Web Scraping

Jennifer Helsby, Eric Potash
Computation for Public Policy
Lecture 11: February 9, 2016
computationforpolicy.github.io

Today

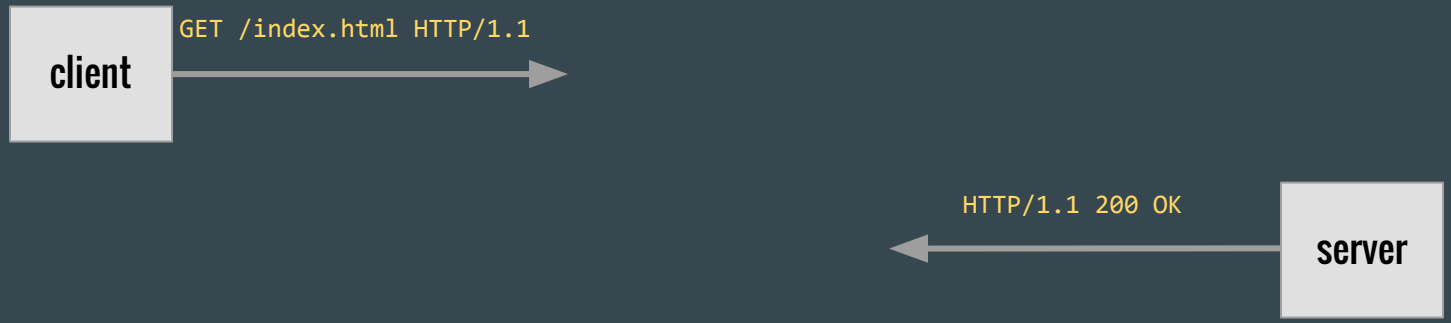
- Web basics: HTML, HTTP requests
- Getting unstructured data by web scraping
- Working with raw HTML to pull out useful data
- **Note:** Homework 4 is on web scraping and is already up on the course site if you want to get started

Web Technologies 101

HTTP, HTML, and friends

HyperText Transfer Protocol (HTTP)

- Core communications protocol for retrieving data
- Consists of messages - *requests* and *responses* - sent between a *client* and a *server*.



HTTP Request

```
GET /index.html HTTP/1.1
```

- First line contains:
  - HTTP method in use. Here “GET”.
  - Requested URL
  - HTTP version
- Rest of request may contain:
  - User-Agent: headers that describes the web browser in use
  - ...

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:17.0) Gecko/17.0
Firefox/17.0
```

HTTP Methods

- HTTP GET
  - Most common HTTP method
  - Used to get data (big surprise)
- HTTP POST
  - Used to send data, e.g. entries in forms, to the server

HTTP Responses

```
HTTP/1.1 200 OK
```

- First line contains:
  - HTTP version
  - HTTP response code
- Rest of response will contain:
  - Additional headers: Server, Content-Type, …
  - Content requested

Common HTTP Response Status Codes

- 1xx: Informational
- 2xx: Success
  - 200: OK
- 3xx: Redirection
  - 301: Redirect
- 4xx: Errors
  - 404: File not found
  - 403: Forbidden

HTTP GET Requests and Parameters

```
/index.php?name1=value1&name2=value2
```

- Query string with parameters sent in the URL of a GET request
- Shouldn’t use with sensitive data

www.theguardian.com/international

headlines

Tuesday
9 February 2016

Now
6°C

08:00	11:00	14:00
4°C	5°C	6°C

London

New Hampshire / 'He's a pussy' – Donald Trump repeats insult from crowd member about Ted Cruz

On the eve of the New Hampshire primary, Donald Trump repeats an offensive remark from the crowd, drawing attention to rival Ted Cruz's position on waterboarding

27

Syria / We will los without support f Arab nations, rebe warn

930

Turban row / American Sikh actor and designer barred from US-bound flight

US news / Passengers t terror as cruise ship ba by 9m waves

view-source:www.theguardian.com/international

```html

<!DOCTYPE html>
<html id="js-context" class="js-off is-not-modern id--signed-out" data-page-path="/international">
<head>
<meta charset="utf-8"/>
<!--
 __        __                  _     _      _
 \ \      / /__    __ _ _ __ ___  | |__ (_)_ __(_)_ __   __ _
  \ \ /\ / / _ \  / _` | '__/ _ \ | '_ \| | '__| | '_ \ / _` |
   \ V  V /  __/ | (_| | | |  __/ | | | | | |  | | | | | (_| |
    \_/\_/ \___|  \__,_|_|  \___| |_| |_|_|_|  |_|_| |_|\__, |
                                                        |___/
Ever thought about joining us?
http://developers.theguardian.com/join-the-team.html
-->
<title>News, sport and opinion from the Guardian's global edition | The Guardian</title>
<meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
<meta name="format-detection" content="telephone=no"/>
<meta name="HandheldFriendly" content="True"/>
<meta name="viewport" content="width=device-width,minimum-scale=1,initial-scale=1">
<link rel="dns-prefetch" href="https://assets.guim.co.uk/"/>
<link rel="dns-prefetch" href="https://i.guim.co.uk"/>
<link rel="dns-prefetch" href="https://api.nextgen.guardianapps.co.uk"/>
<link rel="dns-prefetch" href="https://hits-secure.theguardian.com"/>
<link rel="dns-prefetch" href="//j.ophan.co.uk"/>
<link rel="dns-prefetch" href="//ophan.theguardian.com"/>
<link rel="dns-prefetch" href="//oas.theguardian.com"/>
<link rel="dns-prefetch" href="//beacon.guim.co.uk"/>
<link rel="apple-touch-icon" sizes="152x152" href="https://assets.guim.co.uk/images/favicons/451963ac2e23633472bf48e2856d3f04/152x15
<link rel="apple-touch-icon" sizes="144x144" href="https://assets.guim.co.uk/images/favicons/1a3f98d8491f8cfdc224089b785da86b/144x14
<link rel="apple-touch-icon" sizes="120x120" href="https://assets.guim.co.uk/images/favicons/cf23080600002e50f5869c72f5a904bd/120x12
<link rel="apple-touch-icon" sizes="114x114" href="https://assets.guim.co.uk/images/favicons/f438f6041a4c1d0289e6debd112880c2/114x11
<link rel="apple-touch-icon" sizes="72x72" href="https://assets.guim.co.uk/images/favicons/b5050517955e7cf1e493ccc53e64ca05/72x72.pn
<link rel="apple-touch-icon-precomposed" href="https://assets.guim.co.uk/images/favicons/4fd650035a2cebafea4e210990874c64/57x57.png"
<link rel="manifest" href="/2015-06-24-manifest.json" crossorigin="use-credentials">
<link rel="shortcut icon" type="image/png" href="https://assets.guim.co.uk/images/favicons/79d7ab5a729562cebca9c6a13c324f0e/32x32.ic
<link rel="alternate" href="http://www.theguardian.com/uk" hreflang="en-GB"/>
<link rel="alternate" href="http://www.theguardian.com/us" hreflang="en-US"/>
<link rel="alternate" href="http://www.theguardian.com/au" hreflang="en-AU"/>
<link rel="canonical" href="http://www.theguardian.com/international"/>
<meta name="apple-mobile-web-app-title" content="Guardian"/>
<meta name="application-name" content="The Guardian"/>
<meta name="msapplication-TileColor" content="#005689"/>
```

HTML

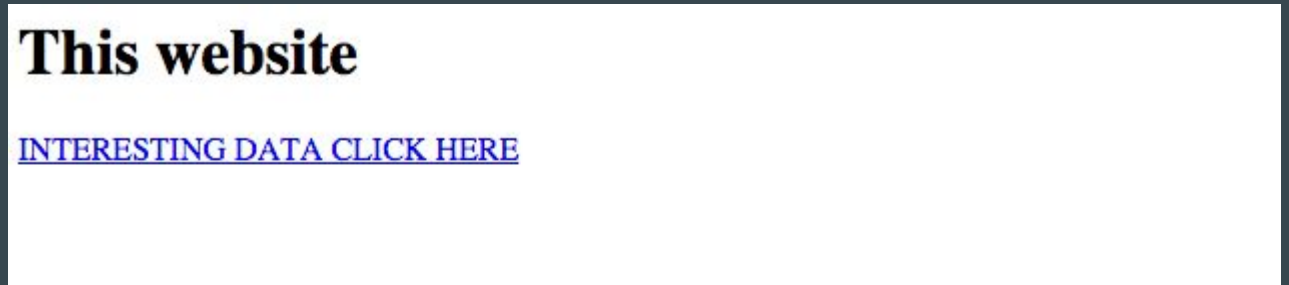
- HyperText Markup Language
- Languages that webpages are written in
- Consists of elements called tags, many of which come in pairs (opening and closing):
  - `<html> </html>`
  - `<head> </head>`
  - `<body> </body>`
  - `<a> </a>`
- But some are not:
  - `<img>`
  - `<br>`
- Whitespace doesn't matter as it does in Python

Basic Structure of an HTML Webpage

```
<html>
<head><title>Baby’s first webpage</title>
</head>
<body>
<h1>This website</h1>
<p>
<a href=”sekretdata.html”>INTERESTING DATA CLICK HERE</a>
</p>
</body>
</html>
```

Basic Structure of an HTML Webpage

```
<html>
<head><title>Baby’s first webpage</title>
</head>
<body>
<h1>This website</h1>
<p>
<a href=”sekretdata.html”>INTERESTING DATA CLICK HERE</a>
</p>
</body>
</html>
```



HTML Links

href attribute used to denote the address of the webpage in the link

```
<a href=”sekretdata.html”>INTERESTING DATA CLICK HERE</a>
```

HTML Tables

```
<table>
<tr>
<td>Header 1</td>
<td>Header 2</td>
</tr>
<tr>
<td>Data 1</td>
<td>Data 2</td>
</tr>
</table>
```

Table rows defined with `<tr>`

Table cells defined with `<td>`

HTML Tables

```
<table>
<tr>
<td>Header 1</td>
<td>Header 2</td>
</tr>
<tr>
<td>Data 1</td>
<td>Data 2</td>
</tr>
</table>
```

Table rows defined with `<tr>`

Table cells defined with `<td>`

Header 1	Header 2
Data 1	Data 2

Other Common Web Technologies

- Javascript
  - *.js files or code embedded in `<script> </script>` tags
  - Used to generate more dynamic content
- CSS
  - *.css files or code embedded in `<style> </style>` or `<link> </link>` tags
- PHP
  - Server-side scripting language

Scraping Content

Web Scraping

- Web Scraping: Process of extracting information from websites
- Anything you can view in your browser is data you can potentially scrape
- Advantages:
  - Does not rely on there being an API to access data (using APIs will be the topic of a future lecture)
  - No limitations on rate / content that can be scraped as with APIs
- Web crawlers / spiders: Programs that browse the web, clicking links, often for the purposes of indexing content for search engines

When to scrape and when not to scrape

Scraping is a good solution for:

- Grabbing all mp3s linked to on a given site
- Constructing a pandas DataFrame from tables or a series of tables over several webpages
- Grabbing all journal articles linked to a current page

Scraping might not be the best solution for:

- Grabbing data from your emails (use email libraries)
- Grabbing tweets (use the Twitter API)

You can always ask the owner for access to save yourself some work.

wget

- Note that for some applications, instead of writing a scraper in Python, you might use wget:

```
wget URL
```

- Recursive flag `-r` enables wget to follow links on the page:

```
wget -r 1 URL
```

Example of where wget is sufficient

www.iprachicago.org/resources.html

City of Chicago

Independent Police Review Authority

About IPRA | File a Complaint | Compliment an Officer | Investigative Process | Investigative Results | Resources | Contact | Home

Resources

Laws and Regulations | Quarterly Reports | Monthly Sustained Cases | Officer-Involved Shootings | Officers-Involved Tasing | Miscellaneous

IPRA Ordinance
Unofficial Version - For Informational Purposes Only

FOIA Disclosures (5 ILCS 140)
FOIA E-mail Request

Pursuant to the Municipal Code of Chicago, Chapter 2-57, the Independent Police Review Authority, through its Chief Administrator, recommends discipline in sustained cases to the Superintendent of the Chicago Police Department. It should be noted that IPRA disciplinary recommendations are subject to a non-concurrence by the Superintendent of Police, or a hearing by the Chicago Police Board, either of which may result in a final disposition different from the IPRA recommendation contained in the posted investigative abstract. The final determination of discipline is made by the Superintendent of Police or the Police Board.

www.iprachicago.org/Investigations_2014.html

City of Chicago

Independent Police Review Authority

Public Records of Investigations

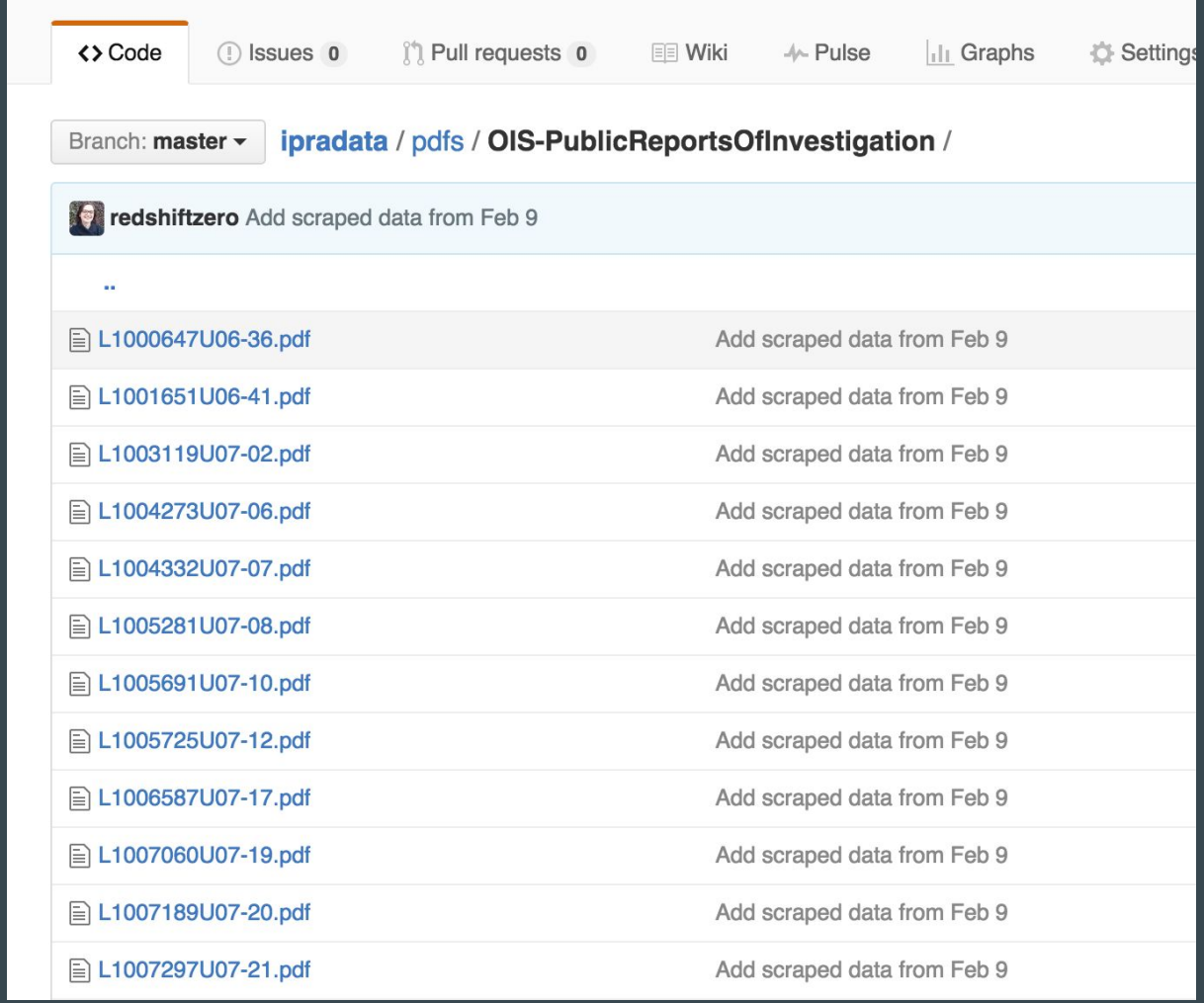
2014 Investigations

L1066834U14-01
L1067652U14-05
L1068262U14-09
L1068458U14-11
L1068793U14-12
L1069086U14-13
L1069378U14-14
L1070142U14-20
L1070144U14-21
L1070169U14-24
L1070864U14-27
L1071085U14-28
L1071166U14-29
L1072212U14-37

INTEGRITY

Grabbing all PDFs

```
wget -r 3 http://www.iprachicago.org/resources.html
```



https://github.com/redshiftzero/ipradata/tree/master/pdfs/OIS-PublicReportsOfInvestigation

Ethical Issues: Be Polite

- Give credit where credit is due
  - You just scraped this data, you didn't produce it
- Don't overload sites
- Obey robots.txt
  - A file describing which areas of the site are "disallowed" for automated crawling and scraping

```
User-agent: *
Disallow: /
```

- ***Remember***: Unless you take explicit steps to be anonymous on the web, ***you are not***.

www.theguardian.com/robots.txt

```
# this is the robots.txt file for theguardian.com

User-agent: *
Disallow: /sendarticle/
Disallow: /Users/
Disallow: /users/
Disallow: /*/print$
Disallow: /email/
Disallow: /contactus/
Disallow: /share/
Disallow: /websearch
Disallow: /*?commentpage=
Disallow: /whsmiths/
Disallow: /external/overture/
Disallow: /discussion/report-abuse/*
Disallow: /discussion/report-abuse-ajax/*
Disallow: /discussion/comment-permalink/*
Disallow: /discussion/report-abuse/*
Disallow: /discussion/user-report-abuse/*
Disallow: /discussion/handlers/*
Disallow: /discussion/your-profile
Disallow: /discussion/your-comments
Disallow: /discussion/edit-profile
Disallow: /discussion/search/comments
Disallow: /discussion/*
Disallow: /search
Disallow: /music/artist/*
Disallow: /music/album/*
Disallow: /books/data/*
Disallow: /settings/
Disallow: /embed/
Disallow: /*styles/js-on.css$
```

Python Web Scraping Libraries

- urllib: for handling URLs
  - urllib.request for opening and reading data from URLs
- BeautifulSoup: for structuring data from HTML files
- Scrapy:

  http://doc.scrapy.org/en/latest/intro/tutorial.html

Let’s try an example

1. Figure out what we want to grab by inspecting the page in a web browser.

The New York Times

The Guantánamo Docket

Documents and research related to the roughly 780 people who have been sent to the Guantánamo Bay prison since 2002.

91 Current Detainees

Of the roughly 780 people who have been detained at the United States military prison at Guantánamo Bay, Cuba, 91 remain.

34 Held in Law-of-War Detention but Recommended for Transfer if Security Conditions Met

Name	Citizenship
Kamin, Mohammed	Afghanistan
Abu Bakr, Omar Khalifa Mohammed	Libya
Ghereby, Salem Abdul Salem	Libya
Abdulayev, Omar Hamzayavich	Tajikistan
al Yazidi, Ridah Bin Saleh	Tunisia
Said Kuman, Ahmed Yaslam	Yemen
al Saleh, Abdul	Yemen
Saleh Naser, Abdul Rahman Mohamed	Yemen
al Bihani, Ghaleb Nassar	Yemen
Suleiman, Fayiz Ahmad Yahia	Yemen
al Raimi, Ali Yahya Mahdi	Yemen
Baada, Tarek Ali Abdullah Ahmed	Yemen
Abd al Wahab, Abd al Malik	Yemen
Sulayman, Abdul Rahman Abdul Abu Ghityh	Yemen
Jarabh, Saeed Ahmed Mohammed Abdullah Sarem	Yemen
al Shabli, Abdullah Yahia Yousf	Yemen
al Hamiri, Mohammed Abdullah	Yemen
al Edah, Mohammed Ahmad Said	Yemen
al Hikimi, Ahmed Umar Abdullah	Yemen
al Mudhaffari, Abdel Qadir Hussein	Yemen
al Bihani, Tolfiq Nassar Ahmed	Yemen
al Shamyri, Mustafa Abdul Qawi Abdul Aziz	Yemen
Bwazir, Mohammed Ali Abdullah	Yemen

7 Charged in Military Commissions System

Name	Citizenship
al Iraqi, Abd al Hadi	Iraq
Mohammed, Khalid Shaikh	Pakistan
Ali, Abd al Aziz	Pakistan
al Nashiri, Abd al Rahim	Saudi Arabia
al Hawsawi, Mustafa Ahmed	Saudi Arabia
Bin al Shibh, Ramzi	Yemen
Bin Attash, Walid	Yemen

3 Convicted in Military Commissions System

Name	Citizenship
Khan, Majid	Pakistan
al Darbi, Ahmed Muhammed Haza	Saudi Arabia
al Bahlul, Ali Hamza Ahmad Suliman*	Yemen

* Conviction vacated on appeal, litigation still pending.

47 Held in Indefinite Law-of-War Detention and not Recommended for Transfer

Name	Citizenship
al Afghani, Muhammad Rahim	Afghanistan
al Afghani, Haroon	Afghanistan
Hamidullah	Afghanistan
Obaidullah	Afghanistan
Zahir, Abdul	Afghanistan

The New York Times

The Guantánamo Docket

Documents and research related to the roughly 780 people who have been sent to the Guantánamo Bay prison since 2002.

Ali Yahya Mahdi al Raimi

Ali Yahya Mahdi al Raimi is a 32- or 33-year-old citizen of Yemen. As of January 2010, the Guantánamo Review Task Force had recommended him for transfer. As of Feb. 9, 2016, he has been held at Guantánamo for 13 years nine months. As of January 2010, the Guantánamo Review Task Force had recommended him for transfer to Yemen provided that certain security conditions were met.*

Note: These documents include some assertions that cannot be independently verified. Many allegations have been contested by detainees and their lawyers, and some have been undercut by other evidence.

Internment Serial Number: 167

Alternate names:

Ali Yahya Mahdi al Rimi
Ali Yahia Mahdi al Yarimi

Related links:

"At Guantánamo, Big Threats Found in Small Clues," NPR

DOCUMENT | PAGES | TEXT Zoom Search

SECRET//NOFORN//20291029

DEPARTMENT OF DEFENSE
JOINT TASK FORCE GUANTANAMO
GUANTANAMO BAY, CUBA
APO AE 09360

REPLY TO

4 DOCUMENTS

Combatant Status Review Tribunals Summaries
2 pages

Combatant Status Review Tribunal Transcripts
11 pages

projects.nytimes.com/guantanamo/detainees/167-ali-yahya-mahdi-al-raimi

The New York Times

Share this view on Twitter or Facebook

The Guantánamo Docket

Documents and research related to the roughly 780 people who have been sent to the Guantánamo Bay prison since 2002.

Search the docket Go

Overview | **All Detainees** | Current Detainees | Transfer Countries | Citizenship | Timeline | About

Related Coverage From the Times »

Ali Yahya Mahdi al Raimi

Ali Yahya Mahdi al Raimi is a 32- or 33-year-old citizen of Yemen. As of January 2010, the Guantánamo Review Task Force had recommended him for transfer. As of Feb. 9, 2016, he has been held at Guantánamo for 13 years nine months. As of January 2010, the Guantánamo Review Task Force had recommended him for transfer to Yemen provided that certain security conditions were met.*

Note: These documents include some assertions that cannot be independently verified. Many allegations have been contested by detainees and their lawyers, and some have been undercut by other evidence.

Internment Serial Number: 167

Alternate names:

Ali Yahya Mahdi al Rimi
Ali Yahia Mahdi al Yarimi

Related links:

"At Guantánamo, Big Threats Found in Small Clues," NPR

DOCUMENT | PAGES | TEXT Zoom Search

SECRET//NOFORN//20291029

DEPARTMENT OF DEFENSE
JOINT TASK FORCE GUANTANAMO
GUANTANAMO BAY, CUBA
APO AE 09360

REPLY TO

4 DOCUMENTS

Combatant Status Review Tribunals Summaries
2 pages

Combatant Status Review Tribunal Transcripts
11 pages

2. Grab and inspect the source code for the page you're interested in.

```
from urllib.request import urlopen

from bs4 import BeautifulSoup

base_url = "http://projects.nytimes.com"

index_ref = "/guantanamo/detainees/current"

index_html = urlopen(base_url + index_ref)

index = BeautifulSoup(index_html, "lxml")
```

2. Grab and inspect the source code for the page you're interested in.

```
<table class="s-full s-datatable">
<tr class="bar"><th>Name</th><th>Citizenship</th></tr>
<tr class="divide nytint-first">
<td><a href="/guantanamo/detainees/1045-mohammed-kamin">Kamin, Mohammed</a></td>
<td><a href="/guantanamo/country/afghanistan">Afghanistan</a></td>
</tr>
<tr class="divide ">
<td><a href="/guantanamo/detainees/695-omar-khalifa-mohammed-abu-bakr">Abu Bakr, Omar Khali
fa Mohammed</a></td>
<td><a href="/guantanamo/country/libya">Libya</a></td>
</tr>
<tr class="divide ">
<td><a href="/guantanamo/detainees/189-salem-abdul-salem-ghereby">Ghereby, Salem Abdul Sale
m</a></td>
<td><a href="/guantanamo/country/libya">Libya</a></td>
</tr>
<tr class="divide ">
```

2. Grab and inspect the source code for the page you're interested in.

```
<table class="s-full s-datatable">
<tr class="bar"><th>Name</th><th>Citizenship</th></tr>
<tr class="divide nytint-first">
<td><a href="/guantanamo/detainees/1045-mohammed-kamin">Kamin, Mohammed</a></td>
<td><a href="/guantanamo/country/afghanistan">Afghanistan</a></td>
</tr>
<tr class="divide ">
<td><a href="/guantanamo/detainees/695-omar-khalifa-mohammed-abu-bakr">Abu Bakr, Omar Khali
fa Mohammed</a></td>
<td><a href="/guantanamo/country/libya">Libya</a></td>
</tr>
<tr class="divide ">
<td><a href="/guantanamo/detainees/189-salem-abdul-salem-ghereby">Ghereby, Salem Abdul Sale
m</a></td>
<td><a href="/guantanamo/country/libya">Libya</a></td>
</tr>
<tr class="divide ">
```

Prettify

```
print(index.prettify())
```

```
</tr>
<tr class="divide ">
 <td>
  <a href="/guantanamo/detainees/91-abdul-al-saleh">
   al Saleh, Abdul
  </a>
 </td>
 <td>
  <a href="/guantanamo/country/yemen">
   Yemen
  </a>
 </td>
</tr>
<tr class="divide ">
 <td>
  <a href="/guantanamo/detainees/115-abdul-rahman-mohamed-saleh-naser">
   Saleh Naser, Abdul Rahman Mohamed
  </a>
 </td>
 <td>
  <a href="/guantanamo/country/yemen">
```

Look at elements of the HTML

index.head

```
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<meta content="width=device-width, user-scalable=no" name="viewport"/>
<title>The Detainees - The Guantánamo Docket</title>
<meta content="The Detainees - The Guantánamo Docket" name="hdl"/>
<link href="http://int.nyt.com/applications/guantanamo/assets/v2-893fb2078582e8a85263fa41144e
124f.css" media="screen" rel="stylesheet" type="text/css"/>
<!--[if lt IE 9]>
      <link embed_assets="false" href="http://int.nyt.com/applications/guantanamo/assets/ie_o
nly.css" media="screen" rel="stylesheet" type="text/css" />
    <![endif]-->
<script src="http://typeface.nytimes.com/zam5nzz.js" type="text/javascript"></script>
<script type="text/javascript">try{Typekit.load();}catch(e){}</script>
<script src="http://graphics8.nytimes.com/projects/assets/shared/assets/vendor.js"></script>
<script src="http://int.nyt.com/applications/guantanamo/assets/application-27c8c1649b272a64ba
cf45d81c17b032.js" type="text/javascript"></script>
</head>
```

Look at elements of the HTML

index.body

```
<body>
<div class="nytint-collab" id="main">
<div id="nytint-logoNYT">
<a href="http://www.nytimes.com" target="_blank"><img alt="The New York Times" src="htt
p://int.nyt.com/applications/guantanamo/assets/nytlogo-dfd6516357d69a501a0626de5027c60f.pn
g"/></a>
</div>
<div id="nytint-shareViewTools">
<div class="nytint-shareViewToolsText">Share this view on
          <a class="nytint-sharetoolsShareLink nytint-sharetoolsShareLinkTwitter nytint-sha
reViewToolsLink" href="#" id="nytint-shareViewToolsLinkTwitter" target="_blank">Twitter</a>
           or
          <a class="nytint-sharetoolsShareLink nytint-sharetoolsShareLinkFacebook nytint-sh
areViewToolsLink" href="#" id="nytint-shareViewToolsLinkFacebook" target="_blank">Faceboo
k</a>
</div>
```

Get all links in the page

```
index.find_all('a')
```

```
 <a href="/guantanamo/detainees/10020-majid-khan">Khan, Majid</a>,
 <a href="/guantanamo/country/pakistan">Pakistan</a>,
 <a href="/guantanamo/detainees/768-ahmed-muhammed-haza-al-darbi">al Darbi, Ahmed Muhammed
Haza</a>,
 <a href="/guantanamo/country/saudi-arabia">Saudi Arabia</a>,
 <a href="/guantanamo/detainees/39-ali-hamza-ahmad-suliman-al-bahlul">al Bahlul, Ali Hamza
Ahmad Suliman</a>,
 <a href="/guantanamo/country/yemen">Yemen</a>,
 <a href="/guantanamo/detainees/10030-muhammad-rahim-al-afghani">al Afghani, Muhammad Rahi
m</a>,
 <a href="/guantanamo/country/afghanistan">Afghanistan</a>,
 <a href="/guantanamo/detainees/10028-haroon-al-afghani">al Afghani, Haroon</a>,
 <a href="/guantanamo/country/afghanistan">Afghanistan</a>,
 <a href="/guantanamo/detainees/1119-hamidullah">Hamidullah</a>,
 <a href="/guantanamo/country/afghanistan">Afghanistan</a>,
 <a href="/guantanamo/detainees/762-obaidullah">Obaidullah</a>,
 <a href="/guantanamo/country/afghanistan">Afghanistan</a>,
```

Regular Expressions

- Regular expressions are used to match text patterns
- In Python, use the `re` module for regular expressions
- Can pull out patterns of interest using:

```
re.compile(pattern)
```

3. Follow links of interest and grab their HTML.

Let's just grab the links that have links pointing to pages that match /guantanamo/detainees/*:

```
import re

prisoner_links = index.find_all("a", href=re.compile
("/guantanamo/detainees/\d+"))
```

Regular Expressions with re

`.` Matches any **single** character except \n (newline)

`*` Matches 0 or more repetitions of preceding regex

`+` Matches 1 or more repetitions of preceding regex

`\d` Match decimal digits [0-9]

`\w` Match words (alphanumerics)

Many more at `https://docs.python.org/3/library/re.html`

3. Follow links of interest and grab their HTML.

```
[<a href="/guantanamo/detainees/1045-mohammed-kamin">Kamin, Mohammed</a>,
 <a href="/guantanamo/detainees/695-omar-khalifa-mohammed-abu-bakr">Abu Bakr, Omar Khalifa
Mohammed</a>,
 <a href="/guantanamo/detainees/189-salem-abdul-salem-ghereby">Ghereby, Salem Abdul Sale
m</a>,
 <a href="/guantanamo/detainees/257-omar-hamzayavich-abdulayev">Abdulayev, Omar Hamzayavic
h</a>,
 <a href="/guantanamo/detainees/38-ridah-bin-saleh-al-yazidi">al Yazidi, Ridah Bin Sale
h</a>,
 <a href="/guantanamo/detainees/321-ahmed-yaslam-said-kuman">Said Kuman, Ahmed Yaslam</a>,
 <a href="/guantanamo/detainees/91-abdul-al-saleh">al Saleh, Abdul</a>,
 <a href="/guantanamo/detainees/115-abdul-rahman-mohamed-saleh-naser">Saleh Naser, Abdul Ra
hman Mohamed</a>,
 <a href="/guantanamo/detainees/128-ghaleb-nassar-al-bihani">al Bihani, Ghaleb Nassar</a>,
 <a href="/guantanamo/detainees/153-fayiz-ahmad-yahia-suleiman">Suleiman, Fayiz Ahmad Yahi
a</a>,
 <a href="/guantanamo/detainees/167-ali-yahya-mahdi-al-raimi">al Raimi, Ali Yahya Mahd
i</a>,
```

3a. Slow down requests

Add in some time delay between each link by creating a wrapper to urlopen():

```
import time


def try_request(url):
    html = urlopen(url)
    time.sleep(5)
    return html
```

3b. Extract details from each individual page

```
def extract_details(prisoner_html):
   pris_details = BeautifulSoup(prisoner_html, "lxml")
   divs = pris_details.find_all("div", {"class": "nytint-detainee-fullcol"})

   for test in divs:
         try:
           name = test.find("h1").get_text().strip()
           country = test.find("a", href=re.compile("/country/")).get_text()
           time_in_gitmo = test.find(text=re.compile("for \d+ years")).lstrip(
               "for ").strip().rsplit(".\n\n", 1)[0]
       except:
           return None

   return {"name": name,
           "country": country,
           "time_in_gitmo": time_in_gitmo}
```

3c. Put it all together

Follow each link:

```
for each in prisoner_links:

    prisoner_html = try_request(base_url + each[“href”])

    details = extract_details(prisoner_html)
```

Summary

- We used Python's urllib to submit HTTP requests
- We processed the response with BeautifulSoup
- We parsed the HTML to acquire additional links
- We follow those new links in the same manner
- Finally, we parse the resulting HTML for each link to pull out the data elements of interest

https://github.com/redshiftzero/guantanamo/blob/master/guantanabot/scrapey.py

Where to go from here?

- Deployment [topic of future lecture]
  - Automate this scraping such that it will run every day
- Convert PDFs to text
  - For these PDFs we could grab the text
  - If an image PDF, could perform Optical Character Recognition (OCR) to get the text
    - Tesseract is a standard tool for OCRing documents
- Web application development [topic of future lecture]
  - Create a web interface to display the data derived from this or the result of some computations using this data

Assignment 4

Web Scraping

You'll be scraping data from this website, which contains a list of incidents involving commercial aircraft listed by year.

Part a

Write a scraper that will produce a pandas dataframe containing the following columns:

- When the accident occurred (year, month, and day - use a datetime object)
- The short text description (everything to the right of the date)
- The link to the detail page.

Part b

Now write a code that clicks each link and scrapes additional content from the detailed page associated with each individual crash. **How will you ensure that you rate limit your requests to the target web server?** Once you have implemented this feature, scrape the content located in the right column of each details page and add it to the pandas dataframe:

- Number of passengers
- Number of crew
- Number of fatalities
- Number of survivors
- Registration
- Flight origin
- Destination

Part c

Which were the top 5 most deadly aviation incidents? Report the number of fatalities and the flight origin for each.

Part d

Which flight origin has the highest number of aviation incidents in the last 25 years?

Part e

Save this Dataframe as JSON and commit to your repo, along with the notebook / python code used to do this assignment.

Index Page

1928 [edit]

- July 13 – An **Imperial Airways Vickers Vulcan crashes** on a test flight from Croydon Airport, England, with a pilot and five passengers near Purley, Surrey, 3 miles (4.8 km) from the airport, with the loss of four passengers. As a result of the crash Imperial Airways stopped the flying of staff (so-called joy rides) on test flights.

1929 [edit]

- June 17 – An **Imperial Airways Handley Page W.10 ditches** in the English Channel due to engine failure, killing 7 of 13 on board.
- November 6 – A **Junkers G 24 crashes** near Marden Park, Godstone, Surrey, England; of the 8 on board, only 1 passenger survives.

1930s [edit]

1930 [edit]

- February 10 – An **Air Union Farman F.63 Goliath crashes** while attempting an emergency landing at Marden Airfield, Marden, Kent, England, due to structural failure, killing 2 of 6 on board.

1931 [edit]

- March 21 – Australian National Airways **Southern Cloud**, an Avro 618 Ten, crashes in the Snowy Mountains while flying from Sydney to Melbourne, killing all eight on board, in Australia's first significant airline disaster; the crash site remained undiscovered for 27 years; severe weather at the time of the flight is the likely cause of the accident.

Individual Pages

1929 Imperial Airways Handley Page W.10 crash

From Wikipedia, the free encyclopedia

Coordinates: 50°45′00″N 1°07′0″E

The **1929 Imperial Airways Handley Page W.10 crash** happened on 17 June 1929 when Handley Page W.10 G-EBMT suffered an engine failure and subsequently ditched in the English Channel off Dungeness with the loss of seven lives. The aircraft was operating an international scheduled flight from Croydon to Le Bourget Airport, Paris, France.

Contents [hide]

Aircraft [edit]

The accident aircraft was Handley Page W.10 G-EBMT *City of Ottawa*, c/n W10-4. It had been delivered to Imperial Airways on 25 December 1925.[1]

Accident [edit]

G-EBMT was operating an international scheduled flight from Croydon Airport to Zurich Airport, Switzerland[2] via

1929 Imperial Airways Handley Page W.10 crash

A Handley Page W.8, similar to the accident aircraft

Accident summary	
Date	17 June 1929
Summary	Engine failure, ditching at sea
Site	English Channel, off Dungeness 50°45′00″N 1°07′0″E
Passengers	11
Crew	2
Injuries (non-fatal)	6
Fatalities	7

Logging in and Scraping

- Need to authenticate to the site:
  - Can pass cookies
    - Set-Cookie response header sets cookies:
      - Set-Cookie: tracking=YW5hcmNoaXN0
    - Browser adds the cookie in future requests to the server:
      - Cookie: tracking=YW5hcmNoaXN0
- For more details on logging in and scraping:
  - Requests: http://docs.python-requests.org/en/latest/user/advanced/#session-objects
  - Mechanize: https://stockrt.github.io/p/emulating-a-browser-in-python-with-mechanize/