Scrape Job Details using Selenium

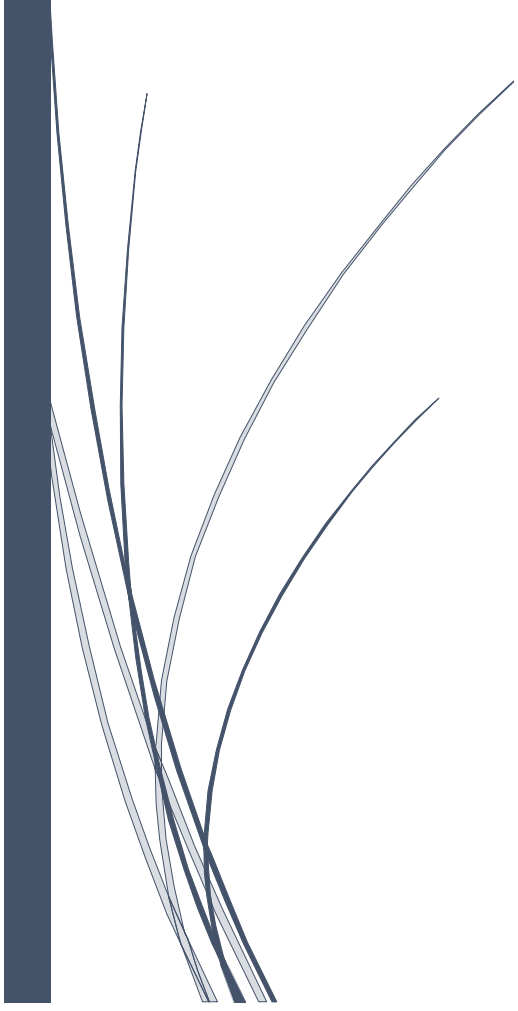


Table of Contents

Learning Outcome

At the end of this lab, you will be able to scrape job posts from foundit.sg using python and selenium and convert the data into CSV file using pandas.

Introduction

This practice covers the steps on how to scrape jobs from Indeed. Using python, selenium, and pandas, we'll be able to extract information from foundit.sg and construct apandas data frame. Before we begin, let's understand web scraping simply.

Imagine if you are trying to get much information about something from various web pages and articles that need to be stored in a suitable format, for instance, an excel file. One way is to go through all those websites and write useful information to the excel sheets manually. But programmers tend to do it in an easy way which is web scraping. Web scrapingis the technique of extracting a large amount of data from different web pages that can be stored in a suitable format.

Scraping job details from Indeed

The foundit (formerly Monster) Job Search App is a platform for freshers & experienced job seekers to find their perfect career opportunities.

Here are the steps involved:

1. Install and import necessary modules
2. Send some basic queries like like job title and location to theIndeed website using selenium
3. Fetch the current URL after sending the queries to the website using selenium
4. Parse the page and fetch the information about job title, company name, salary, etc
5. Store this information into a CSV file using pandas

Download **Selenium Scrape Job Details Starter.ipynb** and open it in Jupyter Notebook.

Load Libraries

First of all, we need to install some specific modules including a chrome driver for selenium. After installing the chrome driver move it to the working directory.

We need to import the libraries that will be used for this practical. Here requests help to send an HTTP request using python, Selenium is an automation tool that helps here to send

queries to the website, lxml can convert the page into XML or HTML format. bs4 module for parsing the web page and pandas to convert the data into a CSV file.

```
# Load packages
import requests
import pandas as pd
import time

from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
from selenium import webdriver

from selenium.webdriver.common.by import By
```

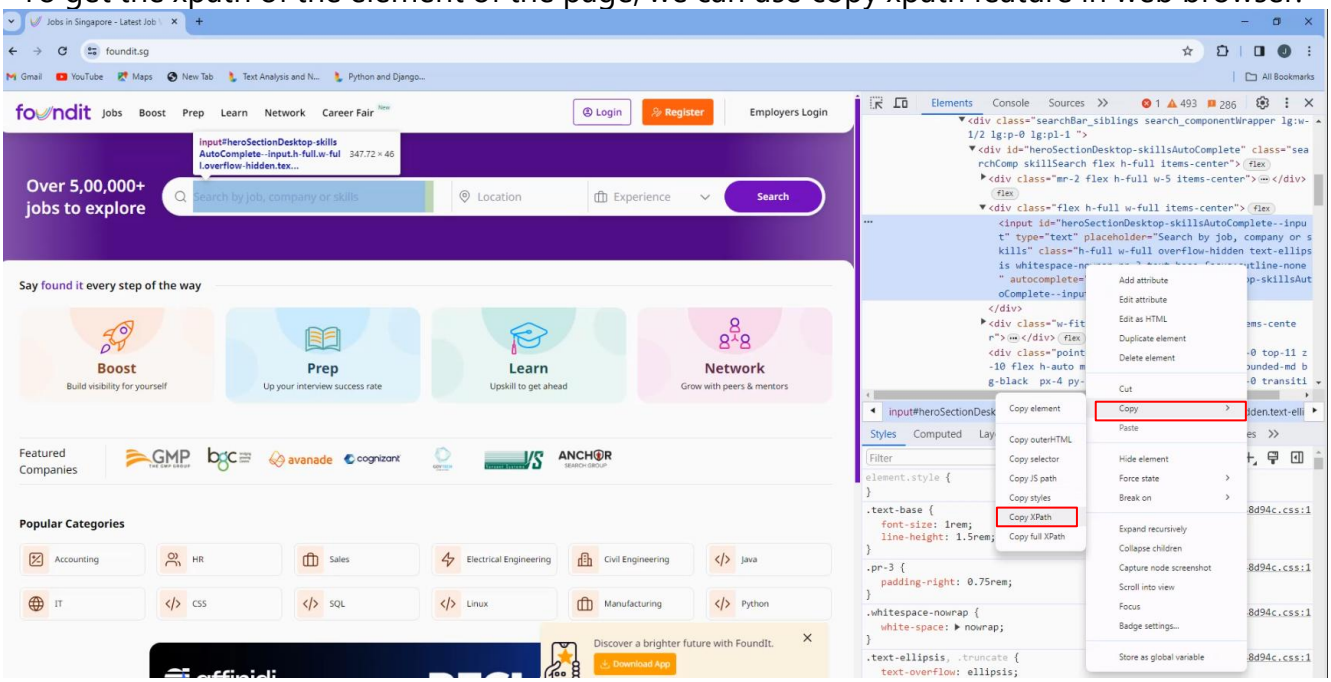
Sending job title and location using selenium

Now let's create a function that sends queries to the web page and returns the current URL. This function opens indeed.com using the specified URL as one of its parameters. Then it sends the job title or company name and location to the site using selenium. After that, we'll get a new page and its URL which consists of all the job details related to the job title and location you have specified as its parameters. Lastly, it returns the current URL which consists of jobs and their details so that we can simply scrape it using Beautiful Soup.

```
def get_current_url(url,job_title, location):

    # adjust your code if you are using Microsoft Edge web browser
    driver=webdriver.Chrome(service=Service(ChromeDriverManager().install()))
    driver.maximize_window()

    time.sleep(3)
    driver.get(url)

    time.sleep(3)
    driver.find_element("xpath", '//*[@id="heroSectionDesktop-skillsAutoComplete--
input"]').send_keys(job_title)

    time.sleep(3)
    driver.find_element("xpath", '//*[@id="heroSectionDesktop-locationAutoComplete--
input"]').send_keys(location)

    time.sleep(3)
    driver.find_element("xpath", '//*[@id="heroSectionDesktop-expAutoComplete--input"]').click()

    time.sleep(3)
    driver.find_element("xpath",'//*[@id="searchDropDown"]/ul/li[4]/span').click()

    time.sleep(3)
    driver.find_element("xpath", '//*[@id="searchForm"]/div/button').click()
```

```
    current_url = driver.current_url
    driver.quit()

    return current_url

current_url = get_current_url('https://foundit.sg/',"Data Scientist","Singapore" )
print(current_url)
```

To get the xpath of the element of the page, we can use copy xpath feature in web browser.



Scraping jobs details

Now let's get into the scraping part. We can use BeautifulSoup to scape data that we require like what we have covered in the previous lab. Try the following code.

```
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) '
                  'AppleWebKit/537.36 (KHTML, like Gecko) '
                  'Chrome/50.0.2661.102 Safari/537.36'
}

resp = requests.get(current_url)

print(resp.status_code)
```

Note that the foundit web page has block scraper scraping the webpage.

Let's try using Selenium to scrap the job details. Let us define another function that will do the scraping stuff for us. This function will retrieve data like the job title, company name and experience.

```
def scrape_job_details(url):
    driver=webdriver.Chrome(service=Service(ChromeDriverManager().install()))
    driver.get(url)
    content = driver.find_elements(By.CLASS_NAME, 'srpResultCardContainer')

    jobs_list = []
    for post in content:
        try:
            data = {
                "job_title": post.find_element(By.CLASS_NAME, 'jobTitle').text,
                "company": post.find_element(By.CLASS_NAME, 'companyName').text,
                "date": post.find_element(By.CLASS_NAME, 'timeText').text,
                "experience": post.find_elements(By.CLASS_NAME, 'details')[0].text
            }
        except IndexError:
            continue
        jobs_list.append(data)

    driver.quit()

    return pd.DataFrame(jobs_list)

df_jobs = scrape_job_details(current_url)
df_jobs.head()
```

The **driver.get()** function returns the data of the entire webpage. The next step is to find the CSS selectors and retrieve the raw text inside the tags that contain these CSS selectors. The CSS selectors given in the code are probably the same on the web page but sometimes it may change.

By looping through all the job posts we'll get much information about it. Lastly, we converted the data into a pandas data frame and simply returned it. You'll get the details about the job title, company name, rating, location, date of posting, and a simple job description. You can save it as a CSV file using df.to_csv("jobs.csv").

```
df_jobs.to_csv('jobs.csv', index=False)
```

~The End~