Introduction to Web Scraping

In python™

PyMalta, 11th June 2019

🔥 https://trailblaze.software/

My Details

- Simon Agius Muscat
- Freelance Software Engineering @ https://trailblaze.software/
- Available for:
  - Full-stack Web Development
  - Mobile Application Development
  - Prototype Development
  - Bespoke tech talks and workshops
  - Technical mentorship
- Software Engineer @ RightBrain http://rightbrain.com.mt/
- GitHub: https://github.com/purrcat259
- Twitter: https://twitter.com/purrcat259

Before we start…
A Legal Disclaimer

I am not a lawyer
This is not legal advice
Scraping can exist in a legally grey area
Your use of scraping is **your responsibility**

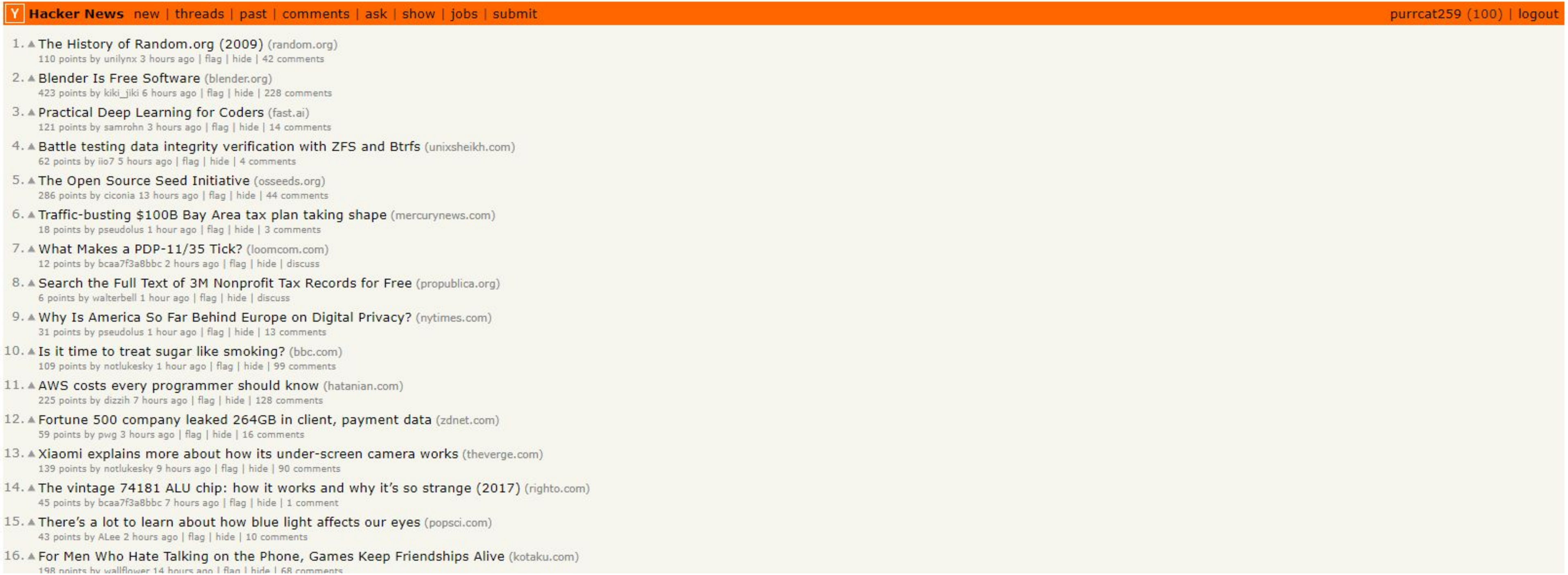
Introduction to Web Scraping

Making Eggs from a Cake

🔥 https://trailblaze.software/

The World Wide Web *is* data

Most of it is optimised for *human* consumption https://news.ycombinator.com/



Y **Hacker News** new | threads | past | comments | ask | show | jobs | submit purrcat259 (100) | logout

1. ▲ The History of Random.org (2009) (random.org)
   110 points by unilynx 3 hours ago | flag | hide | 42 comments
2. ▲ Blender Is Free Software (blender.org)
   423 points by kiki_jiki 6 hours ago | flag | hide | 228 comments
3. ▲ Practical Deep Learning for Coders (fast.ai)
   121 points by samrohn 3 hours ago | flag | hide | 14 comments
4. ▲ Battle testing data integrity verification with ZFS and Btrfs (unixsheikh.com)
   62 points by iio7 5 hours ago | flag | hide | 4 comments
5. ▲ The Open Source Seed Initiative (osseeds.org)
   286 points by ciconia 13 hours ago | flag | hide | 44 comments
6. ▲ Traffic-busting $100B Bay Area tax plan taking shape (mercurynews.com)
   18 points by pseudolus 1 hour ago | flag | hide | 3 comments
7. ▲ What Makes a PDP-11/35 Tick? (loomcom.com)
   12 points by bcaa7f3a8bbc 2 hours ago | flag | hide | discuss
8. ▲ Search the Full Text of 3M Nonprofit Tax Records for Free (propublica.org)
   6 points by walterbell 1 hour ago | flag | hide | discuss
9. ▲ Why Is America So Far Behind Europe on Digital Privacy? (nytimes.com)
   31 points by pseudolus 1 hour ago | flag | hide | 13 comments
10. ▲ Is it time to treat sugar like smoking? (bbc.com)
    109 points by notlukesky 1 hour ago | flag | hide | 99 comments
11. ▲ AWS costs every programmer should know (hatanian.com)
    225 points by dizzih 7 hours ago | flag | hide | 128 comments
12. ▲ Fortune 500 company leaked 264GB in client, payment data (zdnet.com)
    59 points by pwg 3 hours ago | flag | hide | 16 comments
13. ▲ Xiaomi explains more about how its under-screen camera works (theverge.com)
    139 points by notlukesky 9 hours ago | flag | hide | 90 comments
14. ▲ The vintage 74181 ALU chip: how it works and why it’s so strange (2017) (righto.com)
    45 points by bcaa7f3a8bbc 7 hours ago | flag | hide | 1 comment
15. ▲ There’s a lot to learn about how blue light affects our eyes (popsci.com)
    43 points by ALee 2 hours ago | flag | hide | 10 comments
16. ▲ For Men Who Hate Talking on the Phone, Games Keep Friendships Alive (kotaku.com)
    198 points by wallflower 14 hours ago | flag | hide | 68 comments

It is not always available for *machine* consumption https://hacker-news.firebaseio.com/v0/askstories.json?print=pretty

https://hacker-news.firebaseio.com/v0/item/20141052.json

```
[
   20141052,
   20138189,
   20138607,
   20137475,
   20123659,
   20132561,
   20137429,
   20133806,
   20138436,
   20134221,
   20118963,
   20136555,
   20136093,
   20129998,
   20137462,
   ...
```

```
{
   "by": "eterps",
   "descendants": 1,
   "id": 20141052,
   "kids": [
      20141071
   ],
   "score": 2,
   "time": 1560106131,
   "title": "Ask HN: Favorite cross platform
   lang/framework for command line apps?",
   "type": "story"
}
```

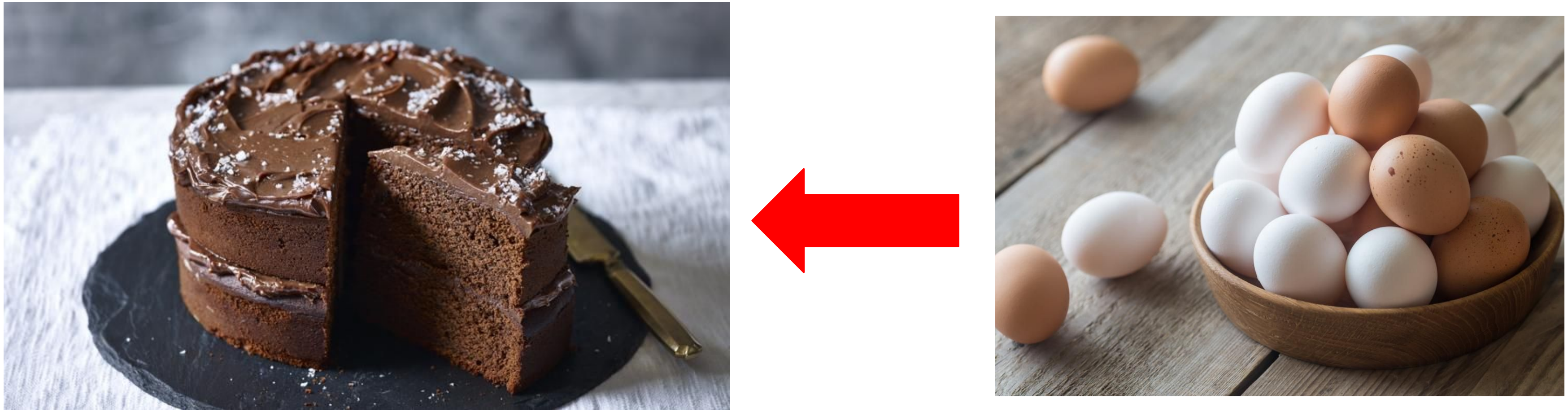
🔥 https://trailblaze.software/

Enter: **Web Scraping**

Reverse engineering the transformation of the data in a database (or other source) to the final view visible within the browser when visiting a website

(My definition)

Normal Web Request



Web Scraping

Definitions

Definitions

1. Web: The World Wide Web, which is a method of delivering electronic documents. Most people refer to this as *the internet.*
2. Web Scraping: Extraction of data from websites. This can be manual or automated, both from an API or a website. **For this talk, let us assume this refers to the automated kind, where an API is not available**
3. API: **Application Programming Interface**. A set of rules and methods defining how two machines can interact with each other (typically referred to as “I’ll send you some JSON”)

Website Basics

The *boring* bits you never find in a web dev tutorial

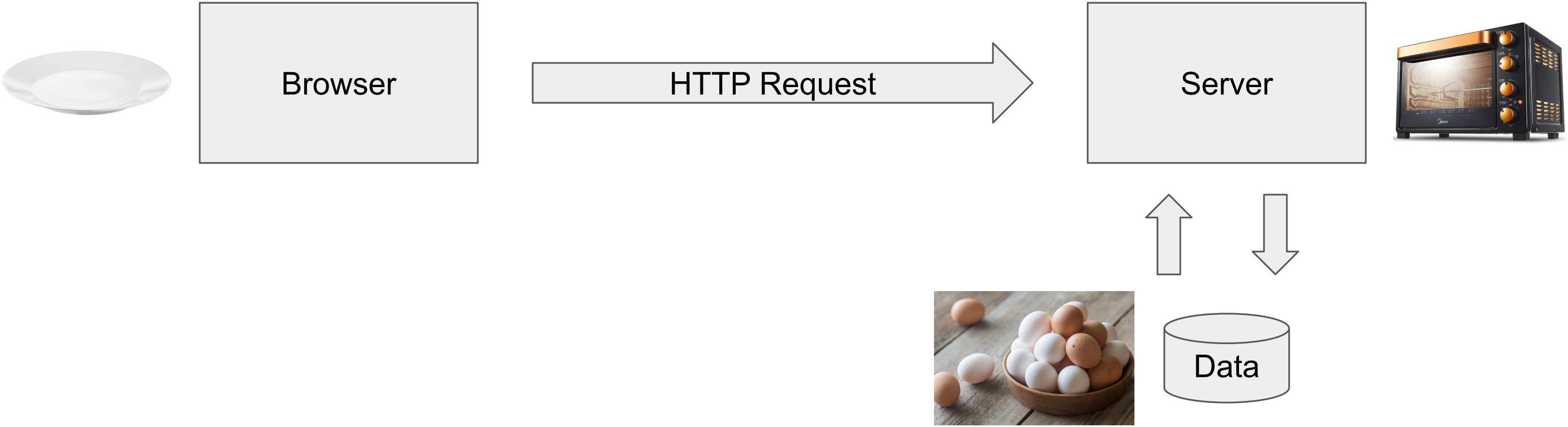
Sir Tim Berners Lee

Invented the World Wide Web at CERN

The WWW needed:

1. HTTP
   a. HyperText Transfer Protocol
   b. A standardised way for machines to have a conversation about documents
2. HTML
   a. HyperText Markup Language
   b. A way to annotate documents with information beyond the textual content

The Request Response cycle



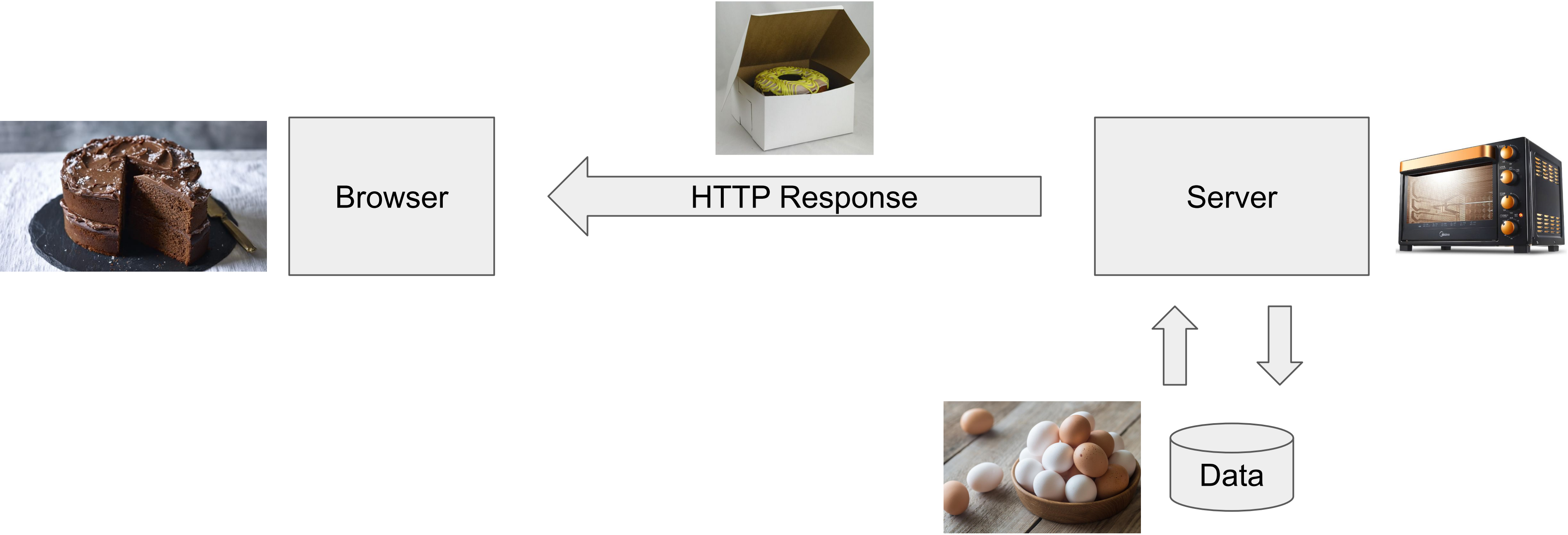
Browser

HTTP Request

Server

Data

The Request Response cycle



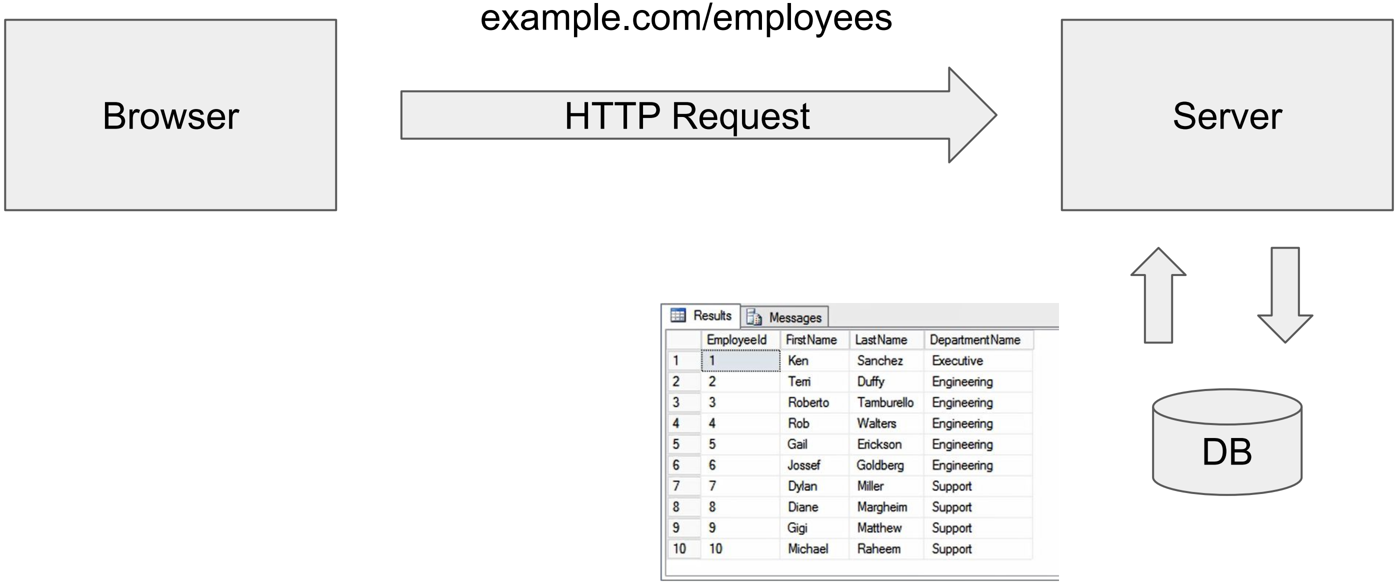
Browser

HTTP Response

Server

Data

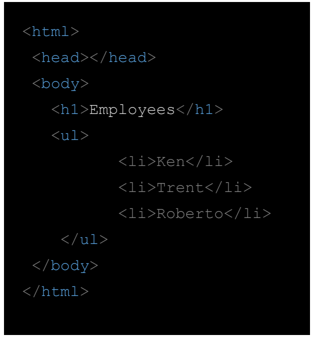
The Request Response cycle



	EmployeeId	FirstName	LastName	DepartmentName
1	1	Ken	Sanchez	Executive
2	2	Terri	Duffy	Engineering
3	3	Roberto	Tamburello	Engineering
4	4	Rob	Walters	Engineering
5	5	Gail	Erickson	Engineering
6	6	Jossef	Goldberg	Engineering
7	7	Dylan	Miller	Support
8	8	Diane	Margheim	Support
9	9	Gigi	Matthew	Support
10	10	Michael	Raheem	Support

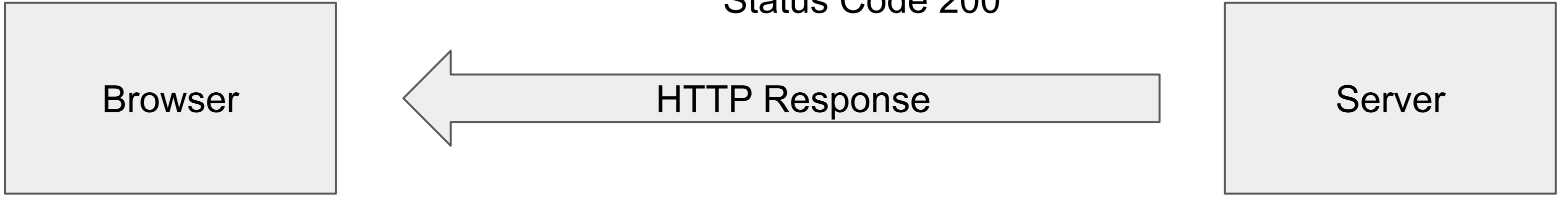
Process highly simplified

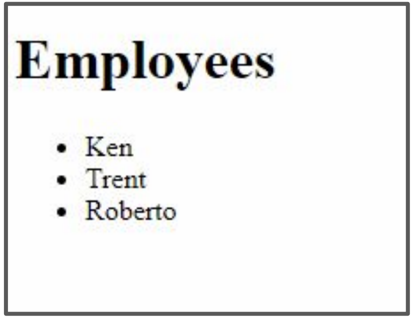
The Request Response cycle



```
<html>
 <head></head>
 <body>
   <h1>Employees</h1>
   <ul>
          <li>Ken</li>
          <li>Trent</li>
          <li>Roberto</li>
    </ul>
 </body>
</html>
```

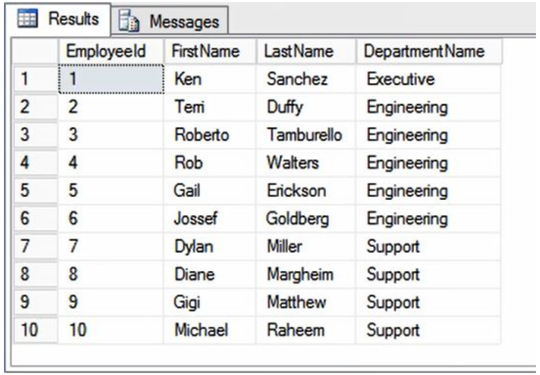
Status Code 200



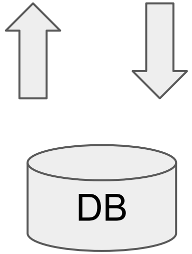


Employees

- Ken
- Trent
- Roberto


Results
Messages

	EmployeeId	FirstName	LastName	DepartmentName
1	1	Ken	Sanchez	Executive
2	2	Terri	Duffy	Engineering
3	3	Roberto	Tamburello	Engineering
4	4	Rob	Walters	Engineering
5	5	Gail	Erickson	Engineering
6	6	Jossef	Goldberg	Engineering
7	7	Dylan	Miller	Support
8	8	Diane	Margheim	Support
9	9	Gigi	Matthew	Support
10	10	Michael	Raheem	Support



🔥 https://trailblaze.software/

Process highly simplified

Time to write a program to do that process instead

We can start by making a simple HTTP request

Web Scraping

1. Scrape the content
    a. Make a request, receive a response
2. Parse the received content
    a. Make sure we can make sense of the data received depending on its format
3. Extract relevant data from the parsed content
    a. Get only what we need out of what we received
4. Store the relevant data in an easier to use format
    a. Such as in a CSV, a Database, etc

Web Scraping

1. **Scrape the content**
    a. **Make a request, receive a response**
2. Parse the received content
    a. Make sure we can make sense of the data received depending on its format
3. Extract relevant data from the parsed content
    a. Get only what we need out of what we received
4. Store the relevant data in an easier to use format
    a. Such as in a CSV, a Database, etc

HTTP Request

1. URL for the resource to be requested
    a. http://www.example.com
2. A verb for the action being performed
    a. GET, POST, PUT, etc

HTTP Response

1. Status Code, indicating success or not
   a. 200, 404, 500, etc
2. Body, which is the returned data
   a. In our case, we are expecting HTML

simple_request.py

```
import requests

url = 'http://info.cern.ch/hypertext/WWW/TheProject.html'

response = requests.get(url)

print(response.status_code)
print(response.text)
```

simple_request_to_file.py

```
import requests

url = 'http://info.cern.ch/hypertext/WWW/TheProject.html'

response = requests.get(url)

with open('first-website.html', 'w') as file:
    file.write(response.text)
```

Web Scraping

1. Scrape the content
    a. Make a request, receive a response
2. **Parse the received content**
    a. **Make sure we can make sense of the data received depending on its format**
3. **Extract relevant data from the parsed content**
    a. **Get only what we need out of what we received**
4. Store the relevant data in an easier to use format
    a. Such as in a CSV, a Database, etc

We have our data

Now we need to parse it

HTML

HyperText Markup Language

HTML

```
<html>
      I think you should <b>learn Python</b>. It
      is <b>very easy</b> to learn.
</html>
```

I think you should **learn Python**.

It is **very easy** to learn.

HTML

```
<html>
      I think you should <b>learn Python</b>. It
      is <b>very easy</b> to learn.
</html>
```

I think you should **learn Python**.

It is **very easy** to learn.

```
<b>learn Python</b>
<b>very easy</b>
```

learn Python

very easy

simple_parse.py

```python
# Notice how we can even parse hardcoded HTML strings!
from bs4 import BeautifulSoup



print('Parsing the following:')
html_document = '<html>I think you should <b>learn Python</b>. It is <b>very easy</b> to learn.</html>'

print(html_document)
```

simple_parse.py (continued)

```python
# First we feed our document into BeautifulSoup
soup = BeautifulSoup(html_document, 'html.parser')

# Then we tell it to find all of the bold tags
bold_tags = soup.find_all('b')
print(bold_tags)

for bold_tag in bold_tags:
   # .text gives us the text inside the tags
   print(bold_tag.text)
```

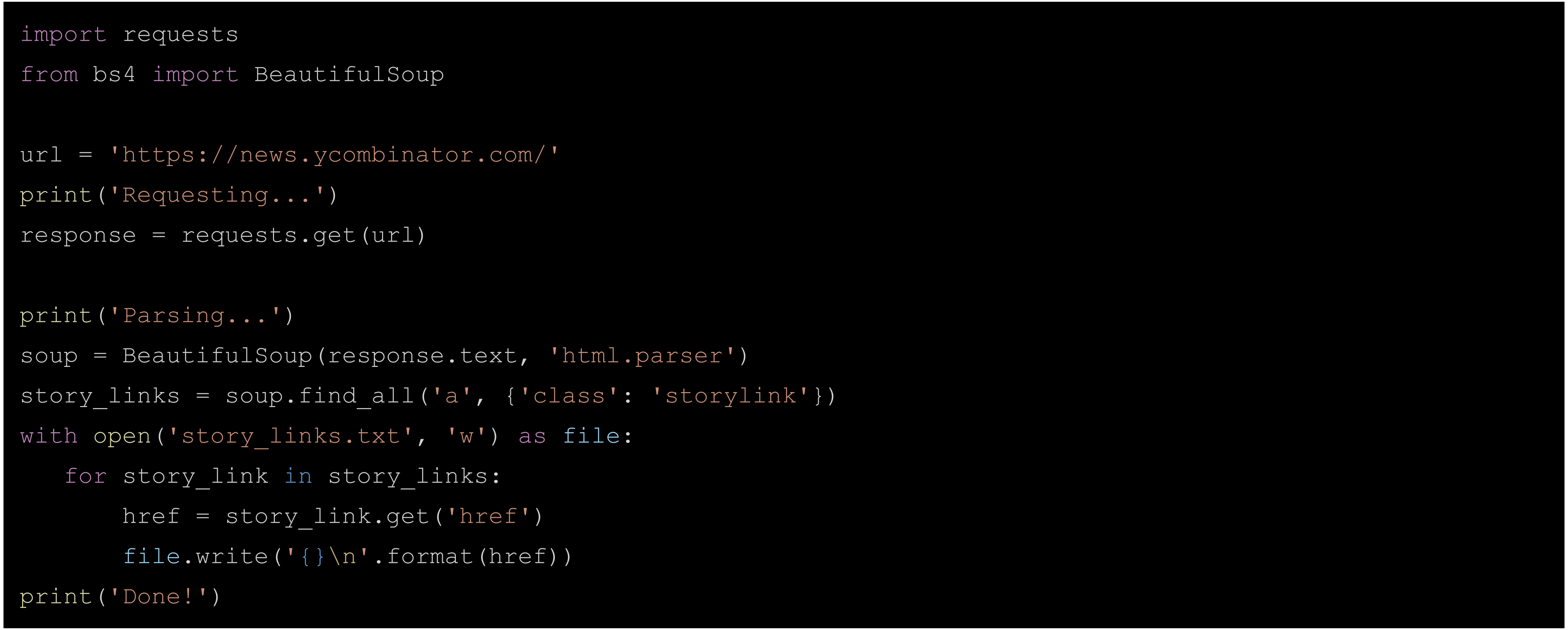
Web Scraping

1. Scrape the content
   a. Make a request, receive a response
2. Parse the received content
   a. Make sure we can make sense of the data received depending on its format
3. Extract relevant data from the parsed content
   a. Get only what we need out of what we received
4. **Store the relevant data in an easier to use format**
   a. **Such as in a CSV, a Database, etc**

The following program combines the following:

1. Send a request to https://news.ycombinator.com/
2. Parse the returned HTML for anchor tags (<a><a/>), also known as hyperlinks
    a. But only the anchor tags with the class *storylink* on them
3. Store the resulting hyperlinks in a text file, with a new link on each line

request_parse_store.py



```python
import requests
from bs4 import BeautifulSoup

url = 'https://news.ycombinator.com/'
print('Requesting...')
response = requests.get(url)

print('Parsing...')
soup = BeautifulSoup(response.text, 'html.parser')
story_links = soup.find_all('a', {'class': 'storylink'})
with open('story_links.txt', 'w') as file:
    for story_link in story_links:
        href = story_link.get('href')
        file.write('{}\n'.format(href))
print('Done!')
```

Livecoding a Web Scraper

AKA, do GiG need a Barista? ☕

🔥 https://trailblaze.software/

Responsible Web Scraping

Conclusion

This is a special talk for me...

Example Starter Projects

1. Retrieve products from supermarket websites
   a. Try to match them together, to see which one is cheaper
   b. **Bonus challenge:** Input your shopping list and export a list of which products to buy from where
2. Animal shelter aggregator
   a. Scrape names and photos from various animal shelter websites
   b. **Bonus challenge:** Display them on one website, with backlinks and shelter contact details
3. Create a web crawler
   a. This is a scraper which scrapes links, then follows those links to get more links
   b. **Bonus challenge:** Find a way to visualise the contents of these pages

Thank you for listening 🙇‍♂️

- This talk will be available in the coming days as a blog post at:
  - https://blog.trailblaze.software
- Slides and talk recording will be available on the PyMalta website and YouTube channel
- Feel free to come up after if you have any questions, want me to clarify something or just to have a chat