
IMDb®
WEB SCRAPING
250
movies

LIBRARIES USE

request

```
pip install requests
```

```
Requirement already satisfied: requests in c:\users\roari\anaconda3\lib\site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\roari\anaconda3\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\roari\anaconda3\lib\site-packages (from requests) (2024.2.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\roari\anaconda3\lib\site-packages (from requests) (1.26.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\roari\anaconda3\lib\site-packages (from requests) (2.10)
Note: you may need to restart the kernel to use updated packages.
```

beautifulSoup4 pandas

```
pip install requests beautifulsoup4 pandas
```

```
Requirement already satisfied: requests in c:\users\roari\anaconda3\lib\site-packages (2.31.0)
Requirement already satisfied: beautifulsoup4 in c:\users\roari\anaconda3\lib\site-packages (4.9.3)
Requirement already satisfied: pandas in c:\users\roari\anaconda3\lib\site-packages (2.0.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\roari\anaconda3\lib\site-packages (from beautifulsoup4) (2.2.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\roari\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\roari\anaconda3\lib\site-packages (from pandas) (2021.1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\roari\anaconda3\lib\site-packages (from pandas) (2023.4)
```

html5lib

```
pip install html5lib
```

```
Requirement already satisfied: html5lib in c:\users\roari\anaconda3\lib\site-packages (1.1)
Requirement already satisfied: six>=1.9 in c:\users\roari\anaconda3\lib\site-packages (from html5lib) (1.15.0)
Requirement already satisfied: webencodings in c:\users\roari\anaconda3\lib\site-packages (from html5lib) (0.5.1)
Note: you may need to restart the kernel to use updated packages.
```

WEB SCRAPING

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

```
url="https://www.imdb.com/chart/top/"
```

Get the HTML

```
r=requests.get(url)
htmlcontent=r.content
print(htmlcontent)
```

```
b'<html>\r\n<head><title>403 Forbidden</title></head>\r\n<body>\r\n<center><h1>403 Forbidden</h1></center>\r\n</body>\r\n</html
>\r\n'
```

Creating Soup

```
soup=BeautifulSoup(htmlcontent,'html.parser')
print(soup)
```

```
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
</body>
</html>
```

```
import urllib3

http = urllib3.PoolManager()
response = http.request('GET', 'https://www.imdb.com/chart/top/')
print(response.status)
```

```
403
```

```
http = urllib3.PoolManager()

response = http.request("GET",'https://www.imdb.com/chart/top/')
x=response.data.decode("utf-8")
print(x)
```

```
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
</body>
</html>
```

```
from bs4 import BeautifulSoup
response=http.request("GET",'https://www.imdb.com/chart/top/')
html=response.data.decode("utf-8")
soup=BeautifulSoup(html,"html.parser")
print(soup.get_text())
```

```
403 Forbidden

403 Forbidden
```

SCRAPING URL

WEB SCRAPPING URL

```python
import requests
from bs4 import BeautifulSoup

# Define the URL to scrape
url = "https://www.imdb.com/chart/top/"

# Define user-agent to mimic a web browser
user_agent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36"

# Define headers with user-agent
headers = {
    "User-Agent": user_agent
}

try:
    # Send a GET request with the specified headers
    response = requests.get(url, headers=headers)

    # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # Parse the HTML content of the response
        soup = BeautifulSoup(response.content, 'html.parser')

        # Extract the data you need from the HTML using BeautifulSoup methods
        # For example, to find all <a> tags, you can use:
        links = soup.find_all('a')

        # Process the extracted data as needed
        if links:
            for link in links:
                print(link.get('href'))
        else:
            print("No links found on the webpage.")
```

```python
        # You can also extract other elements or data from the HTML

    else:
        # If the request was not successful, print an error message with the status code
        print("Failed to retrieve the webpage. Status code:", response.status_code)

except Exception as e:
    # If an exception occurs during the request, print the error message
    print("An error occurred:", e)
```

OUTPUT

```
https://slyb.app.link/SKdyQ6A449
https://www.tiktok.com/@imdb
https://instagram.com/imdb
https://twitter.com/imdb
https://youtube.com/imdb/
https://facebook.com/imdb
https://slyb.app.link/SKdyQ6A449
https://help.imdb.com/imdb
https://help.imdb.com/article/imdb/general-information/imdb-site-index/GNCX7BHNSPBTFALQ#so
https://pro.imdb.com?ref_=cons_tf_pro&rf=cons_tf_pro
https://www.boxofficemojo.com
https://developer.imdb.com/
https://www.imdb.com/pressroom/?ref_=ft_pr
https://advertising.amazon.com/resources/ad-specs/imdb/
https://www.amazon.jobs/en/teams/imdb
/conditions?ref_=ft_cou
/privacy?ref_=ft_pvc
/privacy/redirect/?ref_=ft_redir
/privacy/redirect/?ref_=ft_redir
```

SCRAPING HOME PAGE DATA

WEB SCRAPPING DATA

```python
import requests
from bs4 import BeautifulSoup

# Function to fetch all text from a given URL
def fetch_all_text(url):
    # Define a user-agent header to mimic a web browser
    headers = {'User-Agent': 'Mozilla/5.0'}

    # Send a GET request to the URL with the user-agent header
    response = requests.get(url, headers=headers)

    # Check if the request was successful
    if response.status_code == 200:
        # Parse the HTML content using BeautifulSoup
        soup = BeautifulSoup(response.content, 'html.parser')

        # Extract all text from the parsed HTML
        all_text = soup.get_text()

        return all_text
    else:
        print("Failed to fetch content from the URL:", url)
        return None

# IMDb Top 250 Movies URL
url = "https://www.imdb.com/chart/top/"

# Fetch all text from the IMDb Top 250 Movies page
all_text = fetch_all_text(url)

# Print the extracted text
if all_text:
    print(all_text)
```

OUTPUT

```
IMDb Top 250 Movies


MenuMoviesRelease CalendarTop 250 MoviesMost Popular MoviesBrowse Movies by GenreTop Box OfficeShowtimes & TicketsMovie NewsI
ndia Movie SpotlightTV ShowsWhat's on TV & StreamingTop 250 TV ShowsMost Popular TV ShowsBrowse TV Shows by GenreTV NewsWatch
What to WatchLatest TrailersIMDb OriginalsIMDb PicksIMDb SpotlightIMDb PodcastsAwards & EventsOscarsSXSW Film FestivalWomen's
History MonthSTARmeter AwardsAwards CentralFestival CentralAll EventsCelebsBorn TodayMost Popular CelebsCelebrity NewsCommuni
tyHelp CenterContributor ZonePollsFor Industry ProfessionalsLanguageEnglish (United States)LanguageFully supportedEnglish (Un
ited States)Partially supportedFrançais (Canada)Français (France)Deutsch (Deutschland)हिंदी (भारत)Italiano (Italia)Português (B
rasil)Español (España)Español (México)AllAllTitlesTV EpisodesCelebsCompaniesKeywordsAdvanced SearchWatchlistSign InSign InNew
Customer? Create accountENFully supportedEnglish (United States)Partially supportedFrançais (Canada)Français (France)Deutsch
(Deutschland)हिंदी (भारत)Italiano (Italia)Português (Brasil)Español (España)Español (México)Use app
```

```
IMDb ChartsShareIMDb Top 250 MoviesAs rated by regular IMDb voters.250 TitlesSort byRankingRankingIMDb ratingRelease dateNumb
er of ratingsAlphabeticalPopularityRuntime1. The Shawshank Redemption19942h 22mA9.3 (2.9M)Rate2. The Godfather19722h 55mA9.
2 (2M)Rate3. The Dark Knight20082h 32mUA9.0 (2.9M)Rate4. The Godfather: Part II19743h 22mA9.0 (1.4M)Rate5. 12 Angry Men19571h
36mU9.0 (859K)Rate6. Schindler's List19933h 15mA9.0 (1.4M)Rate7. The Lord of the Rings: The Return of the King20033h 21mU9.
0 (2M)Rate8. Pulp Fiction19942h 34mA8.9 (2.2M)Rate9. The Lord of the Rings: The Fellowship of the Ring20012h 58mU8.9 (2M)Rate
10. Il Buono, Il Brutto, Il Cattivo19662h 41mA8.8 (807K)Rate11. Forrest Gump19942h 22mUA8.8 (2.2M)Rate12. The Lord of the Rin
gs: The Two Towers20022h 59mUA8.8 (1.8M)Rate13. Fight Club19992h 19mA8.8 (2.3M)Rate14. Dune: Part Two20242h 46mPG-138.8 (252
K)Rate15. Inception20102h 28mUA8.8 (2.5M)Rate16. Star Wars: Episode V - The Empire Strikes Back19802h 4mUA8.7 (1.4M)Rate17.
he Matrix19992h 16mA8.7 (2M)Rate18. GoodFellas19902h 25mA8.7 (1.3M)Rate19. One Flew Over the Cuckoo's Nest19752h 13mA8.7 (1.
M)Rate20. Se7en19952h 7mA8.6 (1.8M)Rate21. Interstellar20142h 49mUA8.7 (2.1M)Rate22. It's a Wonderful Life19462h 10mPG8.6 (49
7K)Rate23. Shichinin No Samurai19543h 27mU8.6 (365K)Rate24. The Silence of the Lambs19911h 58mA8.6 (1.5M)Rate25. Saving Priva
te Ryan19982h 49mA8.6 (1.5M)Rate26. City of God20022h 10mA8.6 (798K)Rate27. Life Is Beautiful19971h 56mU8.6 (740K)Rate28. The
Green Mile19993h 9mUA8.6 (1.4M)Rate29. Terminator 2: Judgment Day19912h 17mA8.6 (1.2M)Rate30. Star Wars: Episode IV - A New H
ope19772h 1mU8.6 (1.4M)Rate31. Back to the Future19851h 56mU8.5 (1.3M)Rate32. Spirited Away20012h 5mU8.6 (843K)Rate33. The Pi
anist20022h 30m138.5 (906K)Rate34. Parasite20192h 12mA8.5 (948K)Rate35. Spider-man: Across the Spider-verse20232h 20mU8.6 (31
9K)Rate36. Psycho19601h 49mA8.5 (715K)Rate37. Gladiator20002h 35mUA8.5 (1.6M)Rate38. The Lion King19941h 28mU8.5 (1.1M)Rate3
9. Léon19941h 50mA8.5 (1.2M)Rate40. The Departed20062h 31mA8.5 (1.4M)Rate41. American History X19981h 59mR8.5 (1.2M)Rate42.
```

SAVING THE WHOLE DATASET TO EXCEL FILE

FETCHING ALL DATA

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Function to fetch all text from a given URL
def fetch_all_text(url):
    # Define a user-agent header to mimic a web browser
    headers = {'User-Agent': 'Mozilla/5.0'}

    # Send a GET request to the URL with the user-agent header
    response = requests.get(url, headers=headers)

    # Check if the request was successful
    if response.status_code == 200:
        # Parse the HTML content using BeautifulSoup
        soup = BeautifulSoup(response.content, 'html.parser')

        # Extract all text from the parsed HTML
        all_text = soup.get_text()

        return all_text
    else:
        print("Failed to fetch content from the URL:", url)
        return None

# IMDb Top 250 Movies URL
url = "https://www.imdb.com/chart/top/"

# Fetch all text from the IMDb Top 250 Movies page
all_text = fetch_all_text(url)
```

SAVE THE EXTRACTED TEXT TO AN EXCEL FILE AND CHECKING STATUS

```
# Fetch all text from the IMDb Top 250 Movies page
all_text = fetch_all_text(url)

# Save the extracted text to an Excel file
if all_text:
    # Split the text by newlines to get individual lines
    lines = all_text.split('\n')

    # Filter out empty lines
    lines = [line.strip() for line in lines if line.strip()]

    # Create a DataFrame with the lines
    df = pd.DataFrame({'Text': lines})

    # Save the DataFrame to an Excel file
    df.to_excel('imdb_top_250_movies.xlsx', index=False)
    print("Data saved to 'imdb_top_250_movies.xlsx' successfully.")
else:
    print("No data to save.")
```

OUTPUT

```
Data saved to 'imdb_top_250_movies.xlsx' successfully.
```

FINDING THE PATH IN MY LAPTOP

```python
import os

# Get the current working directory
current_directory = os.getcwd()

# Specify the filename
file_name = "imdb_top_250_movies.xlsx"

# Join the current directory with the filename to get the full path
file_path = os.path.join(current_directory, file_name)

# Print the path of the file
print("Path of the Excel file:", file_path)
```

```
Path of the Excel file: C:\Users\roari\imdb_top_250_movies.xlsx
```

LOADING DATASET

```python
import pandas as pd

file_path = r"C:\Users\roari\Downloads\imdb_top_250_movies.xlsx"

# Read the Excel file
df = pd.read_excel(file_path)

# Display the first few rows of the dataframe
#print(df.head())
```

```python
import pandas as pd

# File path
file_path = r"C:\Users\roari\Downloads\C__Users_roari_Downloads_imdb_languages.xlsx"

# Read the Excel file
data = pd.read_excel(file_path)

# Display the DataFrame
#print(data)
```

OUTPUT

```
df
```

	Ranking	Movie Title	Release Year	IMDb Rating	Number of Ratings
0	1	The Shawshank Redemption	1994	9.3	2.9M
1	2	The Godfather	1972	9.2	2M
2	3	The Dark Knight	2008	9.0	2.9M
3	4	The Godfather: Part II	1974	9.0	1.4M
4	5	12 Angry Men	1957	9.0	859K
...	...	...	...	...	...
245	246	Castle in the Sky	1986	8.0	146K
246	247	Memories of Murder	2003	7.9	157K
247	248	Gone Girl	2014	7.9	922K
248	249	Before Sunrise	1995	8.0	278K
249	250	The Best Years of Our Lives	1946	8.0	53K

250 rows × 5 columns

DATA WRANGLING

Finding is my dataset have null values or not

```
import pandas as pd

file_path = r"C:\Users\roari\Downloads\imdb_top_250_movies.xlsx"

# Read the Excel file
df = pd.read_excel(file_path)

# Check for null values in the DataFrame
null_values = df.isnull().sum()

print("Null values in the dataset:")
print(null_values)
```

```
Null values in the dataset:
Ranking              0
Movie Title          0
Release Year         0
IMDb Rating          0
Number of Ratings    0
dtype: int64
```

Our dataset dont have any null values

INSIGHTS

Top 10 movies by Number of Ratings

```
df.head(10)
```

	Ranking	Movie Title	Release Year	IMDb Rating	Number of Ratings
0	1	The Shawshank Redemption	1994	9.3	2.9M
1	2	The Godfather	1972	9.2	2M
2	3	The Dark Knight	2008	9.0	2.9M
3	4	The Godfather: Part II	1974	9.0	1.4M
4	5	12 Angry Men	1957	9.0	859K
5	6	Schindler's List	1993	9.0	1.4M
6	7	The Lord of the Rings: The Return of the King	2003	9.0	2M
7	8	Pulp Fiction	1994	8.9	2.2M
8	9	The Lord of the Rings: The Fellowship of the Ring	2001	8.9	2M
9	10	Il Buono, Il Brutto, Il Cattivo	1966	8.8	807K

As all movies are arranged by Number of Ratings wise

Finding out total Distinct IMDB rating available for 250 Movies

```
import pandas as pd

file_path = r"C:\Users\roari\Downloads\imdb_top_250_movies.xlsx"

# Read the Excel file
df = pd.read_excel(file_path)

# Get the total distinct count of IMDb ratings
distinct_count = df['IMDb Rating'].nunique()

print("Total distinct count of IMDb ratings:", distinct_count)
```

```
Total distinct count of IMDb ratings: 14
```

Finding out total count of distinct Number of Ratings available for IMDB 250 Movies

```
import pandas as pd

file_path = r"C:\Users\roari\Downloads\imdb_top_250_movies.xlsx"

# Read the Excel file
df = pd.read_excel(file_path)

# Get the distinct count of the number of ratings
distinct_count_ratings = df['Number of Ratings'].nunique()

print("Distinct count of number of ratings:", distinct_count_ratings)
```

```
Distinct count of number of ratings: 177
```

Finding top 10 movies based on IMDB Rating

```
# Drop rows with any missing values
df_clean = df.dropna()

# Sort the DataFrame by IMDb rating in descending order
df_sorted = df_clean.sort_values(by='IMDb Rating', ascending=False)

# Get the top 10 movies
top_10_movies = df_sorted.head(10)

# Display top 10 movies
#print(top_10_movies)
top_10_movies
```

	Ranking	Movie Title	Release Year	IMDb Rating	Number of Ratings
0	1	The Shawshank Redemption	1994	9.3	2.9M
1	2	The Godfather	1972	9.2	2M
2	3	The Dark Knight	2008	9.0	2.9M
3	4	The Godfather: Part II	1974	9.0	1.4M
4	5	12 Angry Men	1957	9.0	859K
5	6	Schindler's List	1993	9.0	1.4M
6	7	The Lord of the Rings: The Return of the King	2003	9.0	2M
54	55	12th Fail	2023	9.0	105K
7	8	Pulp Fiction	1994	8.9	2.2M
8	9	The Lord of the Rings: The Fellowship of the Ring	2001	8.9	2M

Finding highest Number of IMDB movies release per year for 5 years

```
import pandas as pd

# Assuming df is your DataFrame containing movie information

# Count the number of movies released per year
movie_release_counts = df['Release Year'].value_counts().sort_values(ascending=False)

# Create a DataFrame from the result
movie_release_counts_df = pd.DataFrame({'Release Year': movie_release_counts.index, 'Number of Movies Released': movie_release_c

# Display the DataFrame
movie_release_counts_df.head(5)
```

	Release Year	Number of Movies Released
0	1995	10
1	2014	8
2	2016	8
3	2002	7
4	1984	7

Language supported by IMDB for Movies and their count

```
import pandas as pd

# File path
file_path = r"C:\Users\roari\Downloads\C__Users_roari_Downloads_imdb_languages.xlsx"

# Read the Excel file
data = pd.read_excel(file_path)

# Display the DataFrame
#print(data)
data
```

	Language Supported by IMDB Site	Support Status
0	English (United States)	Fully supported
1	English (United States)	Partially supported
2	Français (Canada)	Fully supported
3	Français (France)	Fully supported
4	Deutsch (Deutschland)	Fully supported
5	हिंदी (भारत)	Fully supported
6	Italiano (Italia)	Fully supported
7	Português (Brasil)	Fully supported
8	Español (España)	Fully supported
9	Español (México)	Fully supported

```
#print(data)
print(data.count())
```

```
Language Supported by IMDB Site    10
Support Status                     10
dtype: int64
```

How many movies are there under distinct IMDB Movie Ratings

```
# Count movies under distinct IMDb ratings
rating_counts = df['IMDb Rating'].value_counts().reset_index()
rating_counts.columns = ['IMDb Rating', 'Count of Movies']

# Sort the ratings in descending order
rating_counts_sorted = rating_counts.sort_values(by='IMDb Rating', ascending=False)

# Display the table without the index column
print(rating_counts_sorted.to_string(index=False))
```

```
 IMDb Rating  Count of Movies
         9.3                1
         9.2                1
         9.0                6
         8.9                3
         8.8                5
         8.7                5
         8.6               13
         8.5               22
         8.4               27
         8.3               39
         8.2               28
         8.1                8
         8.0               66
         7.9               26
```

So it can be concluded that under top IMDB 250 Movies 9.3 and 9.2 rating movies are only one while 9 movies have rating above 9 and highest movies are under 8.0 ratings.

Finding Top Movies For every year from IMDB 250 Movies

```python
import pandas as pd

file_path = r"C:\Users\roari\Downloads\imdb_top_250_movies.xlsx"

# Read the Excel file
df = pd.read_excel(file_path)

# Convert 'Number of Ratings' column to numeric
df['Number of Ratings'] = pd.to_numeric(df['Number of Ratings'], errors='coerce')

# Replace NaN values with zeros
df['Number of Ratings'].fillna(0, inplace=True)

# Group the data by release year and find the movie with the maximum number of ratings for each year
most_famous_movies = df.loc[df.groupby('Release Year')['Number of Ratings'].idxmax()]

# Selecting relevant columns (Movie Title and Release Year)
most_famous_movies = most_famous_movies[['Movie Title', 'Release Year']].reset_index(drop=True)

most_famous_movies
```

	Movie Title	Release Year
0	The Kid	1921
1	The Gold Rush	1925
2	Metropolis	1927
3	City Lights	1931
4	Modern Times	1936
...	...	...
76	Parasite	2019
77	Hamilton	2020
78	Top Gun: Maverick	2022
79	Spider-man: Across the Spider-verse	2023
80	Dune: Part Two	2024

81 rows × 2 columns

Finding Top movies of perticular year from IMDB 250 Movies

```
import pandas as pd

file_path = r"C:\Users\roari\Downloads\imdb_top_250_movies.xlsx"

# Read the Excel file
df = pd.read_excel(file_path)

# Convert 'Number of Ratings' column to numeric
df['Number of Ratings'] = pd.to_numeric(df['Number of Ratings'], errors='coerce')

# Replace NaN values with zeros
df['Number of Ratings'].fillna(0, inplace=True)

# Filter the DataFrame for the specific year (e.g., 2023)
year = 2023
movies_2023 = df[df['Release Year'] == year]

# Find the movie with the maximum number of ratings for the year 2023
most_famous_movie_2023 = movies_2023.loc[movies_2023['Number of Ratings'].idxmax()]

print("Most famous movie of", year, ":", most_famous_movie_2023['Movie Title'])
```

```
Most famous movie of 2023 : Spider-man: Across the Spider-verse
```

Thank You!