Python-Web scraping

Web scraping is the process of gathering data from the web. As a novice in data science, you've probably come across CSV files released online by well-known websites like Kaggle and other governmental websites. Either traditional methods of data collection and writing are used to prepare the data, or information is scraped from the Internet. I'll walk you through web scraping using Python and Beautiful Soup in this article. I'll also use the Pandas library to create a data frame and later convert it into a CSV file.

1)Importing libraries and extracting various kinds of data.

```
import requests
from bs4 import BeautifulSoup

#Make A GET Request
r = requests.get('https://lingarajtechhub.com/python-object-oriented-programming/')

soup = BeautifulSoup(r.content,'html.parser')

print(soup.prettify())

print(soup.title)#title tag

print(soup.title.name)#getting the name of the tag

print(soup.title.parent.name) #getting name of the parent tag


#Finding Elements (.class, #id name) #paragraph
import requests
from bs4 import BeautifulSoup

#Make A GET Request
r = requests.get('https://lingarajtechhub.com/what-is-object-oriented-programming-in-python/')

soup = BeautifulSoup(r.content,'html.parser')

s = soup.find('main',class_='site-main hfeed')

content = s.find_all('p')

print(content)
```

```
get all the images of a particular web page using the
coresponding url and the following library
"""

import re #regular expression
import requests
from bs4 import BeautifulSoup

site = 'https://lingarajtechhub.com/python-object-oriented-programming/'

response = requests.get(site)

soup = BeautifulSoup(response.text, 'html.parser')
img_tags = soup.find_all('img')

urls = [img['src'] for img in img_tags]


for url in urls:
    filename = re.search(r'/([\w_-]+[.](jpg|gif|png|webp))$', url)
    if not filename:
         print("Regex didn't match with the url: {}".format(url))
         continue
    with open(filename.group(1), 'wb') as f:
        if 'http' not in url:
            # sometimes an image source can be relative
            # if it is provide the base url which also happens
            # to be the site variable atm.
            url = '{}{}'.format(site, url)
        response = requests.get(url)
        f.write(response.content)
```

```
#Finding Elements ('image src download')
import requests
from bs4 import BeautifulSoup

#Make A GET Request
r = requests.get('https://lingarajtechhub.com/what-is-object-oriented-programming-in-python/')

soup = BeautifulSoup(r.content,'html.parser')

s = soup.find('div',class_='ct-image-container')

content = s.find('img')

ig = content['src']

#download the image
img_data = requests.get(ig)
img_data.raise_for_status()

#save the image to an file
with open('ig.jpg', 'wb') as handler:
    handler.write(img_data.content)
```

```
#Finding Elements (Extract The Links)
import requests
from bs4 import BeautifulSoup

#Make A GET Request
r = requests.get('https://lingarajtechhub.com/what-is-object-oriented-programming-in-python/')

soup = BeautifulSoup(r.content,'html.parser')

s = soup.find('div',class_='hero-section')

content = s.find_all('a')

print(content)


#Finding Elements ("More Links")
import requests
from bs4 import BeautifulSoup

#Make A GET Request
r = requests.get('https://lingarajtechhub.com/what-is-object-oriented-programming-in-python/')

soup = BeautifulSoup(r.content,'html.parser')

s = soup.find('div',class_='hero-section')

for a in s.find_all('a', href=True):
    print("Found the URL:", a['href'])
```

The codes here demonstrating the use of the requests library to make HTTP requests (GET, POST, PUT/PATCH, DELETE) to web pages and the Beautiful Soup library to parse the HTML content of the web pages and extract information such as titles, paragraphs, links, and images. The code also includes an example of downloading all the images from a web page using regular expressions to extract the image URLs.

2) Scraping from Flipkart Website

To get idea on scraping a little bit more let's start with an easy flipkart website scrape.

```
##scrapping datas of samsung mobile more than 30 thousand from all the 23 pages.

import pandas as pd
import requests
from bs4 import BeautifulSoup

Product_Name = []
Prices = []
Description = []
Reviews = []

for i in range(2, 25):  #for the number of pages we want to scrap the data.
    try:
        url = "https://www.flipkart.com/search?q=mobiles+above+30000&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=off&as=off&p%5B%5D=facets.brand%255B%255D%3DSAMSUNG&page=" + str(i)
        r = requests.get(url)
        r.raise_for_status()  # Raise an exception for non-2xx response codes
        soup = BeautifulSoup(r.text, 'html.parser')
        particular = soup.find('div', class_="_1YokD2 _3Mn1Gg")

        n_o_p = particular.find_all('div', class_='_4rR01T')
        for i in n_o_p:
            n = i.text
            Product_Name.append(n)

        get_price = particular.find_all('div', class_='_30jeq3 _1_WHN1')
        for i in get_price:
            n = i.text
            Prices.append(n)

        get_desc = particular.find_all('ul', class_='_1xgFaf')
        for i in get_desc:
            n = i.text
            Description.append(n)

        o_p_r = particular.find_all('div', '_3LWZlK')
        for i in o_p_r:
            n = i.text
            Reviews.append(n)

        if len(o_p_r) < len(n_o_p):
            missing_reviews = len(n_o_p) - len(o_p_r)
            for _ in range(missing_reviews):
                Reviews.append("No reviews available")
    except requests.exceptions.RequestException as e:
        print("Error occurred while requesting the page:", e)

df = pd.DataFrame({"Products": Product_Name, "Prices": Prices, "Description": Description, "Reviews": Reviews})

df.to_csv("E:/python programs/Python_Project/Samsung_phones_above_30K.csv")

print("Data scraping and CSV export completed successfully.")
```

This code is scraping data from the Flipkart website for Samsung phones above 30,000 rupees and storing the product name, price, description, and reviews in separate lists. It then creates a pandas data frame from these lists and exports it to a CSV file. The code is using the requests library to make HTTP requests to the website and the Beautiful Soup library to parse the HTML response.

3) Utility tool for Flipkart web scraping

From the perspective of software engineering, utility tool is simply the process of writing a programme (such as a stand-alone binary, script, library function, class method, etc.) to carry out some specific, very well-defined function(s). It can be very simple and only take a few hours, or it can be extremely difficult and take a few months.

So, what does the utility tool that I have made do? It takes any product name as user input for example say phones, t-shirts, accessories for bikes or anything and scrape the related data into a data frame and later on into a csv file.

As the codes are too long, I am mentioning URL to check the codes

URL - https://github.com/Kai1817/Python_WebScraping/blob/main/utility_tool_webscraping_flipkart.py

So, we infer from the above codes in the given link that, the function scrapes product information from Flipkart website based on the user inputted product name and returns a pandas dataframe.

goods_name: The name of the product that the user wants to search for on Flipkart

return: The function `scrape_flipkart_product` returns a pandas DataFrame containing information about the products searched for on Flipkart, including the product name, price, description, and reviews.

after searching the products name, you get the desired results. If you want to save the product's details that you have got as the result in the form of a csv file, then follow the same process as mentioned in the above section.