scraping-github-topic-repositories

Scraping the top repositories for Topic on Github

TODO (Introduction): -Introduction about web scraping -Introduction about Github and problem statement -Mention the tools you're using(Python,requests,BeautifulSoup,Pandas)

Here are the steps we follow: Project outline: -we,re going to scrape the page https://github.com/topics -we will get list of topic,for each topic ,we will get topic title,topic page URL and topic description -For each topic ,we'll get the top 25 repositories in the topic from the topic page -For each repository we'll grab the repo name,user name,stars and URL -for each topic we will create a csv file in the format:

Repo Name,Username,Stars,Repo URL three.js,mrdoob,97300,https://github.com/mrdoob/three.js libgdx,libgdx,22500,https://github.com/libgdx

Scrape the list of topics from Github

Explain how you'll do it.

- use requests to download the page
- user BS4 to parse and extract information
- convert to a Pandas dataframe

Let's write a function to download the page

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

def get_topics_page():
    topics_url = 'https://github.com/topics'
    #Dowload the page
    response = requests.get(topics_url)
    #Check successful response
    if response.status_code != 200:
        raise Exception('Failed to load page {}'.format(topic_url))
    #Parse using BeautifulSoup
    doc = BeautifulSoup(response.text, 'html.parser')
    return doc
```

Add some explanation

In [ ]:
```
doc = get_topics_page()
```

In [ ]:
```
doc.find('a')
```

Let's create some helper functions to parse informations from the page. To get topic titles,we can pick 'p' tags with the 'class' ...

In [ ]:
```
def get_topic_titles(doc):
    selection_class = 'f3 lh-condensed mb-0 mt-1 Link--primary'
    topic_title_tags = doc.find_all('p', {'class': selection_class})
    topic_titles = []
    for tag in topic_title_tags:
        topic_titles.append(tag.text)
    return topic_titles
```

get_topic_titles can be used to get the list of titles

In [ ]:
```
titles = get_topic_titles(doc)
```

In [ ]:
```
len(titles)
```

In [ ]:
```
titles[:5]
```

Similarly we have defined functions for descriptions and URLs

In [ ]:
```
def get_topic_description(doc):
    desc_selector ='f5 color-fg-muted mb-0 mt-1'
    topic_desc_tags = doc.find_all('p',{'class':desc_selector})


    topic_descs = []
    for tag in topic_desc_tags :
            topic_descs.append(tag.text.strip())
    return topic_descs
```

TODO - example and explanation

In [ ]:
```
def get_topic_urls(doc):
    topic_link_tags = doc.find_all('a',{'class': 'no-underline flex-1 d-flex flex-column'})

    topic_urls = []
    base_url='https://github.com'
    for tag in topic_link_tags:
        topic_urls.append(base_url + tag['href'])
    return topic_urls
```

Let's this put all together into a single function

In [ ]:
```
def scrape_topics():
    topics_url = 'https://github.com/topics'
    response = requests.get(topics_url)
    if response.status_code != 200:
        raise Exception('Failed to load page {}'.format(topic_url))
    doc = BeautifulSoup(response.text, 'html.parser')
    topics_dict = {
        'title': get_topic_titles(doc),
        'description': get_topic_descs(doc),
        'url': get_topic_urls(doc)
    }
    return pd.DataFrame(topics_dict)
```

Get the top repositories from topic page

TODO-Explanation and step

In [ ]:
```
def get_topics_page():
    #Dowload the page
    response = requests.get(topics_url)
    #Check successful response
    if response.status_code != 200:
        raise Exception('Failed to load page {}'.format(topic_url))
    #Parse using BeautifulSoup
    doc = BeautifulSoup(response.text, 'html.parser')
    return doc
```

In [ ]:
```
doc = get_topics_page("https://github.com/topics/3d")
```

TODO - talk about the h3 tags

In [ ]:
```
def get_repo_info(h3_tag, star_tag):
    #returns all the required info about repository
    a_tags = h3_tag.find_all('a')
    username = a_tags[0].text.strip()
    repo_name = a_tags[1].text.strip()
    repo_url = base_url + a_tags[1]['href']
    stars = parse_star_count(star_tags[0].text.strip())
    return username, repo_name, stars,repo_url
```

TODO-show a example

In [ ]:
```
def get_topic_repos(topic_doc):

        #get the h3 tag containing repo title,repo URL and username
        h3_selection_class = 'f3 color-fg-muted text-normal lh-condensed'
        repo_tags = topic_doc.find_all('h3',{'class': h3_selection_class})
        #get star tags
        star_tags = topic_doc.find_all('span',{'class':'Counter js-social-count'})

        topic_repos_dict = { 'username':[],'repo_name':[],'stars':[],'repo_url':[]}


        #get repo info
        for i in range(len(repo_tags)):
            repo_info = get_repo_info(repo_tags[i],star_tags[i])
            topic_repos_dict['username'].append(repo_info[0])
            topic_repos_dict['repo_name'].append(repo_info[1])
            topic_repos_dict['stars'].append(repo_info[2])
            topic_repos_dict['repo_url'].append(repo_info[3])

        return pd.DataFrame(topic_repos_dict)##Pandas dataframe
```

TODO-show a example

In [ ]:
```
def scrape_topic(topic_url,path):
    if os.path.exists(path):
        print("The file {} already exists.Skipping...".format(path))
    topic_df = get_topic_repos(get_topic_page(topic_url))
    topic_df.to_csv(path)
```

TODO-show a example

In [ ]:
```

```

Putting it all together

-We have a funciton to get the list of topics -We have a function to create a CSV file for scraped repos from a topics page -Let's create a function to put them together

In [ ]:
```
def scrape_topics():
    topic_url = 'https://github.com/topics'
    response = requests.get(topics_url)
    if response.status_code !=200:
            raise Exception('failed to load page{}'.format(topic_url))
    doc = BeautifulSoup(response.text,'html.parser')
    topic_dict = {
        'title': get_topic_titles(doc),
        'description': get_topic_description(doc),
        'url':get_topic_urls(doc)
    }
    return pd.DataFrame(topics_dict)
```

In [ ]:
```
##megafunction we put everything in single function for taking infos
def scrape_topic_repos():
    print('Scraping list of topics')
    topic_df = scrape_topics()
    #Create a folder

    os.makedirs('data',exist_ok=True)

    for index,row in topic_df.iterrows():
        print('Scraping top Repositories for "{}" '.format(row['title']))
        scrape_topic(row['url'], 'data/{}.csv'.format(row['title']))
```

In [ ]:
```

```

Let's run it to scrape the top repos for the all the topics on the first page of https://github.com/topics

In [ ]:
```
scrape_topic_repos()
```

We can check that the CSVs were created properly

In [ ]:
```
#read and display  a CSV using pandas
```

In [ ]:
```

```

In [ ]:
```

```

Reference and Future Work

Summary :

-scraped top repositories from github. -Used BeautifulSoup to extract. -Converted extracted Datas into Pandas Dataframe.

References:

-https://www.crummy.com/software/BeautifulSoup/bs4/doc/ -https://www.geeksforgeeks.org/create-a-pandas-dataframe-from-lists/ -https://github.com/topics