**Web scraping from web pages**

We are going to understand in this lab, how to extract the information from a website. There are two types of websites static and dynamic. More than 70% of websites contain dynamic pages.

If the content we are viewing in our browser does not match the content we see in the HTML source code we are retrieving from the site, then we are encountering a dynamic website. Otherwise, if the browser and source code content match each other, the website is static. A mismatch of content would be due to the execution of JavaScript that changes the HTML elements on the page. Using the Chrome browser, we can view the original HTML via the View page source. We can view the revised HTML in our browser if it executes JavaScript in the Elements window via Inspect the web page.

For scraping a website, the first thing you need to do is to visit the website to see its Web-Address. In our example, we are going to use the INDEED.com job website. But it can be any website. The important thing to know here is you can scrap a website using different libraries like BeautifulSoup, Selenium, scrapy, etc. For static pages, regular expressions, beautifulsoap, Lxml can be used. For dynamic pages, beautifulsoap, scrapy, Selenium, and AJAX requests can be used. In all cases, the browser drivers will be required to navigate through the pages automatically.

We are going to use the requests, BeautifulSoup, and Pandas libraries.

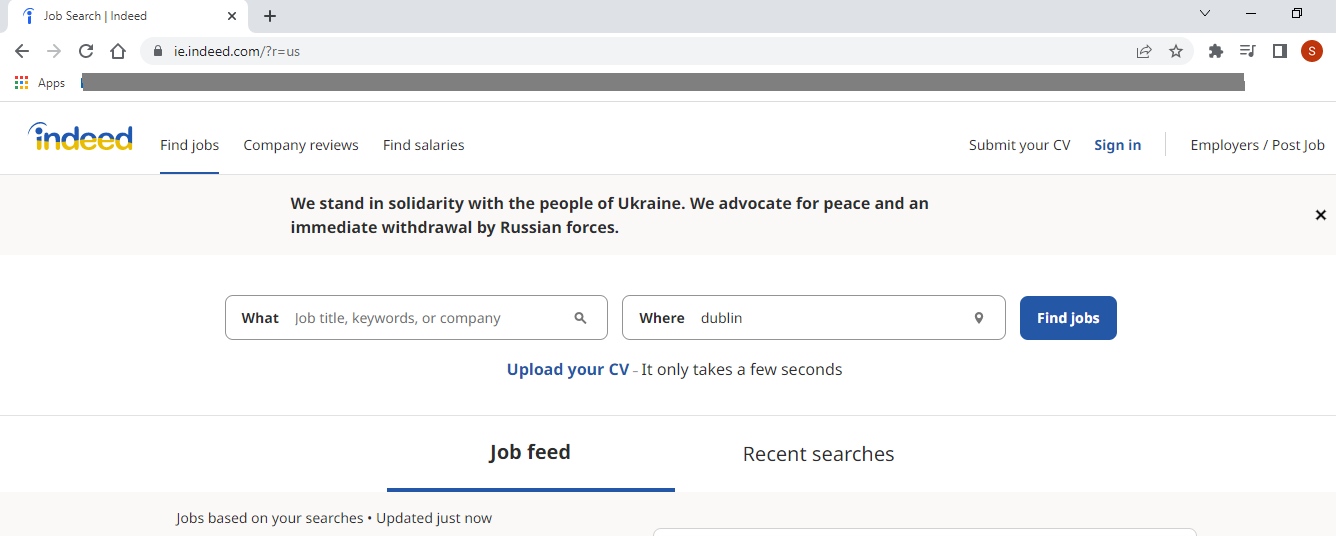
**Understanding, using, and normalization of URL:**

Visit the Indeed job website page.

[www.indeed.com](http://www.indeed.com)

you will see it will take you to your regional website, which is normal.

<https://ie.indeed.com/?r=us>

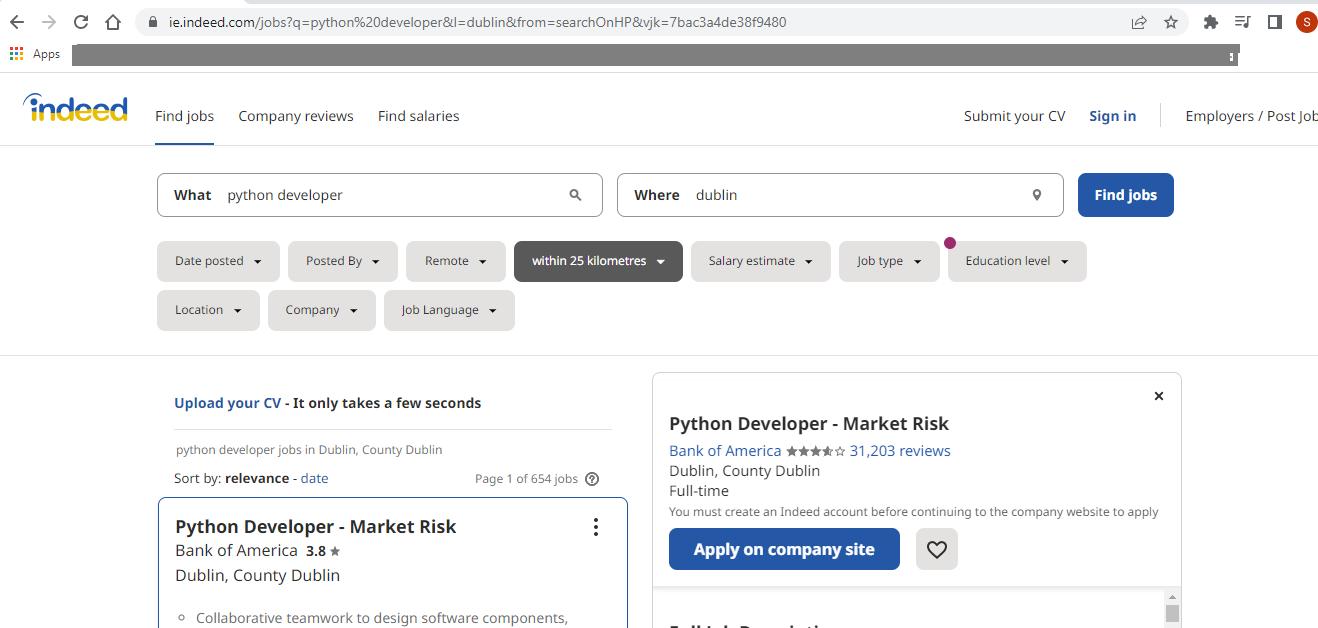


From here you can automate the process of searching for any job title and location. But that will be the advanced stage. For now, we will scrap a particular job title and location for our understanding and later we can do it for searching any particular job title. For now, let’s say we are interested in the python developer job title for the Dublin location.

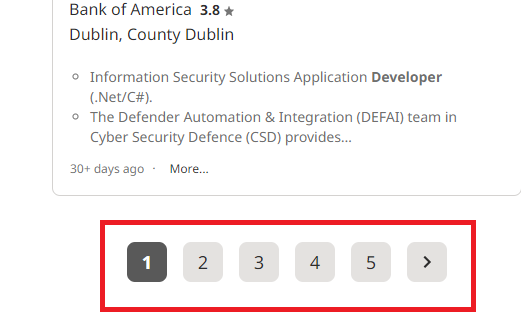
Job title: Python developer

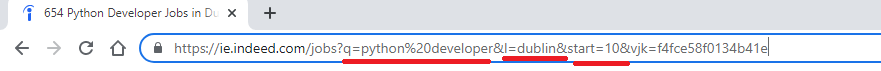
Location: Dublin

Enter this title and job location on the above page in what and where. Click Find Jobs. You will see a list of python related jobs. As shown below. Scroll the page down for all entries.



Click on any search page from the bottom as shown in the figure below. Click on page 2, and then look closely at the web address.





Three fields will be shown which are important for dynamic pages.

q = job title

l = location

start = 10

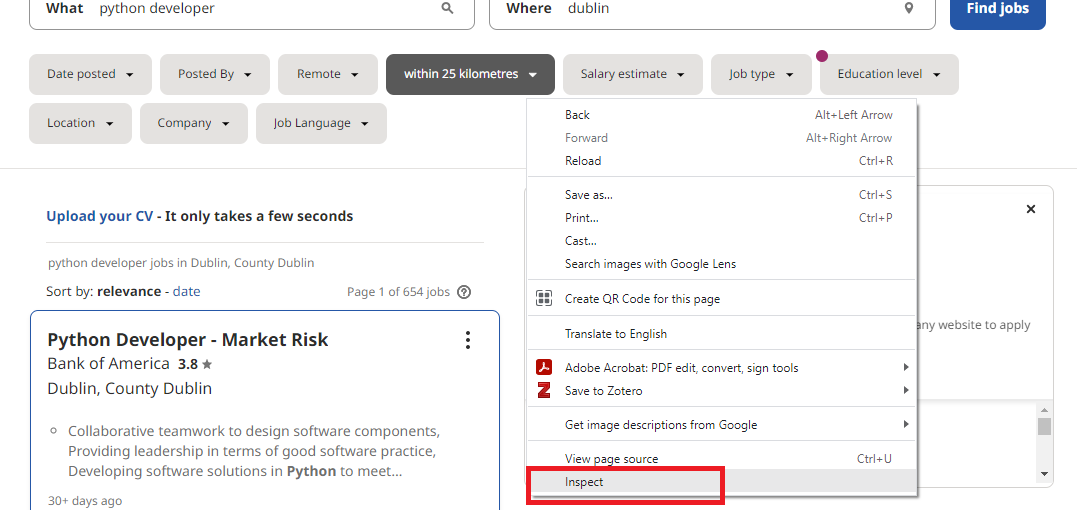
This start is the page number. The pattern here is, that it jumps with +10 here. So the first page will be 0, the second page will be 10, the third will be 20, and so on. When we will write the script to navigate, this pattern we will use to navigate and scroll the pages or any particular search page with given fields of search.

Now ignoring the last hexadecimal values, we can write the URL or web address as below.

|  |
| --- |
| <https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start=0>  This URL we will use for our scrapping script. |

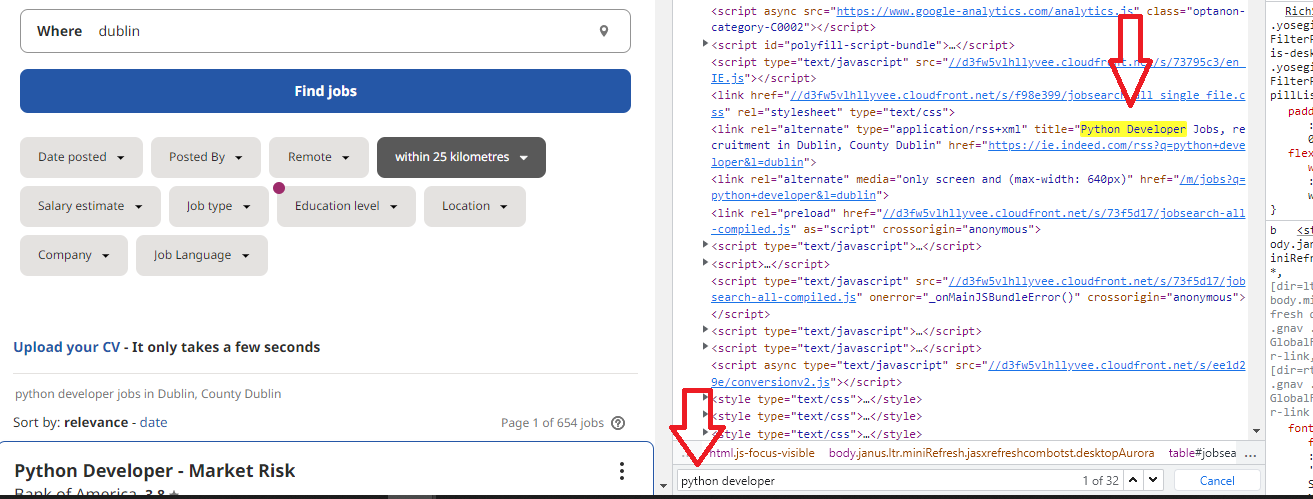
**Practicing HTML code:**

Right-click on the web page and click on INSPECT. Now search there for the work of python developer to see, if it exists there or not.



After clicking on this inspect, you will see a layout pane on the right. Search there the word python developer to see the results.

For searching press CTRL+F on the HTML code interface.



This is how you can inspect any page or part of a page for its HTML code. For web scraping, locating a particular attribute for any button or space on the page is very important, which we will look at later in this manual.

**Importing Python Libraries:**

We will be using three libraries in this script,

Requests:

Requests is an HTTP library for the Python programming language. The goal of the project is to make HTTP requests simpler and more human-friendly.

<https://docs.python-requests.org/en/latest/>

BeaurtifulSoup:

Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Pandas:

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

<https://pandas.pydata.org/>

**Starting to script:**

First import the libraries

|  |
| --- |
| Import requests  from bs4 import BeautifulSoup |

Since we are working on dynamic pages. We need a function that can get to each different web page to extract the information. In the following code, we are defining a function to pass this value of the page, which we later use as well.

|  |
| --- |
| Import requests  from bs4 import BeautifulSoup  def extract(page): |

Now include the URL which I explained above and which includes the q, l, and start parameters. Remember, it was skipping with 10s as a value. We can start with 0, as explained above. But, as we want an automated script, we can use this ‘page’ string in place of numbers (0, 10, 20), as follows.

The actual link as mentioned above was like this,

<https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start=0>

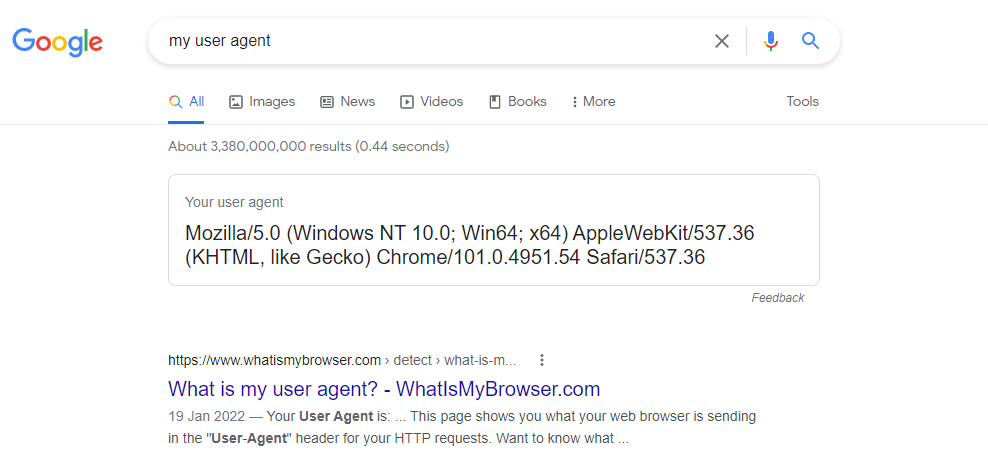
We can use curly brackets to fit the page in there.

[https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}](https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start=%7bpage%7d)

We can use f-sting for format. Do some research on this f string and why we use it.

|  |
| --- |
| Import requests  from bs4 import BeautifulSoup  def extract(page):  url = f‘[https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}](https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start=%7bpage%7d)’ |

Now, we need to include the user-agents. Search on google as my user agent and copy-paste it into the python file.



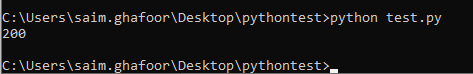
|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36'}  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}' |

Now let’s use the requests and beautiful soup.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser') |

For testing this part, use the following code.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  return r. status\_code  print(extract(0)) #this is zero |



This shows that the code is working, as you can see the HTML/HTTP code for OK.

Now remove these two checking lines.

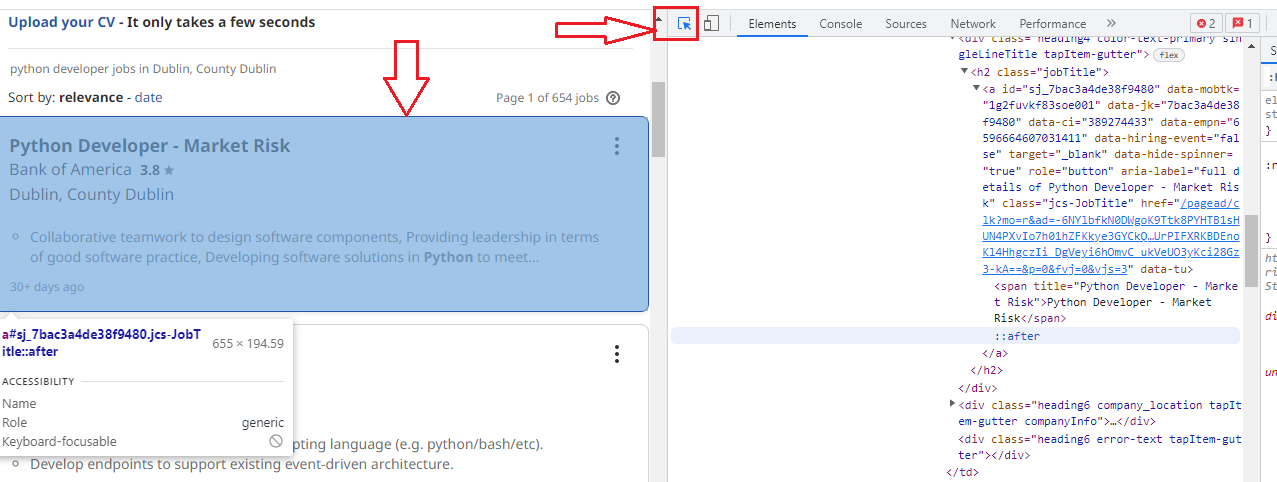
Now we are going to introduce our soup variable, as below, and return the soup.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup |

Now we are going to use the transform, where we are passing these soup values and we will use our HTML code parameters. For this, we need to define the transform function.

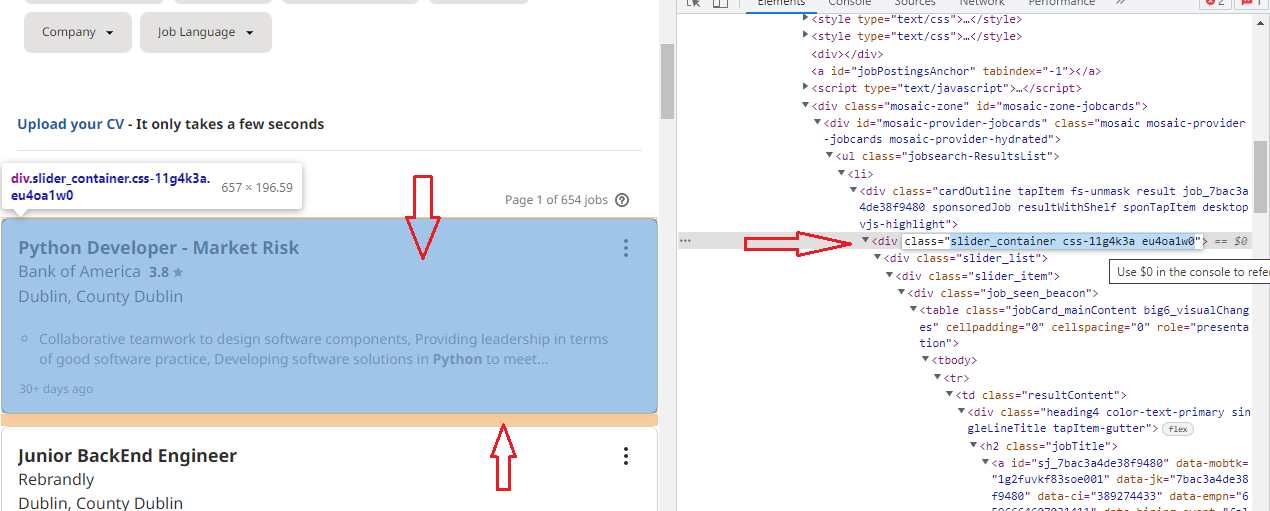
|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup): |

We need now to navigate through our HTML page to look at intro classes where the JOB CARDS code is used.



The down arrow is showing the box for JobCards. The right direction arrow is the navigator for HTML code. Click on this and then place your mouse cursor on any job card. It will direct you to the related HTML code of that particular job card.

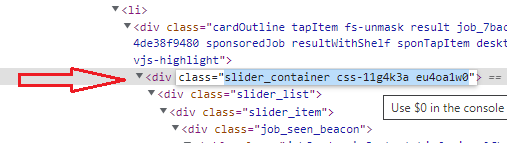
You need to spend some time on this HTML code tags understanding. You should look for similar tags for each job card.



We need to look at the SLIDER CONTAINER, which is common for each job title. You can navigate to verify it in other job cards.

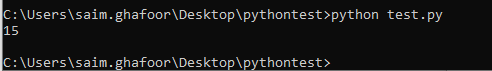
For our function, we are going to use this div variable which is commonly representing the job slider container of all the searched jobs.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = "slider\_container css-11g4k3a eu4oa1w0") |



Let’s now check our code, if it is returning any values.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = "slider\_container css-11g4k3a eu4oa1w0")  return len(divs)  c = extract(0)  print(transform(c)) |



We can see it is showing us this value as 15. It means on the first page we have 15 records.

Now delete those two lines, which we used for testing, the return len() and print(transform).

**FIELDS/INFORMATION extraction:**

Now is the time to extract the information from the webpage.

We will use a for loop, because, it’s a dynamic page and we don’t want to write a separate code for each different page. Remember we use the ‘page’ variable above.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = "slider\_container css-11g4k3a eu4oa1w0")  for item in divs:    c = extract(0) |

But before that let’s see again, how we can extract the exact tag to extract the information which we need.

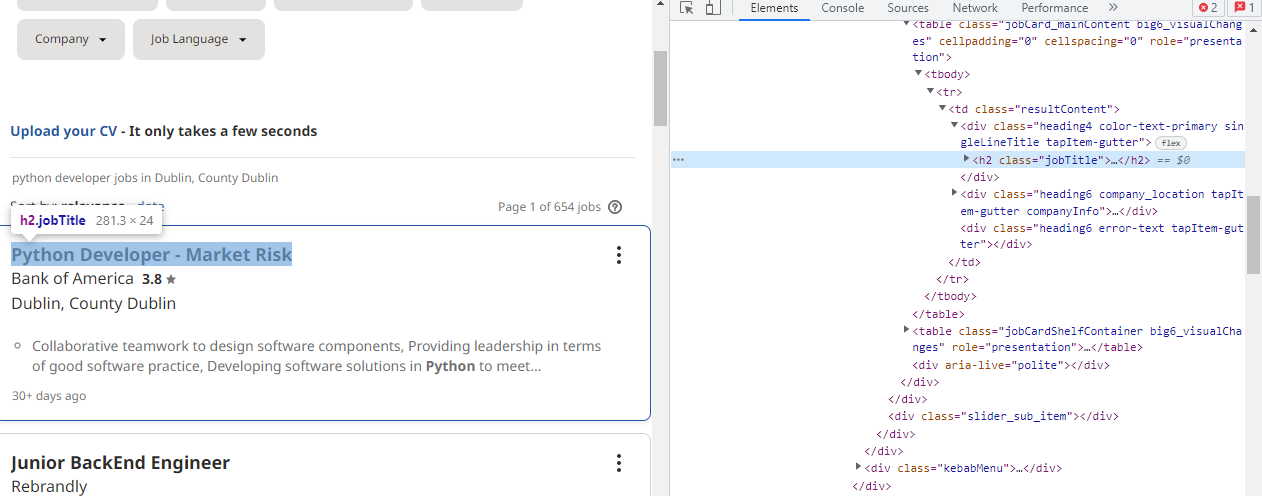
First, look at the below figure to see what is available to us.



In the above figure, we can see the job title, company name, rating, requirements, country, and days.

Let’s see the tags of these fields in HTML and how we can use them.

When you place a cursor on the HTML tags, it will highlight the particular field in the job card on the left. In the below figure I have highlighted the JOB TITLE field. It’s written there if you look closely.

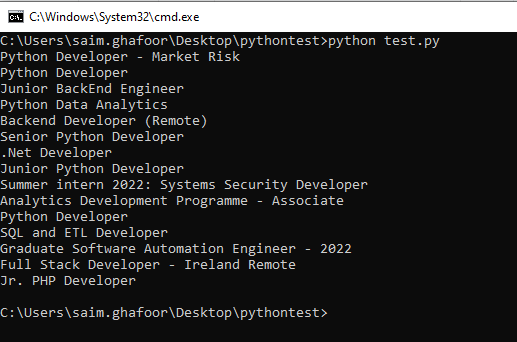


Expand it further to see the HTML code for the title.



Here you can see the ‘a’ variable. Now if you look into other job cards’ HTML code, you will see, that this ‘a’ is common for all of them. THIS IS THE TRICK. Look for common tags. I am going to use this ‘a’ to identify that extract the information within the DIVS where you will find this ‘a’ tag. Let’s look at the code now and test it.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = 'slider\_container css-11g4k3a eu4oa1w0')  for item in divs:  title = item.find('a').text  print(title)  return  c= extract(0)  transform(c) |



We can see that the code is extracting the information which we require. If you get any other information here. Try different and appropriate tags from the HTML code.

If you see spaces between the line. Then use the strip() function after text in the title code line as,

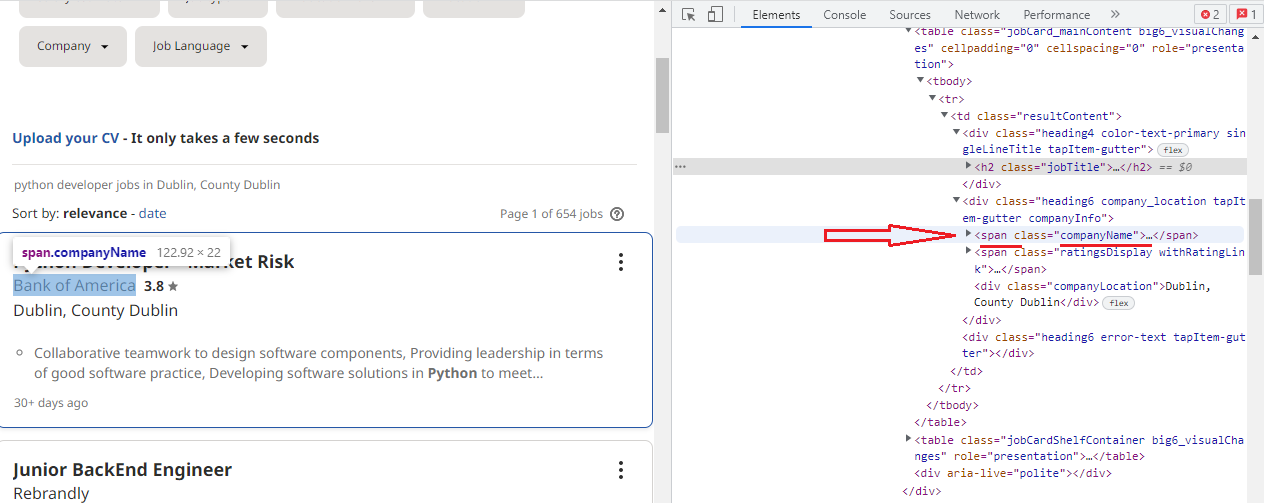
title = item.find('a').text.strip()

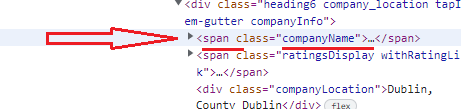
Remove the print line from the above code.

Now we know how to extract the information and locate the tags from the HTML code.

Let’s look at some other fields.

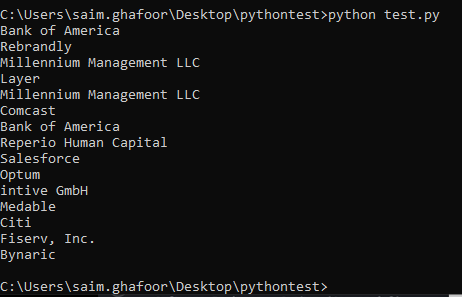
We are going to include the CompanyName now.





From the above, we can see that the tag is Span and the name is ‘companyName’. Let’s use it in our code.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = 'slider\_container css-11g4k3a eu4oa1w0')  for item in divs:  title = item.find('a').text  company = item.find('span', class\_ = 'companyName').text  print(company)  c= extract(0)  transform(c) |



We can extract the company information as well.

Now, we are going to look at salary. But, remember, salary is not given for all the job cards or search results. So, if we simply add the salary tag. It can give you errors. We need to handle that error as well.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = 'slider\_container css-11g4k3a eu4oa1w0')  for item in divs:  title = item.find('a').text  company = item.find('span', class\_ = 'companyName').text  try:  salary = item.find('span', class\_ = 'metadata salary-snippet-container').text  except:  salary = ''  c= extract(0)  transform(c) |

We can also extract summary information as well.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = 'slider\_container css-11g4k3a eu4oa1w0')  for item in divs:  title = item.find('a').text  company = item.find('span', class\_ = 'companyName').text  try:  salary = item.find('span', class\_ = 'metadata salary-snippet-container').text  except:  salary = ''  summary = item.find('div', {'class' : 'job-snippet'}).text.replace('\n', '')  c= extract(0)  transform(c) |

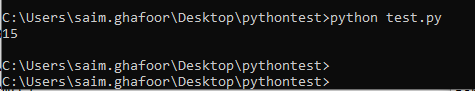
**Creating our Dictionary:**

Let’s create our dictionary to store the extracted values.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = 'slider\_container css-11g4k3a eu4oa1w0')  for item in divs:  title = item.find('a').text  company = item.find('span', class\_ = 'companyName').text  try:  salary = item.find('span', class\_ = 'metadata salary-snippet-container').text  except:  salary = ''  summary = item.find('div', {'class' : 'job-snippet'}).text.replace('\n', '')  job = {  'title': title,  'company': company,  'salary': salary,  'summary': summary  }  joblist.append(job)  c= extract(0)  transform(c) |

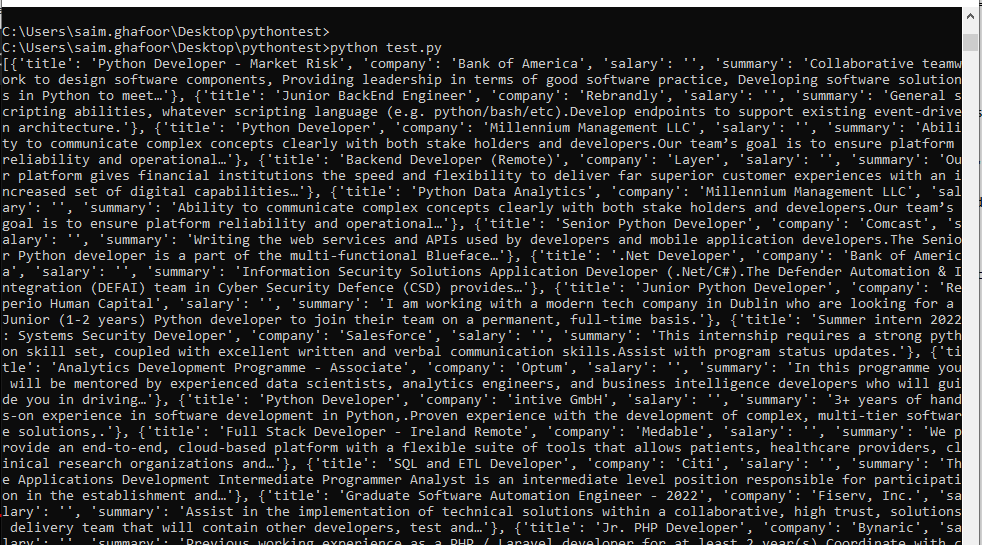
Now to check the code, if it is storing our values. Lets, create a blank list as joblist = [ ] and print the length of the jobs (for testing only)

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = 'slider\_container css-11g4k3a eu4oa1w0')  for item in divs:  title = item.find('a').text  company = item.find('span', class\_ = 'companyName').text  try:  salary = item.find('span', class\_ = 'metadata salary-snippet-container').text  except:  salary = ''  summary = item.find('div', {'class' : 'job-snippet'}).text.replace('\n', '')  job = {  'title': title,  'company': company,  'salary': salary,  'summary': summary  }  joblist.append(job)  joblist = []  c= extract(0)  transform(c)  print(len(joblist)) |



It is showing us 15 records, so we are fine. Now let’s see the contents.

Now in place of print(len(joblist)), try print(joblist) to see the contents of the dictionary.



The values are shown here. You can print them on separate lines as well. The salary is not giving any error, but it does not show the values as well. It may be I am using a wring tag. Figure out the correct tag for the salary. A summary is shown fine, which is a job requirement. In the above lines, the replace function is used for those requirements where multiple lines exist.

**Saving values in CSV file AND extracting info from multiple pages:**

We have verified that we can extract the values and can also store them in the dictionary. Now let’s save the dictionary values in a CSV file.

For this, we are going to use Pandas library.

In the above code top, import it as,

|  |
| --- |
| Import pandas as pd |

Now is the time, when we should introduce the function to extract information from more than one page.

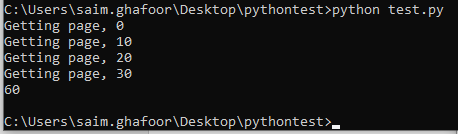
Use the function as below, but remember the pages are of multiples of 10. So we will use the range.

|  |
| --- |
| for i in range(0,40,10):  print(f'Getting page, {i}')  c = extract(0)  transform(c) |

This means go until three pages from 0, to 40 with a jump of 10. So, it will use four pages, 0, 10, 20, and 30.

Now print the joblist length again. Use the below code.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = 'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  #return r. status\_code  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = "slider\_container css-11g4k3a eu4oa1w0")  for item in divs:  title = item.find('a').text.strip()  company = item.find('span', class\_ = 'companyName').text  try:  salary = item.find('span', class\_ = 'metadata salary-snippet-container').text  except:  salary = ''  summary = item.find('div', {'class' : 'job-snippet'}).text.replace('\n', '')  job = {  'title': title,  'company': company,  'salary': salary,  'summary': summary  }  joblist.append(job)  return  joblist = []  for i in range(0,40,10):  print(f'Getting page, {i}')  c = extract(0)  transform(c)  print(len(joblist)) |



The records are now shown as 60, which means the code is working fine for extracting the data from multiple pages.

We have just extracted information from four pages. If you want to extract information from all the pages then use the WHILE loop with the end of the file/loop to see how far you can go with your extraction.

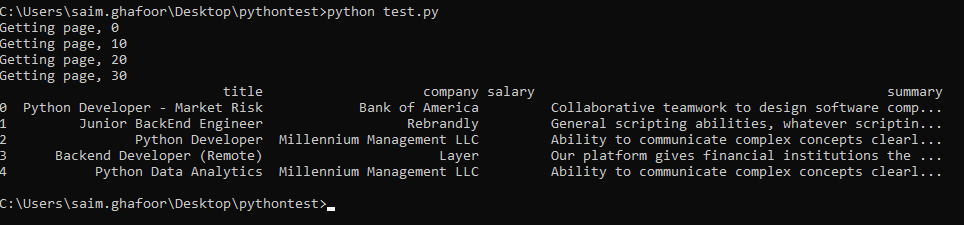
Now lets save our dictionary to CSV file.

|  |
| --- |
| df = pd.DataFrame(joblist)  print(df.head())  df.to\_csv('jobs.csv') |

Head for the first job.

Following is the complete code.

|  |
| --- |
| import requests  from bs4 import BeautifulSoup  import pandas as pd  def extract(page):  headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36' }  url = f'https://ie.indeed.com/jobs?q=python%20developer&l=dublin&start={page}'  r = requests.get(url, headers)  soup = BeautifulSoup(r.content, 'html.parser')  return soup  def transform(soup):  divs = soup.find\_all('div', class\_ = 'slider\_container css-11g4k3a eu4oa1w0')  for item in divs:  title = item.find('a').text  company = item.find('span', class\_ = 'companyName').text  try:  salary = item.find('span', class\_ = 'metadata salary-snippet-container').text  except:  salary = ''  summary = item.find('div', {'class' : 'job-snippet'}).text.replace('\n', '')  job = {  'title': title,  'company': company,  'salary': salary,  'summary': summary  }  joblist.append(job)  return  joblist = []  for i in range(0,40,10):  print(f'Getting page, {i}')  c = extract(0)  transform(c)  df = pd.DataFrame(joblist)  print(df.head())  df.to\_csv('jobs.csv') |



We can see that the data is being extracted successfully with titles such as title, company salary, and summary.

\*\*If you are still getting errors for chromedirver, download it from the chromedriver website and place it into your folder. You should check first your chrome version and download the chromedirver according to your chrome version.

\*\* For those who are using Selenium and encounters chromedriver errors. Download appropriate chromedriver.exe file from the google chromedriver website but first confirm the chorme browser version which you are using. For paths errors wither use r infront of the path as r() or use double slashes like c:\\users\\saim\\pyhton\\chromedriver.exe

**TASK:**

Use any other website to extract information.

Use a while loop to extract information from all available pages.

Use other fields as well to extract, like ratings, city, etc.