Scraping IMDB using Python & BeautifulSoup

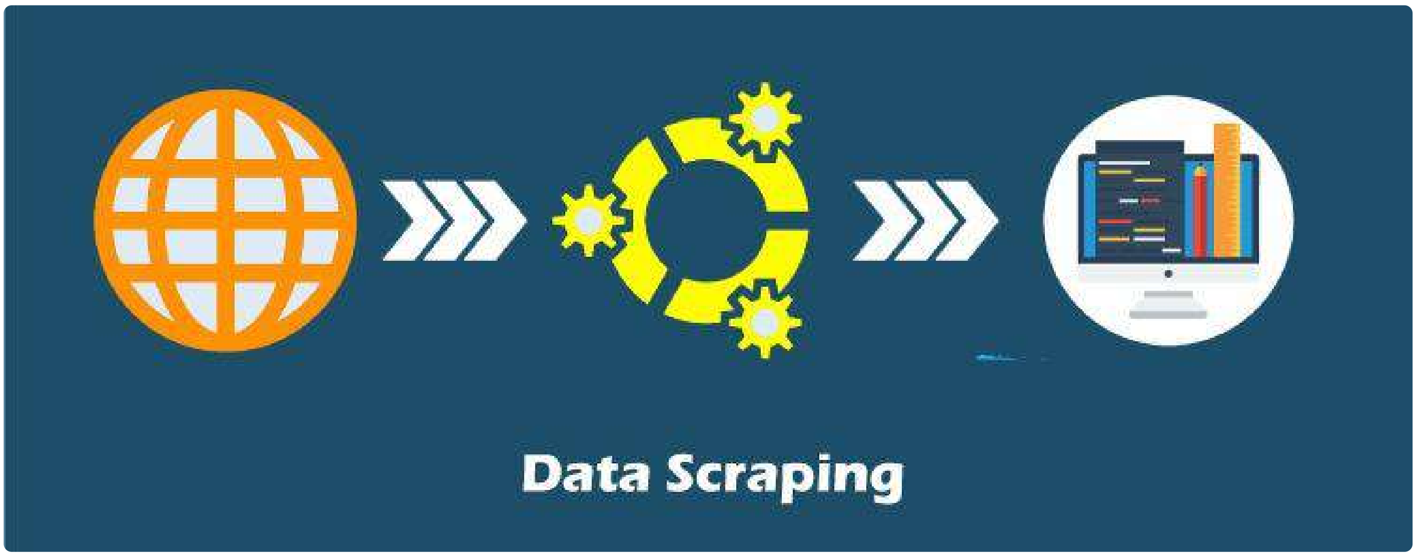
Scraping

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:

Project Steps

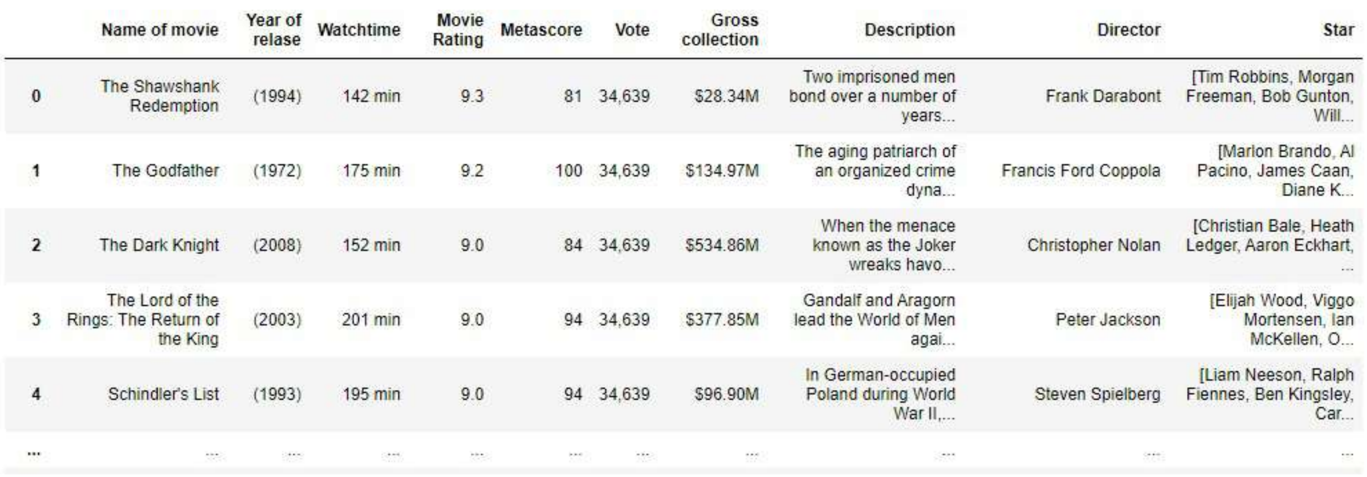
```
-> Download the webpage using requests
-> Parse the HTML source code using BeautifulSoup library
-> Extract the desired infromation
-> Building the scraper components
-> Compile the extracted information into Python list and dictionaries
-> Converting the python dictionaries into Pandas DataFrames
-> 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.
-> Numpy :- It is a general-purpose array-processing package.
```

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:



	Name of movie	Year of relase	Watchtime	Movie Rating	Metascore	Vote	Gross collection	Description	Director	Star
0	The Shawshank Redemption	(1994)	142 min	9.3	81	34,639	$28.34M	Two imprisoned men bond over a number of years...	Frank Darabont	[Tim Robbins, Morgan Freeman, Bob Gunton, Will...
1	The Godfather	(1972)	175 min	9.2	100	34,639	$134.97M	The aging patriarch of an organized crime dyna...	Francis Ford Coppola	[Marlon Brando, Al Pacino, James Caan, Diane K...
2	The Dark Knight	(2008)	152 min	9.0	84	34,639	$534.86M	When the menace known as the Joker wreaks havo...	Christopher Nolan	[Christian Bale, Heath Ledger, Aaron Eckhart, ...
3	The Lord of the Rings: The Return of the King	(2003)	201 min	9.0	94	34,639	$377.85M	Gandalf and Aragorn lead the World of Men agai...	Peter Jackson	[Elijah Wood, Viggo Mortensen, Ian McKellen, O...
4	Schindler's List	(1993)	195 min	9.0	94	34,639	$96.90M	In German-occupied Poland during World War II,...	Steven Spielberg	[Liam Neeson, Ralph Fiennes, Ben Kingsley, Car...
...	...	...	...	...	...	...	...	...	...	...

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="project-python")
```

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

```
'https://jovian.ai/abhineets17/project-python'
```

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.

```
import requests       #to send the request to the URL
```

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.

```
url = 'https://www.imdb.com/search/title/?count=100&groups=top_1000&sort=user_rating'
response = requests.get(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.

```python
# 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.

```python
page_contents = response.text

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

```
686570
```

```
   Boom! We have ~7Lac characters within the HTML that we just downloaded in a second
```

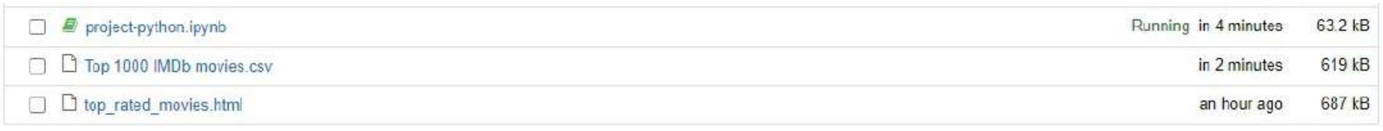
```python
# Its display the first 1000 characters of page contents.
print(page_contents[:1000])
```

```


<!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 1000&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_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
```


project-python.ipynb Running in 4 minutes 63.2 kB
Top 1000 IMDb movies.csv in 2 minutes 619 kB
top_rated_movies.html an hour ago 687 kB

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.
```

Learn more about :(https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

```
from bs4 import BeautifulSoup #to get the content in the form of HTML
```

```
soup = BeautifulSoup(response.content, 'html.parser')
```

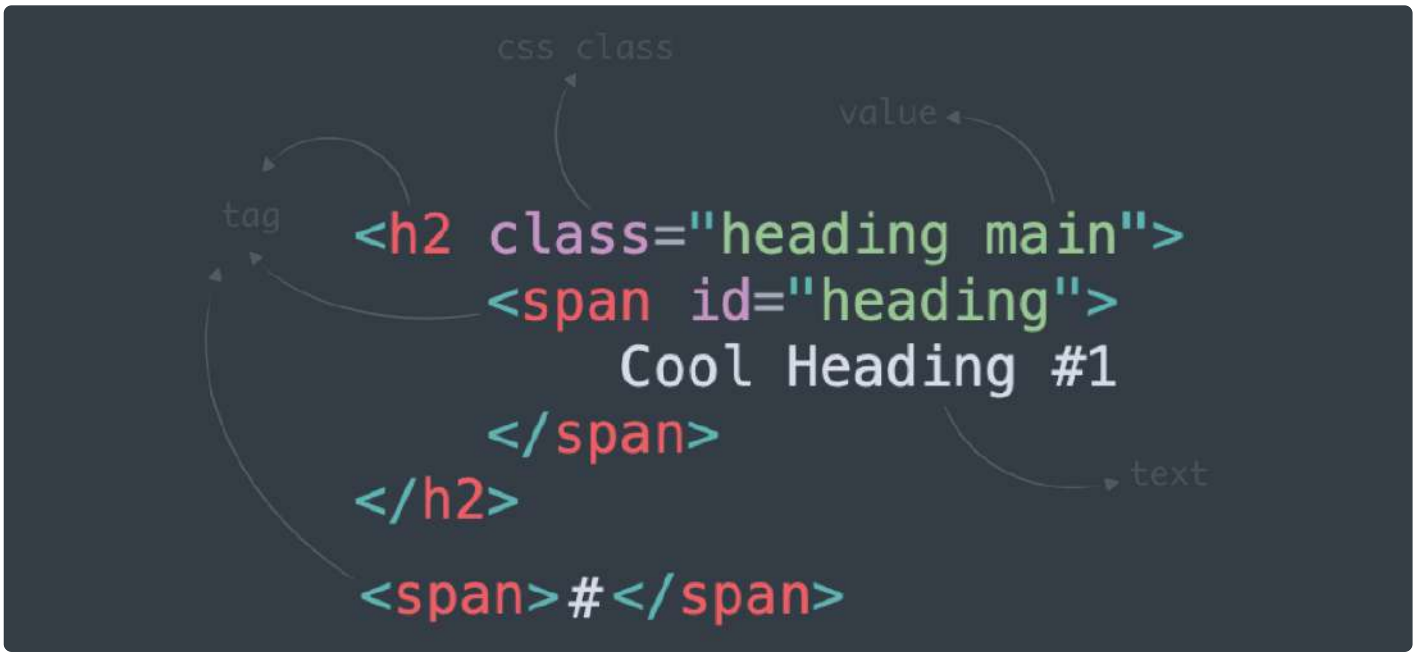
Inspecting the HTML source code of a web page

```
    ->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.
```

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>.
```

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>)
```

Now we will use BeautifulSoup to extract the Details of the top Movies from the HTML Page

```
#creating an empty list, so that we can append the values
movie_name = []
year = []
time = []
rating = []
metascore = []
votes = []
gross = []
description = []
Director = []
Stars = []
```

```
#storing the meaningfull required data in the variable

movie_data = soup.findAll('div', attrs= {'class': 'lister-item mode-advanced'})
#calling one by one using for loop

for store in movie_data:
    name = store.h3.a.text
    movie_name.append(name)

    year_of_release = store.h3.find('span', class_ = 'lister-item-year text-muted unbol
    year.append(year_of_release)

    runtime = store.p.find('span', class_ = 'runtime').text.replace(' min', '')
    time.append(runtime)

    rate = store.find('div', class_ = 'inline-block ratings-imdb-rating').text.replace(
    rating.append(rate)
    meta  = store.find('span', class_ = 'metascore').text.replace(' ', '') if store.fir
    metascore.append(meta)

    #since, gross and votes have same attributes, that's why we had created a common va

    value = store.find_all('span', attrs = {'name': 'nv'})

    vote = value[0].text
    votes.append(vote)

    grosses = value[1].text if len(value) >1 else '*****'
    gross.append(grosses)

    # Description of the Movies

    describe = store.find_all('p', class_ = 'text-muted')
    description_ = describe[1].text.replace('\n', '') if len(describe) >1 else '*****'
    description.append(description_)

    #Cast Details -- Scraping Director name and Stars

    cast = store.find("p", class_ = '')
    cast = cast.text.replace('\n', '').split('|')
    cast = [x.strip() for x in cast]
    cast = [cast[i].replace(j, "") for i,j in enumerate(["Director:", "Stars:"])]
    Director.append(cast[0])
    Stars.append([x.strip() for x in cast[1].split(",")])
```

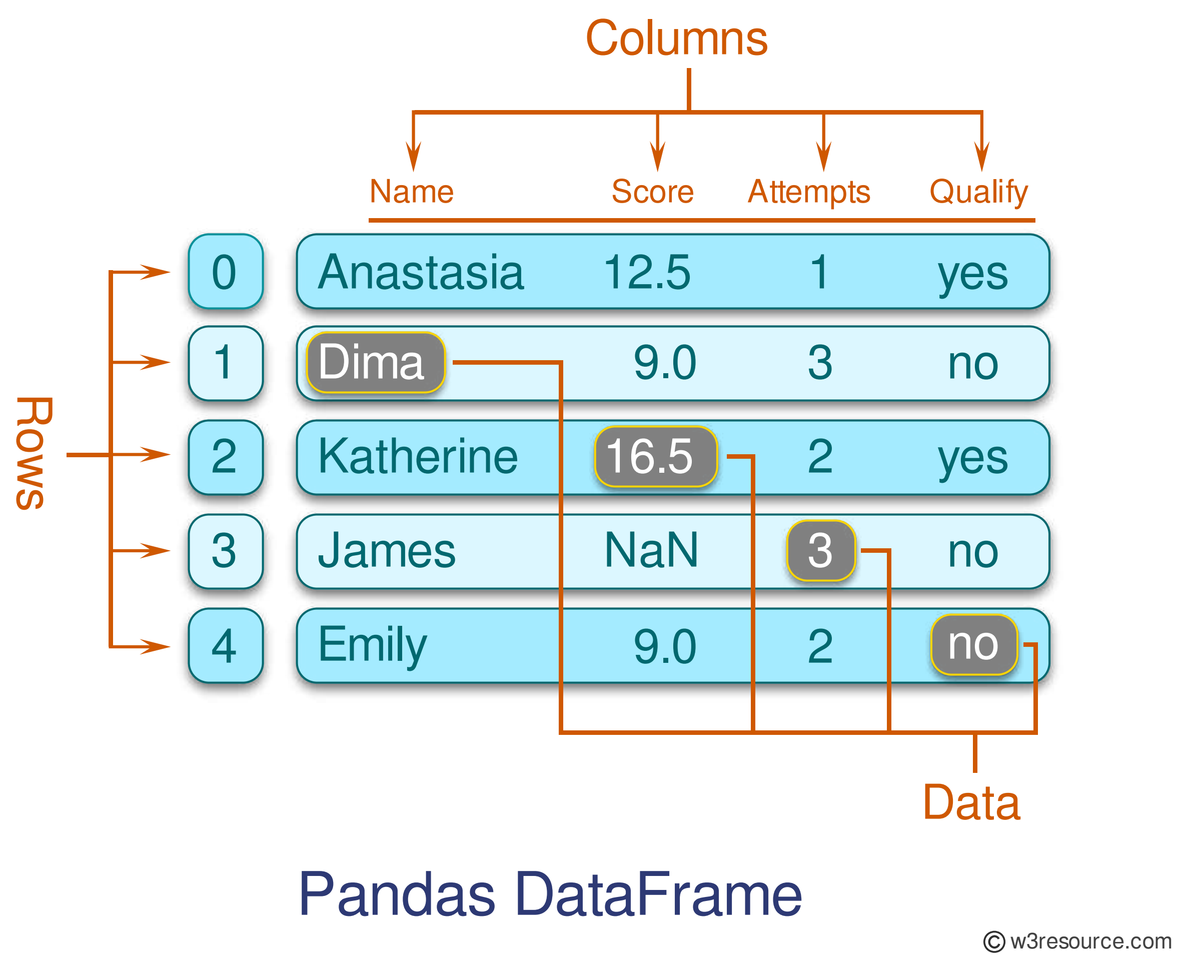
```
doc=BeautifulSoup(page_contents, 'html.parser')
def get_movie_rating(doc):
    rating_selector="inline-block ratings-imdb-rating"
    movie_rating_tags=doc.find_all('div',{'class':rating_selector})
    movie_rating_tagss=[]
    for tag in movie_rating_tags:
        movie_rating_tagss.append(tag.get_text().strip())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

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

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

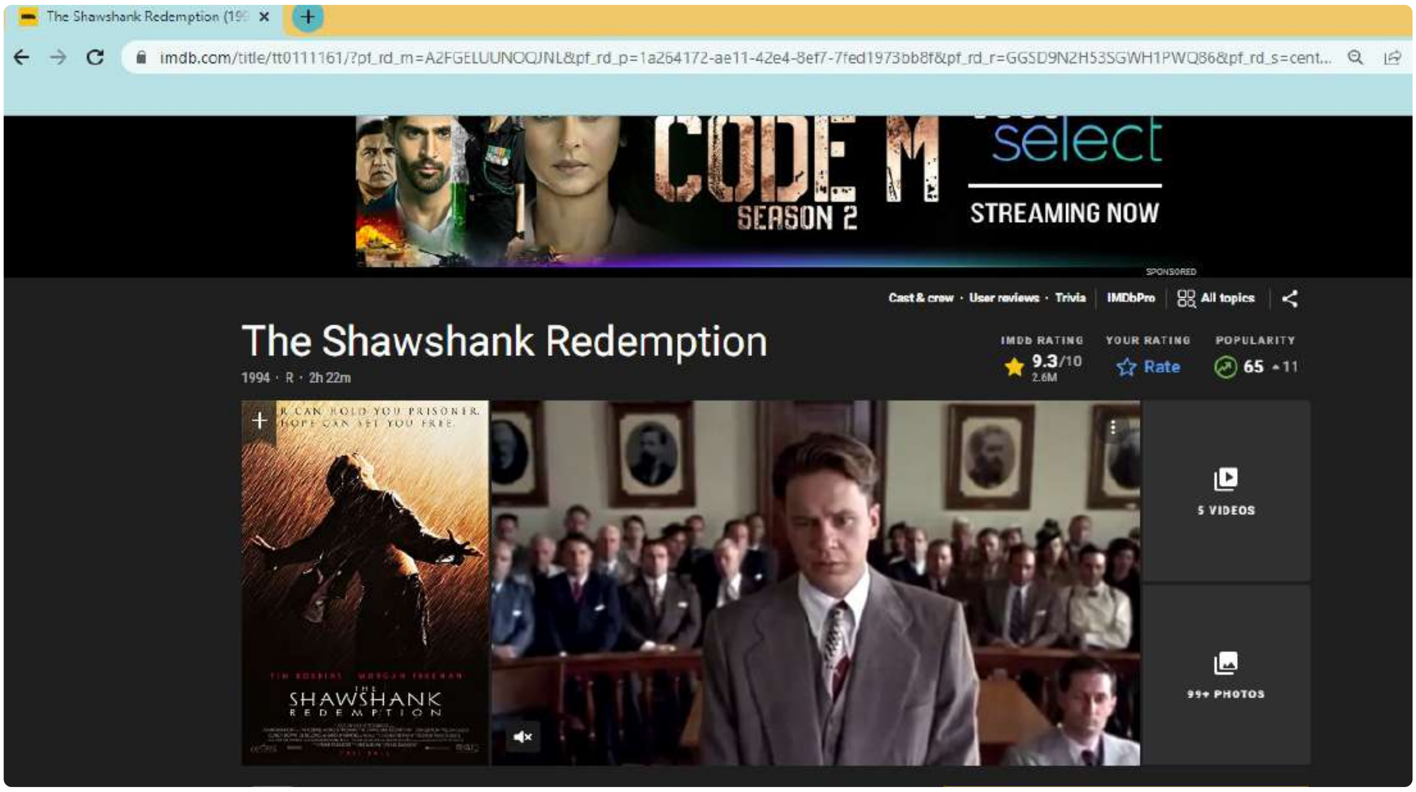
Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.



```
 #to create dataframe
import pandas as pd
```

First Movie(The Shawshank Redemption)

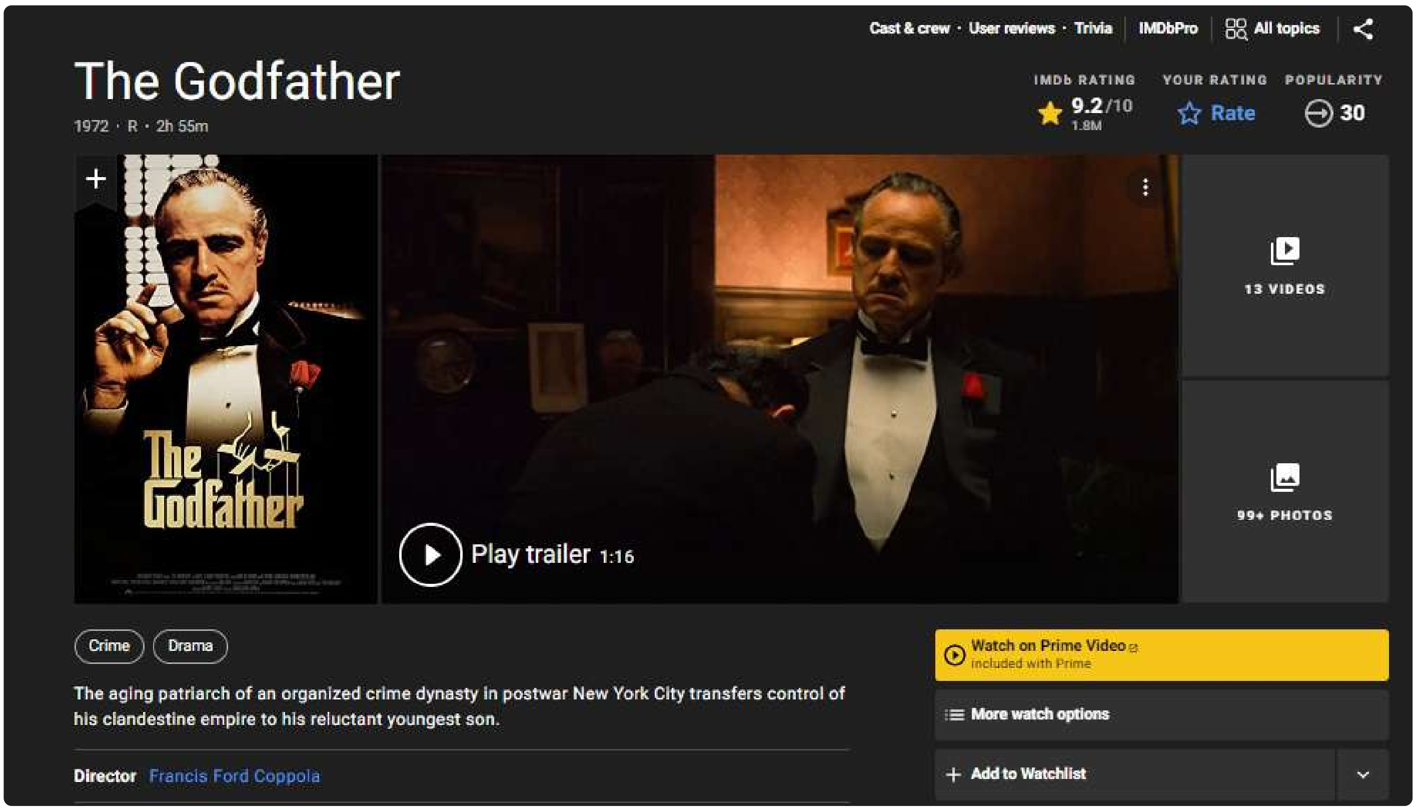

The Shawshank Redemption (19
imdb.com/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=1a264172-ae11-42e4-8ef7-7fed1973bb8f&pf_rd_r=GGSD9N2H53SGWH1PWQ86&pf_rd_s=cent...
CODE M
SEASON 2
select
STREAMING NOW
SPONSORED
Cast & crew · User reviews · Trivia
IMDbPro
All topics
The Shawshank Redemption
1994 · R · 2h 22m
IMDB RATING
9.3/10
2.6M
YOUR RATING
Rate
POPULARITY
65 ▲11
5 VIDEOS
99+ PHOTOS

```python
#creating a dataframe using pandas library
movie_1_DF = pd.DataFrame({'Name of movie': movie_name[0],
                         'Year of relase': year[0],
                         'Watchtime': time[0],
                         'Movie Rating': rating[0],
                         'Metascore':  metascore[0],
                         'Vote':    vote[0],
                         'Gross collection': gross[0],
                         'Description': description[0],
                         "Director": Director[0],
                         'Star': Stars[0]
                        })
```

```python
movie_1_DF.head(1)
```

	Name of movie	Year of relase	Watchtime	Movie Rating	Metascore	Vote	Gross collection	Description	Director	Star
0	The Shawshank Redemption	1994	142	9.3	81	2	$28.34M	Two imprisoned men bond over a number of years...	Frank Darabont	Tim Robbins

Second Movie(The Godfather)


Cast & crew · User reviews · Trivia | IMDbPro | All topics

The Godfather

1972 · R · 2h 55m

IMDb RATING 9.2/10 1.8M | YOUR RATING Rate | POPULARITY 30

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

Watch on Prime Video
included with Prime

More watch options

Add to Watchlist

```python
movie_2_DF = pd.DataFrame({'Name of movie': movie_name[1],
                         'Year of relase': year[1],
                         'Watchtime': time[1],
                         'Movie Rating': rating[1],
                         'Metascore':  metascore[1],
                         'Vote':    vote[1],
                         'Gross collection': gross[1],
                         'Description': description[1],
                         "Director": Director[1],
                         'Star': Stars[1]
                        })
```

```
movie_2_DF.head(1)
```

	Name of movie	Year of relase	Watchtime	Movie Rating	Metascore	Vote	Gross collection	Description	Director	Star
0	The Godfather	1972	175	9.2	100	4	$134.97M	The aging patriarch of an organized crime dyna...	Francis Ford Coppola	Marlon Brando

Total Movie list of First page.

```
movie_DF = pd.DataFrame({'Name of movie': movie_name,
                         'Year of relase': year,
                         'Watchtime': time,
                         'Movie Rating': rating,
                         'Metascore':  metascore,
                         'Vote':    vote,
                         'Gross collection': gross,
                         'Description': description,
                         "Director": Director,
                         'Star': Stars
                        })
```

```
movie_DF.head(100)        #Total 100 movie name on first page.
```

	Name of movie	Year of relase	Watchtime	Movie Rating	Metascore	Vote	Gross collection	Description	Director	Star
0	The Shawshank Redemption	1994	142	9.3	81	242,379	$28.34M	Two imprisoned men bond over a number of years...	Frank Darabont	[Tim Robbins, Morgan Freeman, Bob Gunton, Will...
1	The Godfather	1972	175	9.2	100	242,379	$134.97M	The aging patriarch of an organized crime dyna...	Francis Ford Coppola	[Marlon Brando, Al Pacino, James Caan, Diane K...
2	The Dark Knight	2008	152	9.0	84	242,379	$534.86M	When the menace known as the Joker wreaks havo...	Christopher Nolan	[Christian Bale, Heath Ledger, Aaron Eckhart, ...

	Name of movie	Year of relase	Watchtime	Movie Rating	Metascore	Vote	Gross collection	Description	Director	Star
3	The Lord of the Rings: The Return of the King	2003	201	9.0	94	242,379	$377.85M	Gandalf and Aragorn lead the World of Men agai...	Peter Jackson	[Elijah Wood, Viggo Mortensen, Ian McKellen, O...
4	Schindler's List	1993	195	9.0	94	242,379	$96.90M	In German-occupied Poland during World War II,...	Steven Spielberg	[Liam Neeson, Ralph Fiennes, Ben Kingsley, Car...
...	...	...	...	...	...	...	...	...	...	...
95	Ayla: The Daughter of War	2017	125	8.3	^^^^^^	242,379	*****	In 1950, amid-st the ravages of the Korean War...	Can Ulkay	[Çetin Tekindor, Ismail Hacioglu, Kyung-jin Le...
96	Dangal	2016	161	8.3	^^^^^^	242,379	$12.39M	Former wrestler Mahavir Singh Phogat and his t...	Nitesh Tiwari	[Aamir Khan, Sakshi Tanwar, Fatima Sana Shaikh...
97	Drishyam	2013	160	8.3	^^^^^^	242,379	*****	A man goes to extreme lengths to save his fami...	Jeethu Joseph	[Mohanlal, Meena, Asha Sharath, Ansiba]
98	The Hunt	2012	115	8.3	77	242,379	$0.69M	A teacher lives a lonely life, all the while s...	Thomas Vinterberg	[Mads Mikkelsen, Thomas Bo Larsen, Annika Wedd...
99	A Separation	2011	123	8.3	95	242,379	$7.10M	A married couple are faced with a difficult de...	Asghar Farhadi	[Payman Maadi, Leila Hatami, Sareh Bayat, Shah...

100 rows × 10 columns

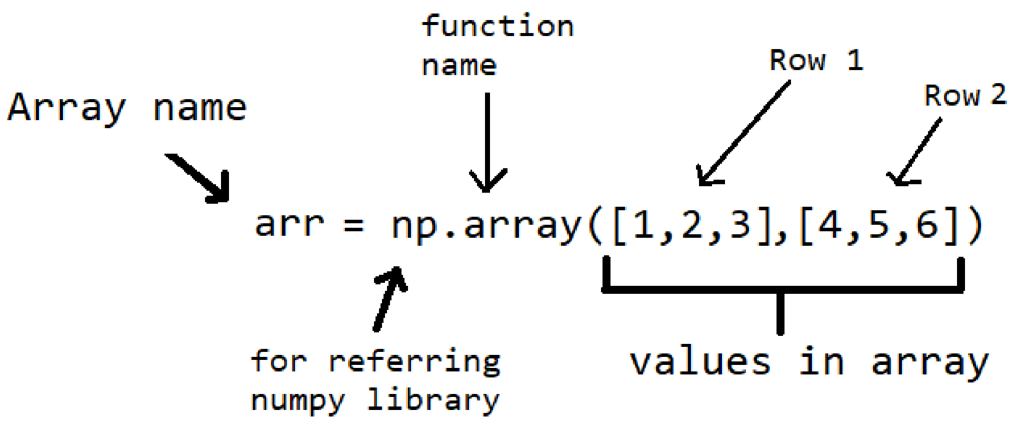
```
b_url='https://www.imdb.com/search/title/?groups=top_1000&sort=user_rating,desc&count=1
def url_building(url):
    urls = []
    for i in range(10):
        urls.append(b_url + str(i*100 + 1))
    return urls

URLs = url_building(b_url)
URLs
```

```
['https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=1',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=101',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=201',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=301',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=401',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=501',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=601',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=701',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=801',
 'https://www.imdb.com/search/title/?
groups=top_1000&sort=user_rating,desc&count=100&start=901']
```

Now We Scrape Multiple Page.

NumPy

NumPy is a Python package. It stands for ‘Numerical Python’. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.



```python
import numpy as np
```

Python sleep()

The sleep() function suspends (waits) execution of the current thread for a given number of seconds.

```
from time import sleep
```

Python randint()

The randint() method returns an integer number selected element from the specified range.

```
from random import randint
```

```
#Declaring the headers
headers = {"Accept-Language": "en-US,en;q=0.5"}
```

```
#declaring the list of empty variables, So that we can append the data overall

movie_name = []
year = []
time = []
rating = []
metascore = []
votes = []
gross = []
description = []
Director = []
Stars = []
```

```
#creating an array of values and passing it in the url for dynamic webpages

pages = np.arange(1,1000,100)

#the whole core of the script

for page in pages:
    page = requests.get("https://www.imdb.com/search/title/?groups=top_1000&sort=user_r
    soup = BeautifulSoup(page.text, 'html.parser')
    movie_data = soup.findAll('div', attrs = {'class': 'lister-item mode-advanced'})
    sleep(randint(2,8))
    for store in movie_data:
        name = store.h3.a.text
        movie_name.append(name)

        year_of_release = store.h3.find('span', class_ = "lister-item-year text-muted u
        year.append(year_of_release)

        runtime = store.p.find("span", class_ = 'runtime').text
        time.append(runtime)

        rate = store.find('div', class_ = "inline-block ratings-imdb-rating").text.repl
        rating.append(rate)

        meta = store.find('span', class_ = "metascore").text if store.find('span', clas
        metascore.append(meta)


        value = store.find_all('span', attrs = {'name': "nv"})

        vote = value[0].text
        votes.append(vote)

        grosses = value[1].text if len(value)>1 else '%^%^%^'
        gross.append(grosses)

        # Description of the Movies

        describe = store.find_all('p', class_ = 'text-muted')
        description_ = describe[1].text.replace('\n', '') if len(describe) >1 else '***
        description.append(description_)


        #Cast Details -- Scraping Director name and Stars
        cast = store.find("p", class_ = '')
        cast = cast.text.replace('\n', '').split('|')
        cast = [x.strip() for x in cast]
        cast = [cast[i].replace(j, "") for i,j in enumerate(["Director:", "Stars:"])]
        Director.append(cast[0])
        Stars.append([x.strip() for x in cast[1].split(",")])
```

```
#creating a dataframe using pandas library

movie_list = pd.DataFrame({'Name of movie': movie_name,
                           'Year of relase': year,
                           'Watchtime':    time,
                           'Movie Rating': rating,
                           'Metascore':  metascore,
                           'Vote':     vote,
                           'Gross collection': gross,
                           'Description': description,
                           "Director": Director,
                           'Star': Stars
                        })
```

```
movie_list.head(1000)
```

	Name of movie	Year of relase	Watchtime	Movie Rating	Metascore	Vote	Gross collection	Description	Director	S
0	The Shawshank Redemption	(1994)	142 min	9.3	81	34,652	$28.34M	Two imprisoned men bond over a number of years...	Frank Darabont	[T Robbi Morg Freem Bob Gunt Wi

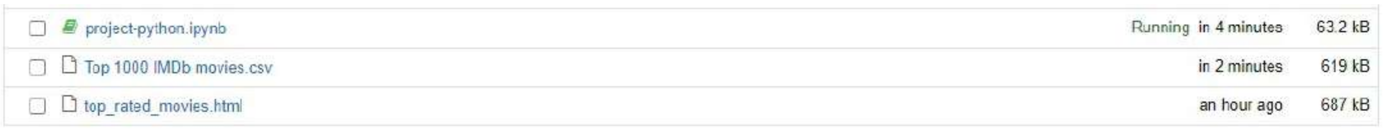
	Name of movie	Year of relase	Watchtime	Movie Rating	Metascore	Vote	Gross collection	Description	Director	S
1	The Godfather	(1972)	175 min	9.2	100	34,652	$134.97M	The aging patriarch of an organized crime dyna...	Francis Ford Coppola	[Marl Brando Paci Jam Caan, Dia
2	The Dark Knight	(2008)	152 min	9.0	84	34,652	$534.86M	When the menace known as the Joker wreaks havo...	Christopher Nolan	[Christi Bale, Hea Ledg Aar Eckhart
3	The Lord of the Rings: The Return of the King	(2003)	201 min	9.0	94	34,652	$377.85M	Gandalf and Aragorn lead the World of Men agai...	Peter Jackson	[Elij Wood, Vig Mortens McKell
4	Schindler's List	(1993)	195 min	9.0	94	34,652	$96.90M	In German-occupied Poland during World War II,...	Steven Spielberg	[Lia Nees Ral Fiennes, E Kings Ca
...	...	...	...	...	...	...	...	...	...	
995	Sabrina	(1954)	113 min	7.6	72	34,652	%^%^%^	A playboy becomes interested in the daughter o...	Billy Wilder	[Humph Boga Aud Hepbu Willia Hol
996	From Here to Eternity	(1953)	118 min	7.6	85	34,652	$30.50M	At a U.S. Army base in 1941 Hawaii, a private ...	Fred Zinnemann	[B Lancas Montgom C Debor Ke
997	Snow White and the Seven Dwarfs	(1937)	83 min	7.6	95	34,652	$184.93M	Exiled into the dangerous forest by her wicked...	Directors:William Cottrell, David Hand, Wilfre...	[Adria Caselo Ha Stockw Lucille
998	The 39 Steps	(1935)	86 min	7.6	93	34,652	%^%^%^	A man in London tries to help a counter-espion...	Alfred Hitchcock	[Rob Dor Madele Carroll, Lu Mannh
999	The Invisible Man	(1933)	71 min	7.6	87	34,652	%^%^%^	A scientist finds a way of becoming invisible,...	James Whale	[Clau Rains, Glo Stu Willia Harriga

1000 rows × 10 columns

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)
```

```python
movie_list.to_csv("Top 1000 IMDb movies.csv", index=None)
```



project-python.ipynb	Running	in 4 minutes	63.2 kB
Top 1000 IMDb movies.csv		in 2 minutes	619 kB
top_rated_movies.html		an hour ago	687 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'
->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
    Year of Release
    Rating
    Watch Time
    Director
    Star
    Gross Collection
    Metascore
```

```
        Description
        Vote
    ->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(files=['Top 1000 IMDb movies.csv'])
```