Web Scraping

Web scraping is the process of extracting specific data on a targeted webpages.

or

we write some code to fetch data from websites in an automated fashion..

web crawling:

is like search engine,it goes through different webpages without specific goal.

Websites with dynamic content(changes with user need) cannot be scraped using BeautifulSoup. One way to scrape dynamic website is by using Selenium.

1. Pick a website and describe your objective

- Browse through different sites and pick on to scrape. Check the "Project Ideas" section for inspiration.
- Identify the information you'd like to scrape from the site. Decide the format of the output CSV file.
- Summarize your project idea and outline your strategy in a Juptyer notebook. Use the "New" button above.

2. Use the requests library to download web pages

- Inspect the website's HTML source and identify the right URLs to download.
- Download and save web pages locally using the requests library
- Create a function to automate downloading for different topics/search queries.

3.Use Beautiful Soup to parse and extract information

- Parse and explore the structure of downloaded web pages using Beautiful soup.
- Use the right properties and methods to extract the required information.
- Create functions to extract from the page into lists and dictionaries.
- (Optional) Use a REST API to acquire additional information if required. #### 4.Create CSV file(s) with the extracted information
- Create functions for the end-to-end process of downloading, parsing, and saving CSVs.
- Execute the function with different inputs to create a dataset of CSV files.
- Verify the information in the CSV files by reading them back using Pandas.

5.Document and share your work

Scraping-Github-topics-repositories

Problem Statement:

Find top 20 repositories in each github topics of 3D,ajax etc..

Before jumping directly for coding its better to prepare output by hands using sheets.new. then we can get basic idea of how final outcome looks as like.

1. Pick a website and describe your objective

- Browse through different sites and pick on to scrape. Check the "Project Ideas" section for inspiration.
- Identify the information you'd like to scrape from the site. Decide the format of the output CSV file.
- Summarize your project idea and outline your strategy in a Juptyer notebook. Use the "New" button above.

Project Outline:

- we're going to scrape github topics https://github.com/topics
- we'll get a list of topics. For each topic, we'll get topic title, topic url & topic description
- for each topic,pick top 25 repositories in the topic from the topic page
- for each repository we grab repo name,username,star,url.
- for each topic we'll create a csv file in the following format:

```
Repo Name,username,stars,repo url
three.js,mrdoob,93.4,https://github.com/mrdoob/three.js
react-three-
fiber,pmndrs,23.3,https://github.com/pmndrs/react-three-
fiber
```

2. Use the requests library to download web pages

In [1]:
```
import requests
```

In [2]:
```
from bs4 import BeautifulSoup
```

3.Use Beautiful Soup to parse and extract information

In [3]:
```
pip install --upgrade pip
```

```
Requirement already satisfied: pip in /opt/anaconda3/lib/python3.9/site-packages (23.2.1)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
pip install beautifulsoup4 --upgrade --quiet
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
import pandas as pd
```

In [6]:
```
def urls():
    url='https://github.com/topics'
    response=requests.get(url)
    content=response.text
    doc=BeautifulSoup(content,'lxml')
    p_tag=doc.find_all('p',class_='f3 lh-condensed mb-0 mt-1 Link--primary')
    des_tag=doc.find_all('p',class_='f5 color-fg-muted mb-0 mt-1')
    url_tag=doc.find_all('a','no-underline flex-1 d-flex flex-column')
    def topic_info(p_tag,url_tag,des_tag):
        repo_url='https://github.com'+url_tag['href']
        des_info=des_tag.text.strip()
        for i in p_tag:
            name=i.text.strip()
        return name,repo_url,des_info
    topics={'Topic name':[],'Description':[],'url':[]}
    for i in range(len(url_tag)):
        topic_link=topic_info(p_tag[i],url_tag[i],des_tag[i])
        topics['Topic name'].append(topic_link[0])
        topics['Description'].append(topic_link[2])
        topics['url'].append(topic_link[1])
    return pd.DataFrame(topics)
```

In [7]:
```
urls().head()
```

Out[7]:

	Topic name	Description	url
0	3D	3D refers to the use of three-dimensional grap...	https://github.com/topics/3d
1	Ajax	Ajax is a technique for creating interactive w...	https://github.com/topics/ajax
2	Algorithm	Algorithms are self-contained sequences that c...	https://github.com/topics/algorithm
3	Amp	Amp is a non-blocking concurrency library for ...	https://github.com/topics/amphp
4	Android	Android is an operating system built by Google...	https://github.com/topics/android

In [8]:
```
for index, row in urls().iterrows():
    print(row["Topic name"], row["url"])
```

```
3D https://github.com/topics/3d
Ajax https://github.com/topics/ajax
Algorithm https://github.com/topics/algorithm
Amp https://github.com/topics/amphp
Android https://github.com/topics/android
Angular https://github.com/topics/angular
Ansible https://github.com/topics/ansible
API https://github.com/topics/api
Arduino https://github.com/topics/arduino
ASP.NET https://github.com/topics/aspnet
Atom https://github.com/topics/atom
Awesome Lists https://github.com/topics/awesome
Amazon Web Services https://github.com/topics/aws
Azure https://github.com/topics/azure
Babel https://github.com/topics/babel
Bash https://github.com/topics/bash
Bitcoin https://github.com/topics/bitcoin
Bootstrap https://github.com/topics/bootstrap
Bot https://github.com/topics/bot
C https://github.com/topics/c
Chrome https://github.com/topics/chrome
Chrome extension https://github.com/topics/chrome-extension
Command line interface https://github.com/topics/cli
Clojure https://github.com/topics/clojure
Code quality https://github.com/topics/code-quality
Code review https://github.com/topics/code-review
Compiler https://github.com/topics/compiler
Continuous integration https://github.com/topics/continuous-integration
COVID-19 https://github.com/topics/covid-19
C++ https://github.com/topics/cpp
```

Getting information out of a topic_url

In [9]:
```
import os
def sub_topic_link(topic_url):
    response=requests.get(topic_url)
    if response.status_code!=200:
        raise Exception('Failed to load page {}'.format(topic_url))
    content=BeautifulSoup(response.text,'lxml')
    h_tag=content.find_all('h3','f3 color-fg-muted text-normal lh-condensed')
    star_tag=content.find_all('span',class_='Counter js-social-count')
    def star(star_tag):
        star_tag.strip()
        if star_tag[-1]=='k':
            return int(float(star_tag[:-1])*1000)
        return int(star_tag)
    def sub_topic_details(h_tag,star_tag):
        a_tag=h_tag.find_all('a',class_='Link')
        username=a_tag[0].text.strip()
        repo_name=a_tag[1].text.strip()
        repo_url='https://github.com'+a_tag[1]['href']
        star_count=star(star_tag.text.strip())
        return username,repo_name,star_count,repo_url
    topics={'user':[],'repository':[],'rating':[],'url':[]}
    for i in range(len(h_tag)):
        topic_details=sub_topic_details(h_tag[i],star_tag[i])
        topics['user'].append(topic_details[0])
        topics['repository'].append(topic_details[1])
        topics['rating'].append(topic_details[2])
        topics['url'].append(topic_details[3])
    return pd.DataFrame(topics)

#help(os.path)

#help(os.path.exists)

def save_topics(topic_url,topic_name): #topic_name--path
    if os.path.exists(topic_name):
        print('The file {} already exists. Skip..'.format(topic_name))
        return
    df=sub_topic_link(topic_url)
    df.to_csv(topic_name,index=None)
```

In [10]:
```
def scrape_topics_infos():
    print('scraping Topics')
    topic_df=urls()

    os.makedirs('Scraping Data',exist_ok=True)

    for index,row in topic_df.iterrows():
        print('Scraping Top repositories of "{}"'.format(row['Topic name']))
        save_topics(row['url'],'Scraping Data/{}.csv'.format(row['Topic name']))
    print('Scraping Done.')
```

In [11]:
```
scrape_topics_infos()
```

```
scraping Topics
Scraping Top repositories of "3D"
Scraping Top repositories of "Ajax"
Scraping Top repositories of "Algorithm"
Scraping Top repositories of "Amp"
Scraping Top repositories of "Android"
Scraping Top repositories of "Angular"
Scraping Top repositories of "Ansible"
Scraping Top repositories of "API"
Scraping Top repositories of "Arduino"
Scraping Top repositories of "ASP.NET"
Scraping Top repositories of "Atom"
Scraping Top repositories of "Awesome Lists"
Scraping Top repositories of "Amazon Web Services"
Scraping Top repositories of "Azure"
Scraping Top repositories of "Babel"
Scraping Top repositories of "Bash"
Scraping Top repositories of "Bitcoin"
Scraping Top repositories of "Bootstrap"
Scraping Top repositories of "Bot"
Scraping Top repositories of "C"
Scraping Top repositories of "Chrome"
Scraping Top repositories of "Chrome extension"
Scraping Top repositories of "Command line interface"
Scraping Top repositories of "Clojure"
Scraping Top repositories of "Code quality"
Scraping Top repositories of "Code review"
Scraping Top repositories of "Compiler"
Scraping Top repositories of "Continuous integration"
Scraping Top repositories of "COVID-19"
Scraping Top repositories of "C++"
Scraping Done.
```

In [ ]:

In [ ]:

In [ ]: