**WEB SCRAPING**

## **WORKSHEET – 1**

**In Q1 to Q9, only one option is correct, Choose the correct option:**

1. Which of the following extracts information from user generated content?

A) Java script tagging B) Web scraping

C) A/B testing D) MROCs

1. Which of the following is not a web scraping library in python?

A) selenium B) Beautiful soup

C) Requests C) scrapy

1. Selenium tests \_\_\_\_\_\_\_\_\_\_?

A) Browser based applications B) DOS applications

C) GUI applications D) All of the above

1. Task of crawling is performed by a complex software which is known as:

A) Scraper B) Crawler

C) Boat D) Spider

1. Which of the following commands is used to access name of a tag in Beautiful Soup?

A) tag.attrs B) tag.name

C) tag,id C) tag[‘id’]

1. Which of the following is the default parser in Beautiful Soup?

A) html.parser B) html5lib

C) lxml D) lxml-xml

1. In selenium the webdriver is used to?

A) design a test using selenese

B) test a web application on firefox only

C) execute tests on HtmlUnit browser

D) to download any content from a webpage

1. In selenium, driver**.find\_elements\_by\_xpath(‘given xpath’)** returns:

A) the first webelement associated with the ‘given xpath’

B) the url of first webelement associated with the ‘given xpath’

C) the list of all webelements associated with the ‘given xpath’

D) all the attributes of the first webelement associated with the ‘given xpath’

1. The script **‘window.scrollBy(0,a)** scrolls the webpage by?

A) **‘a’** number of horizontal spaces

B) **‘a’** number of lines

C) **‘a’** number of pixels horizontally

D) **‘a’** number of pixels vertically

**In Q10, more than one options are correct, Choose all the correct options:**

1. Which of the following is(are) tags of HTML?

A) <a> B) <b>

C) <image> D) <href>

**Q10 to Q13 are subjective answer type questions, Answer them briefly.**

1. What is the main difference between a web scraper and a web crawler?

**Difference between Web Scraping and Web Crawling :**

| **S.NO.** | **WEB SCRAPING** | **WEB CRAWLING** |
| --- | --- | --- |
|  |  |  |
| 1. | The tool used is **Web Scraper**. | The tool used **Web Crawler** or **Spiders**. |
| 2. | It is used for **downloading** information | It is used for **indexing** of Web pages |
| 3. | It **need not visit all the pages** of website for infomation. | It **visits each and every page**, until the last line for information. |
| 4. | A Web Scraper **doesn’t obey robots.txt** in most of the cases. | It always **obeys robots.txt**. |
| 5. | It is done on both **small and large scale**. | It is mostly employed in **large scale**. |
| 6. | Application areas include Retail Marketing, Equity search and Machine learning. | Used in search engines to give search results to the user. |
| 7. | Data de-duplication is not necessarily a part of Web Scraping. | Data de-duplication is and integral part of Web Scraping. |
| 8. | This **needs crawl agent and a parser** for parsing the response. | This only **needs only crawl agent**. |
| 9. | ProWebScraper, Web Scraper.io are the examples | Google, Yahoo or Bing do Web Crawling |

1. What is **‘robots.txt’** file? What is the use of **‘robots.txt’** file?

A robots.txt file tells search engine crawlers which pages or files the crawler can or can't request from your site. This is used mainly to avoid overloading your site with requests; **it is not a mechanism for keeping a web page out of Google.** To keep a web page out of Google, you should use [noindex directives](https://developers.google.com/webmasters/control-crawl-index/docs/robots_meta_tag" \t "_blank), or password-protect your page.

## **What is robots.txt used for?**

robots.txt is used primarily to manage crawler traffic to your site, and usually to keep a page off Google, depending on the file type:

| **Page Type** | **Traffic management** | **Hide from Google** | **Description** |
| --- | --- | --- | --- |
| **Web page** | **✔** | **✗** | For web pages (HTML, PDF, or other [non-media formats that Google can read](https://support.google.com/webmasters/answer/35287)), robots.txt can be used to manage crawling traffic if you think your server will be overwhelmed by requests from Google's crawler, or to avoid crawling unimportant or similar pages on your site.  **You should not use robots.txt as a means to hide your web pages from Google Search results.** This is because, if other pages point to your page with descriptive text, your page could still be indexed without visiting the page. If you want to block your page from search results, use another method such as password protection or a [noindex](https://developers.google.com/webmasters/control-crawl-index/docs/robots_meta_tag" \t "_blank) directive.  **If your web page is blocked with a robots.txt file**, it can still appear in search results, but the search result will not have a description and look [something like this](https://support.google.com/webmasters/answer/7489871). Image files, video files, PDFs, and other non-HTML files will be excluded. If you see this search result for your page and want to fix it, remove the robots.txt entry blocking the page. If you want to hide the page completely from search, use [another method](https://support.google.com/webmasters/answer/6062608?hl=en#robotted-but-indexed). |
| **Media file** | **✔** | **✔** | Use robots.txt to manage crawl traffic, and also to prevent image, video, and audio files from appearing in Google search results. (Note that this won't prevent other pages or users from linking to your image/video/audio file.)   * [Read more about preventing images from appearing on Google.](https://support.google.com/webmasters/answer/35308) * [Read more about preventing video files from appearing on Google.](https://support.google.com/webmasters/answer/156442#block_video) |
| **Resource file** | **✔** | **✔** | You can use robots.txt to block resource files such as unimportant image, script, or style files, **if you think that pages loaded without these resources will not be significantly affected by the loss**. However, if the absence of these resources make the page harder for Google's crawler to understand the page, you should not block them, or else Google won't do a good job of analyzing pages that depend on those resources. |

1. What are static and dynamic web pages?

on the basis of type of response sent to the browser we can classify this response in two categories one is Static web page and other is Dynamic web page.

Following are the important differences between Static Web Page and Dynamic Web Page.

| **Sr. No.** | **Key** | **Static Web Page** | **Dynamic Web Page** |
| --- | --- | --- | --- |
| 1 | Definition | Static web pages are generally simple HTML written pages which serve as response from browser to server in which all the information and data is static in nature and it does not get changed until someone changed it manually. | On other hand Dynamic webpages are the pages written in some more complex language such as ASP.NET in which data is rendered after some interpretation and capacity to produce distinctive content for different calls. |
| 2 | Complexity | As mentioned in above point as data in static web pages is static and do not require any interpretation before rendering so static web pages are simple in complexity. | Dynamic web pages on other hand does the interpretation process which make data dynamic in nature and due to which dynamic web pages become complex in complexity as compare to static web pages. |
| 3 | Language used | Static web pages are generally written in simpler languages such as HTML, JavaScript, CSS, etc. | On other Dynamic web pages are written in more complex languages such as CGI, AJAX, ASP, ASP.NET, etc. |
| 4 | Rendered Data | For static web pages data do not changes until someone changes it manually and hence data is static in nature. | On other hand for Dynamic web page data is first interoperate at server side and due to which it does not remain same on every call and this makes data dynamic in nature.. |
| 5 | Time | Static web pages due to static data take less time to get load. | While Dynamic web pages due to dynamic data take comparatively more time as compare to static web pages. |
| 6 | Database | In Static web pages generally no involvement of database for data redecoration. | On other hand in case of Dynamic web page database is used for data redecoration. |

**Q14 and Q15 are programming practice questions. Solve it using JUPYTER NOTEBOOK and paste the solution in your answer sheets.**

1. Write a python program to check whether a webpage contains a title or not.

from urllib.request import urlopen

from urllib.error import HTTPError

from bs4 import BeautifulSoup

def getTitle(url):

try:

html = urlopen(url)

except HTTPError as e:

return None

try:

bsObj = BeautifulSoup(html.read(), "lxml")

title = bsObj.body.h1

except AttributeError as e:

return None

return title

title = getTitle(url)

if title == None:

return "Title could not be found"

else:

return title

print(getTitle("https://www.facebook.com/"))

print(getTitle("http://www.example.com/"))

1. Write a python program to access the search bar and search button on images.google.com.

import selenium

from selenium import webdriver

import time

import requests

import os

from PIL import Image

import io

import hashlib

# This is the path I use

DRIVER\_PATH = '/Users/hp/chromedriver'

def fetch\_image\_urls(query:str, max\_links\_to\_fetch:int, wd:webdriver, sleep\_between\_interactions:int=1):

def scroll\_to\_end(wd):

wd.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(sleep\_between\_interactions)

# build the google query

search\_url = "https://www.google.com/search?safe=off&site=&tbm=isch&source=hp&q={q}&oq={q}&gs\_l=img"

# load the page

wd.get(search\_url.format(q=query))

image\_urls = set()

image\_count = 0

results\_start = 0

while image\_count < max\_links\_to\_fetch:

scroll\_to\_end(wd)

# get all image thumbnail results

thumbnail\_results = wd.find\_elements\_by\_css\_selector("img.Q4LuWd")

number\_results = len(thumbnail\_results)

print(f"Found: {number\_results} search results. Extracting links from {results\_start}:{number\_results}")

for img in thumbnail\_results[results\_start:number\_results]:

# try to click every thumbnail such that we can get the real image behind it

try:

img.click()

time.sleep(sleep\_between\_interactions)

except Exception:

continue

# extract image urls

actual\_images = wd.find\_elements\_by\_css\_selector('img.n3VNCb')

for actual\_image in actual\_images:

if actual\_image.get\_attribute('src') and 'http' in actual\_image.get\_attribute('src'):

image\_urls.add(actual\_image.get\_attribute('src'))

image\_count = len(image\_urls)

if len(image\_urls) >= max\_links\_to\_fetch:

print(f"Found: {len(image\_urls)} image links, done!")

break

else:

print("Found:", len(image\_urls), "image links, looking for more ...")

time.sleep(30)

return

load\_more\_button = wd.find\_element\_by\_css\_selector(".mye4qd")

if load\_more\_button:

wd.execute\_script("document.querySelector('.mye4qd').click();")

# move the result startpoint further down

results\_start = len(thumbnail\_results)

return image\_urls

def persist\_image(folder\_path:str,file\_name:str,url:str):

try:

image\_content = requests.get(url).content

except Exception as e:

print(f"ERROR - Could not download {url} - {e}")

try:

image\_file = io.BytesIO(image\_content)

image = Image.open(image\_file).convert('RGB')

folder\_path = os.path.join(folder\_path,file\_name)

if os.path.exists(folder\_path):

file\_path = os.path.join(folder\_path,hashlib.sha1(image\_content).hexdigest()[:10] + '.jpg')

else:

os.mkdir(folder\_path)

file\_path = os.path.join(folder\_path,hashlib.sha1(image\_content).hexdigest()[:10] + '.jpg')

with open(file\_path, 'wb') as f:

image.save(f, "JPEG", quality=85)

print(f"SUCCESS - saved {url} - as {file\_path}")

except Exception as e:

print(f"ERROR - Could not save {url} - {e}")

if \_\_name\_\_ == '\_\_main\_\_':

wd = webdriver.Chrome(executable\_path=DRIVER\_PATH)

queries = ["CORONA-VIRUS","oneplus 8"]

for query in queries:

wd.get('https://google.com')

search\_box = wd.find\_element\_by\_css\_selector('input.gLFyf')

search\_box.send\_keys(query)

links = fetch\_image\_urls(query,2,wd)

images\_path = 'E:/scraping\_demo/scraped data'

for i in links:

persist\_image(images\_path,query,i)

wd.quit()