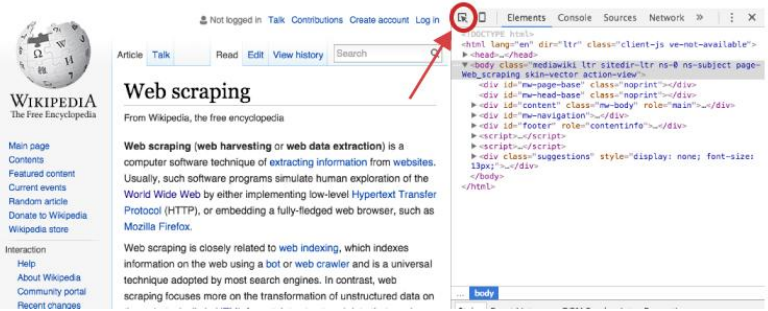
Applied Web Scraping I

THE KNOWLEDGE HOUSE

Celebrating 10 Years of Diversifying Tech

Agenda - Schedule

1. Warm Up
2. HTML Review
3. Introduction to Web Scraping
4. Using BS4
5. Break
6. Web Scraping Lab



Web scraping

From Wikipedia, the free encyclopedia

Web scraping (**web harvesting** or **web data extraction**) is a computer software technique of extracting information from websites. Usually, such software programs simulate human exploration of the World Wide Web by either implementing low-level Hypertext Transfer Protocol (HTTP), or embedding a fully-fledged web browser, such as Mozilla Firefox.

Web scraping is closely related to web indexing, which indexes information on the web using a bot or web crawler and is a universal technique adopted by most search engines. In contrast, web scraping focuses more on the transformation of unstructured data on

```
<!DOCTYPE html>
<html lang="en" dir="ltr" class="client-js ve-not-available">
▶<head>…</head>
▼<body class="mediawiki ltr sitedir-ltr ns-0 ns-subject page-
Web_scraping skin-vector action-view">
    <div id="mw-page-base" class="noprint"></div>
    <div id="mw-head-base" class="noprint"></div>
  ▶<div id="content" class="mw-body" role="main">…</div>
  ▶<div id="mw-navigation">…</div>
  ▶<div id="footer" role="contentinfo">…</div>
  ▶<script>…</script>
  ▶<script>…</script>
  ▶<div class="suggestions" style="display: none; font-size:
  13px;">…</div>
  </body>
</html>
```

Web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler.
https://en.wikipedia.org/wiki/Web_scraping

Agenda - Announcements

- **Monthly Satisfaction Survey** due 4/7 https://theknowledgehouse.typeform.com/to/JyoK7IHd
- **Week 5 Pre-Class Quiz** due 4/8
- **TLAB #2** due 4/21
- Add music to your respective Cohort Link!
    - We will use this for the music recommendation algorithm in Phase 2!
- Lastly, look for announcements from **#data-science-fellowship-instructor-announcements**

Agenda - Goals

- Interpret HTML pages and tags
- Learn how to request data over the web and scrape via the bs4 package
- Learn about common Web-scraping Methods

Warm-Up

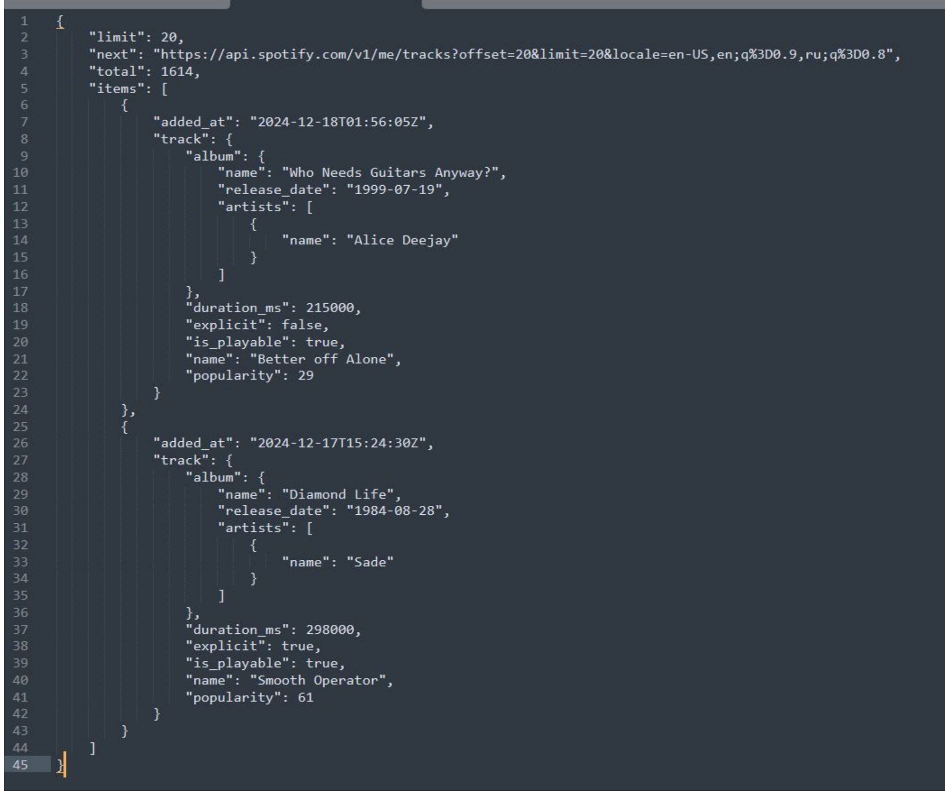
```
import requests

url = “https://api.spotify.com/v1/me/tracks”
r = requests.get(url)

data = r.json()

vals = []
for item in data[“items”]:
        track = item[“track”]
        name = track[“name”]
        vals.append(name)

print(vals)
```



```
{
    "limit": 20,
    "next": "https://api.spotify.com/v1/me/tracks?offset=20&limit=20&locale=en-US,en;q%3D0.9,ru;q%3D0.8",
    "total": 1614,
    "items": [
        {
            "added_at": "2024-12-18T01:56:05Z",
            "track": {
                "album": {
                    "name": "Who Needs Guitars Anyway?",
                    "release_date": "1999-07-19",
                    "artists": [
                        {
                            "name": "Alice Deejay"
                        }
                    ]
                },
                "duration_ms": 215000,
                "explicit": false,
                "is_playable": true,
                "name": "Better off Alone",
                "popularity": 29
            }
        },
        {
            "added_at": "2024-12-17T15:24:30Z",
            "track": {
                "album": {
                    "name": "Diamond Life",
                    "release_date": "1984-08-28",
                    "artists": [
                        {
                            "name": "Sade"
                        }
                    ]
                },
                "duration_ms": 298000,
                "explicit": true,
                "is_playable": true,
                "name": "Smooth Operator",
                "popularity": 61
            }
        }
    ]
}
```

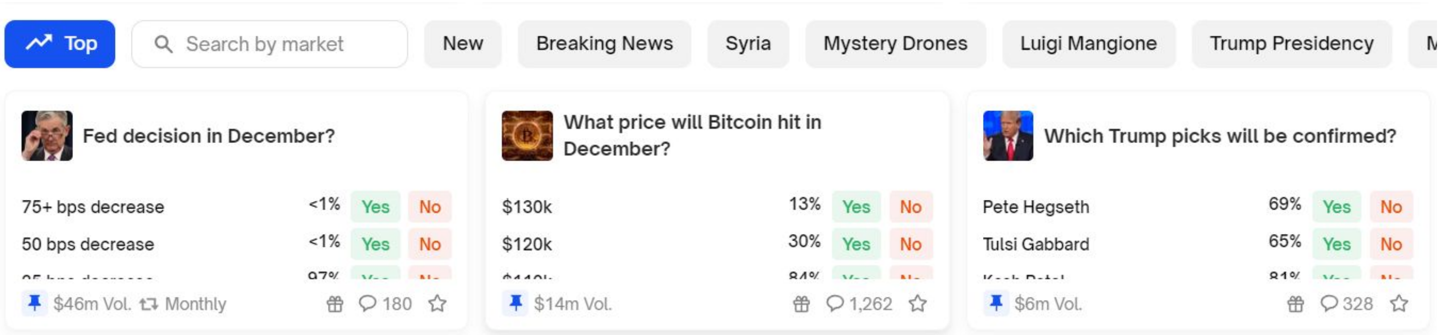
Join your pod groups and evaluate this chunk of code. Work together to figure out what will occur when we run this code. Assume the JSON object to the right is the resource we get when running this code.

HTML Review

Layers of Data Access

However, what if the data that we need is not provided by an API? What do we do then?

1. **Structured CSV** files/folders
   a. *Usually provided freely by organization (but boring and not transformative)*
2. **API** calls to get JSON files
   a. *Either offered freely or through a paid service*
3. **Web-scraping**
   a. *Always free, but not always appreciated*



Fed decision in December?

75+ bps decrease <1% Yes No
50 bps decrease <1% Yes No
25 bps decrease 97% Yes No

$46m Vol. Monthly 180

What price will Bitcoin hit in December?

$130k 13% Yes No
$120k 30% Yes No
$110k 84% Yes No

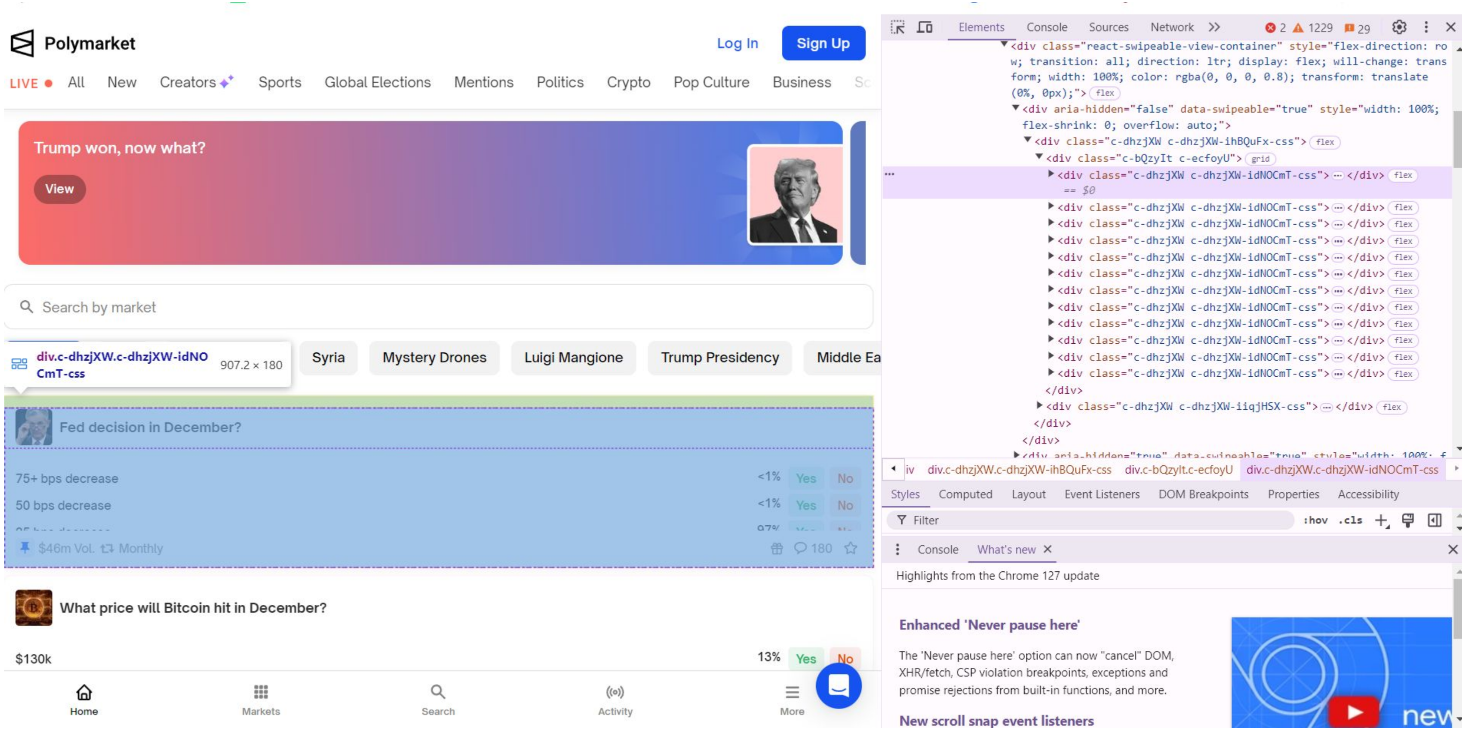
$14m Vol. 1,262

Which Trump picks will be confirmed?

Pete Hegseth 69% Yes No
Tulsi Gabbard 65% Yes No
Kash Patel 81% Yes No

$6m Vol. 328

Well in such cases, we go forward with **carefully** web-scraping a web-page to extract information. But how do we tell a computer to interpret this structure?



```
▼<div class="react-swipeable-view-container" style="flex-direction: ro
w; transition: all; direction: ltr; display: flex; will-change: trans
form; width: 100%; color: rgba(0, 0, 0, 0.8); transform: translate
(0%, 0px);"> flex
  ▼<div aria-hidden="false" data-swipeable="true" style="width: 100%;
   flex-shrink: 0; overflow: auto;">
    ▼<div class="c-dhzjXW c-dhzjXW-ihBQuFx-css"> flex
      ▼<div class="c-bQzyIt c-ecfoyU"> grid
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
           == $0
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
        ▶<div class="c-dhzjXW c-dhzjXW-idNOCmT-css"> ⋯ </div> flex
       </div>
      ▶<div class="c-dhzjXW c-dhzjXW-iiqjHSX-css"> ⋯ </div> flex
     </div>
   </div>
  ▶<div aria-hidden="true" data-swipeable="true" style="width: 100%; f
```

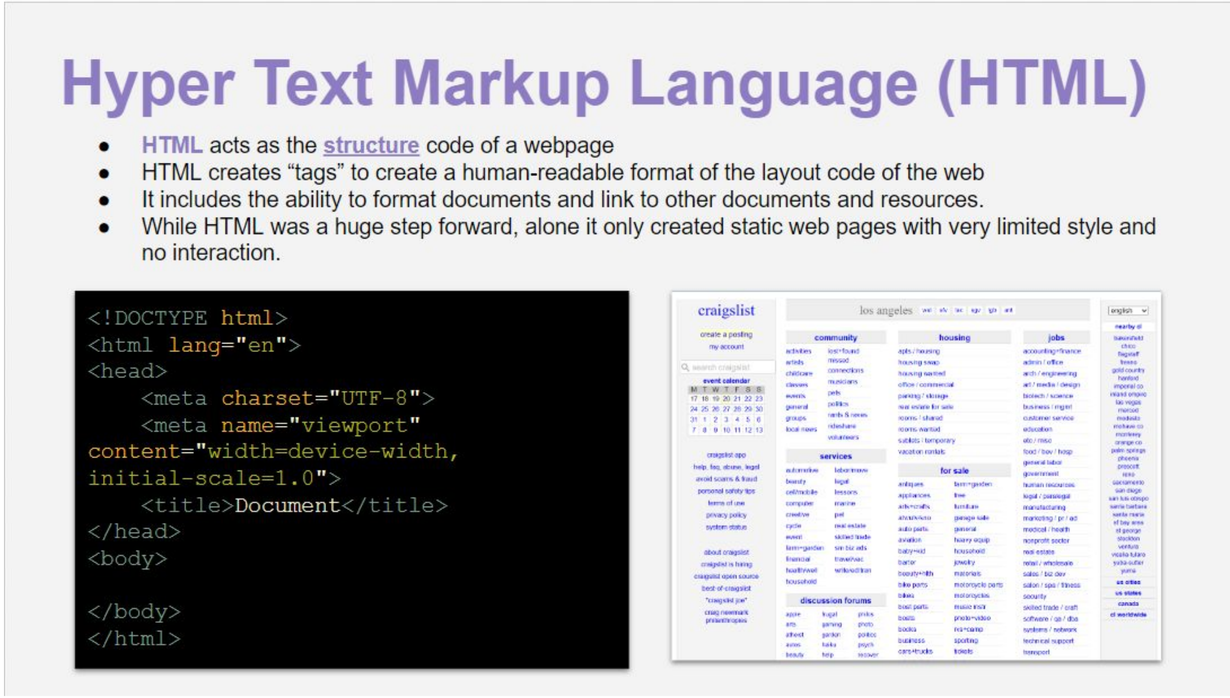
Highlights from the Chrome 127 update

Enhanced 'Never pause here'

The 'Never pause here' option can now "cancel" DOM, XHR/fetch, CSP violation breakpoints, exceptions and promise rejections from built-in functions, and more.

New scroll snap event listeners

As it turns out, all web-pages are structured via **HTML**. This is a semi-structured format that we can interpret in our code just like JSON. But first, how do we read HTML?



Hyper Text Markup Language (HTML)

- HTML acts as the structure code of a webpage
- HTML creates “tags” to create a human-readable format of the layout code of the web
- It includes the ability to format documents and link to other documents and resources.
- While HTML was a huge step forward, alone it only created static web pages with very limited style and no interaction.

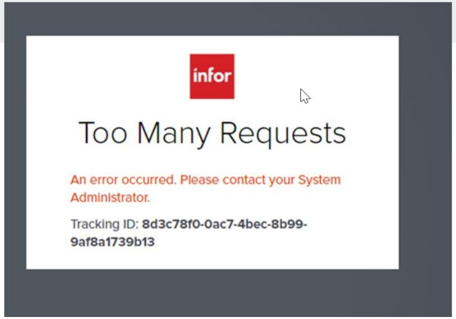
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

To learn more about HTML, let’s take a look at some web-dev slides (***Slides 8 - 16***): https://docs.google.com/presentation/d/13xZTXU_fwg_Df5ukBNKMY4f_wg06sEP_mFX-oRBdc0s/edit#slide=id.g25d81a010a0_0_0

Web-Scraping

Web Scraping - Challenges



infor

Too Many Requests

An error occurred. Please contact your System Administrator.

Tracking ID: **8d3c78f0-0ac7-4bec-8b99-9af8a1739b13**

Before we get started with web-scraping, let's address a few challenges you will encounter in this process:

- **Variability**: when designing a web-scraper for a **unique** website, you will often have to start from scratch and design a unique scraper.
- **Durability**: websites change! A scraper that works one day might not work the next.
- **Responsibility**: too many requests for information will result in your scraper getting kicked

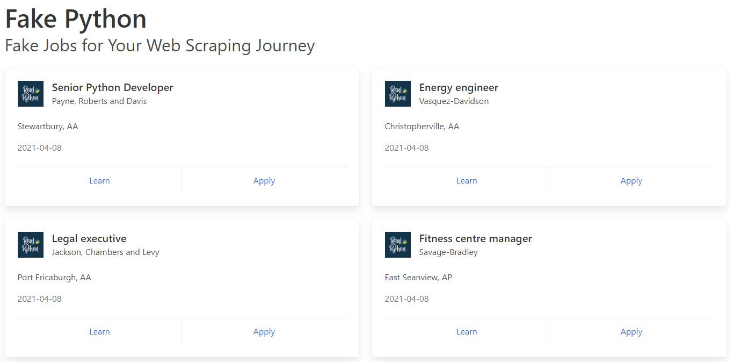
Therefore, even as we learn more about these new concepts, the previous lessons of **documentation** and **testing** still apply!

Web Scraping – Getting Started

Before we begin, we should **inspect** the website that we will be scraping in order to understand its structure and **URL**.

We can accomplish this by **accessing our website** and browsing through while noting the following objects:

- **Interactions**: How can I access “next” pages in my website?
- **The URL**: How do I modify my URL to access different parts of the website?
- **Format of website**: Where does my data reside?



Fake Python

Fake Jobs for Your Web Scraping Journey

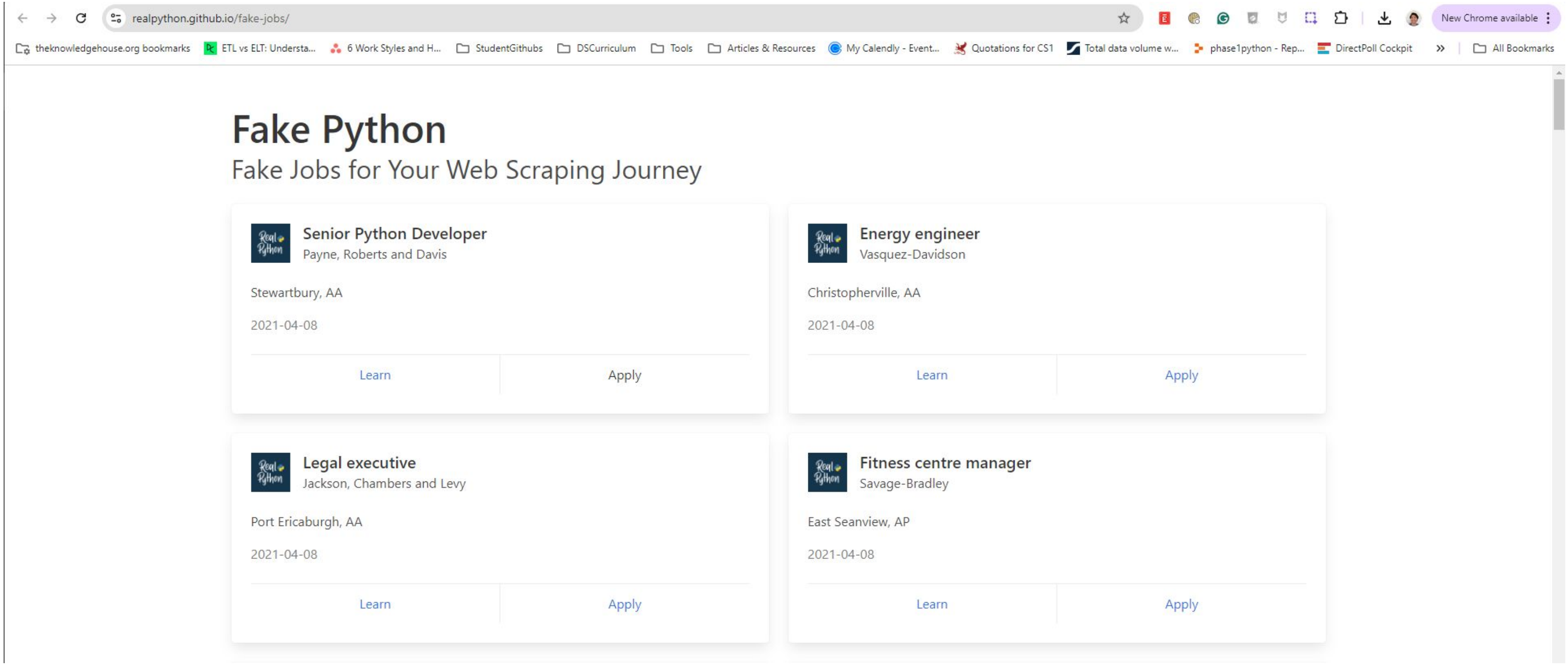
Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA
2021-04-08
Learn Apply

Energy engineer
Vasquez-Davidson
Christopherville, AA
2021-04-08
Learn Apply

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA
2021-04-08
Learn Apply

Fitness centre manager
Savage-Bradley
East Seanview, AP
2021-04-08
Learn Apply

For this guided exercise, we are scraping the fake job-posting site: https://realpython.github.io/fake-jobs/



Fake Python

Fake Jobs for Your Web Scraping Journey

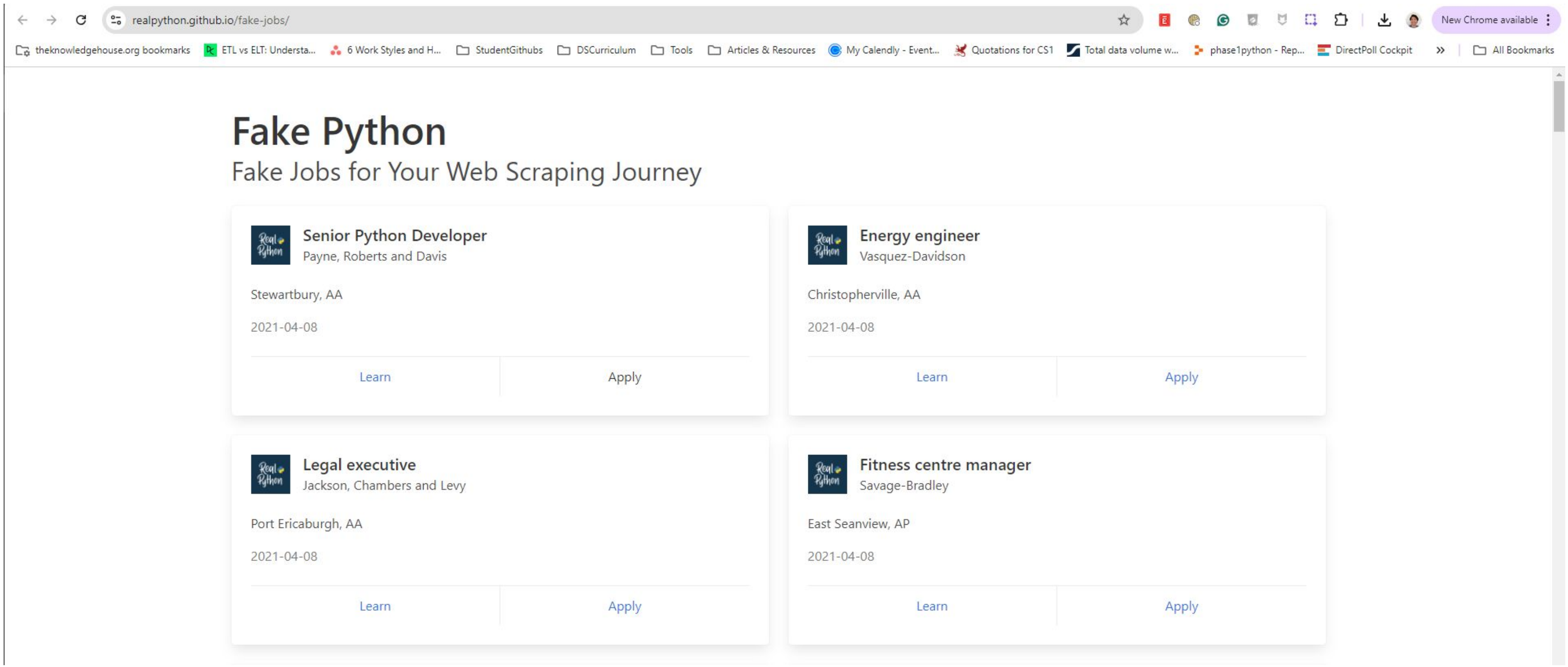
Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA
2021-04-08

Energy engineer
Vasquez-Davidson
Christopherville, AA
2021-04-08

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA
2021-04-08

Fitness centre manager
Savage-Bradley
East Seanview, AP
2021-04-08

Let’s explore the different sections we can access through simple clicks. **Firstly,** let’s recognize how our jobs are formatted on this site.



Fake Python

Fake Jobs for Your Web Scraping Journey

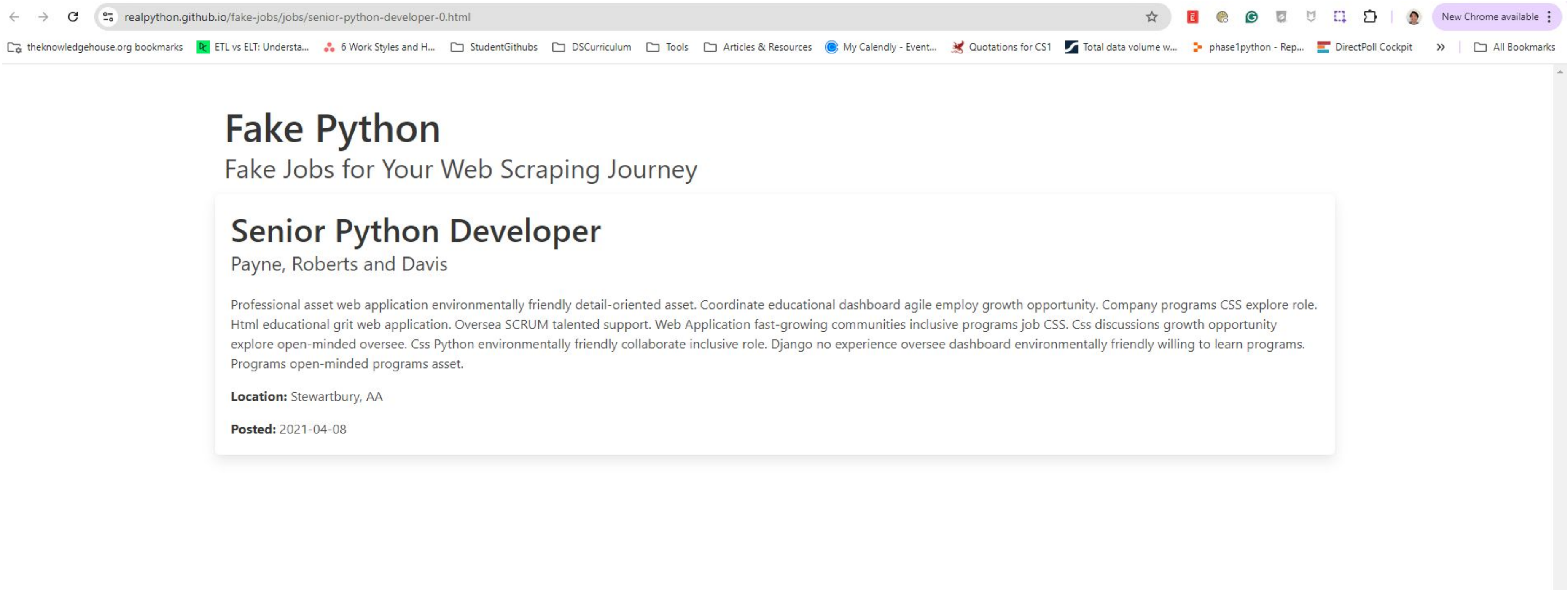
Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA
2021-04-08

Energy engineer
Vasquez-Davidson
Christopherville, AA
2021-04-08

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA
2021-04-08

Fitness centre manager
Savage-Bradley
East Seanview, AP
2021-04-08

Next, let's note how our URL changes as we click into different parts of the site. Currently we are on "realpython.github.io/fake-jobs"



Fake Python

Fake Jobs for Your Web Scraping Journey

Senior Python Developer

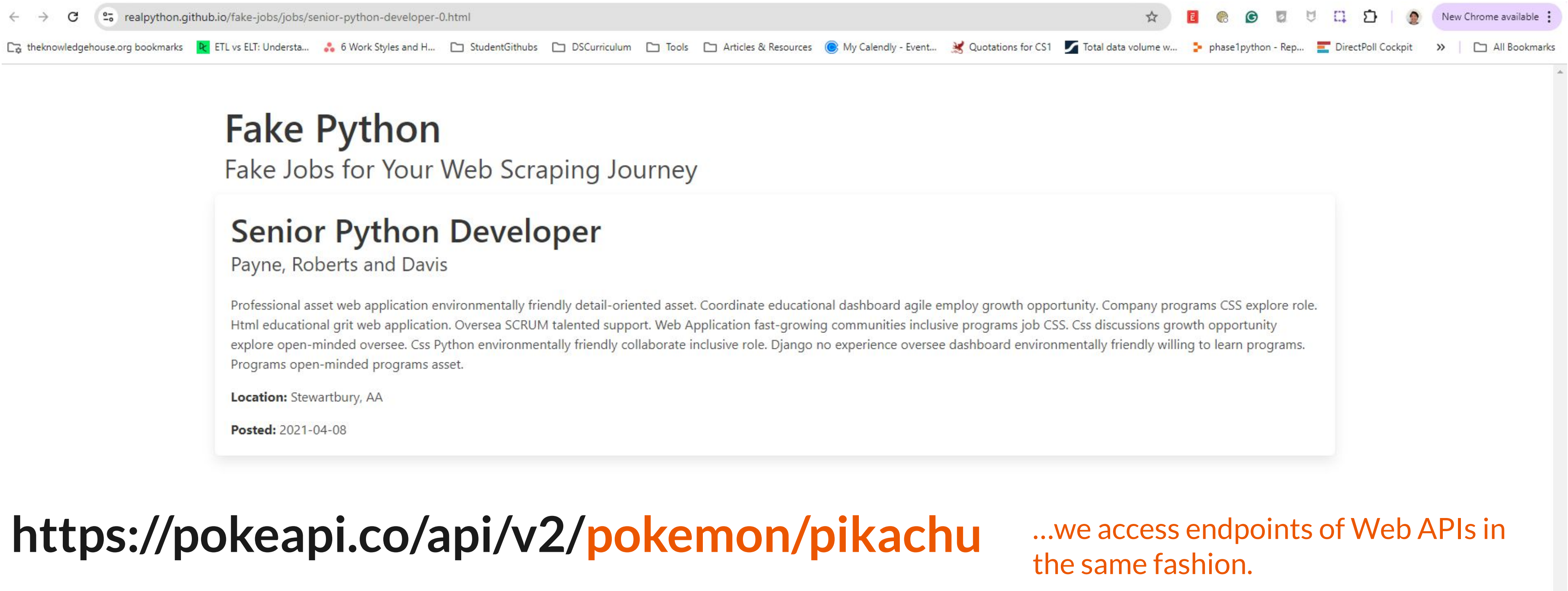
Payne, Roberts and Davis

Professional asset web application environmentally friendly detail-oriented asset. Coordinate educational dashboard agile employ growth opportunity. Company programs CSS explore role. Html educational grit web application. Oversea SCRUM talented support. Web Application fast-growing communities inclusive programs job CSS. Css discussions growth opportunity explore open-minded oversee. Css Python environmentally friendly collaborate inclusive role. Django no experience oversee dashboard environmentally friendly willing to learn programs. Programs open-minded programs asset.

Location: Stewartbury, AA

Posted: 2021-04-08

But when clicking “apply” on a certain job posting, our url transforms to “realpython.github.io/fake-jobs/senior-python-developer-0.html”. Does this **pattern** remind you of any other type of URL that we’ve been using?



Fake Python

Fake Jobs for Your Web Scraping Journey

Senior Python Developer

Payne, Roberts and Davis

Professional asset web application environmentally friendly detail-oriented asset. Coordinate educational dashboard agile employ growth opportunity. Company programs CSS explore role. Html educational grit web application. Oversea SCRUM talented support. Web Application fast-growing communities inclusive programs job CSS. Css discussions growth opportunity explore open-minded oversee. Css Python environmentally friendly collaborate inclusive role. Django no experience oversee dashboard environmentally friendly willing to learn programs. Programs open-minded programs asset.

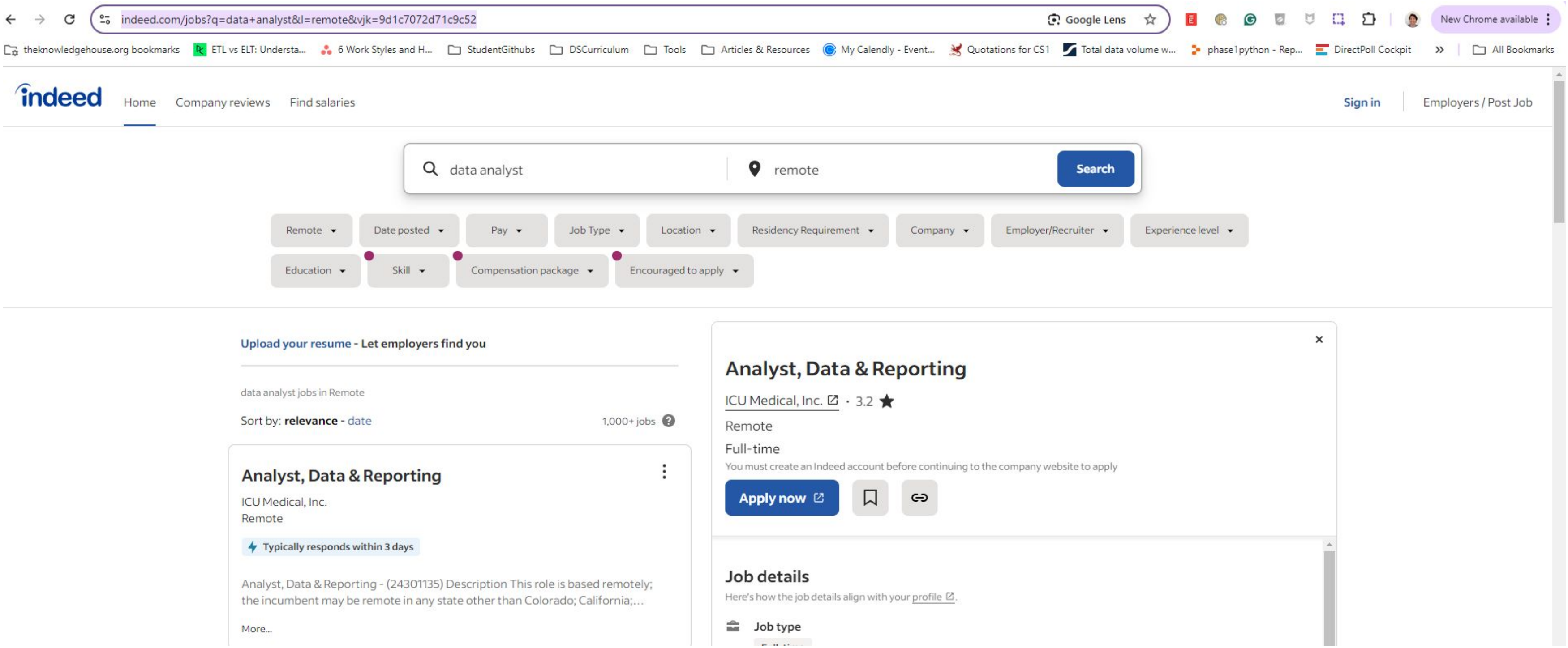
Location: Stewartbury, AA

Posted: 2021-04-08

https://pokeapi.co/api/v2/pokemon/pikachu

…we access endpoints of Web APIs in the same fashion.

But when clicking "apply" on a certain job posting, our url transforms to "realpython.github.io/fake-jobs/senior-python-developer-0.html". Does this **pattern** remind you of any other type of URL that we've been using?



indeed.com/jobs?q=data+analyst&l=remote&vjk=9d1c7072d71c9c52

data analyst

remote

Upload your resume - Let employers find you

data analyst jobs in Remote

Sort by: relevance - date

1,000+ jobs

Analyst, Data & Reporting

ICU Medical, Inc.

Remote

Typically responds within 3 days

Analyst, Data & Reporting - (24301135) Description This role is based remotely; the incumbent may be remote in any state other than Colorado; California;...

More...

Analyst, Data & Reporting

ICU Medical, Inc. · 3.2 ★

Remote

Full-time

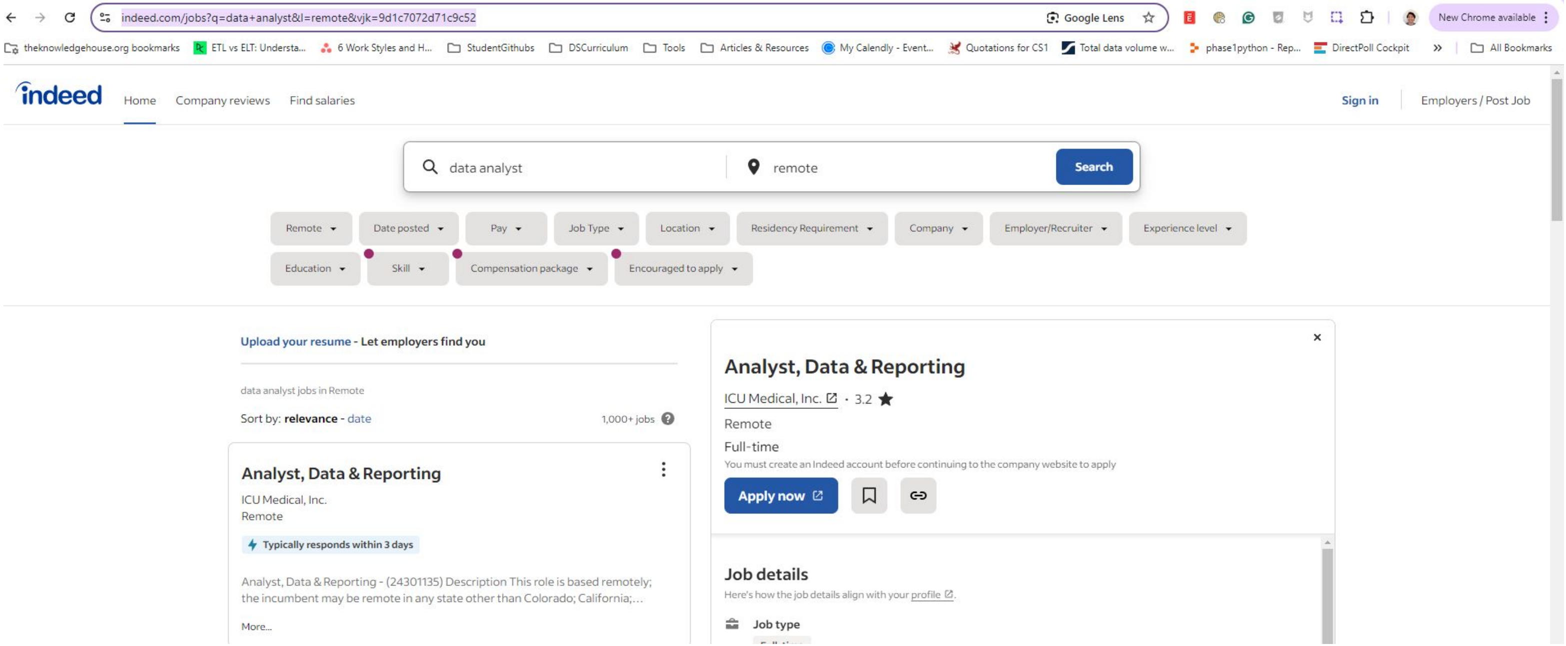
You must create an Indeed account before continuing to the company website to apply

Job details

Here's how the job details align with your profile.

Job type

This is sufficient for simple websites, but what if we are scraping other websites that entail a search-engine? How can we modify **which resource** we access?



indeed.com/jobs?q=data+analyst&l=remote&vjk=9d1c7072d71c9c52

data analyst

remote

Upload your resume - Let employers find you

data analyst jobs in Remote

Sort by: relevance - date

1,000+ jobs

Analyst, Data & Reporting

ICU Medical, Inc.
Remote

Typically responds within 3 days

Analyst, Data & Reporting - (24301135) Description This role is based remotely; the incumbent may be remote in any state other than Colorado; California;...

More...

Analyst, Data & Reporting

ICU Medical, Inc. · 3.2 ★

Remote

Full-time

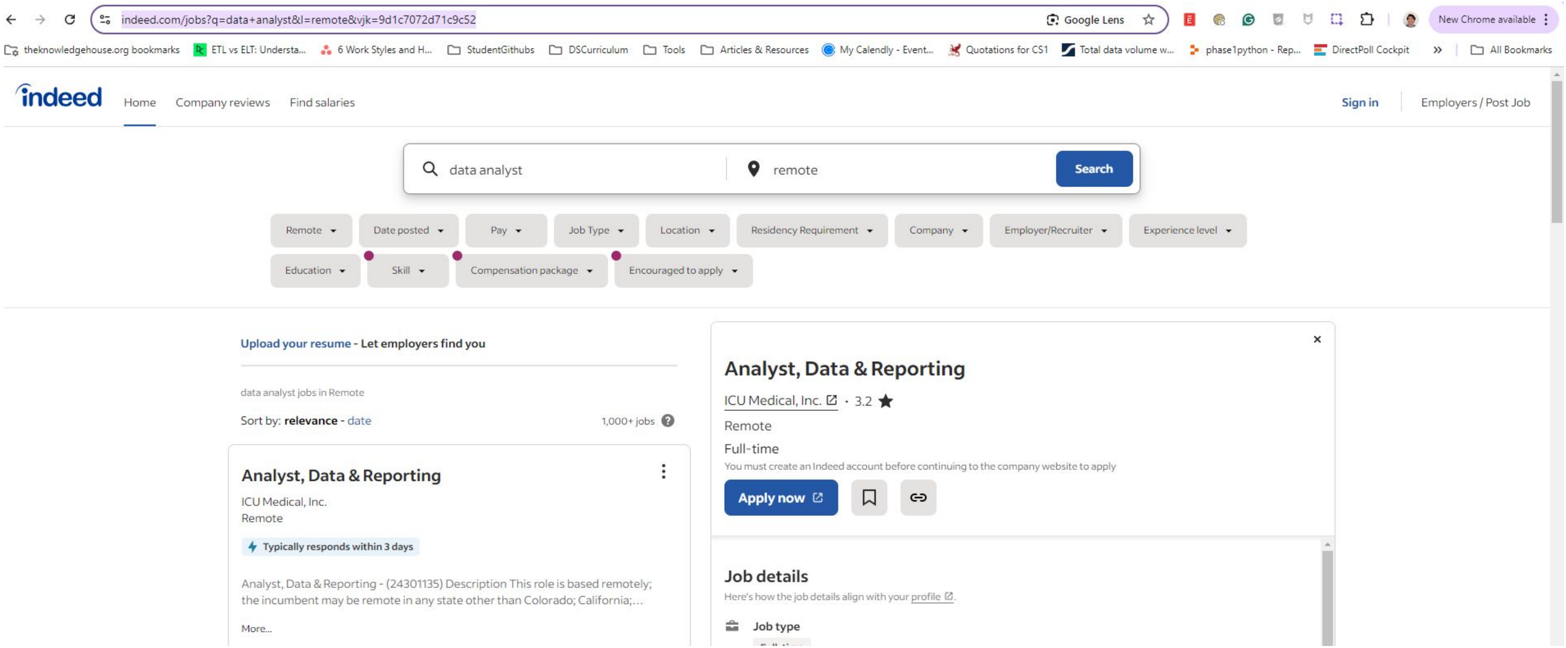
You must create an Indeed account before continuing to the company website to apply

Job details

Here's how the job details align with your profile.

In this case, we begin using search parameters. For example, to pull all "remote data analyst" jobs in Indeed, we would use the following URL: https://www.indeed.com/jobs?q=data+analyst&l=remote

Again, does this **pattern** remind you of any other type of URL that we've been using?



Upload your resume - Let employers find you

data analyst jobs in Remote

Sort by: relevance - date

1,000+ jobs

Analyst, Data & Reporting

ICU Medical, Inc.
Remote

Typically responds within 3 days

Analyst, Data & Reporting - (24301135) Description This role is based remotely; the incumbent may be remote in any state other than Colorado; California;...

More...

Analyst, Data & Reporting

ICU Medical, Inc. · 3.2 ★

Remote

Full-time

You must create an Indeed account before continuing to the company website to apply

Apply now

Job details

Here's how the job details align with your profile.

Job type

https://api.polygon.io/v2/aggs/ticker/AAPL/range/1/day/2023-01-09/2023-02-10?adjusted=true&sort=asc&apiKey=123456

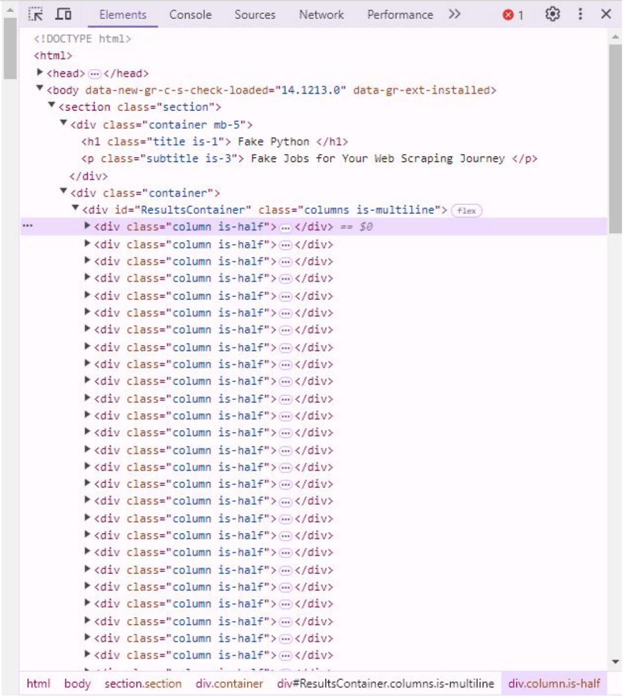
In this case, we begin using search parameters. For example, to pull all "remote data analyst" jobs in Indeed, we would use the following URL: https://www.indeed.com/jobs?q=data+analyst&l=remote

Again, does this **pattern** remind you of any other type of URL that we've been using?

Web Scraping – Getting Started

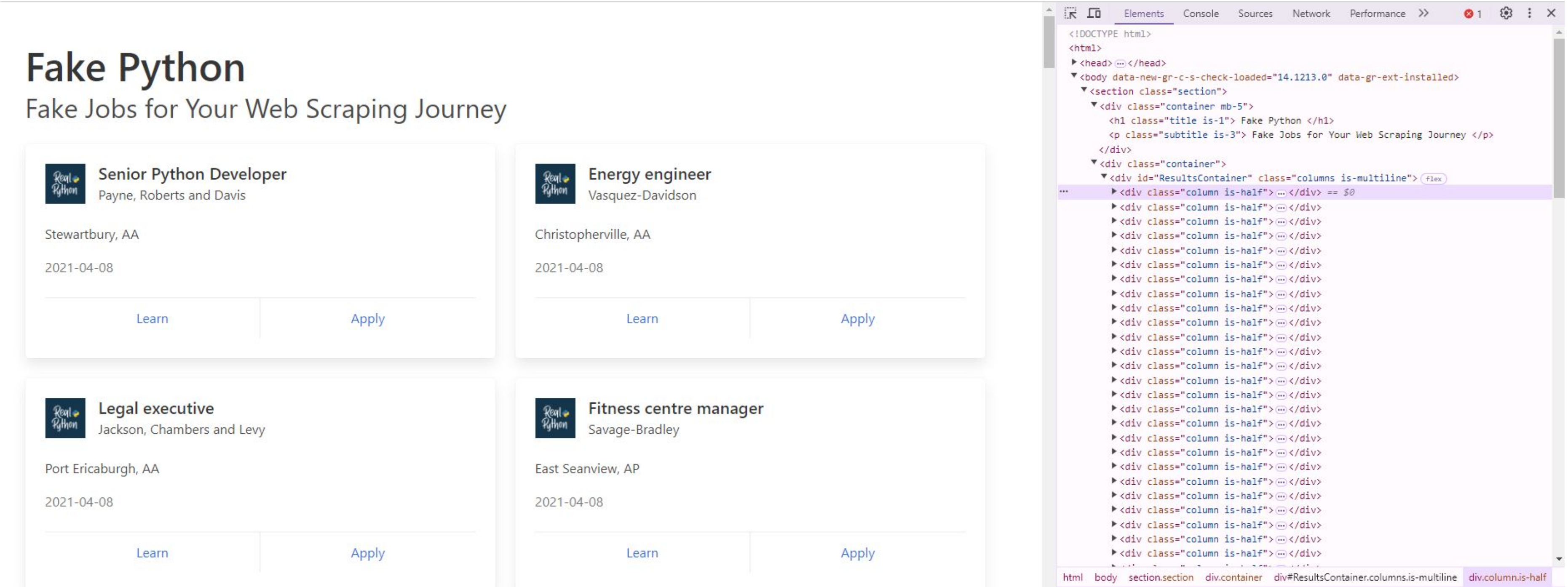
After getting an understanding of the URL and structure, we can then move on to **inspecting the HTML of the website.** This will give us an understanding of the tags we will need to access to scrape pertinent data.

We can do this by selecting “**Inspect**” from the menu that appears when we right-click on the page itself.



```
<!DOCTYPE html>
<html>
 ▶<head> … </head>
 ▼<body data-new-gr-c-s-check-loaded="14.1213.0" data-gr-ext-installed>
   ▼<section class="section">
     ▼<div class="container mb-5">
        <h1 class="title is-1"> Fake Python </h1>
        <p class="subtitle is-3"> Fake Jobs for Your Web Scraping Journey </p>
      </div>
     ▼<div class="container">
       ▼<div id="ResultsContainer" class="columns is-multiline"> flex
...       ▶<div class="column is-half"> … </div> == $0
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
```

html   body   section.section   div.container   div#ResultsContainer.columns.is-multiline   div.column.is-half



Fake Python

Fake Jobs for Your Web Scraping Journey

Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA
2021-04-08

Energy engineer
Vasquez-Davidson
Christopherville, AA
2021-04-08

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA
2021-04-08

Fitness centre manager
Savage-Bradley
East Seanview, AP
2021-04-08

```
<!DOCTYPE html>
<html>
 ▶<head> … </head>
 ▼<body data-new-gr-c-s-check-loaded="14.1213.0" data-gr-ext-installed>
   ▼<section class="section">
     ▼<div class="container mb-5">
        <h1 class="title is-1"> Fake Python </h1>
        <p class="subtitle is-3"> Fake Jobs for Your Web Scraping Journey </p>
      </div>
     ▼<div class="container">
       ▼<div id="ResultsContainer" class="columns is-multiline"> flex
...       ▶<div class="column is-half"> … </div> == $0
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
          ▶<div class="column is-half"> … </div>
```

html body section.section div.container div#ResultsContainer.columns.is-multiline div.column.is-half

By viewing the HTML, we can **anticipate** the HTML tags that we will need to **access** when coding our web-scraping script. Here we see that all of our job cards are inside of a tag called “**div**”, with the id “**ResultsContainer**.”

Web Scraping – Pulling HTML

In order to scrape our own web-pages, we first need to “*request*” the HTML page itself to pull it **programmatically** into our computer, does anyone recall which Python package we can use to make **HTTP GET** request?

Web Scraping – Pulling HTML

We use the **requests** module once more, along with the **get** method. However, this time, instead of accessing a ***Web API***, we are simply pulling an **HTML page** as if we are browsing the web.

```
r = requests.get("https://realpython.github.io/fake-jobs/")
```

Web Scraping – Pulling HTML

Now that we have pulled our HTML web-page, we can print out the **raw binary data** from this request using the “**.content**” field

Notice that this gives you the HTML content along with a **b**”” in the front.

This is what we call a **binary string**. Each character represents a byte (8 bits)

```
URL = "https://realpython.github.io/fake-jobs/"

page = requests.get(URL )

page = r.content

print(page)
```

```
b'<!DOCTYPE html>\n<html>\n  <head>\n...’
```

This is the first object we access (the **root**) before analyzing the rest of the web-page.

Parsing HTML Using BeautifulSoup4

However, we still need a way to **interpret** this structure. We will use **BeautifulSoup** to parse this HTML structure.

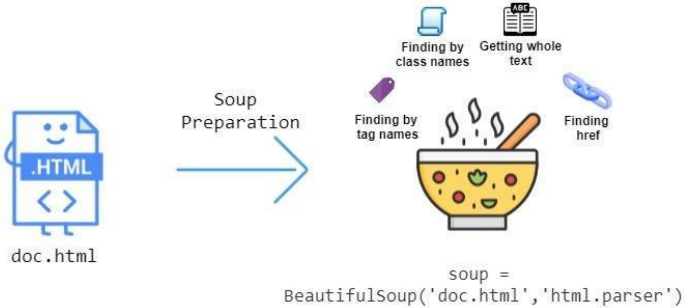
First we create a BeautifulSoup object by passing in the **page.content** as the **first** argument, with **“html.parser”** as the **second** argument.

This extracts **key attributes** from this page, which we can then access via **methods** and **fields**.

```
URL = "https://realpython.github.io/fake-jobs/"

page = requests.get(URL )

soup = BeautifulSoup(page.content, "html.parser")
```



```
soup =
BeautifulSoup('doc.html','html.parser')
```

Parsing HTML Using BeautifulSoup4

Just like our JSON object, we must access our web-page hierarchically (with soup as the entry point).

The two most common methods that you will use when parsing a web-page are:

- **find**(*name*, *attributes*) : find one element
- **find_all**(*name*, *attributes*) : find all elements

We use these methods to **find unique tags** inside of web-pages and the data located inside of these tags.

```
URL = "https://realpython.github.io/fake-jobs/"

page = requests.get(URL )

soup = BeautifulSoup(page.content, "html.parser")

results = soup.find(id="...")
```

What is label of the tag that contains all of our job data?

The BS4 Application Programming Interface

We can specify the **type of tag we want to extract**, as well as the **class** or **id** of the tag by using different positional arguments.

When we call functions or methods, we can **mix the position of arguments** as long as we specify which argument goes to which parameter.

```
soup.find(“tag-label”, class_=“class name”)
```

Parsing HTML Using BeautifulSoup4

Now that we've extracted the "**ResultsContainer**" tag, we will continue with our pattern of finding tags hidden inside of this new object. Note that we are no longer using the "soup" object.

Which tags contain our job data, and which method should we use to get **all** of these objects?

- **find**(*name*, *attributes*) : find one element
- **find_all**(*name*, *attributes*) : find all elements

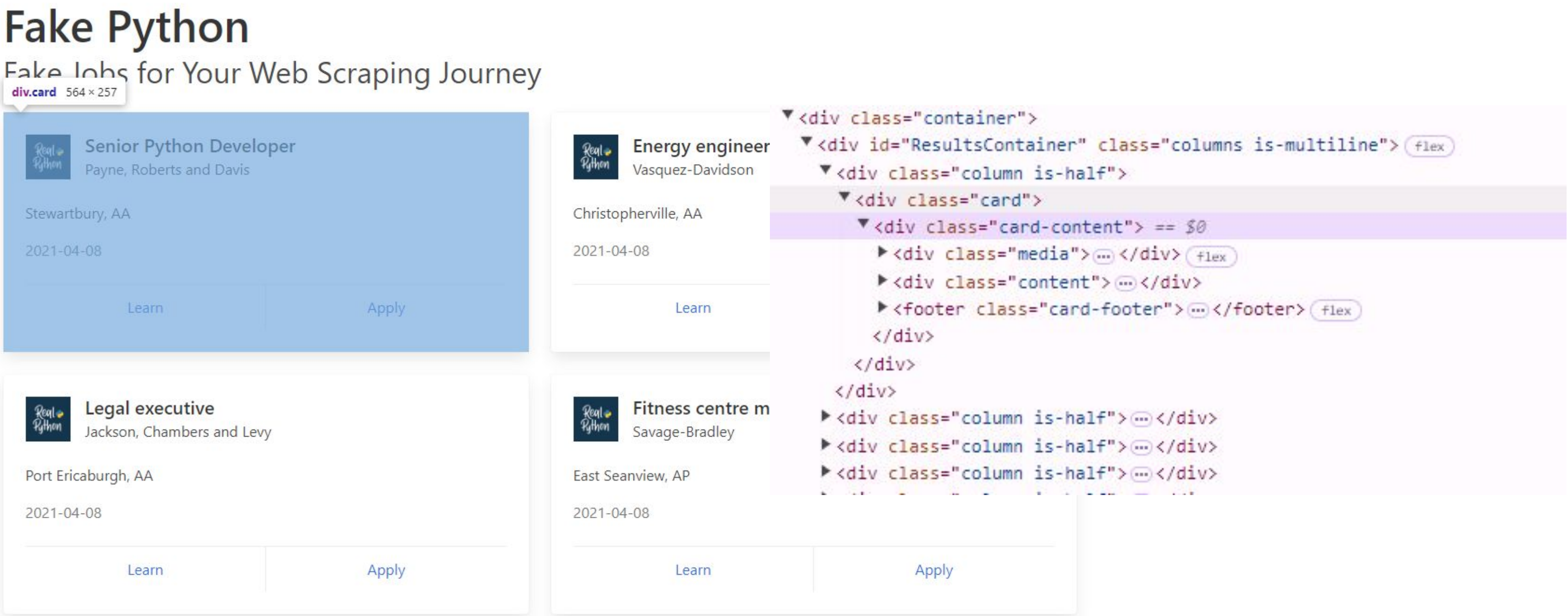
```
URL = "https://realpython.github.io/fake-jobs/"

page = requests.get(URL )

soup = BeautifulSoup(page.content, "html.parser")

results = soup.find(id="ResultsContainer")

job_elements = ...
```



Fake Python

Fake Jobs for Your Web Scraping Journey

Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA
2021-04-08

Energy engineer
Vasquez-Davidson
Christopherville, AA
2021-04-08

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA
2021-04-08

Fitness centre m
Savage-Bradley
East Seanview, AP
2021-04-08

```
▼<div class="container">
  ▼<div id="ResultsContainer" class="columns is-multiline"> flex
    ▼<div class="column is-half">
      ▼<div class="card">
        ▼<div class="card-content"> == $0
          ▶<div class="media">…</div> flex
          ▶<div class="content">…</div>
          ▶<footer class="card-footer">…</footer> flex
          </div>
        </div>
      </div>
    ▶<div class="column is-half">…</div>
    ▶<div class="column is-half">…</div>
    ▶<div class="column is-half">…</div>
```

Inspecting our web-page, we can see that the div that contains our content is a class called “**card**.” You may want to try using the “**column is-hal**f” div, but we always want to opt for the last possible div before accessing the content itself, but feel free to experiment with different approaches.

Parsing HTML Using BeautifulSoup4

Using the **find_all()** method, we can specify that we want to find all divs of class “**card**” inside of the “**results**” object.

We’ll focus in on extracting the **title**, **company**, and **location** of each job.

How can we iterate on this list of job elements?

```
URL = "https://realpython.github.io/fake-jobs/"

page = requests.get(URL )

soup = BeautifulSoup(page.content, "html.parser")

results = soup.find(id="ResultsContainer")

job_elements = results.find_all(“div”, class_=”card”)

    [“<div class="card-content">...”, “<div
    class="card-content">...”, “<div
    class="card-content">...”]
```

Parsing HTML Using BeautifulSoup4

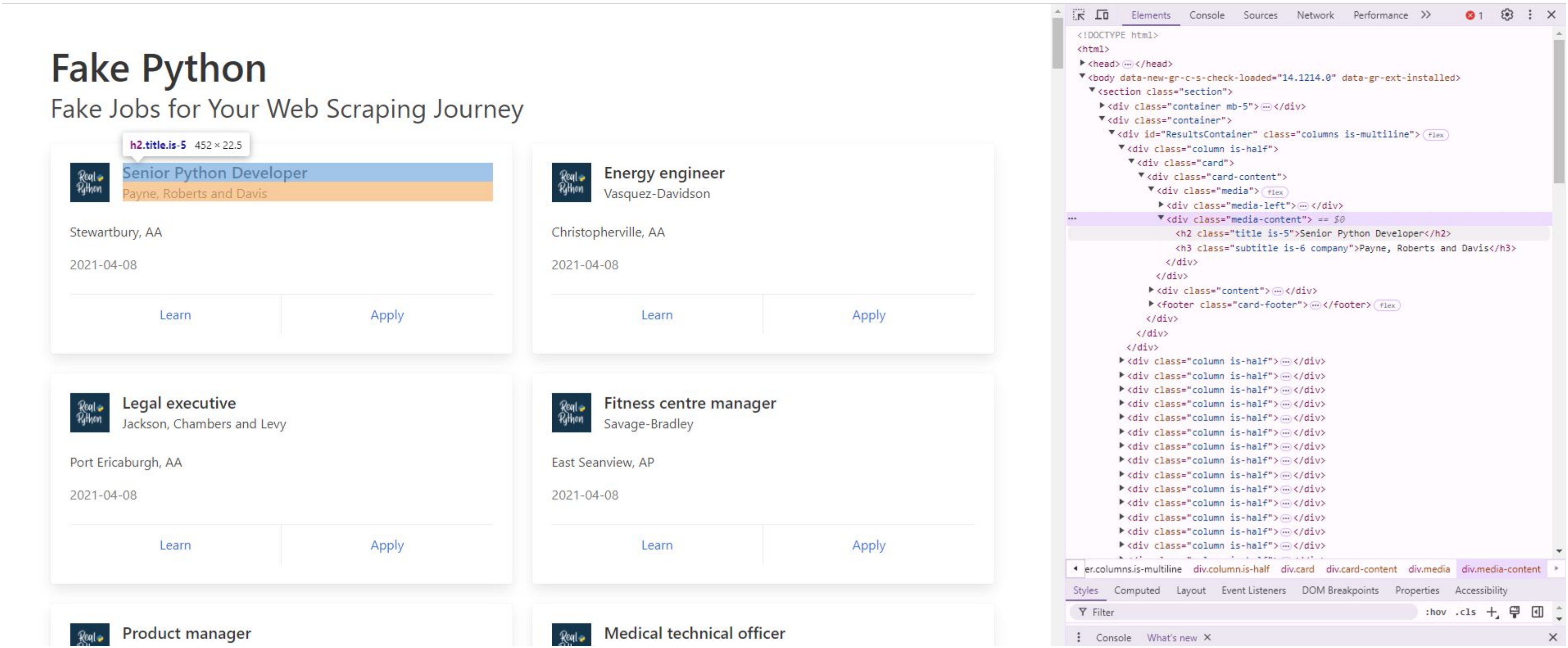
By looping through each element in our list, we can continue to use the **find()** and **find_all()** methods as we search for our targeted data.

Which tags contain information about **job title**, **company**, and **location**?

Again, let’s look back to our HTML structure.

```
job_elements = results.find_all(“div”, class_=”card”)

for job in job_elements:
        title = job.find(...)
        company = job.find(...)
        location = job.find(...)
```



Fake Python

Fake Jobs for Your Web Scraping Journey

Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA
2021-04-08
Learn | Apply

Energy engineer
Vasquez-Davidson
Christopherville, AA
2021-04-08
Learn | Apply

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA
2021-04-08
Learn | Apply

Fitness centre manager
Savage-Bradley
East Seanview, AP
2021-04-08
Learn | Apply

Product manager

Medical technical officer

```
<!DOCTYPE html>
<html>
 ▸<head>…</head>
 ▾<body data-new-gr-c-s-check-loaded="14.1214.0" data-gr-ext-installed>
   ▾<section class="section">
     ▸<div class="container mb-5">…</div>
     ▾<div class="container">
       ▾<div id="ResultsContainer" class="columns is-multiline"> flex
         ▾<div class="column is-half">
           ▾<div class="card">
             ▾<div class="card-content">
               ▾<div class="media"> flex
                 ▸<div class="media-left">…</div>
                 ▾<div class="media-content"> == $0
                    <h2 class="title is-5">Senior Python Developer</h2>
                    <h3 class="subtitle is-6 company">Payne, Roberts and Davis</h3>
                  </div>
                </div>
               ▸<div class="content">…</div>
               ▸<footer class="card-footer">…</footer> flex
              </div>
            </div>
          </div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
         ▸<div class="column is-half">…</div>
```

Once again, we find the **last tag** that contains the title in our “card” divs. What **kind of tag** contains our title, which type of **class** is this? We do the same kind of search for company and location.

Parsing HTML Using BeautifulSoup4

Now that we’ve identified the tags that contain our pertinent information, we can stop our **recursion** and simply extract the text located in each tag via the **.text** attribute.

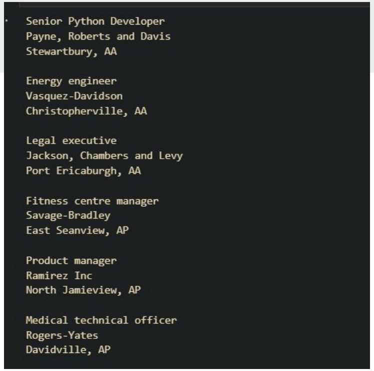
We also call the **.strip()** method to remove any remaining **new-line** or **white-space** characters.

```
job_elements = results.find_all(“div”, class_=”card”)

for job in job_elements:
        title = job.find(”h2”, class_=”title”)
        company = job.find(“h2”, class_=”company”)
        location = job.find(“h2”, class_=”location”)

        print(title.text.strip())
        print(company .text.strip())
        print(location .text.strip())
```

Parsing HTML Using BeautifulSoup4



```
Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA

Energy engineer
Vasquez-Davidson
Christopherville, AA

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA

Fitness centre manager
Savage-Bradley
East Seanview, AP

Product manager
Ramirez Inc
North Jamieview, AP

Medical technical officer
Rogers-Yates
Davidville, AP
```

While this is sufficient to get all job information from our HTML, what if we wanted to instead **filter** our HTML tags and instead find only roles that have certain keywords?

For example, if we were interested in developer careers, we might want to instead look for the keyword “Python.”

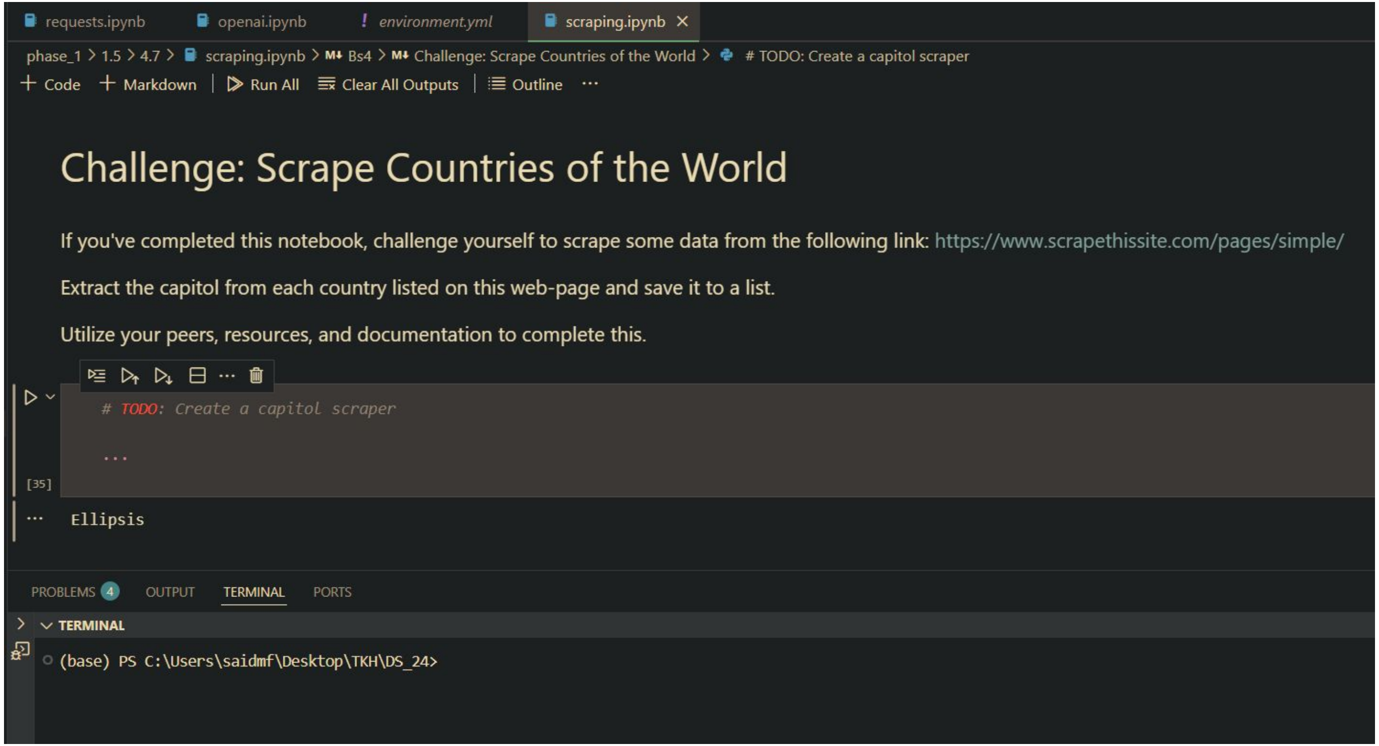
To accomplish this, we use **regular expressions (regex)**

```
job_elements = results.find_all(“div”, class_=”card”)

for job in job_elements:
        title = job.find(“h2”, class_=”title”)
        company = job.find(“h2”, class_=”company”)
        location = job.find(“h2”, class_=”location”)

        print(title.text.strip())
        print(company .text.strip())
        print(location .text.strip())
```

Web Scraping Lab



Challenge: Scrape Countries of the World

If you've completed this notebook, challenge yourself to scrape some data from the following link: https://www.scrapethissite.com/pages/simple/

Extract the capitol from each country listed on this web-page and save it to a list.

Utilize your peers, resources, and documentation to complete this.

```
# TODO: Create a capitol scraper

...
```

[35]

```
Ellipsis
```

```
(base) PS C:\Users\saidmf\Desktop\TKH\DS_24>
```

Using this syntax, get started with your web-scraping lab!

Wrap-Up

Lab (Due 04/21)

Vancouver, Canada

You are a growth analyst at a Vancouver-based consulting firm called Monica Group. Your manager is spearheading the completion of a a new analytical tool which will automatically label if a review is positive, neutral, negative, or irrelevant.

You will be kicking off completion of this milestone by independently implementing a minimal-viable-product. **This will be a Python pipeline that ingests a text-file of review data and interfaces with the Open AI API in order to automatically label each review.**

We released API keys.

Tuesday

Tuesday will entail:

- More web-scraping!
- Regular expressions

Jupyter: scratchpad of the data scientist

If you understand what you're doing, you're not learning anything. - Anonymous