Web Scraping using Selenium

Objectives:

- How to use Selenium to login a website
- How to use Selenium to extract HTML
- How to use Selenium to interact with website before extract HTML

Scape NEA Weather Data

Perform extraction of data from NEA website without any interaction on webpage.

- https://www.nea.gov.sg/weather (https://www.nea.gov.sg/weather)

Install Python library `selenium` and `webdriver_manager` using `pip` .

In [2]:
```
!pip install selenium
!pip install webdriver_manager
```

```
Requirement already satisfied: selenium in c:\users\isszq\anaconda3\lib\sit
e-packages (3.141.0)
Requirement already satisfied: urllib3 in c:\users\isszq\anaconda3\lib\site
-packages (from selenium) (1.25.9)
Requirement already satisfied: webdriver_manager in c:\users\isszq\anaconda
3\lib\site-packages (3.2.2)
Requirement already satisfied: requests in c:\users\isszq\anaconda3\lib\sit
e-packages (from webdriver_manager) (2.24.0)
Requirement already satisfied: crayons in c:\users\isszq\anaconda3\lib\site
-packages (from webdriver_manager) (0.4.0)
Requirement already satisfied: configparser in c:\users\isszq\anaconda3\lib
\site-packages (from webdriver_manager) (5.0.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\isszq\anaconda3\lib
\site-packages (from requests->webdriver_manager) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\isszq\anaconda
3\lib\site-packages (from requests->webdriver_manager) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\isszq\anaconda3\lib\site-packages (from requests->webdriver_manage
r) (1.25.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\isszq\anacond
a3\lib\site-packages (from requests->webdriver_manager) (2020.6.20)
Requirement already satisfied: colorama in c:\users\isszq\anaconda3\lib\sit
e-packages (from crayons->webdriver_manager) (0.4.3)
```

Import libraries

In [3]:
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

from webdriver_manager.chrome import ChromeDriverManager
```

2. Extract Data without Interaction

We will demonstrate on how to extract company announcements and company news from SGX website.

Open Website and Get HTML

Get an instance of web browser.

- The `webdriver_manager` provides managers for different browsers. It will download the correct version of driver for your browser.

In [4]:
```
browser = webdriver.Chrome(ChromeDriverManager().install())
```

```
[WDM] - Current google-chrome version is 85.0.4183
[WDM] - Get LATEST driver version for 85.0.4183



[WDM] - There is no [win32] chromedriver for browser 85.0.4183 in cache
[WDM] - Get LATEST driver version for 85.0.4183
[WDM] - Trying to download new driver from http://chromedriver.storage.goog
leapis.com/85.0.4183.87/chromedriver_win32.zip (http://chromedriver.storag
e.googleapis.com/85.0.4183.87/chromedriver_win32.zip)
[WDM] - Driver has been saved in cache [C:\Users\isszq\.wdm\drivers\chromed
river\win32\85.0.4183.87]
```

Use the `browser` object to open a webpage.

In [5]:
```
url = 'https://www.nea.gov.sg/weather'
browser.get(url)
# Wait for 10 seconds before timeout
wait = WebDriverWait(browser, 10)
# Wait until an element is present
wait.until(EC.presence_of_element_located((By.ID, 'fourDayOutlook')))
# Receive cookies and HTML
cookies = browser.get_cookies()
html = browser.page_source
```

Close web browser since we have already gotten the HTML code.

In [9]:
```
1 browser.close()
```

Examine HTML Code and Make Soup

Save the HTML code to a file and examine it. Examine the file to make sure it contains the data which you are interested in.

In [10]:
```
1 with open('_temp.html', 'w') as f:
2     s = html.encode("utf-8")
3     f.write(str(s))
```

Let's "make a soup" from the downloaded HTML code.

In [11]:
```
1 from bs4 import BeautifulSoup
2
3 soup = BeautifulSoup(html)
```

3. Extract "4-day Outlook"

In Chrome, inspect the element of the `4-day Outlook`. It uses a `<div>` tag with `id="fourDayOutlook"`.

Use `soup.find()` to find above element by its tag name.

In [24]:
```
1 outlook = soup.find('div', {'id':"fourDayOutlook"})
2 # print(raw)
```

Each of the 4 elements inside the `4-day Outlook` uses a `<div>` tag with `class="stats-data--4days__item"`.

Use `findAll()` method to find all matching elements.

In [29]:

```
days = outlook.findAll('div', {'class':"stats-data--4days__item"})
print(len(days))
# print(days)
print(days[0])
```

```
4
<div class="stats-data--4days__item">
<div class="icon"><img alt="weather icon" src="/assets/images/icons/weathe
r/pc.png"/></div>
<div class="content">
<div class="weather-4-outlook">
<span class="day">FRI</span>
<span class="info">Partly cloudy.</span>
</div>
<div class="temperature">
<div class="info">
<i class="icon icon-thermometer"></i>
<span>23 - 33°C</span>
</div>
<div class="info">
<i class="icon icon-wind-direction" id="icon_wind_direction" style="transfo
rm:rotate(312deg);-ms-transform:rotate(312deg);"></i>
<span>SSE 15 - 25km/h</span>
</div>
</div>
</div>
</div>
```

In each day, our target data are all in `<span>` .

In [41]:

```
rows = [ [span.text for span in day.findAll('span')] for day in days ]
print(rows)
```

```
[['FRI', 'Partly cloudy.', '23 - 33°C', 'SSE 15 - 25km/h'], ['SAT', 'Pre-da
wn hours and early morning thundery showers.', '24 - 33°C', 'SSE 15 - 25km/
h'], ['SUN', 'Afternoon and evening thundery showers.', '24 - 33°C', 'ENE 1
0 - 20km/h'], ['MON', 'Afternoon thundery showers.', '24 - 34°C', 'NE 5 - 1
5km/h']]
```