**Libraries used for Web Scraping**

As we know, Python is has various applications and there are different libraries for different purposes. In our further demonstration, we will be using the following libraries:

**Selenium**:  Selenium is a web testing library. It is used to automate browser activities

**Pandas:**  pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

**Web Scraping Example : Scraping linkedin Website**

Pre-requisites:

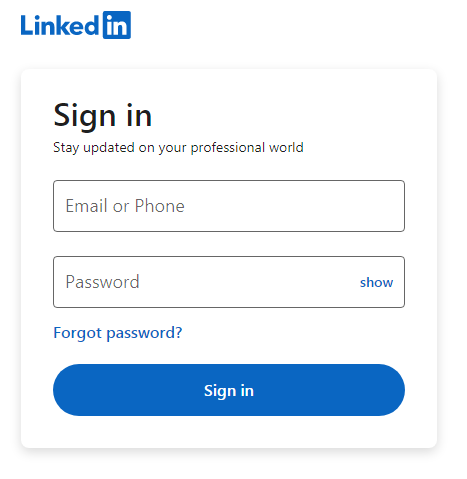
* Python 3.x with **Selenium, pandas** libraries installed
* Google-chrome browser

### ****Step 1: Find the URL that you want to scrape****

For this example, we are going scrape **linkedin** website to extract the Title, Company Name, and Location of Company. The URL for this page is <https://www.linkedin.com/login?fromSignIn=true&trk=guest_homepage-basic_nav-header-signin> and  <https://www.linkedin.com/jobs>

### ****Step 1: Inspecting the page****

### The data is usually nested in tags. So, we inspect the page to see, under which tag the data we want to scrape is nested. To inspect the page, just right click on the element and click on “Inspect”.



When you click on the “Inspect” tab, you will see a “Browser Inspector Box” open.



### ****Step 3: Find the data you want to extract****

Let’s extract the Title, Company Name, and Location which is in the “div” tag respectively.

### ****Step 4: Write the code****

First, let’s create a Python file. To do this, open the file in sublime and type python <my file name> with .py extension.

I am going to name my file “LinkedIn\_data”. Here’s the command:

python LinkedIn\_data.py

Now, let’s write our code in this file.

First, let us import all the necessary libraries:

1. from selenium import \*
2. from selenium import webdriver
3. import pandas as pd
4. import selenium.webdriver.common.keys

To configure webdriver to use Chrome browser, we have to set the path to chromedriver

1. browser = webdriver.Chrome("C:/Users/cyber/Downloads/chromedriver\_win32/chromedriver.exe")

Refer the below code to open the URL:

1. browser.get("https://www.linkedin.com/login?fromSignIn=true&trk=guest\_homepage-basic\_nav-header-signin")
2. username=browser.find\_element\_by\_id("username")
3. username.send\_keys("\*\*\*\*\*\*\*@gmail.com")
4. password=browser.find\_element\_by\_id("password")
5. password.send\_keys("\*\*\*\*\*\*\*")
6. login\_button=browser.find\_element\_by\_class\_name("login\_\_form\_action\_container ")
7. login\_button.click()

Now that we have written the code to open the URL, it’s time to extract the data from the website. As mentioned earlier, the data we want to extract is nested in <div> tags. So, I will find the div tags with those respective class-names, extract the data and store the data in a variable. Refer the code below

1. browser.get("https://www.linkedin.com/jobs")
2. job=browser.find\_elements\_by\_class\_name("job-card-square\_\_title")
3. c=[]
4. for i in job:
5. #print(i.text)
6. c.append(i.text)
7. job\_title=[]#List to store title of the company
8. for i in range(len(c)):
9. job\_title.append(c[i].strip("Job Title\n"))
10. job\_title
11. job2=browser.find\_elements\_by\_class\_name("job-card-container\_\_company-name")
12. comp\_name=[]#List to store company name of the company
13. for i in job2:
14. #print(i.text)
15. comp\_name.append(i.text)
16. job3=browser.find\_elements\_by\_class\_name("job-card-container\_\_metadata-wrapper")
17. loc\_name=[]#List to store location of the company
18. for i in job3:
19. loc\_name.append(i.text)

### ****Step 5: Run the code and extract the data****

To run the code, use the below command:

1. python LinkedIn\_data.py

### ****Step 6: Store the data in a required format****

After extracting the data, you might want to store it in a format. This format varies depending on your requirement.

1. col=["Company Name","job title","Location"]
2. df=pd.DataFrame({"Company Name":comp\_name[slice(10)],"job title":job\_title[slice(10)],"Location":loc\_name[slice(10)]})
3. print(df)
4. df.head()

Now, I’ll run the whole code again.

