Web Scraping Using Selenium

References

https://gist.github.com/korakot/5c8e21a5af63966d80a676af0ce15067

https://www.selenium.dev/documentation/webdriver/

https://www.selenium.dev/documentation/support_packages/working_with_select_elements/

https://selenium-python.readthedocs.io/getting-started.html#simple-usage

https://www.guru99.com/xpath-selenium.html

(ctrl) + (shift) + (j) to open chrome develop tools

```
!apt update
!apt install chromium-chromedriver
!pip install selenium
!apt-get install -y chromium-browser
```

```
The following additional packages will be installed:
  apparmor chromium-browser libfuse3-3 liblzo2-2 libudev1 snapd squashfs-tools systemd-hwe-hwdb
  udev
Suggested packages:
  apparmor-profiles-extra apparmor-utils fuse3 zenity | kdialog
The following NEW packages will be installed:
  apparmor chromium-browser chromium-chromedriver libfuse3-3 liblzo2-2 snapd squashfs-tools
  systemd-hwe-hwdb udev
The following packages will be upgraded:
  libudev1
1 upgraded, 9 newly installed, 0 to remove and 46 not upgraded.
Need to get 28.5 MB of archives.
After this operation, 118 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 apparmor amd64 3.0.4-2ubuntu2.3 [595 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 liblzo2-2 amd64 2.10-2build3 [53.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 squashfs-tools amd64 1:4.5-3build1 [159 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libudev1 amd64 249.11-0ubuntu3.12 [78.2 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 udev amd64 249.11-0ubuntu3.12 [1,557 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfuse3-3 amd64 3.10.5-1build1 [81.2 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 snapd amd64 2.63+22.04ubuntu0.1 [25.9 MB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 chromium-browser amd64 1:85.0.4183.83-0ubuntu2.22.04.1 [49.2 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 chromium-chromedriver amd64 1:85.0.4183.83-0ubuntu2.22.04.1 [2,3
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 systemd-hwe-hwdb all 249.11.5 [3,228 B]
Fetched 28.5 MB in 1s (19.1 MB/s)
Preconfiguring packages ...
```

```
# set options to be headless, ..
from selenium import webdriver
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
```

```
#import needed libs
from selenium.webdriver.support.select import Select
from selenium.webdriver.common.by import By
import pandas as pd
import time
```

```
# open it, go to a website, and get results
wd = webdriver.Chrome(options=options)

wd.get("https://www.pttor.com/en/oil_price")
```

What is XPath?

XPath stands for XML Path Language
XPath is a major element in the XSLT standard.
XPath can be used to navigate through elements and attributes in an XML document.

How to find elements by using xpath in Selenium:

```
#select a search option in the web page
radio_search_filter = wd.find_element(By.XPATH,'//*[@id="__layout"]/div/div/section/div[2]/div/div/div[1]/div[1]/div[2]/input')
radio_search_filter.click()
```

```
province_filter = wd.find_element(By.XPATH,'//*[@id="__layout"]/div/div/section/div[2]/div/div/div[2]/form/div/select')
province_object = Select(province_filter)
```

```
Start coding or generate with AI.
```

```
<selenium.webdriver.support.select.Select at 0x216e4083eb0>
```

In the field Province, Month and Year, select the values Nonthaburi, January and 2022

```
province_filter = wd.find_element(By.XPATH,'//*[@id="__layout"]/div/div/section/div[2]/div/div/div[2]/form/div/select')
province_object = Select(province_filter)
# al_optios= province_object.options
# for option in al_optios:
#     print(option.text)
province_object.select_by_visible_text("Nonthaburi")
province_filter.send_keys()
time.sleep(2)

month_filter = wd.find_element(By.XPATH,'//*[@id="__layout"]/div/div/section/div[2]/div/div/div[2]/form/div[2]/select')
month_object = Select(month_filter)
month_object.select_by_visible_text("January")
month_filter.send_keys()
time.sleep(2)
```

```
year_filter = wd.find_element(By.XPATH,'//*[@id="__layout"]/div/div/section/div[2]/div/div/div[2]/form/div[3]/select')
year_object = Select(year_filter)
year_object.select_by_visible_text("2022")
year_filter.send_keys()
time.sleep(2)
```

```
#get the data table by class attribute
filter_table = wd.find_element(By.CLASS_NAME,'section-filter__table')


#show data in the table
filter_table.text
```

```
'Date - Time\n28-01-2022 05:00 29.94 26.14 32.84 33.78 34.05 41.46 35.96 -\n25-01-2022 05:00 29.94 25.54 32.24 33.28 33.55 40.96 35.96
-\n20-01-2022 05:00 29.94 24.94 31.64 32.88 33.15 40.56 35.96 -\n14-01-2022 05:00 29.84 24.64 31.14 32.38 32.65 40.06 35.86 -\n11-01-2
022 05:00 29.84 24.34 30.64 31.88 32.15 39.56 35.86 -\n08-01-2022 05:00 29.44 24.14 30.24 31.48 31.75 39.16 35.46 -\n05-01-2022 05:00
```

```
#search with 'tr' tag to get rows from the table
table_rows = filter_table.find_elements(By.TAG_NAME, "tr")


#Because the data table use images as a header
#create a list of table header
oil_type = ["DateTime","diesel","E85","E20","91","95","Benzene","Diesel Gold","Gasohol"]
row_list = []


##Save the value from table data into an array
for rows in table_rows:
    oil_item = list()
    for col in rows.find_elements(By.TAG_NAME, 'td'):
        oil_item.append(col.text)
    # print(oil_item)
    row_list.append(oil_item)



row_list[0]
row_list.pop(0)
```

```
[]
```

```
row_list
```

```
[['28-01-2022 05:00',
  '29.94',
  '26.14',
  '32.84',
  '33.78',
  '34.05',
  '41.46',
  '35.96',
  '-'],
 ['25-01-2022 05:00',
  '29.94',
  '25.54',
  '32.24',
  '33.28',
  '33.55',
  '40.96',
  '35.96',
  '-'],
 ['20-01-2022 05:00',
  '29.94',
  '24.94',
  '31.64',
  '32.88',
  '33.15',
  '40.56',
  '35.96',
  '-'],
 ['14-01-2022 05:00',
  '29.84',
  '24.64',
  '31.14',
  '32.38',
  '32.65',
  '40.06',
  '35.86',
  '-'],
 ['11-01-2022 05:00',
  '29.84',
  '24.34',
  '30.64',
  '31.88',
  '32.15',
```

```
        '39.56',
        '35.86',
        '-'],
       ['08-01-2022 05:00',
        '29.44',
        '24.14',
        '30.24',
        '31.48',
        '31.75',
        '39.16',
        '35.46',
        '-'],
       ['05-01-2022 05:00',
        '29.04',
        '24.14',
        '30.24',
```

```
df = pd.DataFrame(row_list, columns=oil_type)
```

```
print(df)
```

```
            DateTime diesel    E85    E20     91     95 Benzene Diesel Gold  \
0  28-01-2022 05:00  29.94  26.14  32.84  33.78  34.05   41.46       35.96
1  25-01-2022 05:00  29.94  25.54  32.24  33.28  33.55   40.96       35.96
2  20-01-2022 05:00  29.94  24.94  31.64  32.88  33.15   40.56       35.96
3  14-01-2022 05:00  29.84  24.64  31.14  32.38  32.65   40.06       35.86
4  11-01-2022 05:00  29.84  24.34  30.64  31.88  32.15   39.56       35.86
5  08-01-2022 05:00  29.44  24.14  30.24  31.48  31.75   39.16       35.46
6  05-01-2022 05:00  29.04  24.14  30.24  31.48  31.75   39.16       35.06

  Gasohol
0       -
1       -
2       -
3       -
4       -
5       -
6       -
```

```
df = df.drop(columns=["Gasohol"])
```

```
df.isna().sum()
```

	0
DateTime	0
diesel	0
E85	0
E20	0
91	0
95	0
Benzene	0
Diesel Gold	0

dtype: int64

```
wd.quit()
```

Scraping the IMDB Top 250 Movie webpage

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

import time
import pandas as pd
import re

options = webdriver.ChromeOptions()
```

```
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.162 Safar
```

Scrap the top 250 movie data from the IMDB website and create a dataframe with the columns: Title, Year, Duration, MPA Rating, Stars

```
# open it, go to a website, and get results
wd = webdriver.Chrome(options=options)

wd.get("https://www.imdb.com/chart/top/")

imdb_filter = wd.find_element(By.XPATH, '//body/div[2]/main/div/div[3]/section/div/div[2]/div/ul')

list_films = imdb_filter.find_elements(By.TAG_NAME, 'li')


column_names = ['Title', 'Year', 'Duration', 'MPA Rating', 'Stars']
data = []
for i in range(250):
# single_film = list_films[0]
    time.sleep(1)
    row = []
    info_filter = WebDriverWait(wd, 10).until(
        EC.element_to_be_clickable((By.XPATH, './/div[3]/button'))
    )
    info_filter = list_films[i].find_element(By.XPATH, './/div[3]/button')
    # info_filter.click()

    wd.execute_script("arguments[0].click();", info_filter)

    time.sleep(2)
    # Wait for the modal to appear
    description_box = WebDriverWait(wd, 10).until(
        EC.visibility_of_element_located((By.XPATH, '//div[@class="ipc-promptable-base__content"]'))
    )

    # Extract text and process it
    cur_data = description_box.text.split('\n')
    row.append(cur_data[0])
    row.append(int(cur_data[1][:4]))

    last_index = cur_data[1][4:].find('min')
    duration = ''
    if last_index != -1:
        duration = cur_data[1][4:4+last_index+3]

    row.append(duration)
    row.append(float(cur_data[3]))
    # row.append(cur_data[8][6:])
    index = next((i for i, entry in enumerate(cur_data) if entry.startswith('Stars')), None)

    if index is not None:
        stars = cur_data[index][6:]
        stars = re.findall(r'[A-Z][a-z]*\s?[A-Z][a-z]*', stars)
        stars = ', '.join(stars)
    else:
        stars = ''

    row.append(stars)
    data.append(row)

    # Wait for the modal to be visible and clickable
    try:
        # Wait for the close button to become clickable
        close_box = WebDriverWait(wd, 10).until(
            EC.element_to_be_clickable((By.XPATH, '//div[@class="ipc-promptable-base__close"]/button'))
        )

        # # Scroll into view and click using JavaScript if needed
        # wd.execute_script("arguments[0].scrollIntoView(true);", close_box)
        # time.sleep(1)  # Allow some time for scroll to finish
```

```
        # Attempt to click using JavaScript if normal click fails
        wd.execute_script("arguments[0].click();", close_box)
    except Exception as e:
        print(f"An error occurred: {e}")
    # time.sleep(1)
# province_filter.text
# for film in list_films:
#     print(film.text)
#     break


column_names = [['Title', 'Year', 'Duration', 'MPA Rating', 'Stars']]
df = pd.DataFrame(data, columns=column_names)
df
```

	0
Title	0
Year	0
Duration	0
MPA Rating	0
Stars	0

```
df.to_csv('imdb_movie_250_st125457.csv', index=False)
```

Save the data in a dataframe as `imdb_movie_250_{student_id}.csv`. Submit this file as well when submitting the assignment

Some simple ideas to play around with using pandas:

1. Check for any null values in the data frame and filter out those values
2. min, max and average movie rating, count of movie released in each year, count of movie in each MPA Rating

```
#null values in the data frame
df.isna().sum()
```

	0
Title	0
Year	0
Duration	0
MPA Rating	0
Stars	0

```
# min, max and average movie rating
df['MPA Rating'].describe()
```

	MPA Rating
count	250.000000
mean	8.309600
std	0.235222
min	8.000000
25%	8.100000
50%	8.200000
75%	8.400000
max	9.300000

```
# count of movie released in each year
df.Year.value_counts().sort_index()
```

(Year,)	count
1921	1
1924	1
1925	1
1926	1
1927	1
...	...
2020	2
2021	2
2022	1
2023	3
2024	2

87 rows × 1 columns

```
df['MPA Rating'].value_counts()
```

(MPA Rating,)	count
8.1	74
8.2	50
8.3	39
8.4	25
8.5	23
8.6	14
8.7	6
8.8	5
9.0	5
8.0	4
8.9	3
9.2	1
9.3	1

```
wd.quit()
```