INT 303 BIG DATA ANALYTICS

Lecture4: Get the Data

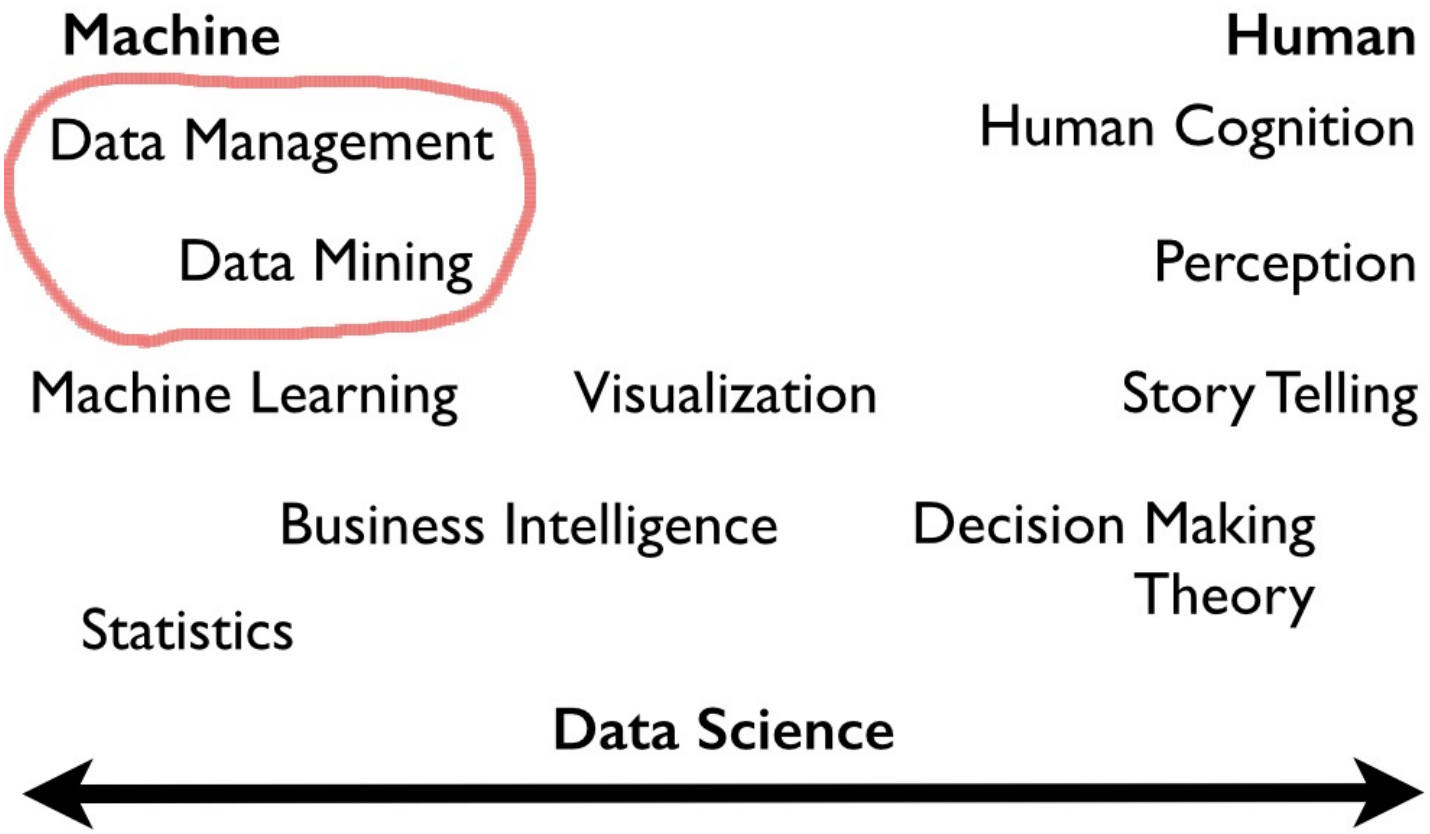
Jia WANG

Jia.wang02@xjtlu.edu.cn

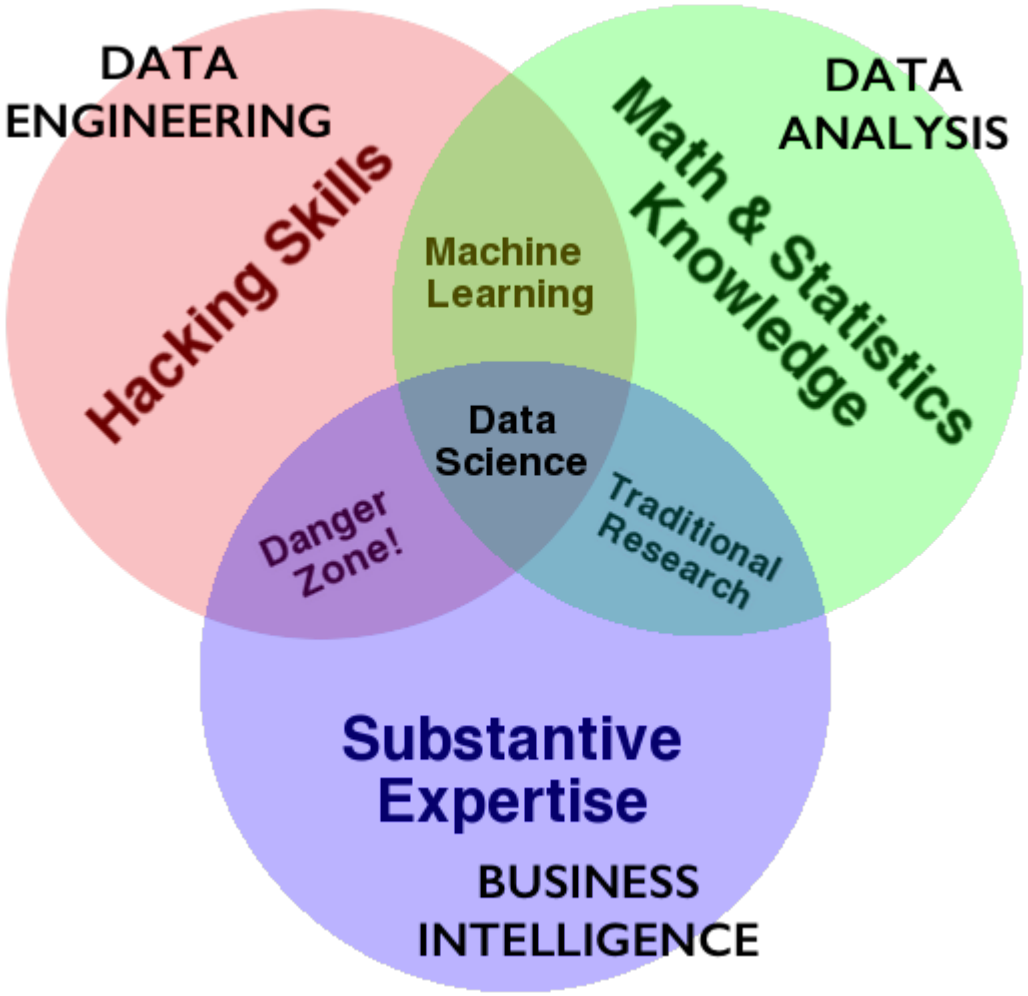
Xi'an Jiaotong-Liverpool University
西交利物浦大學

OUTLINE

- What is Web Service?
- Data Scraping
- Gathering data from APIs



Inspired by Daniel Keim, "Visual Analytics: Definition, Process, and Challenges"

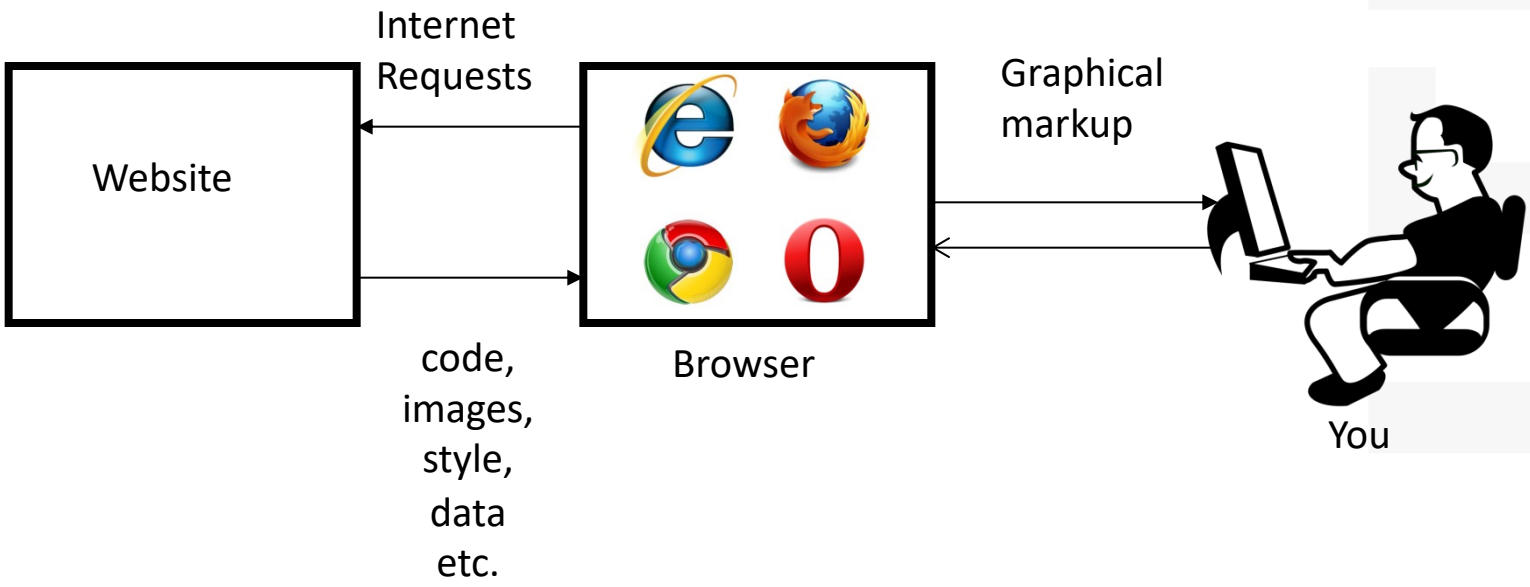

DATA
ENGINEERING
Hacking Skills
DATA
ANALYSIS
Math & Statistics
Knowledge
Machine
Learning
Data
Science
Danger
Zone!
Traditional
Research
Substantive
Expertise
BUSINESS
INTELLIGENCE

Web Servers

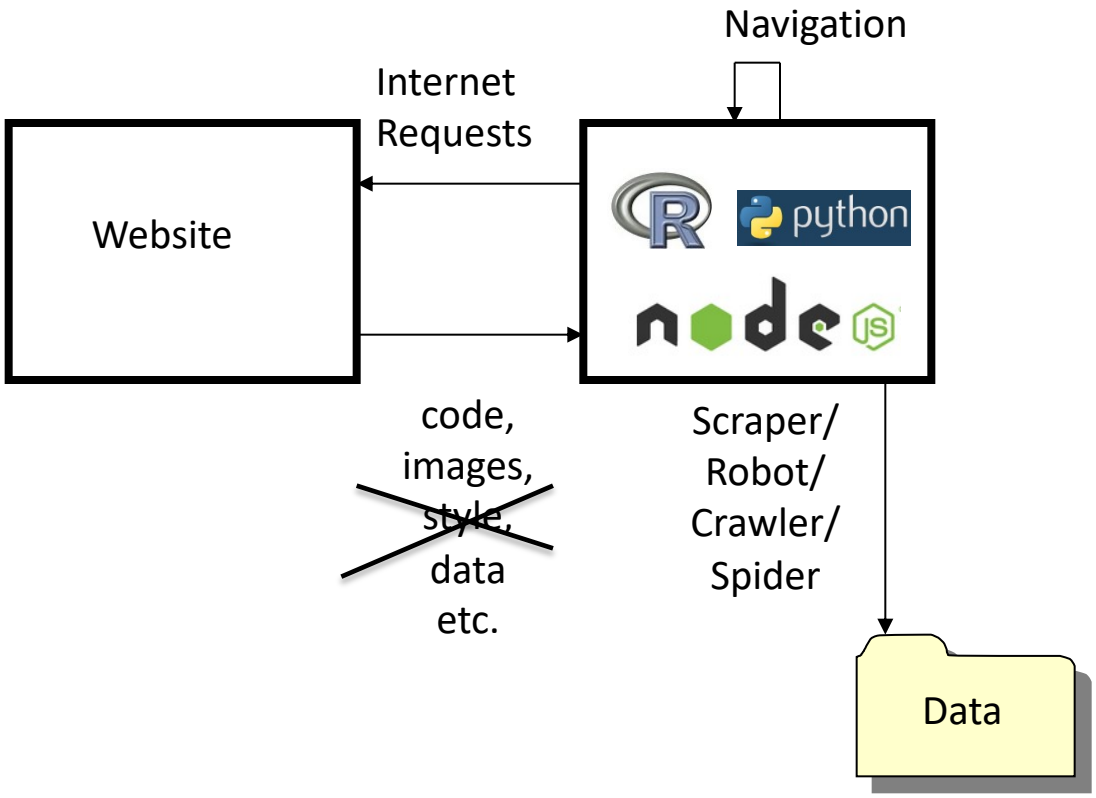
WEB SERVERS

- A server is a long running process (also called daemon) which listens on a pre-specified port
- and responds to a request, which is sent using a protocol called HTTP
- A browser parses the url.

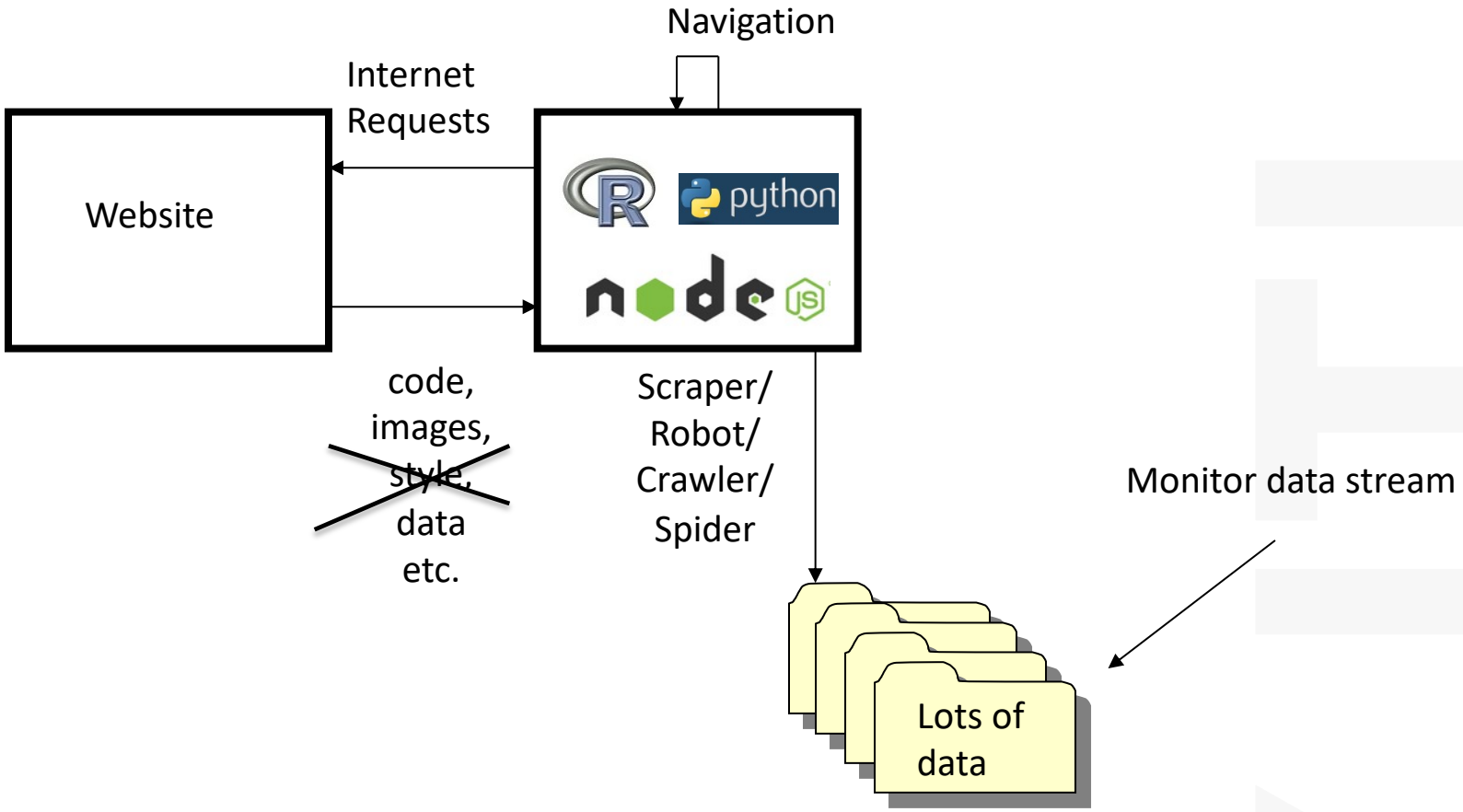
HOW IT WORKS (1)



HOW IT WORKS (1)



HOW IT WORKS (3)



WEB SERVERS

- **Example:**
- Our notebooks also talk to a local web server on our machines: `http://localhost:8888/Documents/cs109/BLA.ipynb#something`
- protocol is http, hostname is localhost, port is 8888
- url is /Documents/cs109/BLA.ipynb
- url fragment is #something
- Request is sent to localhost on port 8888. It says:
- `Request: GET /request-URI HTTP/version`

Example with Response: Google

```
GET / HTTP/1.0
Host: www.google.com

HTTP/1.0 200 OK
Date: Mon, 14 Nov 2016 04:49:02 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is ..."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=90=gb5q7b0...; expires=Tue, 16-May-2017 04:49:02 GMT;
path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding

<!doctype html><html itemscope=""
itemtype="http://schema.org/WebPage" lang="en">
<head><meta content="Search the world's information,
```

HTTP STATUS CODES[1]

200 OK:

Means that the server did whatever the client wanted it to, and all is well.

400: Bad request

The request sent by the client didn't have the correct syntax.

401: Unauthorized

Means that the client is not allowed to access the resource. This may change if the client retries with an authorization header.

403: Forbidden

The client is not allowed to access the resource and authorizaton will not help.

404: Not found

Seen this one before? :) It means that the server has not heard of the resource and has no further clues as to what the client should do about it. In other words: dead link.

500: Internal server error

Something went wrong inside the server.

501: Not implemented

The request method is not supported by the server.

[1]from http://www.garshol.priv.no/download/text/http-tut.htm)

WEB SERVERS

- **Requests:**
- great module built into python for http requests
- `req=requests.get("https://en.wikipedia.org/wiki/Harvard_University")`

- <Response [200]>
- page = req.text
- `'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta charset="UTF-8"/>\n<title>Harvard University - Wikipedia</title>\n<script>document.documentElement.className=document.documentElement.className.replace( /(^|\\s)client-nojs(\\s|$)/,"$1client-js$2"`
- `);</script>\n<script>(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({`
- `"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber"`
- `:0,"wgPageName":"Harvard_University","wgTitle":"Harva...'`

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Article Talk Read View source View history Search Wikipedia

Wiki Loves Monuments: The world's largest photography competition is now open!
Photograph a historic site, learn more about our history, and win prizes.

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export

Harvard University

From Wikipedia, the free encyclopedia

Coordinates: 42°22′28″N 71°07′01″W

"Harvard" redirects here. For other uses, see Harvard (disambiguation).

Harvard University is a private Ivy League research university in Cambridge, Massachusetts, established in 1636, whose history, influence, and wealth have made it one of the world's most prestigious universities.[7]

Established originally by the Massachusetts legislature and soon thereafter named for John Harvard (its first benefactor), Harvard is the United States' oldest institution of higher learning,[8] and the Harvard Corporation (formally, the *President and Fellows of Harvard College*) is its first chartered corporation. Although

Harvard University

Latin: *Universitas Harvardiana*

Former names	Harvard College
Motto	*Veritas*[1]
Motto in English	Truth
Type	Private research
Established	1636[2]
Endowment	$34.541 billion (2016)[3]

Python data scraping

PYTHON DATA SCRAPING

- Why scrape the web?
  - companies have not provided APIs
  - automate tasks
  - keep up with sites
  - fun!

CHALLENGES IN WEB SCRAPING

- Which data?
  - It is not always easy to know which site to scrape
  - Which data is relevant, up to date, reliable?
- The internet is dynamic
  - Each web site has a particular structure, which may be changed anytime
- Data is volatile
  - Be aware of changing data patterns over time

LEGAL

- Privacy:
  - Legislation on protection of personal information
  - At this moment we only scrape public sources
- Netiquette (practical):
  - respect the Robots Exclusion Protocol also known as the robots.txt (example)
  - identify yourself (user-agent)
  - do not overload servers, use some idle time between requests, run crawlers at night / morning
  - Inform website owners if feasible

NOTICE

- **copyrights and permission:**
  - be careful and polite
  - give credit
  - care about media law
  - don't be evil (no spam, overloading sites, etc.)

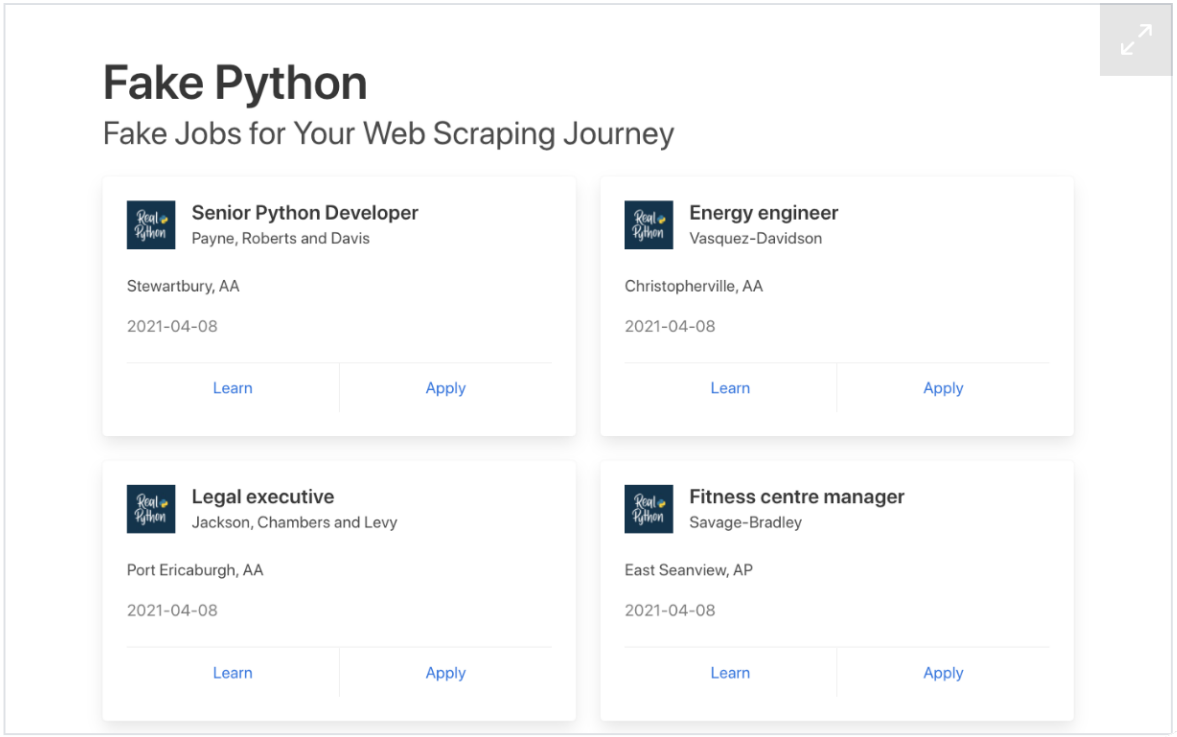
ROBOTS.TXT

- specified by web site owner
- gives instructions to web robots (aka your script)
- is located at the top-level directory of the web server
- e.g.: http://google.com/robots.txt

STEP 1: INSPECT YOUR DATA SOURCE

- Explore the Website

Click through the site and interact with it just like any typical job searcher would. For example, you can scroll through the main page of the website:



https://realpython.github.io/fake-jobs/

DEVELOPER TOOLS

- ctrl/cmd shift- i in chrome
- cmd-option-i in safari
- look for "inspect element"
- locate details of tags

- **Mac:** ⌘ Cmd + Alt + I
- **Windows/Linux:** ^ Ctrl + ⇧ Shift + I

STEP 2: SCRAPE HTML CONTENT FROM A PAGE

Python

```python
import requests

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

print(page.text)
```

This code issues an HTTP GET request to the given URL. It retrieves the HTML data that the server sends back and stores that data in a Python object.

You successfully fetched the static site content from the Internet!

STEP 3: PARSE HTML CODE WITH BEAUTIFUL SOUP

Python

```python
import requests
from bs4 import BeautifulSoup

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

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

FIND ELEMENTS BY ID

The element you're looking for is a `<div>` with an `id` attribute that has the value `"ResultsContainer"`. It has some other attributes as well, but below is the gist of what you're looking for:

HTML

```
<div id="ResultsContainer">
  <!-- all the job listings -->
</div>
```

Beautiful Soup allows you to find that specific HTML element by its ID:

Python

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

FINDALL VS. FIND

- will normalize dirty html
- basic usage

```
import bs4
## get bs4 object
soup = bs4.BeautifulSoup(source)
## all a tags
soup.findAll('a')
## first a
soup.find('a')
## get all links in the page
link_list = [l.get('href') for l in soup.findAll('a')]
```

http://books.toscrape.com/index.html

FIND ELEMENTS BY HTML CLASS NAME

- You've seen that every job posting is wrapped in a element with the class card-content.

Python

```
job_elements = results.find_all("div", class_="card-content")
```

Here, you call `.find_all()` on a Beautiful Soup object, which returns an iterable containing all the HTML for all the job listings displayed on that page.

Take a look at all of them:

Python

```
for job_element in job_elements:
    print(job_element, end="\n"*2)
```

FIND ELEMENTS BY HTML CLASS NAME

Python

```python
for job_element in job_elements:
    title_element = job_element.find("h2", class_="title")
    company_element = job_element.find("h3", class_="company")
    location_element = job_element.find("p", class_="location")
    print(title_element)
    print(company_element)
    print(location_element)
    print()
```

HTML

```html
<h2 class="title is-5">Senior Python Developer</h2>
<h3 class="subtitle is-6 company">Payne, Roberts and Davis</h3>
<p class="location">Stewartbury, AA</p>
```

EXTRACT TEXT FROM HTML ELEMENTS

Python

```python
for job_element in job_elements:
    title_element = job_element.find("h2", class_="title")
    company_element = job_element.find("h3", class_="company")
    location_element = job_element.find("p", class_="location")
    print(title_element.text.strip())
    print(company_element.text.strip())
    print(location_element.text.strip())
    print()
```

- You can add .text to a Beautiful Soup object to return only the **text content** of the HTML elements that the object contains.
- you can .strip() the superfluous whitespace.

FIND ELEMENTS BY CLASS NAME AND TEXT CONTENT

HTML

```
<h2 class="title is-5">Senior Python Developer</h2>
<h3 class="subtitle is-6 company">Payne, Roberts and Davis</h3>
<p class="location">Stewartbury, AA</p>
```

You know that job titles in the page are kept within <h2> elements. To filter for only specific jobs, you can use the `string` argument:

Python

```
python_jobs = results.find_all("h2", string="Python")
```

PASS A FUNCTION TO A BEAUTIFUL SOUP METHOD

Python

```
python_jobs = results.find_all(
    "h2", string=lambda text: "python" in text.lower()
)
```

Now you're passing an **anonymous function** to the `string=` argument. The lambda function looks at the text of each <h2> element, converts it to lowercase, and checks whether the substring `"python"` is found anywhere. You can check whether you managed to identify all the Python jobs with this approach:

Python >>>

```
>>> print(len(python_jobs))
10
```

HTML IS A TREE

```
tree = bs4.BeautifulSoup(source)

## get html root node
root_node = tree.html
## get head from root using contents
head = root_node.contents[0]
## get body from root
body = root_node.contents[1]
## could directly access body
tree.body
```

ACCESS PARENT ELEMENTS

With this information in mind, you can now use the elements in `python_jobs` and fetch their great-grandparent elements instead to get access to all the information you want:

Python

```
python_jobs = results.find_all(
    "h2", string=lambda text: "python" in text.lower()
)

python_job_elements = [
    h2_element.parent.parent.parent for h2_element in python_jobs
]
```

EXTRACT ATTRIBUTES FROM HTML ELEMENTS

HTML

```
    <!-- snip -->
    <footer class="card-footer">
        <a href="https://www.realpython.com" target="_blank"
           class="card-footer-item">Learn</a>
        <a href="https://realpython.github.io/fake-jobs/jobs/senior-python-dev
           target="_blank"
           class="card-footer-item">Apply</a>
    </footer>
  </div>
</div>
```

Start by fetching all the <a> elements in a job card. Then, extract the value of their `href` attributes using square-bracket notation:

Python

```
for job_element in python_job_elements:
    # -- snip --
    links = job_element.find_all("a")
    for link in links:
        link_url = link["href"]
        print(f"Apply here: {link_url}\n")
```

PROJECT EXAMPLE

- https://github.com/alirezamika/autoscraper
- https://github.com/scrapy/scrapy
- https://yasoob.me/posts/github-actions-web-scraper-schedule-tutorial/

Gathering data from APIs

API

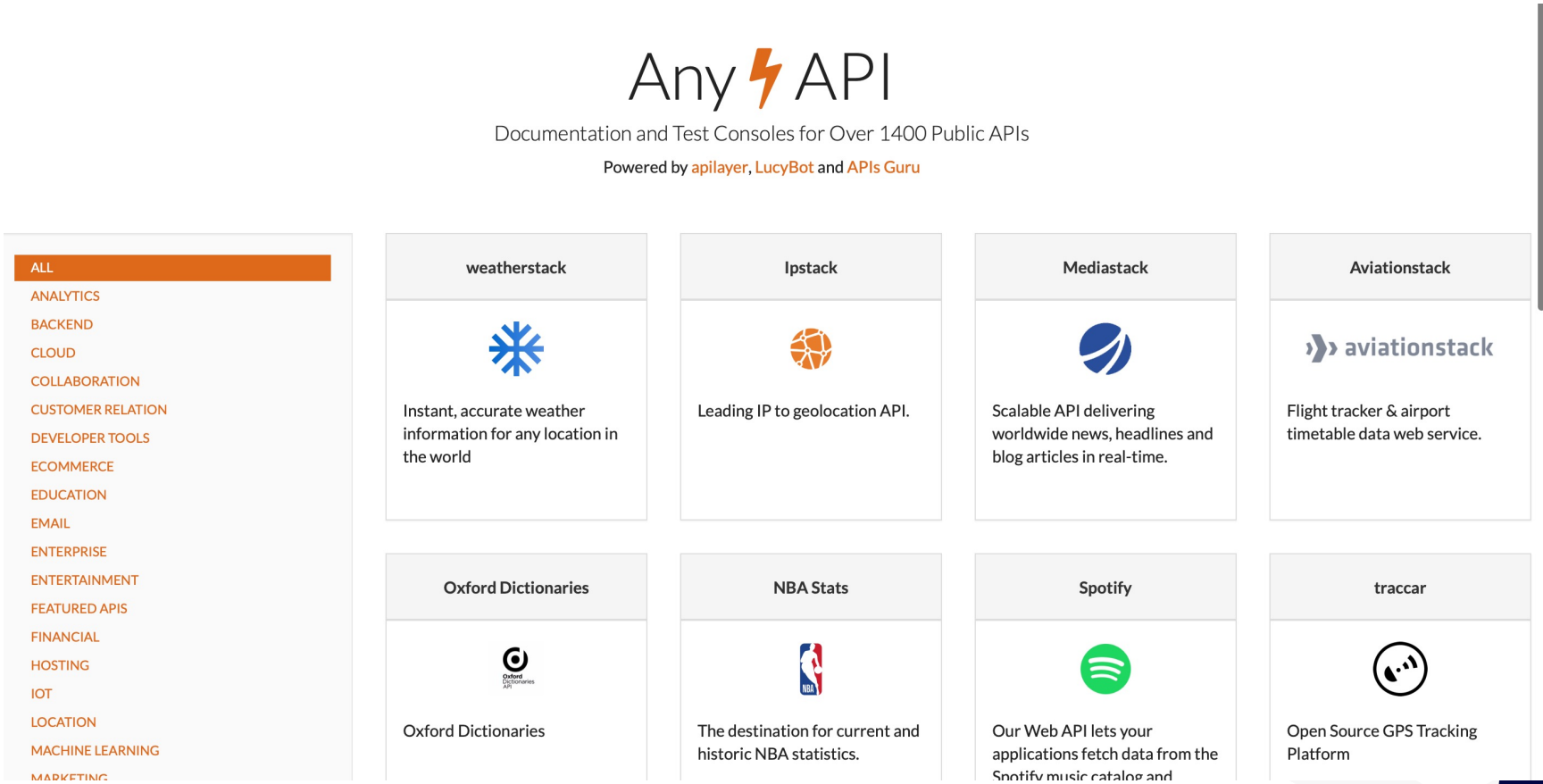
- API = **A**pplication **P**rogram **I**nterface
- Many data sources have API's - largely for talking to other web interfaces

- Consists of a set of methods to search, retrieve, or submit data to, a data source

- Many packages already connect to well-known API's (we'll look at a couple today)

PUBLIC API

https://any-api.com



THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大學