import libraries

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

scrape topics

get username, repo name, repo url and stars

- extract username, repo_name, repo_url and stars from a given tag

In [2]:
```
def get_repo_info(repo_tag, star_tag):
    base_url = "https://github.com"
    atags = repo_tag.find_all('a')
    username = atags[0].text.strip()
    repo_name = atags[1].text.strip()
    repo_url = base_url + atags[1]['href']
    stars = parse_star_tag(star_tag.text.strip())
    return username, repo_name, stars, repo_url
```

- parse stars from "8K" to 8000

In [3]:
```
def parse_star_tag(star):
    if star[-1]=='k':
        return int(float(star[:-1])*1000)
    else:
        return int(star)
```

- create a data frame that contains
- username | repo_name | stars | repo_url

In [4]:
```
def create_repo_info_df(repo_tags, star_tags):
    topic_repos_dict ={
                       'username':[],
                       'repo_name':[],
                       'stars':[],
                       'repo_url':[]
                      }
    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 topic_repos_dict
```

- scrape the url and get the required tags
- use functions above to parse stars and create dataframe

In [5]:
```
def get_topic_repos(topic_url):
    #download the page
    response = requests.get(topic_url)
    #check response
    if response.status_code != 200:
        raise Exception("Failed to load page {}").format(topic_url)
    #parse using Beautiful soup
    topic_doc =  BeautifulSoup(response.text, 'html.parser')

    # get h2 tags containing repo title, url and username
    h1_class = "f3 color-fg-muted text-normal lh-condensed"
    repo_tags = topic_doc.find_all('h3', class_=h1_class)
    # get stars tags
    star_class = 'repo-stars-counter-star'
    star_tags = topic_doc.find_all('span', {'id':star_class})

    topic_repos_dict = create_repo_info_df(repo_tags, star_tags)

    return pd.DataFrame(topic_repos_dict)
```

For every topic create a csv with topic name, description and url

- given topic html extract titles, description and urls

In [6]:
```
def get_topic_titles(doc):
    selection_class = "f3 lh-condensed mb-0 mt-1 Link--primary"
    topic_title_p_tags = doc.find_all('p',
                          {'class': selection_class})
    topic_titles = [ tag.text for tag in topic_title_p_tags]
    return topic_titles

def get_topic_desc(doc):
    selection_class = "f5 color-fg-muted mb-0 mt-1"
    topic_desc_p_tags = doc.find_all('p', class_=selection_class)
    topic_desc = [ tag.text.strip() for tag in topic_desc_p_tags]
    return topic_desc

def get_topic_urls(doc):
    selection_class = "no-underline flex-1 d-flex flex-column"
    url_a_tag = doc.find_all('a', class_=selection_class)
    base_url = "https://github.com"
    topic_urls = [base_url+url['href'] for url in url_a_tag]
    return topic_urls
```

- loop through topic and scrape titles, description, urls
- create data frame that contains
- title | description | url

In [7]:
```
def scrape_topics():
    topic_url = 'https://github.com/topics'
    #download the page
    response = requests.get(topic_url)
    #check response
    if response.status_code != 200:
        raise Exception("Failed to load page {}".format(topic_url))
    base_url = "https://github.com"
    page_content = response.text

    doc = BeautifulSoup(page_content, 'html.parser')


    topic_titles = get_topic_titles(doc)
    topic_desc = get_topic_desc(doc)
    topic_urls = get_topic_urls(doc)

    topic_dict = {"title":topic_titles,
                  "description":topic_desc,
                  "url": topic_urls}

    topics_df = pd.DataFrame(topic_dict)
    return topics_df
```

bring it together

- create csv of the scraped topic

In [8]:
```
def scrape_topic(topic_url, topic_name):
    topic_repo_df = get_topic_repos(topic_url)
    fname = "github_scrape/" + topic_name + '.csv'

    if os.path.exists(fname):
        print("The file {} already exists. Skipping..".format(fname))
        return

    topic_repo_df.to_csv(fname, index = None)
```

- loop through all topics and use funvtions above to create the csv files

In [9]:
```
def scrape_topics_repos():
    print('Scraping list of topics')
    topics_df = scrape_topics()

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

    for index, row in topics_df.iterrows():
        print('scraping top repositories for "{}"'.format(row['title']))
        scrape_topic(row['url'], row['title'])
```

call the function for scraping

In [10]:
```
scrape_topics_repos()
```

```
Scraping list of topics
scraping top repositories for "3D"
The file github_scrape/3D.csv already exists. Skipping..
scraping top repositories for "Ajax"
The file github_scrape/Ajax.csv already exists. Skipping..
scraping top repositories for "Algorithm"
The file github_scrape/Algorithm.csv already exists. Skipping..
scraping top repositories for "Amp"
The file github_scrape/Amp.csv already exists. Skipping..
scraping top repositories for "Android"
The file github_scrape/Android.csv already exists. Skipping..
scraping top repositories for "Angular"
The file github_scrape/Angular.csv already exists. Skipping..
scraping top repositories for "Ansible"
The file github_scrape/Ansible.csv already exists. Skipping..
scraping top repositories for "API"
The file github_scrape/API.csv already exists. Skipping..
scraping top repositories for "Arduino"
The file github_scrape/Arduino.csv already exists. Skipping..
scraping top repositories for "ASP.NET"
The file github_scrape/ASP.NET.csv already exists. Skipping..
scraping top repositories for "Atom"
The file github_scrape/Atom.csv already exists. Skipping..
scraping top repositories for "Awesome Lists"
The file github_scrape/Awesome Lists.csv already exists. Skipping..
scraping top repositories for "Amazon Web Services"
The file github_scrape/Amazon Web Services.csv already exists. Skipping..
scraping top repositories for "Azure"
The file github_scrape/Azure.csv already exists. Skipping..
scraping top repositories for "Babel"
The file github_scrape/Babel.csv already exists. Skipping..
scraping top repositories for "Bash"
The file github_scrape/Bash.csv already exists. Skipping..
scraping top repositories for "Bitcoin"
The file github_scrape/Bitcoin.csv already exists. Skipping..
scraping top repositories for "Bootstrap"
The file github_scrape/Bootstrap.csv already exists. Skipping..
scraping top repositories for "Bot"
The file github_scrape/Bot.csv already exists. Skipping..
scraping top repositories for "C"
The file github_scrape/C.csv already exists. Skipping..
scraping top repositories for "Chrome"
The file github_scrape/Chrome.csv already exists. Skipping..
scraping top repositories for "Chrome extension"
The file github_scrape/Chrome extension.csv already exists. Skipping..
scraping top repositories for "Command line interface"
The file github_scrape/Command line interface.csv already exists. Skipping..
scraping top repositories for "Clojure"
The file github_scrape/Clojure.csv already exists. Skipping..
scraping top repositories for "Code quality"
The file github_scrape/Code quality.csv already exists. Skipping..
scraping top repositories for "Code review"
The file github_scrape/Code review.csv already exists. Skipping..
scraping top repositories for "Compiler"
The file github_scrape/Compiler.csv already exists. Skipping..
scraping top repositories for "Continuous integration"
The file github_scrape/Continuous integration.csv already exists. Skipping..
scraping top repositories for "COVID-19"
The file github_scrape/COVID-19.csv already exists. Skipping..
scraping top repositories for "C++"
```

future work

get more topics by looping through

https://github.com/topics?page= + i -> i from 1 to 7

In [ ]:
```

```