# Scraping Movie Information

USING SCRAPY, BEAUTIFUL SOUP AND SELENIUM

## Joblo.com

The first step is to create a csv file to write to. We need to specify the name of the csv file we would like to save, set writing mode to “w”, which is going to overwrite the old file with the same name. If we the writing mode to “a”, it’s going to append the new rows to the existing file. Once we create the file and open it for writing, we write the first row as the column names. Then, we send html request to the url we would like to parse and use Beautiful soup to extract the information we would like to store in the csv file.

```
import requests
from bs4 import BeautifulSoup
import csv

# Create a file to write to, add headers row
f = csv.writer(open('joblo_poster.csv', 'w', newline=''))
f.writerow(['Title', 'Poster'])

page = requests.get('https://www.joblo.com/movie-posters/archives/ALL/')
# Create a BeautifulSoup object
soup = BeautifulSoup(page.text, 'html.parser')
last_links = soup.find_all("li", attrs={"class":"vertical"})
number_of_poster = len(last_links)

for i in range(0,number_of_poster):
        Title= last_links[i].find("div",class_="bottom-poster").a.string
        Poster = last_links[i].div.a.img.get("src")
        Poster = 'https://www.joblo.com'+Poster
        print(i)
        f.writerow([Title,Poster])

del f
```

The table below shows the result of the above code. The column “Poster” is the link to the image that we will download in the next step.

| Title | Poster |
|---|---|
| The Butterfly Effect | https://www.joblo.com/assets/images/oldsite/posters/images/full/2004-butterfly_effect-1_thumb.jpg |
| The Butterfly Effect | https://www.joblo.com/assets/images/oldsite/posters/images/full/2004-butterfly_effect-2_thumb.jpg |
| Agent Cody Banks 2 | https://www.joblo.com/assets/images/oldsite/posters/images/full/agent-cody-banks-two-poster_thumb.jpg |
| Blade: Trinity | https://www.joblo.com/assets/images/oldsite/posters/images/full/2004-blade_trinity-1_thumb.jpg |
| Blade: Trinity | https://www.joblo.com/assets/images/oldsite/posters/images/full/2004-blade_trinity-2_thumb.jpg |
| Blade: Trinity | https://www.joblo.com/assets/images/oldsite/posters/images/full/2004-blade_trinity-3_thumb.jpg |
| Blade: Trinity | https://www.joblo.com/assets/images/oldsite/posters/images/full/2004-blade_trinity-4_thumb.jpg |
| Blade: Trinity | https://www.joblo.com/assets/images/oldsite/posters/images/full/2004-blade_trinity-5_thumb.jpg |
| Barbershop 2: Back in | https://www.joblo.com/assets/images/oldsite/posters/images/full/barbershop2-poster_thumb.jpg |

The next step is to download all the image from the links that we scraped to csv file earlier. First, we create the name of directory that we are going to store the images. After that, we create a list of all image links by using “Pandas” library.

```
####Download from joblo csv file#####
#Define the name of directory
f_name = "/joblo_poster"

# Create the directory name where the images will be saved
base_dir = os.getcwd()
dir_name = (f_name.split('/')[-1]).split('.')[0]
dir_name
dir_path = os.path.join(base_dir, dir_name)

#Create the directory if already not there
if not os.path.exists(dir_path):
    os.mkdir(dir_path)

# Read the csv with links to all the image pages
os.getcwd()
df = pd.read_csv("CSV_File\joblo_poster.csv")
df.columns
links=df.Poster
```

Once we have the list of image links ready, we define a function to download the image save them in the format “Pic_number_MovieName.jpg”. In order to avoid getting blocked by the site you are scraping or downloading, we change user agents name in the header.

```
# Function to take an image url and save the image in the given directory
def download_image(url,image_number):
    print("[INFO] downloading {}".format(url))
    name = str(url.split('/')[-1])
    name = f"Pic_{image_number}"+name #File name that will be saved
    opener=urllib.request.build_opener()#Add header and chage user-
agent name so we don't get forbidden by the website
    opener.addheaders=[('User-
Agent','Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
me/36.0.1941.0 Safari/537.36')]
    urllib.request.install_opener(opener)
    urllib.request.urlretrieve(url,os.path.join(dir_path, name))

#Download ALL image in URL links list
j=1
for i in links:
    print (j)
    download_image(i,j)
    j+=1
```

## Impawards.com

The information we want from this website are also images, so the process is basically the same as for joblo.com. The difference is just the scaping part where we extract images link which store in different HTML tag format. The code below shows how to get the image links.

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

```
#from time import sleep

# Create a file to write to, add headers row
f = csv.writer(open('impawards.csv', 'a', newline='')) # 'a' to append not overwrite th
e original file ,,  newline='' to write row without blank row betweeb each row
f.writerow(['Designer', 'Number_of_poster',"Links","Poster_name","Year"])


page = requests.get('http://www.impawards.com/designers/index.html')
# Create a BeautifulSoup object
soup = BeautifulSoup(page.text, 'html.parser')
last_links = soup.find_all("div", attrs={"class":"col-md-4"}, limit=3)
number = len(last_links)
number

for i in range(0,number):

        Designersss= last_links[i].ul.find_all("li")
        for j in range(0,len(Designersss)):
            #Designer name
            Designer = Designersss[j].find("b").string
            #Poster
            link_to_poster = Designersss[j].a.get("href")
            link_to_poster = "http://www.impawards.com/designers/"+link_to_poster
            newpage = requests.get(link_to_poster) #Go into the link
            newsoup = BeautifulSoup(newpage.text, 'html.parser')
            new_last_links= newsoup.find("center").find_next_sibling("center")
            Postersss=new_last_links.find_all("a")
            #Number of poster of each designer
            Designersss[j].b.decompose()
            Number_of_poster = Designersss[j].get_text()
            #Poster (continue)
            for k in range(0,len(Postersss)):

                #Get poster link URL
                Poster=Postersss[k].get("href")
                if Poster[0:5]=='/intl':
                    inter=Poster.split("/")
                    MyPoster="http://www.impawards.com"+"/"+inter[1]+"/"+inter[2]+"/"+i
nter[3]+"/posters/"+inter[4][:-4]+"jpg"
                    Poster_name = inter[4][:-5]
                    Year = inter[3]

                elif Poster[0:3]=="/tv" :
                    inter=Poster.split("/")
                    MyPoster="http://www.impawards.com"+"/"+inter[1]+"/posters/"+inter[
2][:-4]+"jpg"
                    Poster_name = Poster[4:-5]
                    Year = "na"
                else:
                    MyPoster="http://www.impawards.com"+Poster[0:5]+"/posters"+Poster[5
:-4]+"jpg"
                    Poster_name = Poster[6:-5]
                    Year = Poster[1:5]

                print(Designer)
                print(MyPoster)
                f.writerow([Designer,Number_of_poster, MyPoster, Poster_name, Year])

del f
```

| Designer | Number | Links | Poster_name | Year | pic_no | tconst | genres | originalTitle |
|---|---|---|---|---|---|---|---|---|
| 11:24 Design Advertising | -52 | http://www.impawa | maynard | 2017 | 1 | tt4375446 | Documentary | maynard |
| 11:24 Design Advertising | -52 | http://www.impawa | true to the gam | 2017 | 2 | tt5116504 | Drama | true to the game |
| 11:24 Design Advertising | -52 | http://www.impawa | sister code | 2015 | 3 | tt3912040 | Comedy,Drama,F | sister code |
| 11:24 Design Advertising | -52 | http://www.impawa | falcon rising | 2014 | 4 | tt2295722 | Action,Adventur | falcon rising |
| 11:24 Design Advertising | -52 | http://www.impawa | oldboy | 2013 | 5 | tt1321511 | Action,Drama,My | oldboy |

The next step is to download the image from the links we scraped earlier in the CSV file. The code below shows how to achieve this task.

```
f_name = "Impawards_updated"

# Create the directory name where the images will be saved
base_dir = os.getcwd()
dir_name = (f_name.split('/')[-1]).split('.')[0]
dir_name
dir_path = os.path.join(base_dir, dir_name)

#Create the directory if already not there
if not os.path.exists(dir_path):
    os.mkdir(dir_path)

# Read the csv with links to all the image pages
os.getcwd()
df = pd.read_csv("impawards_updated.csv")
df.columns
links=df.Links

# Function to take an image url and save the image in the given directory
def download_image(url,image_number):
    print("[INFO] downloading {}".format(url))
    name = f"Pic_{image_number}.jpg" #File name that will be saved
    try:
        opener=urllib.request.build_opener()
        opener.addheaders=[('User-
Agent','Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
me/36.0.1941.0 Safari/537.36')]
        urllib.request.install_opener(opener)
        urllib.request.urlretrieve(url,os.path.join(dir_path, name))
    except Exception as error:
        print ("[~] Error Occured with %s : %s" % (name, error))

for i in links:
    print (j)
    download_image(i,j)
    j+=1
```

## Rottentomatoes.com

We also collect image information, so the process is, again, similar to that of joblo.com. We collect the links save them to csv and download from there. However, this website is back by javascript. We can't send the request regularly and get the full information. We need to

render the javascript part. To be able to render the javascript part, we use selenium. The result in csv file and the code are shown below.

| Title | Metascore | Release_date | Image |
|---|---|---|---|
| Shazam! | 91% | 2-Jul | https://resizing.flixster.com/kuYtu_B1xzuyeXnjzWhEQFtIvM0=/130x0/v1.bTsxMzAxODU5MTtqOzE4MTk1OzEyMDA7Mjc2NDs0MDk2 |
| Avengers: | 94% | 30-Jul | https://resizing.flixster.com/7-hNtSRfnHzzQB6U_IKkDgW1pkk=/130x0/v1.bTsxMzAxOTkzMjtqOzE4MTk1OzEyMDA7MTY4ODsyNTAw |
| John Wick | 90% | 23-Aug | https://resizing.flixster.com/O_CeZ0_x3j2jnw7dlU4GZE45Grg=/130x0/v1.bTsxMzAyNzExOTtqOzE4MTk1OzEyMDA7MzYwMDs1NTUw |
| Long Shot | 81% | 16-Jul | https://resizing.flixster.com/-JUHHrJKNxwvPLg6g3oG_wuV6Wc=/130x0/v1.bTsxMzE1MDg4NDtqOzE4MTk3OzEyMDA7MjAwMDszMDAw |
| Hale Coun | 97% | 18-Jun | https://resizing.flixster.com/gt4X968w0EEMqcRY_Xp2NGYyN34=/130x0/v1.bTsxMjgzNzQ5NTtqOzE4MTkzOzEyMDA7Mzc4OzU1OQ |

```
1.  from bs4 import BeautifulSoup
2.  import csv
3.  from selenium import webdriver
4.
5.  # Create a file to write to, add headers row
6.  f = csv.writer(open('Rt_rating.csv', 'w', newline='' , encoding="utf-8"))
7.  f.writerow(['Title','Metascore','Release_date', 'Image'])
8.
9.  ####Use Selenium to open web browser
10. browser = webdriver.Chrome() #replace with .Firefox(), or with the browser of your choi
    ce
11. url = "https://www.rottentomatoes.com/browse/cf-dvd-streaming-all/"
12. browser.get(url) #navigate to page
13.
14. #Get the script from the site
15. innerHTML = browser.execute_script("return document.body.innerHTML")
16. # execute script to scroll down the page
17. browser.execute_script("window.scrollTo(0, document.body.scrollHeight);var lenOfPage=do
    cument.body.scrollHeight;return lenOfPage;")
18.
19. #Beautiful Soup
20. soup = BeautifulSoup(innerHTML, 'html.parser')
21. last_links = soup.find_all("div", attrs={"class":'mb-movie'})
22. number_of_movies = len(last_links)
23. number_of_movies
24.
25. for i in range(0,number_of_movies):
26.     Image =last_links[i].div.a.img.get('src')
27.     Title =last_links[i].find("div", attrs={"class":"movie_info"}).h3.string
28.     Metascore =last_links[i].find("span",  class_="tMeterScore").get_text(" ", strip=Tr
    ue)
29.     Release_date = last_links[i].find("p",class_="release-date").get_text(" ")[11:]
30.     print(Release_date)
31.     f.writerow([Title,Metascore,Release_date,Image])
32. del f
33.
34. browser.close()
```

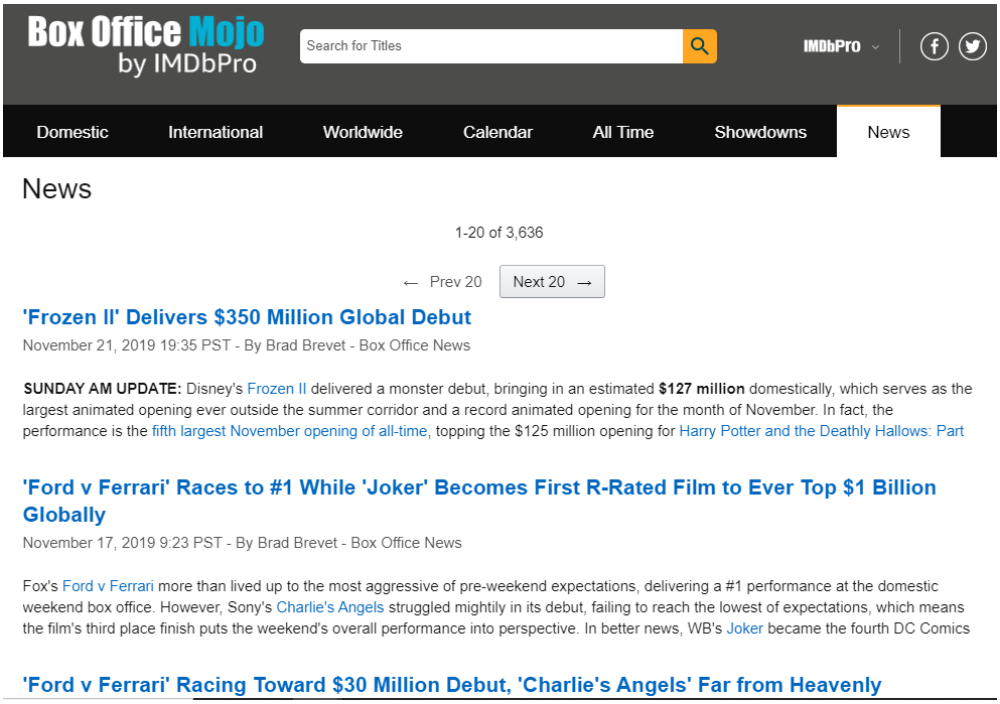
Once we have the image links, we can download from them with the same function as we use for joblo.com. We just need to change how we want to save our image files. The code below shows how to download images and save them to the specified directory.

```
1.  #Define the name of directory
2.  f_name = "Rottentomato"
3.
4.  # Create the directory name where the images will be saved
5.  base_dir = os.getcwd()
6.  dir_name = (f_name.split('/')[-1]).split('.')[0]
```

```
dir_name
dir_path = os.path.join(base_dir, dir_name)

#Create the directory if already not there
if not os.path.exists(dir_path):
    os.mkdir(dir_path)

# Read the csv with links to all the image pages
os.getcwd()
df = pd.read_csv("Rottentomato_rating.csv")
df.columns
links=df.Image

# Function to take an image url and save the image in the given directory
def download_image(url,image_number):
    print("[INFO] downloading {}".format(url))
    name = f"Pic_{image_number}.jpg" #File name that will be saved
    try:
        opener=urllib.request.build_opener()
        opener.addheaders=[('User-
Agent','Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
me/36.0.1941.0 Safari/537.36')]
        urllib.request.install_opener(opener)
        urllib.request.urlretrieve(url,os.path.join(dir_path, name))
    except Exception as error:
        print ("[~] Error Occured with %s : %s" % (name, error))

# Print the number of images
print ("[INFO] Downloading {} images".format(len(links)))
#Download ALL image in URL links list
for i in links:
    print (j)
    download_image(i,j)
    j+=1
```

## Boxofficemojo.com

In this website, we extract movie news title, contents of the news and the date posted. The title and date are not difficult to get because it includes in the page shows below; however; to get the content for each news, it is a little bit harder. We need to click in the article links. The code below shows how to do that task. We first loop into every page to get every news article and in each page, we loop again to send another request to get the news content.




```
# Create a file to write to, add headers row
f = csv.writer(open('Boxofficemojo_news.csv', 'w', newline='' , encoding="utf-8"))
f.writerow(['Date','Article','Content'])


pages = []
for i in range(1, 56):
    url = 'https://www.boxofficemojo.com/news/?view=&page=' + str(i) + '&p=.htm'
    pages.append(url)

for item in pages:
    page = requests.get(item)
    # Create a BeautifulSoup object
    soup = BeautifulSoup(page.text, 'html.parser')
    last_links = soup.find("td", attrs={"colspan": "3"})
    last_links=last_links.find_all('tr')

    number_of_article = len(last_links)

    for i in range(1,number_of_article):

            Date= last_links[i].find_all("td")[0].string

            Article =str(last_links[i].find_all("td")[1].string)

            insidelink = last_links[i].find_all("td")[1].a.get('href')
            insidelink = 'https://www.boxofficemojo.com'+insidelink
            pageinside = requests.get(insidelink)
            soupinside = BeautifulSoup(pageinside.text, 'html.parser')
            soupinside = soupinside.find_all("p", attrs={"align":"justify"})
            soupinside = BeautifulSoup(str(soupinside[0:-
1]),'lxml').get_text(" ",strip=True).replace(" , ", " ").replace("[ " , "").replace(" ]
" , "")
```

```
32.
33.             print(Date)
34.             f.writerow([Date,Article,soupinside])
```

We also extract revenue information from this website. The code below shows how to achieve the task. The information we scraped are also shown in the table below.

| Rank | Title | Studio | Worldwide_revenu | Domestic_revenue | Domestic_revenue_percent | Oversea_revenue | Oversea_revenue_percent | Year |
|---|---|---|---|---|---|---|---|---|
| 1 | igers: Endg | BV | $2,796.30 | $858.40 | 30.70% | $1,937.90 | 69.30% | 2019 |
| 2 | Avatar | Fox | $2,789.70 | $760.50 | 27.30% | $2,029.20 | 72.70% | 2009 |
| 3 | Titanic | Par. | $2,187.50 | $659.40 | 30.10% | $1,528.10 | 69.90% | 1997 |
| 4 | : The Force | BV | $2,068.20 | $936.70 | 45.30% | $1,131.60 | 54.70% | 2015 |

```
1.  # Create a file to write to, add headers row
2.  f = csv.writer(open('Boxofficemojo_revenue.csv', 'w', newline=''))
3.  f.writerow(['Rank', 'Title','Studio','Worldwide_revenue','Domestic_revenue'
4.              ,'Domestic_revenue_percent','Oversea_revenue','Oversea_revenue_percent'
5.              ,'Year'])
6.  pages = []
7.  for i in range(1, 9):
8.      url = 'https://www.boxofficemojo.com/alltime/world/?pagenum=' + str(i) + '.&p=.htm'
9.      pages.append(url)
10.
11. for item in pages:
12.     page = requests.get(item)
13.     # Create a BeautifulSoup object
14.     soup = BeautifulSoup(page.text, 'html.parser')
15.     last_links = soup.table.next_sibling.next_sibling.find_next('table')
16.     number_of_movie = len(last_links.find_all("tr"))
17.
18.     for i in range(1,number_of_movie):
19.
20.             Rank= last_links.find_all("tr")[i].find_all("td")[0].string
21.             Title =last_links.find_all("tr")[i].find_all("td")[1].string
22.             Studio = last_links.find_all("tr")[i].find_all("td")[2].string
23.             Worldwide_revenue = last_links.find_all("tr")[i].find_all("td")[3].string
24.             Domestic_revenue = last_links.find_all("tr")[i].find_all("td")[4].string
25.             Domestic_revenue_percent = last_links.find_all("tr")[i].find_all("td")[5].string
26.             Oversea_revenue = last_links.find_all("tr")[i].find_all("td")[6].string
27.             Oversea_revenue_percent = last_links.find_all("tr")[i].find_all("td")[7].string
28.             Year = last_links.find_all("tr")[i].find_all("td")[8].string[0:4]# [0:4] is to remove
    "^" on some rows
29.             print(Year)
30.
31.             f.writerow([Rank, Title,Studio,Worldwide_revenue,Domestic_revenue,Domestic_revenue_per
    cent,Oversea_revenue,Oversea_revenue_percent,Year])
32. del f
```

## OMDB

This site allows us to send API request with the limit of 1,000 a day. It will take IMBD id to search for information. First, we need to have a list of IMDB ID. After we decide which movie

id we would like to find, we then create list of OMDB URL to send request to. The output in CSV file will be the information of the movie in json format.

| IMDB_ID | Info_json_form |
|---|---|
| tt1000000 | {'Title': 'Rocco Ravishes St. Petersburg', 'Year': '2007', 'Rated': 'X', 'Released': '16 Mar 2007', 'Runtime': '145 min', 'Genre': 'Adult', 'Director': 'Rocco Siffredi', 'Writer': 'Rocco S |
| tt1000001 | {'Title': 'Rough Crossings', 'Year': '2007', 'Rated': 'N/A', 'Released': '23 Mar 2007', 'Runtime': 'N/A', 'Genre': 'Drama', 'Director': 'Steve Condie', 'Writer': 'Simon Schama', 'Acto |
| tt1000010 | {'Title': 'Strap Attack 6', 'Year': '2007', 'Rated': 'N/A', 'Released': 'N/A', 'Runtime': 'N/A', 'Genre': 'Adult', 'Director': 'Joey Silvera', 'Writer': 'N/A', 'Actors': 'Christian, Jada Fire, |
| tt1000017 | {'Title': 'Torture Room', 'Year': '2007', 'Rated': 'N/A', 'Released': '20 Jul 2007', 'Runtime': '98 min', 'Genre': 'Thriller', 'Director': 'Eric Forsberg', 'Writer': 'Eric Forsberg', 'Actors' |

```
1.  import pandas as pd
2.  import requests
3.  from bs4 import BeautifulSoup
4.  import csv
5.  import json
6.
7.  #Craete link for OMDB API Using imdb id
8.  list_imdbid='http://www.omdbapi.com/?i='+list_imdbid+'&apikey=3e46a856'
9.  list_imdbid
10. list_imdbid=list(list_imdbid)
11. len(list_imdbid)
12.
13. list_imdbid[0][26:35]
14.
15. # Create a file to write to, add headers row
16. f = csv.writer(open('OMDB_api.csv', 'w', newline='' , encoding="utf-8"))
17. f.writerow(['IMDB_ID','Info_json_form'])
18.
19. for item in list_imdbid:
20.     page = requests.get(item)
21.     # Create a BeautifulSoup object
22.     soup = BeautifulSoup(page.text, 'html.parser')
23.     soup=str(soup) #Change to string
24.     json_soup = json.loads(soup) #Change to JSON
25.     IMDB_ID=item[26:35] # TO get id
26.
27.     print(IMDB_ID)
28.     f.writerow([IMDB_ID,json_soup])
29.
30. del f
```

## Scraping Top grossing movies of 2019 from the-numbers.com using Scrapy:

The task assigned was to collect top grossing movies in the year 2019. In order to achieve the task, we considered extracting information from the-numbers.com website. The data in tabular format can easily be extracted with scrapy. Links to scrape is given as an argument and scrapy gives you the response variable with all the information of the web page. The data from the web page can be accessed with xpath command of scrapy and tabular data can be obtained by using tr and td elements of the web page and can be parsed through easily in python as a list. The code for it is as follows:

```
1.  import scrapy
2.  import csv
```

```
import logging
import time
import csv
pages_length=0
class tutorial(scrapy.Spider):
    name='tutori'
    def start_requests(self):
        #links for the data to be scrapped
        urls = [
           'https://www.the-numbers.com/box-office-records/worldwide/all-movies/cumulative/released-in-2019'
            ]
        for url in urls:
            #headers are changed to avoid getting 403 response
            yield scrapy.Request(url=url,  headers={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36'},callback=self.parse)
    def parse(self, response):
        global pages_length
        #iterating through multiple pages
        pages_length=pages_length+1
        data=[]
        #response body of the table to be scraped
        body=response.xpath("//tbody")
        #iterating through each rows in the table
        for row in body.xpath('.//tr'):
            #taking each column values
            columns=row.xpath(".//td")
            #extracting movie name
            movie_name=row.xpath(".//td[2]//b//a//text()").extract()
            row_data=[]
            #extracting other info related to the movie
            for col in columns:
                each_column=col.xpath('text()').extract()
                if len(each_column)==1:
                    row_data.append(each_column[0])
                else:
                    row_data.append('')
            row_data[1]=movie_name[0]
            #appending movie info to a list
            data.append(row_data)
            #saving each row to a csv file
        with open('movie_data.csv', 'a',newline='',encoding="utf-8") as fd:
            writer = csv.writer(fd)
            for rows in data:
                writer.writerow(rows)
                #taking the next page data if exists
        next_page=response.xpath('//div[@class="pagination"]//a//@href').extract()
        if len(next_page)>0 and len(next_page)!=pages_length:
            next_page='https://www.the-numbers.com'+next_page[pages_length]
            yield response.follow(next_page,headers={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36'}, callback=self.parse,dont_filter=True)
```

Scraping top grossing production houses from the-numbers site using Scrapy:

The task assigned was to collect top grossing production houses for year 2019 . In order to achieve the task, we considered extracting information from the-numbers.com website. The data in tabular format can easily be extracted with scrapy. Links to scrape is given as an argument and scrapy gives you the response variable with all the information of the web page. The data from the web page can be accessed with xpath command of scrapy and tabular data can be obtained by using tr and td elements of the web page and can be parsed through easily in python as a list. The code for it is as follows:

```
import scrapy
import csv
import logging
import time
import csv
class tutorial(scrapy.Spider):
    name='tutori'
    def start_requests(self):
        #links for the data to be scrapped
        urls = [
           'https://www.the-numbers.com/movies/production-companies/'
           ]
        for url in urls:
            #headers are changed to avoid getting 403 response
            yield scrapy.Request(url=url,  headers={'User-
Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/77.0.3865.90 Safari/537.36'},callback=self.parse)
    def parse(self, response):
        data=[]
        #response body of the table to be scraped
        body=response.xpath("//tbody")
        #iterating through each rows in the table
        for row in body.xpath('.//tr'):
            #taking each column values
            columns=row.xpath(".//td")
            #extracting Production name
            production_name=row.xpath(".//td[1]//b//a//text()").extract()
            row_data=[]
            #extracting other info related to the Production company
            for col in columns:
                each_column=col.xpath('text()').extract()
                if len(each_column)==1:
                    row_data.append(each_column[0])
                else:
                    row_data.append('')
            row_data[0]=production_name[0]
            #appending production info to a list
            data.append(row_data)
            #saving each row to a csv file
        with open('production_data.csv', 'a',newline='',encoding="utf-8") as fd:
            writer = csv.writer(fd)
            for rows in data:
                writer.writerow(rows)
```

## Collecting movie data from IMDB website

Subsets of IMDb data are available for access to customers for personal and non-commercial use

### Data Location

The dataset files can be accessed and downloaded from https://datasets.imdbws.com/. The data is refreshed daily.

### IMDb Dataset Details

Each dataset is contained in a gzipped, tab-separated-values (TSV) formatted file in the UTF-8 character set. The first line in each file contains headers that describe what is in each column. A '*N*' is used to denote that a particular field is missing or null for that title/name. The available datasets are as follows:

**title.akas.tsv.gz** - Contains the following information for titles:

- titleId (string) - a tconst, an alphanumeric unique identifier of the title
- ordering (integer) – a number to uniquely identify rows for a given titleId
- title (string) – the localized title
- region (string) - the region for this version of the title
- language (string) - the language of the title
- types (array) - Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd", "festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning
- attributes (array) - Additional terms to describe this alternative title, not enumerated
- isOriginalTitle (boolean) – 0: not original title; 1: original title

**title.basics.tsv.gz** - Contains the following information for titles:

- tconst (string) - alphanumeric unique identifier of the title
- titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc)
- primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release
- originalTitle (string) - original title, in the original language
- isAdult (boolean) - 0: non-adult title; 1: adult title
- startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year

- endYear (YYYY) – TV Series end year. '\N' for all other title types
- runtimeMinutes – primary runtime of the title, in minutes
- genres (string array) – includes up to three genres associated with the title

**title.crew.tsv.gz** – Contains the director and writer information for all the titles in IMDb. Fields include:

- tconst (string) - alphanumeric unique identifier of the title
- directors (array of nconsts) - director(s) of the given title
- writers (array of nconsts) – writer(s) of the given title

**title.episode.tsv.gz** – Contains the tv episode information. Fields include:

- tconst (string) - alphanumeric identifier of episode
- parentTconst (string) - alphanumeric identifier of the parent TV Series
- seasonNumber (integer) – season number the episode belongs to
- episodeNumber (integer) – episode number of the tconst in the TV series

**title.principals.tsv.gz** – Contains the principal cast/crew for titles

- tconst (string) - alphanumeric unique identifier of the title
- ordering (integer) – a number to uniquely identify rows for a given titleId
- nconst (string) - alphanumeric unique identifier of the name/person
- category (string) - the category of job that person was in
- job (string) - the specific job title if applicable, else '\N'
- characters (string) - the name of the character played if applicable, else '\N'

**title.ratings.tsv.gz** – Contains the IMDb rating and votes information for titles

- tconst (string) - alphanumeric unique identifier of the title
- averageRating – weighted average of all the individual user ratings
- numVotes - number of votes the title has received

**name.basics.tsv.gz** – Contains the following information for names:

- nconst (string) - alphanumeric unique identifier of the name/person
- primaryName (string)– name by which the person is most often credited
- birthYear – in YYYY format
- deathYear – in YYYY format if applicable, else '\N'
- primaryProfession (array of strings)– the top-3 professions of the person
- knownForTitles (array of tconsts) – titles the person is known for

## Conclusion

Web Scraping is the technique that is widely used by researchers to collect the data from websites. Many python libraries are built to achieve this task. Scrapy is powerful and fast but need some learning curve at the beginning. Beautiful soup is easier set up. Selenium is useful for javascript-content website. The next step we would like to suggest is to use Scrapy combined with Splash to scrape javascript-content website. This could be faster than using Selenium.