

# Web Scrapping On IMDB Top Movies Using python



## What is Web Scrapping?

In the most simple terms, Web Scrapping is the process through which we extract data from a website, and save it in a form which is easy to read, to understand and to work on.

When we say 'Easy to work on', we mean to say that the data thus extracted can be used to get a lot of useful insights and answer a lot of questions, finding answers to which would not be such an easy task, if we did not have that data stored with us in a simple and sorted manner, i.e. generally in a CSV File, an Excel File or a Database.

## project steps

Here is an outline of the steps we'll follow :

- 1.Download the webpage using requests
- 2.Parse the HTML source code using BeautifulSoup library and extract the desired information
- 3.Building the scraper components
- 4.Compile the extracted information into Python list and dictionaries
- 5.Converting the python dictionaries into Pandas DataFrames
- 6.Write information to the final CSV file
- 7.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

## 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 prewritten 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 jovian --upgrade --quiet
import jovian
# Execute this to save new versions of the notebook
jovian.commit(project="final-web-scraping-project")
```

```
[jovian] Updating notebook "sunithapachala93/final-web-scraping-project" on
https://jovian.com
```

```
[jovian] Committed successfully! https://jovian.com/sunithapachala93/final-web-scraping-project
```

```
'https://jovian.com/sunithapachala93/final-web-scraping-project'
```

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

```
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 Sort by: Ranking

Rank & Title	IMDb Rating	Your Rating
1. <a href="#">The Shawshank Redemption</a> (1994)	★ 9.2	☆
2. <a href="#">The Godfather</a> (1972)	★ 9.2	☆
3. <a href="#">The Dark Knight</a> (2008)	★ 9.0	☆
4. <a href="#">The Godfather Part II</a> (1974)	★ 9.0	☆
5. <a href="#">12 Angry Men</a> (1957)	★ 9.0	☆
6. <a href="#">Schindler's List</a> (1993)	★ 9.0	☆

amazon prime

**BLACK ADAM**

English | Hindi | Tamil | Telugu

WATCH NOW

You Have Seen

0/250 (0%)

☐ Hide titles I've seen

Top Rated Movies by Genre

- Action
- Adventure
- Animation
- Biography
- Comedy
- Crime
- Drama

```
topic_url='https://www.imdb.com/search/title/?groups=top_250&sort=user_rating'
```

```
response = requests.get(topic_url)
```

## Status code

Now, we have to check if we successfully 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 sent a HTTP response back is the **status code**. In `requests` library, `requests.get` returns a response object, which contains 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**.

```
response.status_code # Here we are checking the status code : 200-299 will mean that the
```

```
200
```

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

```
len(response.text) # The 'len' function tells us the length of response object
```

```
420824
```

```
page_content = response.text
```

```
page_content[:1000] # This display the first 1000 charcters of page_contens
```

```
'\n\n<!DOCTYPE html>\n<html\n    xmlns:og="http://ogp.me/ns#\nxmlns:fb="http://www.facebook.com/2008/fbml">\n    <head>\n        \n\n        <meta\ncharset="utf-8">\n\n\n\n        <script type="text/javascript">var IMDbTimer=\n{starttime: new Date().getTime(),pt:\'java\'};</script>\n\n<script>\n    if (typeof uet\n== \'function\') {\n        uet("bb", "LoadTitle", {wb: 1});\n    }\n</script>\n<script>(function(t){ (t.events = t.events || {})[ "csm_head_pre_title" ] = new\nDate().getTime(); })(IMDbTimer);</script>\n        <title>IMDb &quot;Top\n250&quot;\n(Sorted by IMDb Rating Descending) - IMDb</title>\n    <script>(function(t){\n(t.events = t.events || {})[ "csm_head_post_title" ] = new Date().getTime(); })\n(IMDbTimer);</script>\n<script>\n    if (typeof uet == \'function\') {\n        uet("be",\n"LoadTitle", {wb: 1});\n    }\n</script>\n<script>\n    if (typeof uex == \'function\')\n{\n    uex("ld", "LoadTitle", {wb: 1});\n    }\n</script>\n\n        <link\nrel="canonical" href="https://www.imdb.com/search/title/?groups=top_250" />\n    '
```

```
with open('webpage.html', 'w') as f:\n    f.write(page_content)
```

```
!pip install beautifulsoup4 --upgrade --quiet
```

## Parse the HTML source code using BeautifulSoup library

### What is BeautifulSoup?

Beautiful Soup is a **Python package** for **parsing HTML and XML documents**. BeautifulSoup 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. <https://www.crummy.com/software/BeautifulSoup/bs4/doc/#getting-help>

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

```
from bs4 import BeautifulSoup
```

```
doc = BeautifulSoup(page_content, 'html.parser')
```

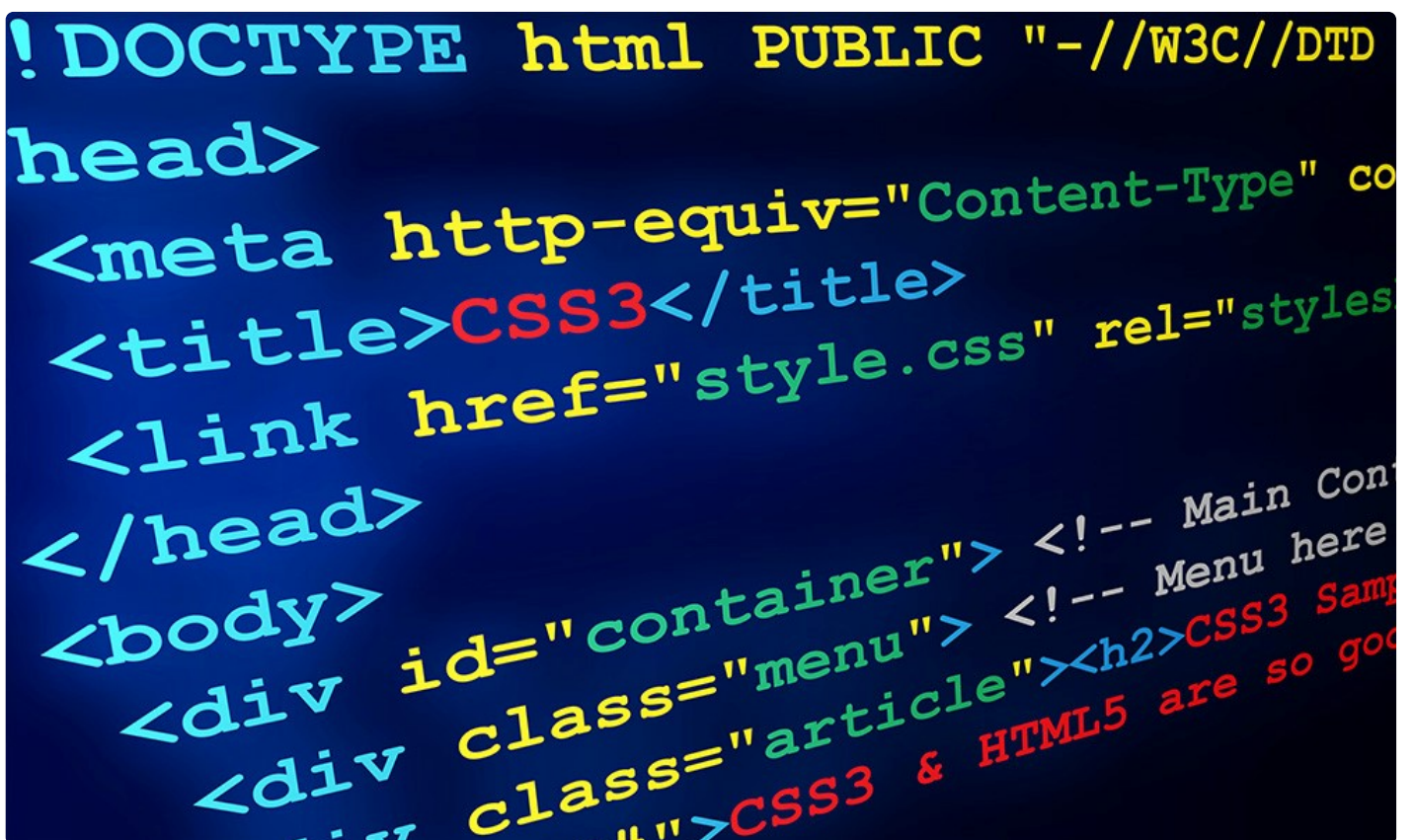
### Inspecting the HTML source code of a web page

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.

## What is 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 and scripting languages such as JavaScript.



```
!DOCTYPE html PUBLIC "-//W3C//DTD
head>
<meta http-equiv="Content-Type" co
<title>CSS3</title>
<link href="style.css" rel="styles
</head>
<body>
<div id="container"> <!-- Main Con
<div class="menu"> <!-- Menu here
<div class="article"><h2>CSS3 Samp
CSS3 & HTML5 are so good
```

### 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 the information inside it.
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.

Of many tags, I wanted to highlight **<a> tag**, which can contain attributes such as `href` (hyperlink reference), because `<a>` tag allows users to click and they would be directed to another site. That's why the name of `<a>` tag is **anchor**.

## 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>`)

```
def get_topics_page(url):  
  
    response=requests.get(url)  
    if response.status_code != 200:  
        raise Exception(f'Failed to load page {topic_url}')  
    doc = BeautifulSoup(response.text, 'html.parser')  
    return doc
```

```
topic_url = 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating'  
doc = get_topics_page(topic_url)
```

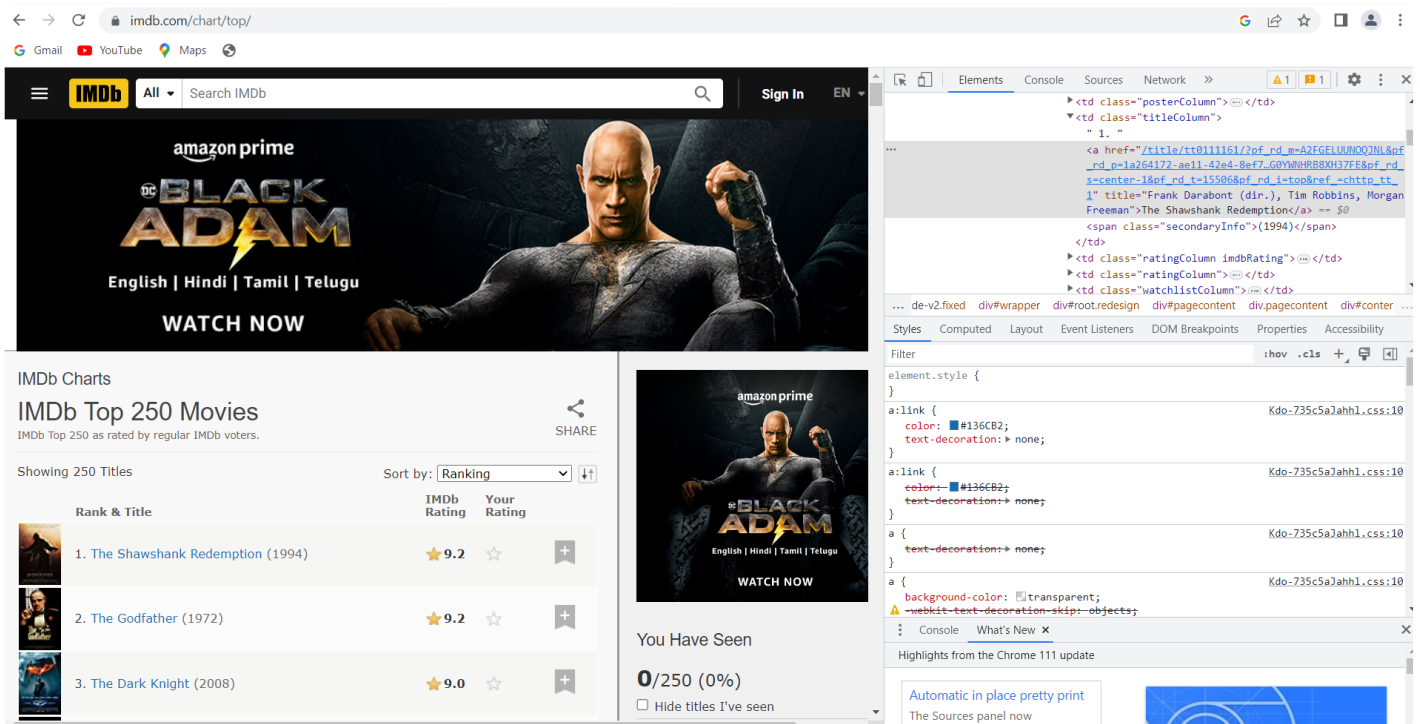
```
doc.find('title')
```

```
<title>IMDb "Top 250"  
(Sorted by IMDb Rating Descending) - IMDb</title>
```

## 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





As shown in the photo above, I've censored over one of the Movie Names to display how the entire content was presented.

I found out that each `movienam` 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>` tags 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 BeautifulSoup library to withdraw the piece of information we want.

```
parent = doc.find_all('h3', {'class' : 'lister-item-header'}) # Here we create a list
movie_names = []
for item in parent:
    movie_names.append(item.find('a').text)
movie_names
```

```
['The Shawshank Redemption',
 'The Godfather',
 'The Dark Knight',
 'Schindler's List',
 '12 Angry Men',
 'The Lord of the Rings: The Return of the King',
 'The Godfather Part II',
 'Spider-Man: Across the Spider-Verse',
 'Pulp Fiction',
 'Inception',
 'The Lord of the Rings: The Fellowship of the Ring',
 'Fight Club',
 'Forrest Gump',
 'Il buono, il brutto, il cattivo',
 'The Lord of the Rings: The Two Towers',
 'Jai Bhim',
 'Interstellar',
 'GoodFellas',
```

'The Matrix',  
"One Flew Over the Cuckoo's Nest",  
'Star Wars: Episode V - The Empire Strikes Back',  
'Saving Private Ryan',  
'The Green Mile',  
'Terminator 2: Judgment Day',  
'Se7en',  
'The Silence of the Lambs',  
'Star Wars',  
'Sen to Chihiro no kamikakushi',  
'Cidade de Deus',  
'La vita è bella',  
'Shichinin no samurai',  
"It's a Wonderful Life",  
'Seppuku',  
'Gladiator',  
'Back to the Future',  
'The Departed',  
'Gisaengchung',  
'Léon',  
'Whiplash',  
'Alien',  
'The Prestige',  
'The Lion King',  
'The Usual Suspects',  
'American History X',  
'The Pianist',  
'Psycho',  
'Casablanca',  
'The Intouchables',  
'Once Upon a Time in the West',  
'Hotaru no haka']


## Get the Movie Name



**"After getting iPhone,  
I got interested in photography."**

Monish B. switched to iPhone


[Learn more](#)



**IMDb "Top 250" (Sorted by IMDb Rating Descending)**

1-50 of 250 titles. [Next »](#) [View Mode: Compact](#) [Detailed](#)

Sort by: [Popularity](#) | [A-Z](#) | [User Rating ▼](#) | [Number of Votes](#) | [US Box Office](#) | [Runtime](#) | [Year](#) | [Release Date](#) | [Date of Your Rating](#) | [Your Rating](#)



**1. The Shawshank Redemption** (1994)


R | 142 min | Drama

★ 9.3 ☆ Rate this 82 Metascore

Over the course of several years, two convicts form a friendship, seeking consolation and, eventually, redemption through basic compassion.

Director: [Frank Darabont](#) | Stars: [Tim Robbins](#), [Morgan Freeman](#), [Bob Gunton](#), [William Sadler](#)

Votes: 2,754,419 | Gross: \$28.34M | [Top 250: #1](#)



**"After I switched to iPhone  
my wife made me her  
personal cameraman"**

Ankit P.  
switched to iPhone

[Learn more](#)

```
<div class="list-item mode-advanced">
  <div class="list-item-image float-left">
  <div class="list-item-content">
    <h3 class="list-item-header">
      <span class="list-item-index unbold text-primary">1.</span>
      <a href="/title/tt0111161/?ref=adv_li_tt">The Shawshank Redemption
    </a> == $0
      <span class="list-item-year text-muted unbold">(1994)</span>
    </h3>
    <p class="text-muted">
    <div class="ratings-bar">
    <p class="text-muted">
    <p class="text-muted">
    <p class="sort-num_votes-visible">
    </div>
  </div>
</div>
```

◀ details.sub-list div.list-item mode-advanced div.list-item-content h3.list-item-header a

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter

element.style {

a:visited {color: #70579D;text-decoration: none;YndQv5DES10eXre.css:10}

a:visited {color: #70579D;text-decoration: none;YndQv5DES10eXre.css:10}

a {text-decoration: none;YndQv5DES10eXre.css:10}



```
doc.find_all('h3',{ 'class': 'lister-item-header'})[0]('a')[0].text
```

'The Shawshank Redemption'

```
def get_movie_name(doc):
    movie_name_tags=doc.find_all('h3',{ 'class': 'lister-item-header'})
    movie_name=[]
    for tag in movie_name_tags:
        movie_name.append(tag('a')[0].text.strip())
    return movie_name
```

```
movie_name = get_movie_name(doc)
```

```
movie_name[:3]
```

['The Shawshank Redemption', 'The Godfather', 'The Dark Knight']

```
len(movie_name)
```

50

## Get the Movie Released Year

The screenshot shows the IMDb website's 'Top 250' list. The first movie is 'The Shawshank Redemption' (1994). A browser developer tool is open on the right, showing the HTML structure of the movie entry. The HTML includes a header with the movie title and year, a ratings bar, and a text-muted year. The developer tool highlights the `span` element with the class `lister-item-year text-muted unbold` containing the year '1994'.

```
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= []
    for tag in movie_year_tags:
        movie_year.append(tag.text.strip())
    return movie_year
```

```
movie_year = get_movie_year(doc)
```

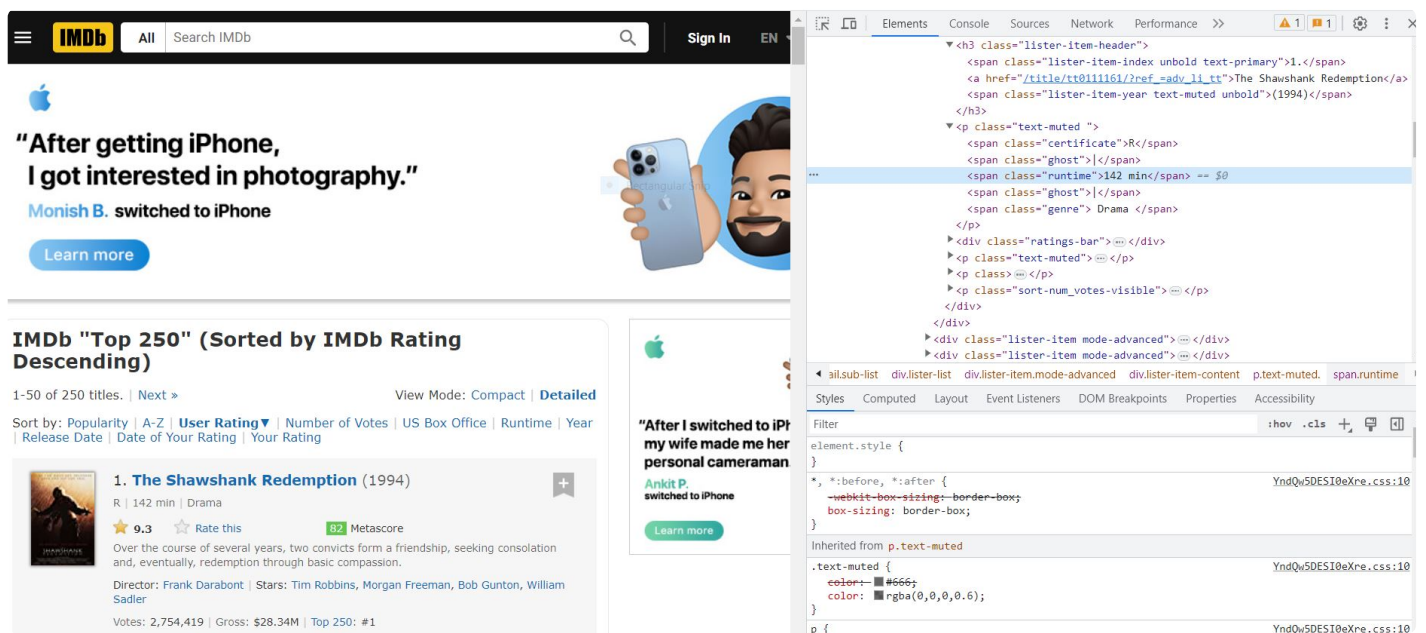
```
movie_year[:3]
```

```
['(1994)', '(1972)', '(2008)']
```

```
len(movie_name)
```

50

## Get the Movie Runtime



The screenshot shows the IMDb website interface. The main content area displays the movie 'The Shawshank Redemption' (1994) with a rating of 9.3 and a runtime of 142 minutes. The browser's developer console is open on the right, showing the HTML structure of the page. The console highlights the 'runtime' attribute of the movie's runtime tag, which is '142 min'.

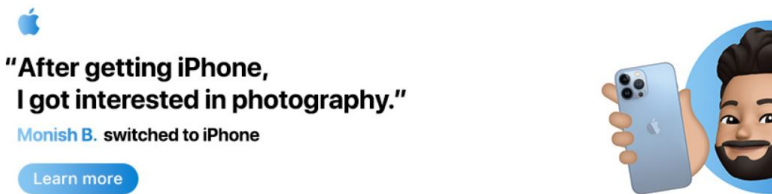
```
def get_movie_runtime(doc):  
    movie_runtime_tags=doc.find_all('span',{'class': 'runtime'})  
  
    movie_runtime=[]  
  
    for tag in movie_runtime_tags:  
        movie_runtime.append(tag.text)  
  
    return movie_runtime
```

```
movie_runtime= get_movie_runtime(doc)
```

```
movie_runtime[:3]
```

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

## Get the movie Rating



## IMDb "Top 250" (Sorted by IMDb Rating Descending)

1-50 of 250 titles. | Next »

View Mode: Compact | Detailed

Sort by: Popularity | A-Z | **User Rating** | Number of Votes | US Box Office | Runtime | Year Release Date | Date of Your Rating | Your Rating



### 1. The Shawshank Redemption (1994)

R | 142 min | Drama

★ 9.3

☆ Rate this

82 Metascore

Over the course of several years, two convicts form a friendship, seeking consolation and, eventually, redemption through basic compassion.

Director: Frank Darabont | Stars: Tim Robbins, Morgan Freeman, Bob Gunton, William Sadler

Votes: 2,754,419 | Gross: \$28.34M | Top 250: #1

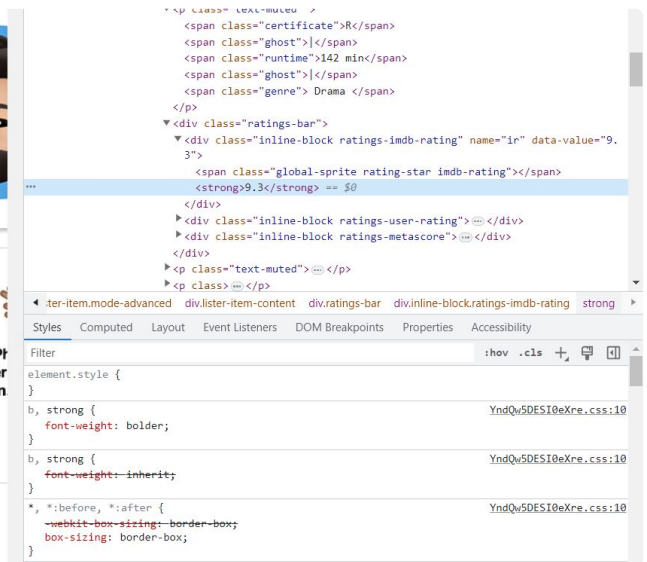


"After I switched to iPhone, my wife made me her personal cameraman"

Ankit P.

switched to iPhone

Learn more



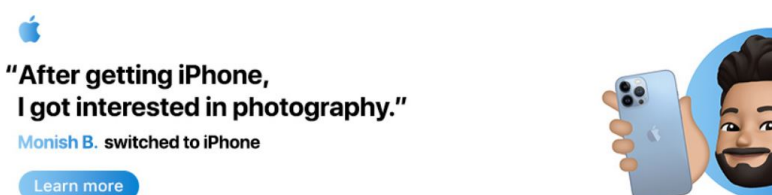
```
def get_movie_rating(doc):
    movie_rating_tags=doc.find_all('div',{'class':'inline-block ratings-imdb-rating'})
    movie_rating = []
    for tag in movie_rating_tags:
        movie_rating.append(tag.text.strip())
    return movie_rating
```

```
movie_rating = get_movie_rating(doc)
```

```
movie_rating[:3]
```

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

## Get the Movie Metascore



## IMDb "Top 250" (Sorted by IMDb Rating Descending)

1-50 of 250 titles. | Next »

View Mode: Compact | Detailed

Sort by: Popularity | A-Z | **User Rating** | Number of Votes | US Box Office | Runtime | Year Release Date | Date of Your Rating | Your Rating



### 1. The Shawshank Redemption (1994)

R | 142 min | Drama

★ 9.3

☆ Rate this

82 Metascore

Over the course of several years, two convicts form a friendship, seeking consolation and, eventually, redemption through basic compassion.

Director: Frank Darabont | Stars: Tim Robbins, Morgan Freeman, Bob Gunton, William Sadler

Votes: 2,754,419 | Gross: \$28.34M | Top 250: #1

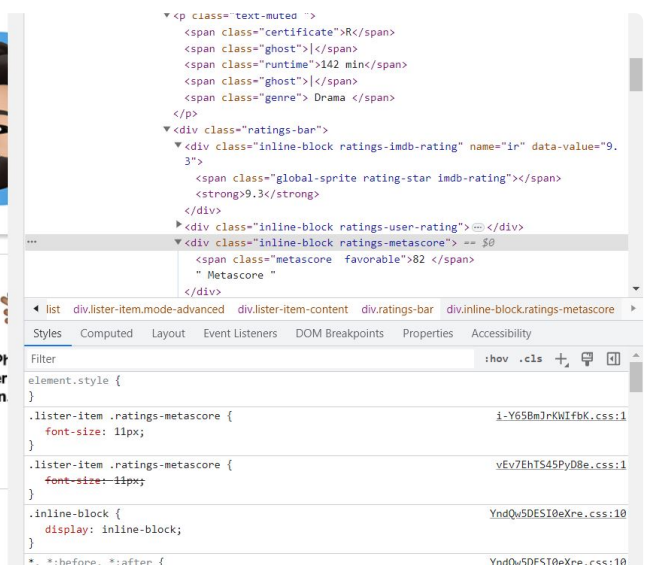


"After I switched to iPhone, my wife made me her personal cameraman"

Ankit P.

switched to iPhone

Learn more



```
def get_movie_metascore(doc):
    movie_metascore_tags=doc.find_all('span',{'class':'metascore'})
    movie_metascore = []
    for tag in movie_metascore_tags:
```

```
        movie_metascore.append(tag.text.strip())
    return movie_metascore
```

```
movie_metascore=get_movie_metascore(doc)
```

```
movie_metascore[:3]
```

```
['82', '100', '84']
```

## Get the Movie Director

```
def get_movie_director(doc):
    director_tags=doc.find_all('div',{'class':'lister-item-content'})
    directors = []
    direc = []
    for tag in director_tags:
        direc.append(tag.find_all('p')[2].get_text().strip('\n'))
    for di in direc:
        directors.append(di.split('\n')[1].strip('Director:').strip())
    return directors
```

```
movie_director=get_movie_director(doc)
```

```
movie_director[:3]
```

```
['Frank Darabon', 'Francis Ford Coppola', 'Christopher Nolan']
```

```
len(movie_director)
```

```
50
```

## Get the Movie Actors

```
def get_movie_actors(doc):
    movie_actors_tags=doc.find_all('div',{'class':'lister-item-content'})
    movie_actors = []
    actors = []
    for tag in movie_actors_tags:
        actors.append(tag.find_all('p')[2].get_text().strip('\n'))
    for actor in actors:
        all_actors = actor.split('\n')
        stars = [star.strip(', ') for star in all_actors[3:] if star.strip(', ').strip()
        movie_actors.append(stars[1:])
    return movie_actors
```

```
movie_actors=get_movie_actors(doc)
```

```
movie_actors[:3]
```

```
[['Morgan Freeman', 'Bob Gunton', 'William Sadler'],  
 ['Al Pacino', 'James Caan', 'Diane Keaton'],  
 ['Heath Ledger', 'Aaron Eckhart', 'Michael Caine']]
```

```
len(movie_actors)
```

50

## Let's put everything in one single function.

```
!pip install pandas --upgrade --quiet
```

```
import pandas as pd
```

```
import requests  
from bs4 import BeautifulSoup
```

```
def all_pages(num=6):  
    # Let's we create a dictionary to store data of all movies  
    movie_dict={  
  
        'movie':[],  
        'year':[],  
        'runtime':[],  
        'rating':[],  
        'metascore':[],  
        'director':[],  
        'actors':[]  
    }  
    # We have to scrap more than one page so we want urls of all pages with the help of  
    for i in [0,51,101]:  
        url='https://www.imdb.com/search/title/?groups=top_250&start={}&ref=adv_nxt'.f  
        doc=get_topics_page(url)  
  
        movie_dict['movie'] += get_movie_name(doc)  
        movie_dict['year'] += get_movie_year(doc)  
        movie_dict['runtime'] += get_movie_runtime(doc)  
        movie_dict['rating'] += get_movie_rating(doc)  
        movie_dict['metascore'] += get_movie_metascore(doc)  
        movie_dict['director'] += get_movie_director(doc)  
        movie_dict['actors'] += get_movie_actors(doc)  
  
    return pd.DataFrame.from_dict(movie_dict, orient='index')
```

```
movies = all_pages()
```

```
movies.to_csv('movies.csv', index=None)
```

```
dataframe = pd.read_csv('movies.csv')
```

dataframe

	0	1	2	3	4	5	6	7	8	
0	Spider-Man: Across the Spider-Verse	Spider-Man: Into the Spider-Verse	Guardians of the Galaxy Vol. 3	Raiders of the Lost Ark	Interstellar	The Godfather	Jurassic Park	Top Gun: Maverick	The Shawshank Redemption	Spider-Man: No Way Home
1	(2023)	(2018)	(2023)	(1981)	(2014)	(1972)	(1993)	(2022)	(1994)	(2002)
2	140 min	117 min	150 min	115 min	169 min	175 min	127 min	130 min	142 min	148 min
3	8.9	8.4	8.2	8.4	8.7	9.2	8.2	8.3	9.3	8.5
4	86	87	64	85	74	100	68	78	82	90
5	Joaquim Dos Santos,	Bob Persichetti,	James Gunn	Steven Spielberg	Christopher Nolan	Francis Ford Coppola	Steven Spielberg	Joseph Kosinski	Frank Darabont	Jon Watts
6	['Shameik Moore', 'Hailee Steinfeld', 'Brian T...	['Shameik Moore', 'Jake Johnson', 'Hailee Stei...	['Chukwudi Iwuji', 'Bradley Cooper', 'Pom Klem...	['Karen Allen', 'Paul Freeman', 'John Rhys-Dav...	['Anne Hathaway', 'Jessica Chastain', 'Mackenz...	['Al Pacino', 'James Caan', 'Diane Keaton']	['Laura Dern', 'Jeff Goldblum', 'Richard Atten...	['Jennifer Connelly', 'Miles Teller', 'Val Kil...	['Morgan Freeman', 'Bob Gunton', 'William Sadl...	['Zendaya', 'Ben Stiller', 'Cumberbatch', 'Jacob El...

7 rows x 150 columns

```
dataframe_transposed=dataframe.T
dataframe_transposed
```

		0	1	2	3	4	5	6
0	Spider-Man: Across the Spider-Verse	(2023)	140 min	8.9	86	Joaquim Dos Santos,	['Shameik Moore', 'Hailee Steinfeld', 'Brian T...	
1	Spider-Man: Into the Spider-Verse	(2018)	117 min	8.4	87	Bob Persichetti,	['Shameik Moore', 'Jake Johnson', 'Hailee Stei...	
2	Guardians of the Galaxy Vol. 3	(2023)	150 min	8.2	64	James Gunn	['Chukwudi Iwuji', 'Bradley Cooper', 'Pom Klem...	
3	Raiders of the Lost Ark	(1981)	115 min	8.4	85	Steven Spielberg	['Karen Allen', 'Paul Freeman', 'John Rhys-Dav...	
4	Interstellar	(2014)	169 min	8.7	74	Christopher Nolan	['Anne Hathaway', 'Jessica Chastain', 'Mackenz...	
...	...	...	...	...	...	...	...	
145	Three Billboards Outside Ebbing, Missouri	(2017)	115 min	8.1	82	Martin McDonagh	['Woody Harrelson', 'Sam Rockwell', 'Caleb Lan...	
146	Salinui chueok	(2003)	131 min	8.1	80	Bong Joon H	['Kim Sang-kyung', 'Roe-ha Kim', 'Jae-ho Song']	



	0	1	2	3	4	5	6
147	Incendies	(2010)	131 min	8.3	88	Denis Villeneuve	['Mélima Désormeaux-Poulin', 'Maxim Gaudette'...
148	Amadeus	(1984)	160 min	8.4	98	Milos Forman	['Tom Hulce', 'Elizabeth Berridge', 'Roy Dotri...
149	Pan's Labyrinth	(2006)	118 min	8.2	NaN	Guillermo del T	['Ariadna Gil', 'Sergi López', 'Maribel Verdú']

150 rows × 7 columns

## Summary

Here is what we covered in this notebook:

- Download the webpage using requests
- Parse the HTML source code using beautiful soup
- Extract movie name, movie released year, runtime, actors and director from webpage
- Compile extracted information using Python lists
- Save the extracted information to a CSV file.

So, we are successful in parsing Web page for imdb movies. We got lots of interesting and useful information from it.

All our work is saved in CSV file. Through this CSV file, we can get all our information in the best and optimal way.

## Future Work

- Now we can easily get details about most trending movies by top rating. Also we can find movies by our favorite genre.
- this data can be use for further analysis into the data and it will be auto update by running all cells again
- Python libraries like matplotlib and seaborn can be used for visualisation of the data we have scraped.
- Analyse the extracted data using pandas library for insights on directors/actors featuring in the list.
- In this project I started to look into some of the websites where I could get the data for movies in IMDm website.

## References

- <https://beautiful-soup-4.readthedocs.io/en/latest/>
- <https://www.youtube.com/watch?v=RKsLLG-bzEY&t=8940s>

```
jovian.commit()
```

[jovian] Updating notebook "sunithapachala93/final-web-scraping-project" on <https://jovian.com>

[jovian] Committed successfully! <https://jovian.com/sunithapachala93/final-web-scraping-project>

<https://jovian.com/sunithapachala93/final-web-scraping-project>