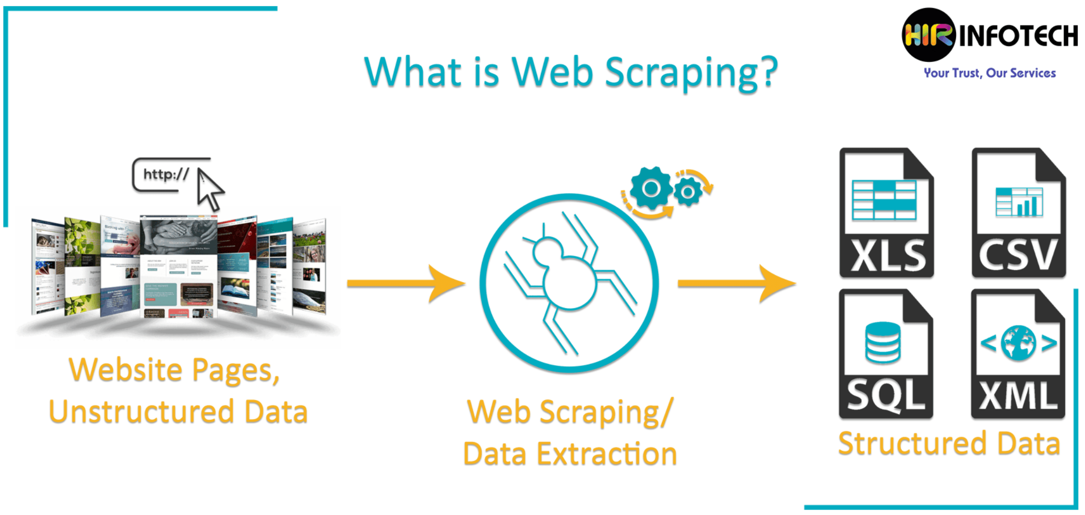
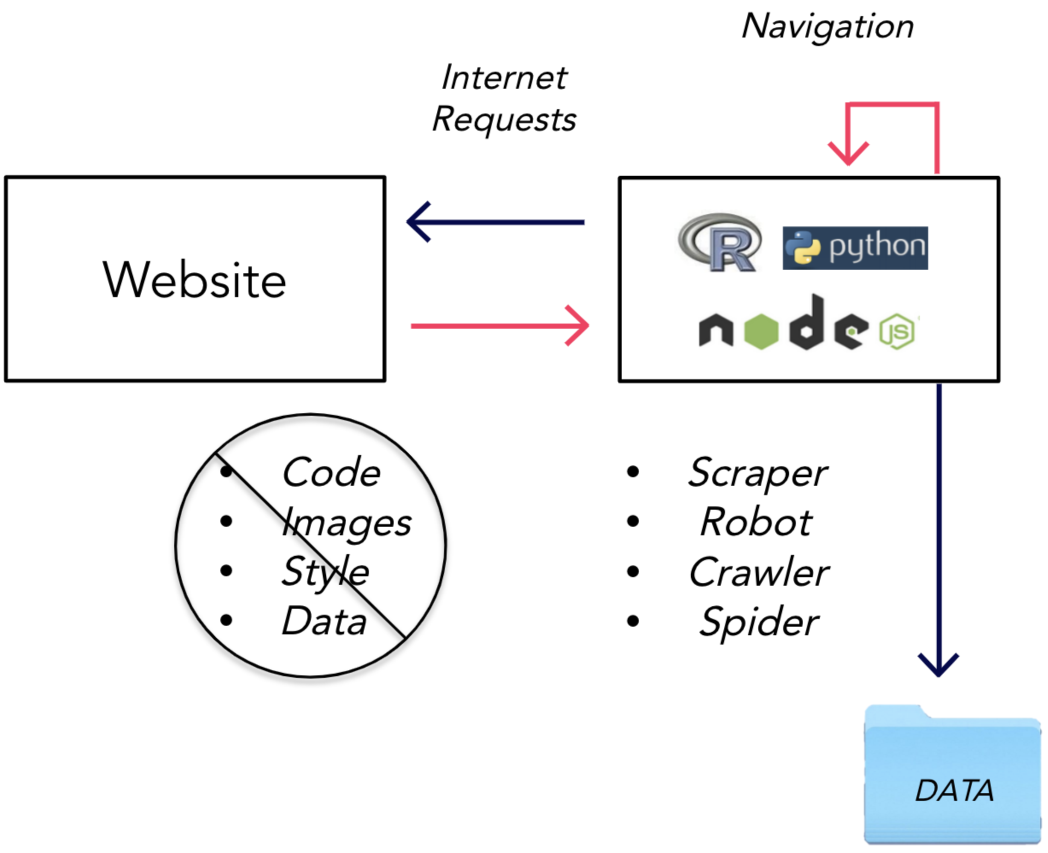
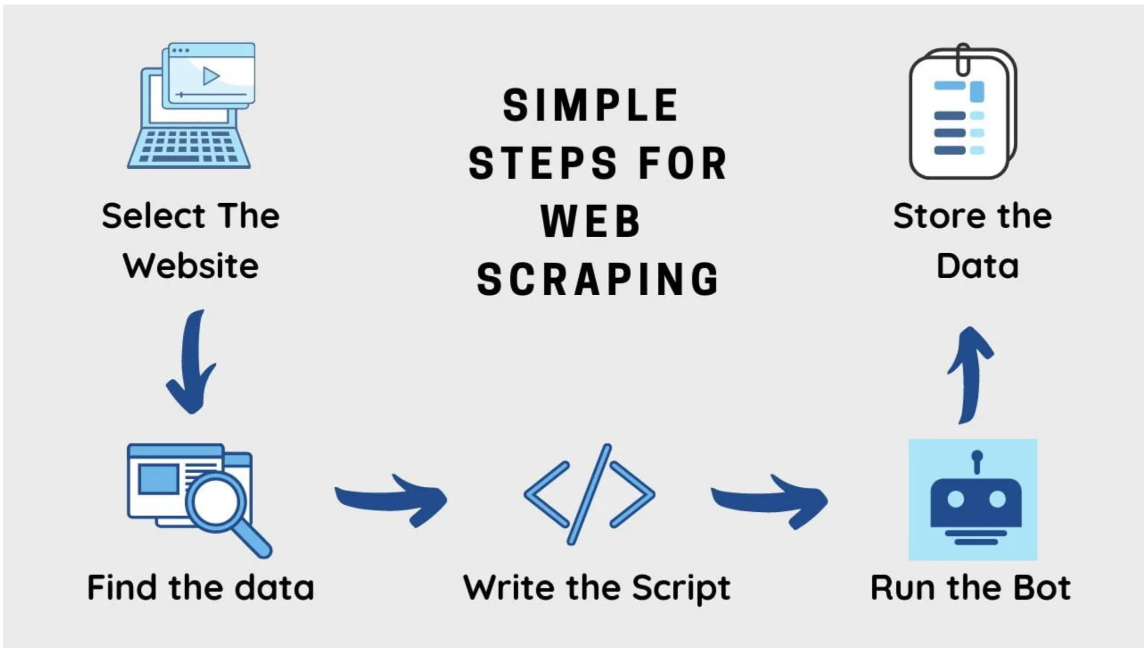
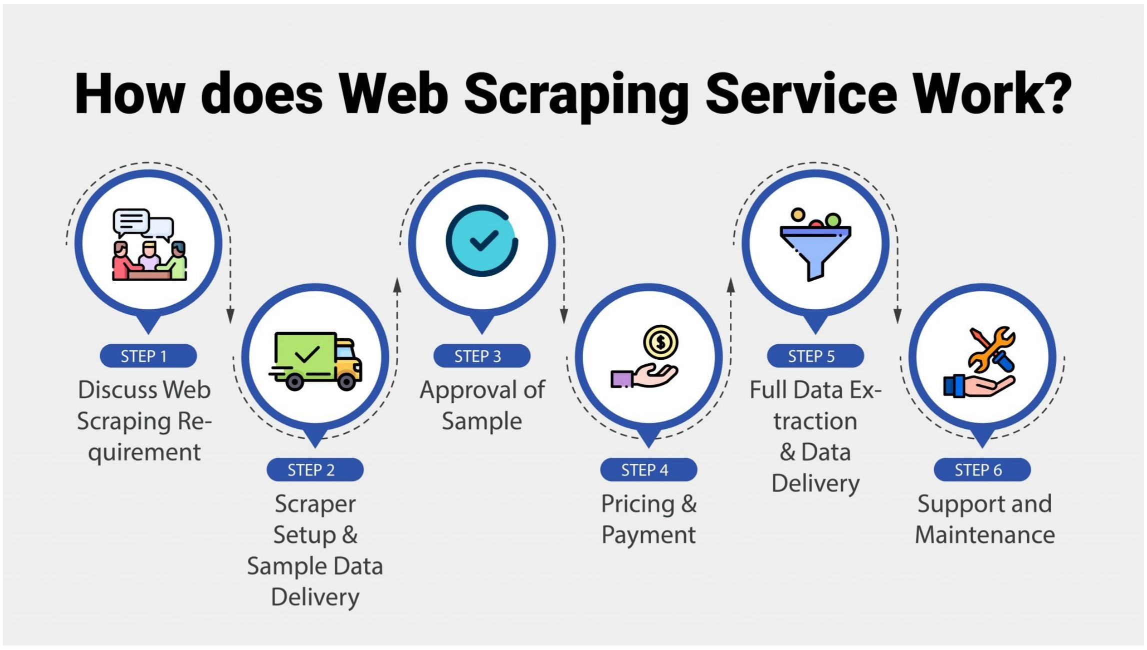
Project - Web Scraping





Scraping Workflow




How does Web Scraping Service Work?
STEP 1 Discuss Web Scraping Requirement
STEP 2 Scraper Setup & Sample Data Delivery
STEP 3 Approval of Sample
STEP 4 Pricing & Payment
STEP 5 Full Data Extraction & Data Delivery
STEP 6 Support and Maintenance

Project 1 - Scraping Topics and Repositories on GitHub

> ### Project Outline:
>
> - Scraping https://github.com/topics
> - Grabing Topic Title, Topic Description, Topic URL to Pandas and CSV file.
> - Grabing Username, Repository name, Stars and Repository URL to Pandas and CSV file.

> ### Project Libraries:
>
> - *import numpy as np*
> - *import pandas as pd*
> - *import requests*

- *from bs4 import BeautifulSoup*

Project Design Thinking

1. Using the requests library to download web pages
2. Using Beautiful Soup to parse and extract information
3. Converting information to Pandas dataframe
4. Creating CSV file with the extracted information

Project Outcome 1 -> Creating Mutiple files

In [1]:
```
import numpy as np
import pandas as pd
import requests
from bs4 import BeautifulSoup
import os

BASE_URL = 'https://github.com'

def scrape_webpage_html(wp_url):
    res = requests.get(wp_url)
    if res.status_code != 200:
        raise Exception(f'Fail to load {wp_url}')
        return
    os.makedirs('data', exist_ok=True)
    if os.path.exists('data/yt_feed_trending.csv'):
        print('Files existed, Skipping...')
    with open('data/github_topic.html', 'w', encoding='utf-8') as f:
        f.write(res.text)
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

def scrape_topic_title(wp_url):
    soup = scrape_webpage_html(wp_url)
    selector = 'f3 lh-condensed mb-0 mt-1 Link--primary'
    tags = soup.find_all('p', class_=selector)
    topic_title_list = []
    for tag in tags:
        topic_title_list.append(tag.text)
    return topic_title_list

def scrape_topic_desc(wp_url):
    soup = scrape_webpage_html(wp_url)
    selector = 'f5 color-fg-muted mb-0 mt-1'
    tags = soup.find_all('p', class_=selector)
    topic_desc_list = []
    for tag in tags:
        topic_desc_list.append(tag.text.strip())
    return topic_desc_list

def scrape_topic_url(wp_url):
    soup = scrape_webpage_html(wp_url)
    selector = 'no-underline flex-1 d-flex flex-column'
    tags = soup.find_all('a', class_=selector)
    topic_link_list = []
    for tag in tags:
        topic_link_list.append(BASE_URL+tag['href'])
    return topic_link_list

def scrape_topics(wp_url):
    dict = {
        'Topic Title': scrape_topic_title(wp_url),
        'Topic Description': scrape_topic_desc(wp_url),
        'Topic URL': scrape_topic_url(wp_url)
    }
    df_topic = pd.DataFrame(dict)
    return df_topic

def scrape_repo_info(wp_url, repo_name, topic_title):
    soup = scrape_webpage_html(wp_url)
    selector_repo = 'f3 color-fg-muted text-normal lh-condensed'
    repo_tags = soup.find_all('h3', class_=selector_repo)
    selector_star = 'Counter js-social-count'
    star_tags = soup.find_all('span', class_=selector_star)
    name_list = []
    repo_list = []
    repo_url_list = []
    star_list = []
    for i in range(len(repo_tags)):
        name = repo_tags[i].find_all('a')[0].text.strip()
        repo = repo_tags[i].find_all('a')[1].text.strip()
        repo_url = repo_tags[i].find_all('a')[1]['href']
        name_list.append(name)
        repo_list.append(repo)
        repo_url_list.append(BASE_URL+repo_url )
        star_list.append(star_tags[i]['title'])
        dict = {
            'Topic Title': topic_title,
            'Name': name_list,
            'Reposity': repo_list,
            'stars': star_list,
            'Reposity URL': repo_url_list
        }
    df_repo = pd.DataFrame(dict)
    os.makedirs('data', exist_ok=True)
    if os.path.exists('data/GitHub_topic_repo.csv'):
        print(f'File is exsited. Skipping...')
        return
    print(f'data/{topic_title}.csv is creating...')
    df_repo.to_csv(f'data/{topic_title}.csv', index=False)
    print(f'data/{topic_title}.csv was done.')

def scrape_topic_repo(wp_rul):
    df_topic = scrape_topics(wp_rul)
    for index, row in df_topic.iterrows():
        scrape_repo_info(row['Topic URL'], row['Topic Title'], topic_title=row['Top
```

In [2]:

```
scrape_topic_repo('https://github.com/topics')
```

```
data/3D.csv is creating...
data/3D.csv was done.
data/Ajax.csv is creating...
data/Ajax.csv was done.
data/Algorithm.csv is creating...
data/Algorithm.csv was done.
data/Amp.csv is creating...
data/Amp.csv was done.
data/Android.csv is creating...
data/Android.csv was done.
data/Angular.csv is creating...
data/Angular.csv was done.
data/Ansible.csv is creating...
data/Ansible.csv was done.
data/API.csv is creating...
data/API.csv was done.
data/Arduino.csv is creating...
data/Arduino.csv was done.
data/ASP.NET.csv is creating...
data/ASP.NET.csv was done.
data/Atom.csv is creating...
data/Atom.csv was done.
data/Awesome Lists.csv is creating...
data/Awesome Lists.csv was done.
data/Amazon Web Services.csv is creating...
data/Amazon Web Services.csv was done.
data/Azure.csv is creating...
data/Azure.csv was done.
data/Babel.csv is creating...
data/Babel.csv was done.
data/Bash.csv is creating...
data/Bash.csv was done.
data/Bitcoin.csv is creating...
data/Bitcoin.csv was done.
data/Bootstrap.csv is creating...
data/Bootstrap.csv was done.
data/Bot.csv is creating...
data/Bot.csv was done.
data/C.csv is creating...
data/C.csv was done.
data/Chrome.csv is creating...
data/Chrome.csv was done.
data/Chrome extension.csv is creating...
data/Chrome extension.csv was done.
data/Command line interface.csv is creating...
data/Command line interface.csv was done.
data/Clojure.csv is creating...
data/Clojure.csv was done.
data/Code quality.csv is creating...
data/Code quality.csv was done.
data/Code review.csv is creating...
data/Code review.csv was done.
data/Compiler.csv is creating...
data/Compiler.csv was done.
data/Continuous integration.csv is creating...
data/Continuous integration.csv was done.
data/COVID-19.csv is creating...
data/COVID-19.csv was done.
data/C++.csv is creating...
data/C++.csv was done.
```

Project Outcome 2 -> Merging to one file

In [3]:
```
import numpy as np
import pandas as pd
import requests
from bs4 import BeautifulSoup
import os

BASE_URL = 'https://github.com'

def scrape_webpage_html(wp_url):
    res = requests.get(wp_url)
    if res.status_code != 200:
        raise Exception(f'Fail to load {wp_url}')
        return
    os.makedirs('data', exist_ok=True)
    if os.path.exists('data/yt_feed_trending.csv'):
        print('Files existed, Skipping...')
    with open('data/github_topic.html', 'w', encoding='utf-8') as f:
        f.write(res.text)
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup

def scrape_topic_title(wp_url):
    soup = scrape_webpage_html(wp_url)
    selector = 'f3 lh-condensed mb-0 mt-1 Link--primary'
    tags = soup.find_all('p', class_=selector)
    topic_title_list = []
    for tag in tags:
        topic_title_list.append(tag.text)
    return topic_title_list

def scrape_topic_desc(wp_url):
    soup = scrape_webpage_html(wp_url)
    selector = 'f5 color-fg-muted mb-0 mt-1'
    tags = soup.find_all('p', class_=selector)
    topic_desc_list = []
    for tag in tags:
        topic_desc_list.append(tag.text.strip())
    return topic_desc_list

def scrape_topic_url(wp_url):
    soup = scrape_webpage_html(wp_url)
    selector = 'no-underline flex-1 d-flex flex-column'
    tags = soup.find_all('a', class_=selector)
    topic_link_list = []
    for tag in tags:
        topic_link_list.append(BASE_URL+tag['href'])
    return topic_link_list

def scrape_topics(wp_url):
    dict = {
        'Topic Title': scrape_topic_title(wp_url),
        'Topic Description': scrape_topic_desc(wp_url),
        'Topic URL': scrape_topic_url(wp_url)
    }
    df_topic = pd.DataFrame(dict)
    return df_topic
```

```
def scrape_repo_info(wp_url, repo_name, topic_title):
    soup = scrape_webpage_html(wp_url)
    selector_repo = 'f3 color-fg-muted text-normal lh-condensed'
    repo_tags = soup.find_all('h3', class_=selector_repo)
    selector_star = 'Counter js-social-count'
    star_tags = soup.find_all('span', class_=selector_star)
    name_list = []
    repo_list = []
    repo_url_list = []
    star_list = []
    for i in range(len(repo_tags)):
        name = repo_tags[i].find_all('a')[0].text.strip()
        repo = repo_tags[i].find_all('a')[1].text.strip()
        repo_url = repo_tags[i].find_all('a')[1]['href']
        name_list.append(name)
        repo_list.append(repo)
        repo_url_list.append(BASE_URL+repo_url )
        star_list.append(star_tags[i]['title'])
        dict = {
            'Topic Title': topic_title,
            'Name': name_list,
            'Reposity': repo_list,
            'stars': star_list,
            'Reposity URL': repo_url_list
        }
        df_repo = pd.DataFrame(dict)
    return df_repo

def scrape_topic_repo(wp_rul):
    df = pd.DataFrame()
    df_topic = scrape_topics(wp_rul)
    for index, row in df_topic.iterrows():
        df_repo = scrape_repo_info(row['Topic URL'], row['Topic Title'], topic_titl
        df = pd.concat([df, df_repo])
    os.makedirs('data', exist_ok=True)
    if os.path.exists('data/GitHub_topic_repo.csv'):
        print(f'File is exsited. Skipping...')
        return
    print('data/GitHub_topic_repo.csv is creating...')
    df.to_csv('data/GitHub_topic_repo.csv', index=False)
    print('data/GitHub_topic_repo.csv was done.')
```

In [4]:
```
scrape_topic_repo('https://github.com/topics')
```

```
data/GitHub_topic_repo.csv is creating...
data/GitHub_topic_repo.csv was done.
```

Project Testing and Validation

In [5]:
```
ml_repo = scrape_repo_info('https://github.com/topics/machine-learning', 'machine-l
ml_repo
```

Out[5]:

	Topic Title	Name	Reposity	stars	Reposity URL
0	machine-learning	tensorflow	tensorflow	168,436	https://github.com/tensorflow/tensorflow
1	machine-learning	huggingface	transformers	72,387	https://github.com/huggingface/transformers
2	machine-learning	pytorch	pytorch	59,641	https://github.com/pytorch/pytorch
3	machine-learning	keras-team	keras	56,418	https://github.com/keras-team/keras
4	machine-learning	scikit-learn	scikit-learn	51,737	https://github.com/scikit-learn/scikit-learn
5	machine-learning	tesseract-ocr	tesseract	47,050	https://github.com/tesseract-ocr/tesseract
6	machine-learning	Developer-Y	cs-video-courses	46,141	https://github.com/Developer-Y/cs-video-courses
7	machine-learning	ageitgey	face_recognition	46,116	https://github.com/ageitgey/face_recognition
8	machine-learning	microsoft	ML-For-Beginners	42,526	https://github.com/microsoft/ML-For-Beginners
9	machine-learning	deepfakes	faceswap	42,497	https://github.com/deepfakes/faceswap
10	machine-learning	aymericdamien	TensorFlow-Examples	42,302	https://github.com/aymericdamien/TensorFlow-Ex...
11	machine-learning	binhnguyennus	awesome-scalability	41,329	https://github.com/binhnguyennus/awesome-scala...
12	machine-learning	JuliaLang	julia	40,646	https://github.com/JuliaLang/julia
13	machine-learning	Avik-Jain	100-Days-Of-ML-Code	38,634	https://github.com/Avik-Jain/100-Days-Of-ML-Code
14	machine-learning	d2l-ai	d2l-zh	35,691	https://github.com/d2l-ai/d2l-zh
15	machine-learning	iperov	DeepFaceLab	35,403	https://github.com/iperov/DeepFaceLab
16	machine-learning	BVLC	caffe	32,921	https://github.com/BVLC/caffe
17	machine-learning	ultralytics	yolov5	31,761	https://github.com/ultralytics/yolov5
18	machine-learning	GokuMohandas	Made-With-ML	31,158	https://github.com/GokuMohandas/Made-With-ML
19	machine-learning	fengdu78	Coursera-ML-AndrewNg-Notes	26,278	https://github.com/fengdu78/Coursera-ML-Andrew...

Project 2 - Crawl IMDB Top 250 and randomly select a movie

In [1]:
```python
import random
import requests
from bs4 import BeautifulSoup

# Crawl IMDB Top 250 and randomly select a movie

URL = 'http://www.imdb.com/chart/top'

def main():
    response = requests.get(URL)

    #soup = BeautifulSoup(response.text, 'html.parser')
    soup = BeautifulSoup(response.text, 'lxml')

    #print(soup.prettify)

    movietags = soup.select('td.titleColumn')
    inner_movietags = soup.select('td.titleColumn a')
    ratingtags = soup.select('td.posterColumn span[name=ir]')

    def get_year(movie_tag):
        moviesplit = movie_tag.text.split()
        year = moviesplit[-1]
        return year

    years = [get_year(tag) for tag in movietags]
    actors_list = [tag['title'] for tag in inner_movietags]
    titles = [tag.text for tag in inner_movietags]
    ratings = [float(tag['data-value']) for tag in ratingtags]

    n_movies = len(titles)

    idx = random.randrange(0, n_movies)

    print(f'{titles[idx]} {years[idx]}, Rating: {ratings[idx]:.1f}, Starring: {acto


if __name__=='__main__':
    main()
```

```
腦筋急轉彎 (2015), Rating: 8.1, Starring: Pete Docter (dir.), Amy Poehler, Bill Had
er
```

In [2]:
```python
import random
import requests
from bs4 import BeautifulSoup

# Crawl IMDB Top 250 and randomly select a movie

URL = 'http://www.imdb.com/chart/top'

def main():
    response = requests.get(URL)

    #soup = BeautifulSoup(response.text, 'html.parser')
    soup = BeautifulSoup(response.text, 'lxml')

    #print(soup.prettify)

    movietags = soup.select('td.titleColumn')
    inner_movietags = soup.select('td.titleColumn a')
    ratingtags = soup.select('td.posterColumn span[name=ir]')

    def get_year(movie_tag):
        moviesplit = movie_tag.text.split()
        year = moviesplit[-1]
        return year

    years = [get_year(tag) for tag in movietags]
    actors_list = [tag['title'] for tag in inner_movietags]
    titles = [tag.text for tag in inner_movietags]
    ratings = [float(tag['data-value']) for tag in ratingtags]

    n_movies = len(titles)


    while(True):
        idx = random.randrange(0, n_movies)

        print(f'{titles[idx]} {years[idx]}, Rating: {ratings[idx]:.1f}, Starring: 

        user_input = input('Do you want another movie (y/[n])?')

        if user_input.lower() != 'y':
            break


if __name__=='__main__':
    main()
```

```
銀翼殺手 (1982), Rating: 8.1, Starring: Ridley Scott (dir.), Harrison Ford, Rutger
Hauer

黑暗騎士：黎明昇起 (2012), Rating: 8.3, Starring: Christopher Nolan (dir.), Christia
n Bale, Tom Hardy

唐人街 (1974), Rating: 8.1, Starring: Roman Polanski (dir.), Jack Nicholson, Faye D
unaway

站在我這邊 (1986), Rating: 8.0, Starring: Rob Reiner (dir.), Wil Wheaton, River Pho
enix

熱情如火 (1959), Rating: 8.2, Starring: Billy Wilder (dir.), Marilyn Monroe, Tony C
urtis

全面啟動 (2010), Rating: 8.7, Starring: Christopher Nolan (dir.), Leonardo DiCapri
o, Joseph Gordon-Levitt
```

Future PLan

> - Using a REST API to retrieve data as JSON
> - Crawling Websites(Scraping Multiple Pages)

> - **`Designing Data Anaylis Project`**
>   -> `Exploratory Analysis` -- *Numpy, Pandas*
>   -> `Visualization` -- *Matploylib, seaborn, plotly*
>   -> `Linking to Power BI`
> - **`Designing Machine Learning Project`**
>   -> `Exploratory Analysis` -- *Numpy, Pandas*
>   -> `Visualization` -- *Matploylib, seaborn, plotly*
>   -> `Model Selections`
>   -> `Linking to Power BI`

-- Memo END --