Introduction to Web Scraping

In 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

https://trailblaze.software/
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

https://trailblaze.software/
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
<table>
<thead>
<tr>
<th>Title</th>
<th>Points</th>
<th>Author</th>
<th>Score</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>The History of Random.org (2009)</td>
<td>110</td>
<td>unlynx</td>
<td>3.9</td>
<td>[Link]</td>
</tr>
<tr>
<td>Blender Is Free Software</td>
<td>430</td>
<td>JPL</td>
<td>3.5</td>
<td>[Link]</td>
</tr>
<tr>
<td>Practical Deep Learning for Coders</td>
<td>123</td>
<td>smirnkh</td>
<td>3.5</td>
<td>[Link]</td>
</tr>
<tr>
<td>Battle testing data integrity verification with ZFS and Btrfs</td>
<td>63</td>
<td>unashikht</td>
<td>3.4</td>
<td>[Link]</td>
</tr>
<tr>
<td>The Open Source Seed Initiative</td>
<td>265</td>
<td>oaseeds.org</td>
<td>3.4</td>
<td>[Link]</td>
</tr>
<tr>
<td>Traffic-busting $100B Bay Area tax plan taking shape</td>
<td>15</td>
<td>mercurynews.com</td>
<td>3.3</td>
<td>[Link]</td>
</tr>
<tr>
<td>What Makes a PDP-11/35 Tick?</td>
<td>16</td>
<td>boomcom.com</td>
<td>3.3</td>
<td>[Link]</td>
</tr>
<tr>
<td>Search the Full Text of 3M Nonprofit Tax Records for Free</td>
<td>12</td>
<td>propublica.org</td>
<td>3.3</td>
<td>[Link]</td>
</tr>
<tr>
<td>Why Is America So Far Behind Europe on Digital Privacy?</td>
<td>31</td>
<td>nytimes.com</td>
<td>3.3</td>
<td>[Link]</td>
</tr>
<tr>
<td>Is it time to treat sugar like smoking?</td>
<td>120</td>
<td>bbc.com</td>
<td>3.3</td>
<td>[Link]</td>
</tr>
<tr>
<td>AWS costs every programmer should know</td>
<td>225</td>
<td>italiansen.com</td>
<td>3.3</td>
<td>[Link]</td>
</tr>
<tr>
<td>Fortune 500 company leaked 264GB in client, payment data</td>
<td>99</td>
<td>odnet.com</td>
<td>3.2</td>
<td>[Link]</td>
</tr>
<tr>
<td>Xiaomi explains more about how its under-screen camera works</td>
<td>139</td>
<td>theverge.com</td>
<td>3.2</td>
<td>[Link]</td>
</tr>
<tr>
<td>The vintage 74181 ALU chip: how it works and why it's so strange</td>
<td>45</td>
<td>righto.com</td>
<td>3.2</td>
<td>[Link]</td>
</tr>
<tr>
<td>There's a lot to learn about how blue light affects our eyes</td>
<td>49</td>
<td>popsci.com</td>
<td>3.1</td>
<td>[Link]</td>
</tr>
<tr>
<td>For Men Who Hate Talking on the Phone, Games Keep Friendships Alive</td>
<td>408</td>
<td>techadc.com</td>
<td>3.1</td>
<td>[Link]</td>
</tr>
</tbody>
</table>
It is not always available for *machine* consumption
https://hacker-news.firebaseio.com/v0/item/20141052.json

{
    "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"
}
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

https://trailblaze.software/
Web Scraping

https://trailblaze.software/
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

https://trailblaze.software/
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

https://trailblaze.software/
The Request Response cycle

Browser → HTTP Request → Server → Data

https://trailblaze.software/
The Request Response cycle

Browser ➞ HTTP Response ➞ Server

https://trailblaze.software/
The Request Response cycle

Browser  ➔ HTTP Request  ➔ Server

example.com/employees

Process highly simplified

https://trailblaze.software/
The Request Response cycle

Browser

HTTP Response

Server

Employees
- Ken
- Trent
- Roberto

Status Code 200

Process highly simplified

https://trailblaze.software/
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

https://trailblaze.software/
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

https://trailblaze.software/
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

```python
import requests

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

response = requests.get(url)

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

https://trailblaze.software/
simple_request_to_file.py

```python
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

https://trailblaze.software/
I think you should learn Python. It is very easy to learn.
I think you should learn Python.

It is very easy to learn.
# Notice how we can even parse hardcoded HTML strings!

```python
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)
```
# 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/](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
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('{}
            .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](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