Movie-Netnaija Web Scrapping

Importing all relevant libraries

- About Netnaija

Netnaija is a website designed for blogging ,music and movie downloads and it was created in 2016.

- Website Main Sections
  - Blogging
  - Music
  - Movies
- Website section scrapped
  - Movies
- Aims of the project

The project aims at scraping data from the movies section that has been uploaded from 2016 till date(March 2023)

In [1]:
```
import pandas as pd
from bs4 import BeautifulSoup#Beautiful Soup is a Python library u
import requests#used to send GET and POST requests to websites and
from tqdm.notebook import tqdm_notebook
```

Creating resuable fuctions

In [2]:
```
#function to load url and parse its function
def parse_html(url):
    #this will get url from the web
    response=requests.get(url)
    soup=BeautifulSoup(response.content,'lxml')#used for processin
    return soup


#Function to print next page link
def nextlink(soup):
    try:
        next_link=soup.find('a', class_="next page-numbers", href=
        return next_link
    except:
        return #return noting if at last page


```

Get the last page of the website(netnaija)

In [3]:
```
1 url='https://www.thenetnaija.net/videos/movies'
2 soup=parse_html(url)
3 paginating=soup.find('ul',class_='pagination')
4 x=int(paginating.findAll('li')[5].text)
```

In [4]:
```
1 x #the number of pages is stored in variable 'x'
```

Out[4]: 226

Loop to get link of all pages

Base url

In [5]:
```
1  url='https://www.thenetnaija.net/videos/movies' #you may open on y
2
3  pagelink_storage=[]
4
5  for i in tqdm_notebook(range(x), desc='Loading....'):
6      soup=parse_html(url)
7      url=nextlink(soup)
8      if not url:
9          break
10     pagelink_storage.append(url)
11
12
13
14
```

Loading....: 100% 225/226 [01:31<00:00, 3.85it/s]

In [6]:
```
1 len(pagelink_storage)
```

Out[6]: 225

Loop to get the links to all movies on each page

In [7]:
```
movie_links=[] # empty list to store movie links form each page
for page in tqdm_notebook(pagelink_storage, desc='Loading...'): #
    soup= parse_html(page) # load each page and parse

    # this series of code get all link to movies on each page and
    video_files=soup.find("div", class_="video-files")
    class_info=video_files.findAll("div", class_="info")
    for x in class_info:
        link=x.find("a", href=True)['href']
        movie_links.append(link)

```

Loading...: 100% 225/225 [01:09<00:00, 4.45it/s]

In [8]:
```
len(movie_links)
```

Out[8]: 4040

In [ ]:
```

```

Empty list to store data we need about each movie

In [9]:
```
titles=[]  # movie titles
movie_linkss=[] # movie links
movie_types=[] #video type
time_of_uplos=[] # date of upload
movie_lengths=[] # lenght of movie
num_of_comments=[] #numbers of comment
mo_summarys=[] # moive summary
Genres=[] #movie genre
Release_Dates=[] #release date
Starss=[] # actors and actress
Languages=[] #movie language
Subtitles=[] #available subtitle
imdb_links=[] #imdb link
```

In [10]:

```
for link in tqdm_notebook(movie_links, desc='Loading'):
    soup= parse_html(link) # browse movie link and parse

    #This series of code get the requried data and append to the d
    try:
        title=soup.find('h1', class_="page-h1").text
        titles.append(title)
    except:
        titles.append(' ')
    post_meta=soup.find("div", class_="post-meta")
    try:
        movie_link=post_meta.find('a', href=True)['href']
        movie_linkss.append(movie_link)
    except:
        movie_links.append(' ')
    meta_one=soup.findAll('span', class_='meta-one')
    try:
        movie_type=meta_one[0].text.split()
        movie_types.append(movie_type)
    except:
        movie_types.append(' ')
    #x=meta_one[1].text.split()
    try:
        x=meta_one[1].text.split()
        time_of_uplo=' '.join(x)
        time_of_uplos.append(time_of_uplo)
    except:
        time_of_uplos.append(' ')
    try:
        movie_length=meta_one[2].text.split()
        movie_lengths.append(movie_length)
    except:
        movie_lengths.append(' ')
    try:
        num_of_comment=meta_one[3].text.split()
        num_of_comments.append(num_of_comment)
    except:
        num_of_comments.append('0')
    try:
        mo_summary=soup.find('p').next_element
        mo_summarys.append(mo_summary)
    except:
        mo_summarys.append(' ')
    try:
        block=soup.find('blockquote', class_='quote-content')
        y=block.findAll('p')

        try:
            Genre=y[1].text.split(':')[1:]
            Genres.append(Genre)
        except:
            Genres.append('missing')
        try:
            Release_Date=y[2].text
            Release_Dates.append(Release_Date)
```

```
        except:
            Release_Dates.append('missing')

        try:
            Stars=y[3].text.split(':')[1:]
            Starss.append(Stars)
        except:
            Starss.append('missing')

        try:
            Language=y[5].text.split(':')[1:]
            Languages.append(Language)
        except:
              Languages.append('missing')
        try:
            Subtitle=y[6].text.split(':')[1:]
            Subtitles.append(Subtitle)
        except:
            Subtitles.append('missing')

    except:
        Genres.append('missing')
        Release_Dates.append('missing')
        Starss.append('missing')
        Languages.append('missing')
        Subtitles.append('missing')


    try:
        imdb_link=block.find('a', href=True)['href']
        imdb_links.append(imdb_link)
    except:
         imdb_links.append('missing')
```

Loading: 100% 4040/4040 [34:06<00:00, 1.90it/s]

Creating a table of all data with pandas dataframe

In [11]:
```
df=pd.DataFrame({"titles":titles,
                 "movie_types":movie_types,
                 "time_of_uplos":time_of_uplos,
                 "movie_lengths":movie_lengths,
                 "num_of_comments":num_of_comments,
                 "Genres":Genres,
                 "Release_Dates":Release_Dates,
                 "Starss":Starss,
                 "Languages":Languages,
                 "Subtitles":Subtitles,
                 "movie_linkss":movie_linkss,
                 "imdb_links":imdb_links,
                 "mo_summarys":mo_summarys,
                })
```

save data in cvs and excel format

In [12]:
```
df.to_csv('netnaija.csv')
df.to_excel('netnaija.xlsx')
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4040 entries, 0 to 4039
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   titles           4040 non-null   object
 1   movie_types      4040 non-null   object
 2   time_of_uplos    4040 non-null   object
 3   movie_lengths    4040 non-null   object
 4   num_of_comments  4040 non-null   object
 5   Genres           4040 non-null   object
 6   Release_Dates    4040 non-null   object
 7   Starss           4040 non-null   object
 8   Languages        4040 non-null   object
 9   Subtitles        4040 non-null   object
 10  movie_linkss     4040 non-null   object
 11  imdb_links       4040 non-null   object
 12  mo_summarys      4040 non-null   object
dtypes: object(13)
memory usage: 410.4+ KB
```

In [15]:
```
1 df.head(8)
```

Out[15]:

	titles	movie_types	time_of_uplos	movie_lengths	num_of_comments	Genres	Relea
0	Kuttey (2023) [Indian]	[Movies]	Mar 20	[01:48:51]	[68]	[Action, Comedy, Crime, Thriller]	Rele Jar
1	Bad City (2022) [Japanese]	[Movies]	Mar 17	[01:58:04]	[50]	[Action, Crime]	Rele Jar
2	In His Shadow (2023) [French]	[Movies]	Mar 17	[01:29:56]	[24]	[Crime, Drama, Family, Thriller]	Rele Mar
3	Noise (2023) [Dutch]	[Movies]	Mar 17	[01:30:34]	[6]	[Drama, Mystery, Thriller]	Rele Mar
4	Boston Strangler (2023)	[Movies]	Mar 17	[01:52:18]	[17]	[Crime, Drama, History, Thriller]	Rele Mar (Unit
5	Vaathi (2023) [Indian]	[Movies]	Mar 17	[05:00:52]	[38]	[Action, Comedy, Drama, Romance]	Rele Fel
6	Haunted Universities 2 (2022) [Thai]	[Movies]	Mar 15	[02:03:59]	[28]	[Comedy, Horror, Thriller]	Rele Mar
7	The Lake (2022) [Thai]	[Movies]	Mar 15	[01:44:20]	[55]	[Drama, Horror, Sci-Fi, Thriller]	Rele Aug