Scraping Indeed jobs in Python

The following notebook demonstrates simple way to scrap jobs data from website named indeed.com

A brief about **Web scraping**: Web scraping is data scraping used for extracting data from websites. Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser.

Source: Wikipedia

We will need following libraries of Python: requests, html5lib, bs4(Beautiful Soup 4). You can install them using the pip package manager.

Let us first import all the libraries that we will use in this scraping:

In [1]:

```
import requests
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
from pandas import Series,DataFrame
```

For scraping data from the website, first we need their HTML data which we can get by a request from their server. Hence we can get the HTML data by passing a URL to get method of requests library. But depending upon the company you're searching for, the URL will be different. So for that, let us closely examing the URL mapping of a search in Indeed website.

As you can see in the image below, when you search for Goldman Sachs jobs, the parameter "q" in URL is set equal to the keywords user typed in the search query. The spaces in the search query are ignored and the individual words of the query are concatenated. Also, the parameter "l" is responsible for collecting the location query of the user who is searching. Therefore, using string manipulation, we can generate URLs for any search done based on the keywords in the query.

For simplicity and demonstration purposes, let us search for first few jobs with keywords "goldman sachs", same as mentioned in the above image.

Therefore, we create the URL accordingly:

In [2]:

```
URL = "https://www.indeed.co.in/jobs?q=goldman+sachs&l="
```

Now that we have our URL ready, for getting the HTML data, we will pass the URL to get method provided by the requests library and pass the response of that method to BeautifulSoup constructor, so that we can get the raw HTML content.

```
BeautifulSoup constructor parameters:
req.content ==> The raw HTML content sent as a response.
html5lib ==> HTML parser to be used for parsing HTML data.
```

An HTML parser parses the raw HTML content and constructs it in the form of HTML tree hierarchy.

In [3]:

```
req = requests.get(URL)

htmlData = BeautifulSoup(req.content, 'html5lib')
```

Now that we have the HTML structured data, we will use the robust BeautifulSoup library to go to each section of the HTML and get the things we need. We're scraping the job titles and their location of the few jobs posted on the website.

In [4]:

```
jobtitles = []
joblocation = []

for table in htmlData.find_all(name='table' , attrs={'id':'resultsBody'}):
    for table1 in htmlData.find_all(name='table' , attrs={'id':'pageContent'}):
        for td in table.find_all(name='td' , attrs={'id':'resultsCol'}):
            for div in td.find_all(name='div' , attrs={'data-tn-component':'organicJob'}):
                for h2 in div.find_all(name='h2', attrs={'class':'jobtitle'}):
                    for a in h2.find_all(name='a', attrs={'class':'turnstileLink'}):
                        jobtitles.append(a['title'])

for table in htmlData.find_all(name='table' , attrs={'id':'resultsBody'}):
    for table1 in htmlData.find_all(name='table' , attrs={'id':'pageContent'}):
        for td in table.find_all(name='td' , attrs={'id':'resultsCol'}):
            for div in td.find_all(name='div' , attrs={'data-tn-component':'organicJob'}):
                for span in div.find_all(name='span', attrs={'class':'location'}):
                    joblocation.append(span.text)
```

In the above code, what we did is we manually inspected the HTML content using an Inspector available in a browser. Then from that, we have an overall idea how the content we are interested in ie. the job postings are saved. Then we simply dive into each hierarchies of the tags and sections, to obtain the desired data and store it an array.

Finally, the data which is needed, is available and what's left is simply to arrange it in a format and print the result:

In [5]:

```
titlesSeries = Series(np.array(jobtitles))
locationSeries = Series(np.array(joblocation))
dataframe = pd.concat([titlesSeries,locationSeries],axis=1)
dataframe.columns = ['Job Title','Location']
print("Goldman Sachs jobs by Indeed:")
dataframe
```

```
Goldman Sachs jobs by Indeed:
```

Out[5]:

	Job Title	Location
0	Research & Development Engineering - Software ...	India
1	Technology Risk - Risk Measurement & Analytics...	Bengaluru, Karnataka
2	Platform - Application Infrastructure - Develo...	Bengaluru, Karnataka
3	Finance - Controllers - Regulatory Capital & R...	Bengaluru, Karnataka
4	Technology Risk - Risk Measurement & Analytics...	Bengaluru, Karnataka
5	Marquee Engineering - Software Engineer	India
6	Finance - Investment Management Division - Pri...	Bengaluru, Karnataka
7	Securities- Equities One Infra Strats - Analys...	Bengaluru, Karnataka
8	Platform - App Bank Operation - Front Line Sup...	India
9	Platform - Foundational Engg - DDI Engineer	Bengaluru, Karnataka

There we have some job postings at Goldman Sachs from indeed.com.

Let us try some other company along with the location:

In [6]:

```
URL = "https://www.indeed.co.in/jobs?q=morgan+stanley&l=mumbai"
req = requests.get(URL)
htmlData = BeautifulSoup(req.content, 'html5lib')

jobtitles = []
joblocation = []
```

```
for table in htmlData.find_all(name='table' , attrs={'id':'resultsBody'}):
    for table1 in htmlData.find_all(name='table' , attrs={'id':'pageContent'}):
        for td in table.find_all(name='td' , attrs={'id':'resultsCol'}):
            for div in td.find_all(name='div' , attrs={'data-tn-component':'organicJob'}):
                for h2 in div.find_all(name='h2', attrs={'class':'jobtitle'}):
                    for a in h2.find_all(name='a', attrs={'class':'turnstileLink'}):
                        jobtitles.append(a['title'])

for table in htmlData.find_all(name='table' , attrs={'id':'resultsBody'}):
    for table1 in htmlData.find_all(name='table' , attrs={'id':'pageContent'}):
        for td in table.find_all(name='td' , attrs={'id':'resultsCol'}):
            for div in td.find_all(name='div' , attrs={'data-tn-component':'organicJob'}):
                for span in div.find_all(name='span', attrs={'class':'location'}):
                    joblocation.append(span.text)

titlesSeries = Series(np.array(jobtitles))
locationSeries = Series(np.array(joblocation))
dataframe = pd.concat([titlesSeries,locationSeries],axis=1)
dataframe.columns = ['Job Title','Location']
print("Morgan Stanley Mumbai jobs by Indeed:")
dataframe
```

```
Morgan Stanley Mumbai jobs by Indeed:
```

Out[6]:

	Job Title	Location
0	Associate	Mumbai, Maharashtra
1	Analyst - Risk Identification	Mumbai, Maharashtra
2	Associate Risk Analytics	Mumbai, Maharashtra
3	Scenario Analytics Associate/Analyst	Mumbai, Maharashtra
4	Analyst - ICAAP Production	Mumbai, Maharashtra
5	Associate GGLC	Mumbai, Maharashtra
6	Analyst - Cash Client Processing	Mumbai, Maharashtra
7	Associate - Payment Investigation, SSBO	Mumbai, Maharashtra
8	MSDE Engineer	Mumbai, Maharashtra
9	Developer - Database	Mumbai, Maharashtra

Here we have scraped jobs at Morgan Stanley in Mumbai.

Hence, a simple web scraper for scraping jobs from Indeed is designed.

Thanks for reading.