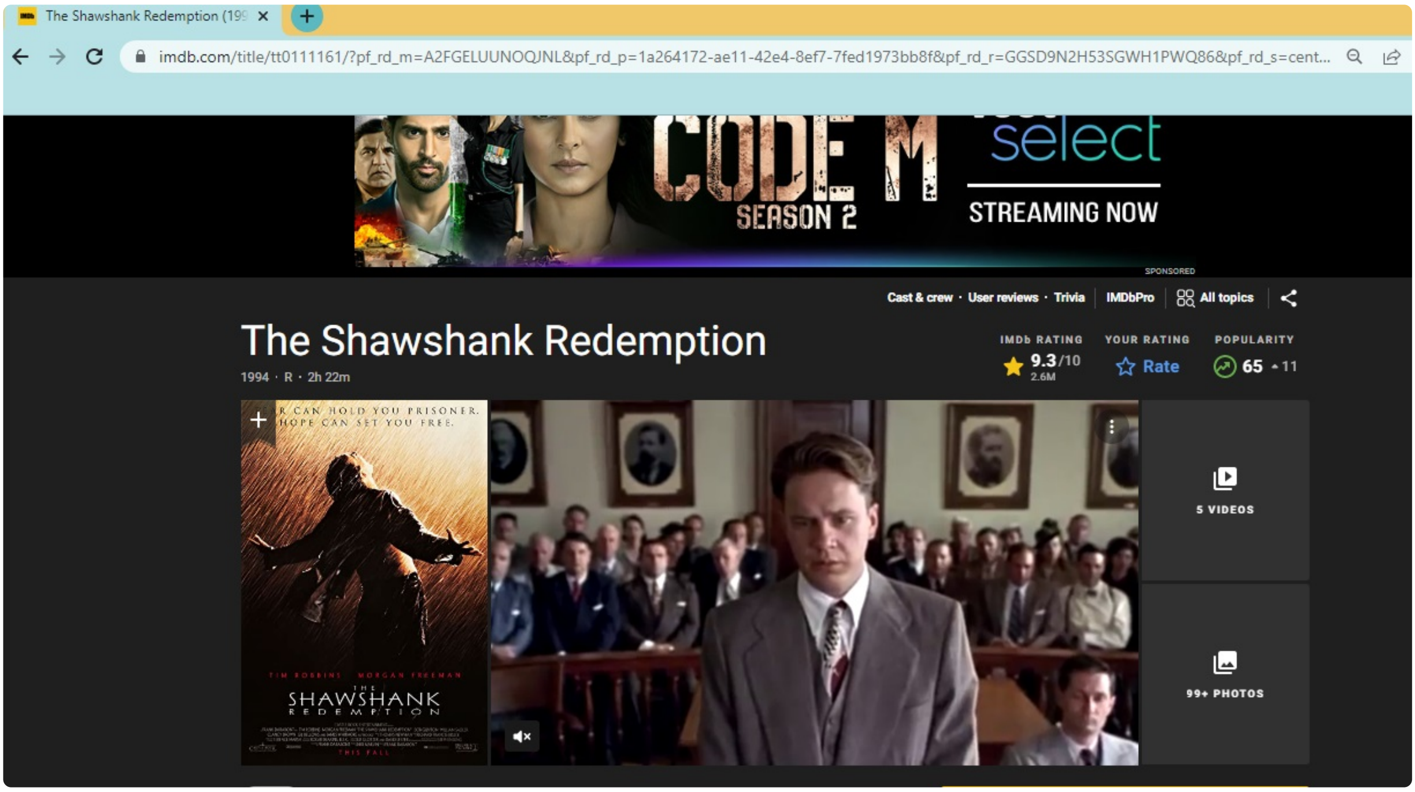
```
rank = get_movie_rank(doc)
rank[:5]
```

```
['1.', '2.', '3.', '4.', '5.']
```

Now, we will go into each individual Movie's page and extract the rest of the required information.

Let's start with extracting all the information for the movie Shawshank Redemption, which is the first movie in our list

1st movie detail(The Shawshank Redemption)

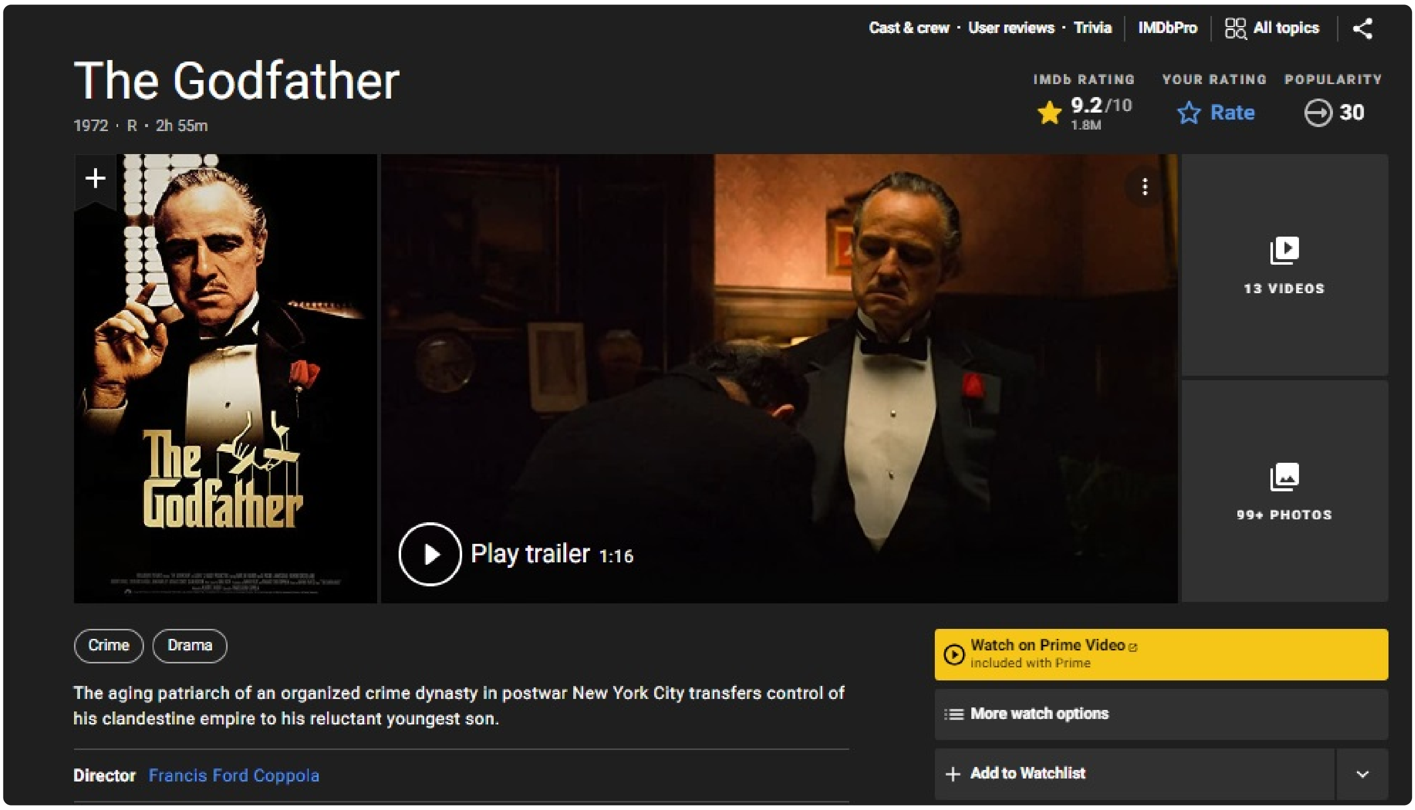

The Shawshank Redemption (199
imdb.com/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=1a264172-ae11-42e4-8ef7-7fed1973bb8f&pf_rd_r=GGSD9N2H53SGWH1PWQ86&pf_rd_s=cent...
CODE M
SEASON 2
select
STREAMING NOW
SPONSORED
Cast & crew · User reviews · Trivia
IMDbPro
All topics
The Shawshank Redemption
1994 · R · 2h 22m
IMDb RATING
9.3/10
2.6M
YOUR RATING
Rate
POPULARITY
65 ▲11
5 VIDEOS
99+ PHOTOS

```
movies_dict={
        'Rank':rank[0],
        'Movie':movi[0],
        'Director':director[0],
        'Rating':ratings[0],
        'Year':years[0],
        'Duration':durations[0],
        'Genre':genre[0],
        'Certification':certificate[0],

        'Url' :movie_urls[0]
    }
```

```
movie_1_df = pd.DataFrame.from_dict(movies_dict, orient='index')
movie_1_df = movie_1_df.transpose()
movie_1_df.head()
```

	Rank	Movie	Director	Rating	Year	Duration	Genre	Certification	Url
0	1.	The Shawshank Redemption	Frank Darabont	9.3	1994	142 min	Drama	R	https://www.imdb.com//title/tt0111161/

Second Movie Detail(The Godfather)



The Godfather

1972 · R · 2h 55m

9.2/10 1.8M

Crime Drama

The aging patriarch of an organized crime dynasty in postwar New York City transfers control of his clandestine empire to his reluctant youngest son.

Director Francis Ford Coppola

```
movies_dict={
        'Rank':rank[1],
        'Movie':movi[1],
        'Director':director[1],
        'Rating':ratings[1],
        'Year':years[1],
        'Duration':durations[1],
        'Genre':genre[1],
        'Certification':certificate[1],

        'Url' :movie_urls[1]
    }
```

Pandas head() method is used to return top n rows of a data frame or series.
The transpose() function is used to transpose index and columns.

```
movie_2_df = pd.DataFrame.from_dict(movies_dict, orient='index')
movie_2_df = movie_2_df.transpose()
movie_2_df.head()
```

```
   Rank  Movie      Director   Rating  Year  Duration  Genre    Certification                                     Url
0    2.  The        Francis    9.2     1972  175 min   Crime,   R             https://www.imdb.com//title/tt0068646/
         Godfather  Ford                               Drama
                    Coppola
```

Page 2

```
home_url1 = 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&st
response_1 = requests.get(home_url1)
```

```
response_1.status_code
```

```
200
```

```
page_1_contents = response_1.text
len(page_1_contents)
```

```
432458
```

```
print(page_1_contents[:1000])
```

```


<!DOCTYPE html>
<html
    xmlns:og="http://ogp.me/ns#"
    xmlns:fb="http://www.facebook.com/2008/fbml">
    <head>
         

        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">

    

    


        <script type="text/javascript">var IMDbTimer={starttime: new
Date().getTime(),pt:'java'};</script>

<script>
    if (typeof uet == 'function') {
      uet("bb", "LoadTitle", {wb: 1});
    }
</script>
  <script>(function(t){ (t.events = t.events || {})["csm_head_pre_title"] = new
Date().getTime(); })(IMDbTimer);</script>
        <title>IMDb &quot;Top 250&quot;
(Sorted by IMDb Rating Descending) - IMDb</title>
  <script>(function(t){ (t.events = t.events || {})["csm_head_post_title"] = new
Date().getTime(); })(IMDbTimer);</script>
<script>
```

```
    if (typeof uet == 'function') {
      uet("be", "LoadTitle", {wb: 1});
    }
</script>
<script>
    if (typeof uex == 'function') {
      uex("ld", "LoadTitle", {wb: 1});
    }
</script>


        <link rel="canonical"
```

```python
with open('top_rated_100_movies.html', 'w') as file:
    file.write(page_1_contents)
```

```python
doc1=BeautifulSoup(page_1_contents, 'html.parser')
```

```python
type(doc1)
```

```
bs4.BeautifulSoup
```

```python
title = doc1.find('title')
print(title.text)
```

```
IMDb "Top 250"
(Sorted by IMDb Rating Descending) - IMDb
```

Movie Names

```python
parent1=doc1.find_all("h3",class_="lister-item-header")
movi_2 = []
for item1 in parent1:
    movi_2.append(item1.find('a').text)
```

```python
movi_2[:5]      # print first 5 movie name
```

```
['Psycho', 'Rear Window', 'Casablanca', 'Modern Times', 'City Lights']
```

Movie Urls

```python
movie_urls_2 = []
base_urls = 'https://www.imdb.com/'
for item1 in parent1:
    a = item1.find('a')
    movie_urls_2.append(base_urls + item1.find('a')['href'])
```

```
movie_urls_2[:5]          # print first 5 url
```

```
['https://www.imdb.com//title/tt0054215/',
 'https://www.imdb.com//title/tt0047396/',
 'https://www.imdb.com//title/tt0034583/',
 'https://www.imdb.com//title/tt0027977/',
 'https://www.imdb.com//title/tt0021749/']
```

```
movies_dict = {
    'Movie Name'  : movi_2,
    'Movie Url'   : movie_urls_2
}
movies_dict
```

```
{'Movie Name': ['Psycho',
  'Rear Window',
  'Casablanca',
  'Modern Times',
  'City Lights',
  'Hamilton',
  'Capernaum',
  'Joker',
  'Your Name.',
  'Spider-Man: Into the Spider-Verse',
  'Avengers: Endgame',
  'Avengers: Infinity War',
  'Coco',
  'Django Unchained',
  'The Dark Knight Rises',
  '3 Idiots',
  'WALL·E',
  'The Lives of Others',
  'Oldboy',
  'Memento',
  'American Beauty',
  'Princess Mononoke',
  'Braveheart',
  'Come and See',
  'Aliens',
  'Amadeus',
  'Indiana Jones and the Raiders of the Lost Ark',
  'The Boat',
  'The Shining',
  'High and Low',
  'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb',
  'Witness for the Prosecution',
  'Paths of Glory',
  'Sunset Blvd.',
  'The Great Dictator',
  'Everything Everywhere All at Once',
  'Dangal',
  'The Hunt',
  'A Separation',
  'Incendies',
  'Spider-Man: No Way Home',
  'Up',
  'Like Stars on Earth',
  'Toy Story 3',
  'Inglourious Basterds',
  'Eternal Sunshine of the Spotless Mind',
  'Amélie',
  'Snatch',
  'Requiem for a Dream',
  'Good Will Hunting'],
 'Movie Url': ['https://www.imdb.com//title/tt0054215/',
  'https://www.imdb.com//title/tt0047396/',
  'https://www.imdb.com//title/tt0034583/',
  'https://www.imdb.com//title/tt0027977/',
  'https://www.imdb.com//title/tt0021749/',
  'https://www.imdb.com//title/tt8503618/',
  'https://www.imdb.com//title/tt8267604/',
  'https://www.imdb.com//title/tt7286456/',
  'https://www.imdb.com//title/tt5311514/',
  'https://www.imdb.com//title/tt4633694/',
  'https://www.imdb.com//title/tt4154796/',
  'https://www.imdb.com//title/tt4154756/',
  'https://www.imdb.com//title/tt2380307/',
  'https://www.imdb.com//title/tt1853728/',
  'https://www.imdb.com//title/tt1345836/',
  'https://www.imdb.com//title/tt1187043/',
  'https://www.imdb.com//title/tt0910970/',
  'https://www.imdb.com//title/tt0405094/',
  'https://www.imdb.com//title/tt0364569/',
  'https://www.imdb.com//title/tt0209144/',
  'https://www.imdb.com//title/tt0169547/',
  'https://www.imdb.com//title/tt0119698/',
  'https://www.imdb.com//title/tt0112573/',
  'https://www.imdb.com//title/tt0091251/',
  'https://www.imdb.com//title/tt0090605/',
  'https://www.imdb.com//title/tt0086879/',
  'https://www.imdb.com//title/tt0082971/',
  'https://www.imdb.com//title/tt0082096/',
  'https://www.imdb.com//title/tt0081505/',
  'https://www.imdb.com//title/tt0057565/',
  'https://www.imdb.com//title/tt0057012/',
  'https://www.imdb.com//title/tt0051201/',
  'https://www.imdb.com//title/tt0050825/',
  'https://www.imdb.com//title/tt0043014/',
  'https://www.imdb.com//title/tt0032553/',
  'https://www.imdb.com//title/tt6710474/',
  'https://www.imdb.com//title/tt5074352/',
  'https://www.imdb.com//title/tt2106476/',
  'https://www.imdb.com//title/tt1832382/',
  'https://www.imdb.com//title/tt1255953/',
  'https://www.imdb.com//title/tt10872600/',
  'https://www.imdb.com//title/tt1049413/',
  'https://www.imdb.com//title/tt0986264/',
  'https://www.imdb.com//title/tt0435761/',
  'https://www.imdb.com//title/tt0361748/',
  'https://www.imdb.com//title/tt0338013/',
  'https://www.imdb.com//title/tt0211915/',
  'https://www.imdb.com//title/tt0208092/',
  'https://www.imdb.com//title/tt0180093/',
  'https://www.imdb.com//title/tt0119217/']}
```

```
movie_df = pd.DataFrame(movies_dict)
```

```
movie_df
```

	Movie Name	Movie Url
0	Psycho	https://www.imdb.com//title/tt0054215/
1	Rear Window	https://www.imdb.com//title/tt0047396/
2	Casablanca	https://www.imdb.com//title/tt0034583/
3	Modern Times	https://www.imdb.com//title/tt0027977/
4	City Lights	https://www.imdb.com//title/tt0021749/
5	Hamilton	https://www.imdb.com//title/tt8503618/
6	Capernaum	https://www.imdb.com//title/tt8267604/
7	Joker	https://www.imdb.com//title/tt7286456/
8	Your Name.	https://www.imdb.com//title/tt5311514/
9	Spider-Man: Into the Spider-Verse	https://www.imdb.com//title/tt4633694/
10	Avengers: Endgame	https://www.imdb.com//title/tt4154796/
11	Avengers: Infinity War	https://www.imdb.com//title/tt4154756/
12	Coco	https://www.imdb.com//title/tt2380307/
13	Django Unchained	https://www.imdb.com//title/tt1853728/
14	The Dark Knight Rises	https://www.imdb.com//title/tt1345836/
15	3 Idiots	https://www.imdb.com//title/tt1187043/
16	WALL·E	https://www.imdb.com//title/tt0910970/
17	The Lives of Others	https://www.imdb.com//title/tt0405094/
18	Oldboy	https://www.imdb.com//title/tt0364569/
19	Memento	https://www.imdb.com//title/tt0209144/
20	American Beauty	https://www.imdb.com//title/tt0169547/

	Movie Name	Movie Url
21	Princess Mononoke	https://www.imdb.com//title/tt0119698/
22	Braveheart	https://www.imdb.com//title/tt0112573/
23	Come and See	https://www.imdb.com//title/tt0091251/
24	Aliens	https://www.imdb.com//title/tt0090605/
25	Amadeus	https://www.imdb.com//title/tt0086879/
26	Indiana Jones and the Raiders of the Lost Ark	https://www.imdb.com//title/tt0082971/
27	The Boat	https://www.imdb.com//title/tt0082096/
28	The Shining	https://www.imdb.com//title/tt0081505/
29	High and Low	https://www.imdb.com//title/tt0057565/
30	Dr. Strangelove or: How I Learned to Stop Worr...	https://www.imdb.com//title/tt0057012/
31	Witness for the Prosecution	https://www.imdb.com//title/tt0051201/
32	Paths of Glory	https://www.imdb.com//title/tt0050825/
33	Sunset Blvd.	https://www.imdb.com//title/tt0043014/
34	The Great Dictator	https://www.imdb.com//title/tt0032553/
35	Everything Everywhere All at Once	https://www.imdb.com//title/tt6710474/
36	Dangal	https://www.imdb.com//title/tt5074352/
37	The Hunt	https://www.imdb.com//title/tt2106476/
38	A Separation	https://www.imdb.com//title/tt1832382/
39	Incendies	https://www.imdb.com//title/tt1255953/
40	Spider-Man: No Way Home	https://www.imdb.com//title/tt10872600/
41	Up	https://www.imdb.com//title/tt1049413/
42	Like Stars on Earth	https://www.imdb.com//title/tt0986264/
43	Toy Story 3	https://www.imdb.com//title/tt0435761/
44	Inglourious Basterds	https://www.imdb.com//title/tt0361748/
45	Eternal Sunshine of the Spotless Mind	https://www.imdb.com//title/tt0338013/
46	Amélie	https://www.imdb.com//title/tt0211915/
47	Snatch	https://www.imdb.com//title/tt0208092/
48	Requiem for a Dream	https://www.imdb.com//title/tt0180093/
49	Good Will Hunting	https://www.imdb.com//title/tt0119217/

```
len(movie_df)
```

```
50
```

Rating

```
def get_movie_rating1(doc1):
    rating1_selector="inline-block ratings-imdb-rating"
    movie_rating1_tags=doc1.find_all('div',{'class':rating1_selector})
    movie_rating1_tagss=[]
    for tag1 in movie_rating1_tags:
        movie_rating1_tagss.append(tag1.get_text().strip())
    return movie_rating1_tagss
```

```
ratings2 = get_movie_rating1(doc1)
ratings2[:5]
```

```
['8.5', '8.5', '8.5', '8.5', '8.5']
```

Duration

```
def get_movie_duration1(doc1):

    selection_class="runtime"
    movie_duration1_tags=doc1.find_all('span',{'class':selection_class})
    movie_duration1=[]

    for tag in movie_duration1_tags:
#         duration1 = tag.text[:-4]
        movie_duration1.append(tag.get_text().strip())


    return movie_duration1
```

```
durations2 = get_movie_duration1(doc1)
durations2[:5]
```

```
['109 min', '112 min', '102 min', '87 min', '87 min']
```

Year

```
def get_movie_year1(doc1):
    year_selector = "lister-item-year text-muted unbold"
    movie_year1_tags=doc1.find_all('span',{'class':year_selector})
    movie_year1_tagss=[]
    for tag in movie_year1_tags:
        movie_year1_tagss.append(tag.get_text().strip()[1:5])
    return movie_year1_tagss
```

```
years2 = get_movie_year(doc1)
years2[:5]
```

```
['1960', '1954', '1942', '1936', '1931']
```

Director

```
def get_movie_director1(doc1):
    movie_director1_tags=doc1.find_all('p',{'class':""})
    movie_director1=[]

    for tag in movie_director1_tags:
        directors1 = tag.find('a').text
        movie_director1.append(directors1)


    return movie_director1
```

```
director2 = get_movie_director1(doc1)
director2[:5]
```

```
['Alfred Hitchcock',
 'Alfred Hitchcock',
 'Michael Curtiz',
 'Charles Chaplin',
 'Charles Chaplin']
```

Genre

```
def get_movie_genre1(doc1):
    genre_selector="genre"
    movie_genre1_tags=doc1.find_all('span',{'class':genre_selector})
    movie_genre1_tagss=[]
    for tag in movie_genre1_tags:
        movie_genre1_tagss.append(tag.get_text().strip())
    return movie_genre1_tagss
```

```
genre2 = get_movie_genre1(doc1)
genre2[:5]
```

```
['Horror, Mystery, Thriller',
 'Mystery, Thriller',
 'Drama, Romance, War',
 'Comedy, Drama, Romance',
 'Comedy, Drama, Romance']
```

Certification

```
def get_movie_certificate1(doc1):
    certificate_selector="certificate"
    movie_certificate1_tags=doc1.find_all('span',{'class':certificate_selector})
    movie_certificate1_tagss=[]
    for tag in movie_certificate1_tags:
        movie_certificate1_tagss.append(tag.get_text().strip())
    return movie_certificate1_tagss
```

```
certificate2 = get_movie_certificate1(doc1)
certificate2[:5]
```

```
['R', 'PG', 'PG', 'G', 'G']
```

Rank

```
def get_movie_rank1(doc1):
    rank_selector="lister-item-index unbold text-primary"
    movie_rank1_tags=doc1.find_all('span',{'class':rank_selector})
    movie_rank1_tagss=[]
    for tag in movie_rank1_tags:
        movie_rank1_tagss.append(tag.get_text())
    return movie_rank1_tagss
```

```
rank2 = get_movie_rank1(doc1)
rank2[:5]
```

```
['51.', '52.', '53.', '54.', '55.']
```

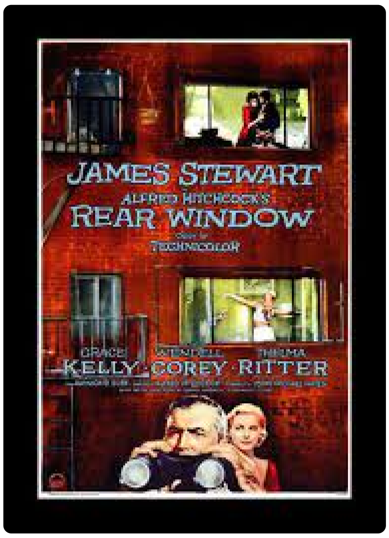
Get 51st Movie detail(Psycho)

```
movies_dict={
        'Rank'         :rank2[0],
        'Movie'        :movi_2[0],
        'Director'     :director2[0],
        'Rating'       :ratings2[0],
        'Year'         :years2[0],
        'Duration'     :durations2[0],
        'Genre'        :genre2[0],
        'Certification':certificate2[0],

        'Url'          :movie_urls_2[0]
    }
```

```
movie_51_df = pd.DataFrame.from_dict(movies_dict, orient='index')
movie_51_df = movie_51_df.transpose()
movie_51_df.head()
```

	Rank	Movie	Director	Rating	Year	Duration	Genre	Certification	Url
0	51.	Psycho	Alfred Hitchcock	8.5	1960	109 min	Horror, Mystery, Thriller	R	https://www.imdb.com//title/tt0054215/

Get 52nd Movie detail(Rear Window)



```
movies_dict={
        'Rank'         :rank2[1],
        'Movie'        :movi_2[1],
        'Director'     :director2[1],
        'Rating'       :ratings2[1],
        'Year'         :years2[1],
        'Duration'     :durations2[1],
        'Genre'        :genre2[1],
        'Certification':certificate2[1],

        'Url'          :movie_urls_2[1]
    }
```

```
movie_52_df = pd.DataFrame.from_dict(movies_dict, orient='index')
movie_52_df = movie_52_df.transpose()
movie_52_df.head()
```

	Rank	Movie	Director	Rating	Year	Duration	Genre	Certification	Url
0	52.	Rear Window	Alfred Hitchcock	8.5	1954	112 min	Mystery, Thriller	PG	https://www.imdb.com//title/tt0047396/

The dictionary will store the names of the movies, and for each movie the dictionary value will be a list of show times.

Pandas DataFrame.transpose() function transpose index and columns of the dataframe. It reflect the DataFrame over its main diagonal by writing rows as columns and vice-versa.

```
movies_dict1={                                                     #page1
        'Rank':rank,
        'Movie':movi,
        'Director':director,
        'Rating':ratings,
        'Year':years,
        'Duration':durations,
        'Genre':genre,
        'Certification':certificate,

        'Url' :movie_urls
    }
```

```
df1 = pd.DataFrame.from_dict(movies_dict1, orient='index')
df1 = df1.transpose()
 #df1.head()
```

```
movies_dict2={                                                           #page2
        'Rank':rank2,
        'Movie':movi_2,
        'Director':director2,
        'Rating':ratings2,
        'Year':years2,
        'Duration':durations2,
        'Genre':genre2,
        'Certification':certificate2,

        'Url' :movie_urls_2
    }
```

```
df2 = pd.DataFrame.from_dict(movies_dict2, orient='index')
df2 = df2.transpose()
#df2.head()
```

```
b_url='https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&start='
def url_building(url):
    urls = []
    for i in range(5):
        urls.append(b_url + str(i*50 + 1))
    return urls

URLs = url_building(b_url)
URLs
```

```
['https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&start=1',
 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&start=51',
 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&start=101',
 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&start=151',
 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&start=201']
```

```
# df = pd.DataFrame.from_dict(dict3, orient='index')
# df = df.transpose()
# df.head()
```

Concatenating Objects

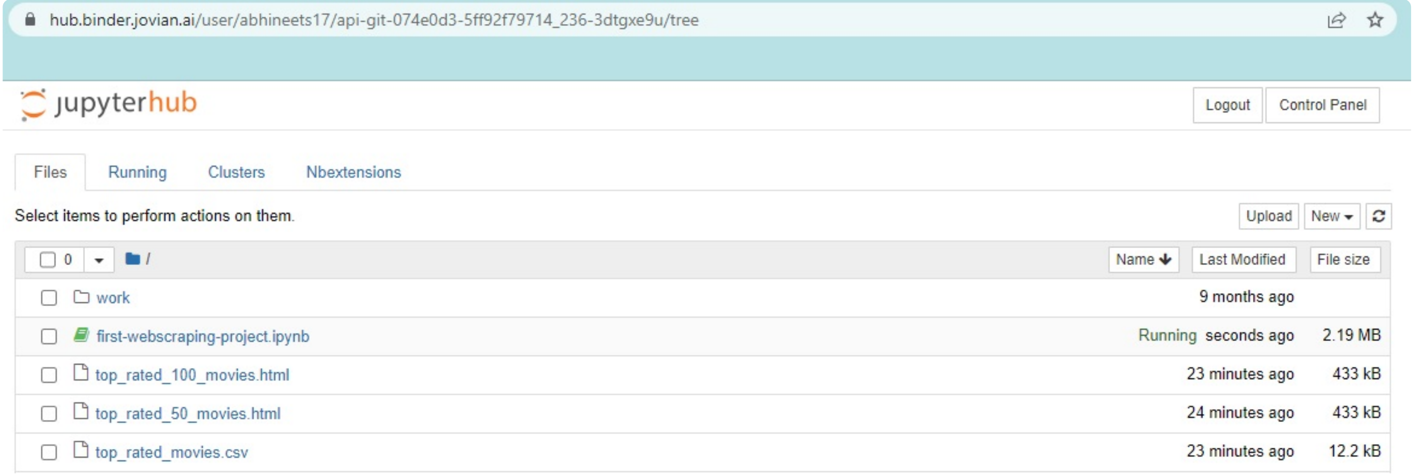
The concat function does all of the heavy lifting of performing concatenation operations along an axis.
image

```
final_df = pd.concat([df1,df2], axis = 0)
final_df
```

Converting the final Dataframe to a CSV File

```
->To write to a CSV file in Python, we can use the csv. writer() function.
->The csv. writer() function returns a writer object,
   that converts the user's data into a delimited string.
->CSV (Comma Separated Values)
```

```
final_df.to_csv('top_rated_movies.csv', index=None)
```



hub.binder.jovian.ai/user/abhineets17/api-git-074e0d3-5ff92f79714_236-3dtgxe9u/tree

Select items to perform actions on them.

Name	Last Modified	File size
work	9 months ago	
first-webscraping-project.ipynb	Running seconds ago	2.19 MB
top_rated_100_movies.html	23 minutes ago	433 kB
top_rated_50_movies.html	24 minutes ago	433 kB
top_rated_movies.csv	23 minutes ago	12.2 kB

Summary

Finally, we have managed to parse 'IMDB - Top Rated Movies' to get our hands on very interesting and insightful data when it comes to the world of entertainment. We have saved all the information we could extract from that website for our needs in a CSV file using which we can further get answers to a lot of questions we may want to ask, e.g - Which director has directed the most movies which are top ranked in the world?

Let us look at the steps that we took from start to finish :

```
    ->We downloaded the webpage using requests
    ->We parsed the HTML source code using BeautifulSoup library and extracted the desire
      infromation, i.e.
    ->The names of 'Top Rated Movies'
    ->URLs of each of those movies
    ->We created a DataFrame using Pandas for Python Lists that we derived from
       the previous step
    ->We extracted detailed information for each movie among the list of Top Rated Movies
       such as :
        Movie Name
        Url
        Year of Release
        Genre
        Rating
        Rank
        Director Name
    Certificate
    Duration
  ->We then created a Python Dictionary to save all these details
  ->We converted the python dictionary into Pandas DataFrames
  ->With DataFrame in hand,we then converted it into a single CSV file,
   which was the goal of our project.
```

Future Work

We can now work forward to explore this data more and more to fetch meaningful information out of it.

With all the insights , and further analysis into the data, we can have answers to a lot of questions like -

Which actor has worked in most top rated movies across the world? The Top Rated Movies as per the Genre of our interest? Which Director has directed the most top rated movies? Which year gave us the most Top Rated Movies till date? And the list goes on..

In the future, I would like to work to make this DataSet even richer with more data from other lists created by IMDB like - Most Trending Movies, Top Rated Indian Movies, Lowest Rated Movies etc. I would then like to work on analysing the entire data, to know a lot more about movies than I currently know.

References

[1] Python offical documentation. https://docs.python.org/3/

[2] Requests library. https://pypi.org/project/requests/

[3] Beautiful Soup documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[4] Aakash N S, Introduction to Web Scraping. https://jovian.ai/aakashns/python-web-scraping-and-rest-api

[5] Pandas library documentation. https://pandas.pydata.org/docs/

[6] IMDB Website. https://www.imdb.com/chart/top

[7] Web Scraping Article. https://www.toptal.com/python/web-scraping-with-python

[8] Web Scraping Image. https://morioh.com/p/431153538ecb

[8] Working with Jupyter Notebook https://towardsdatascience.com/write-markdown-latex-in-the-jupyter-notebook-10985edb91fd

```
jovian.commit()
```

```
jovian.commit(files=['top_rated_movies.csv'])
```