

# Python Data Collection and Management for Public Policy Research

## Day 10: Obtaining Data from the Web

---

Blake Miller<sup>†</sup>

12 July, 2024

<sup>†</sup>Assistant Professor, Department of Methodology, London School of Economics and Political Science (E-mail: [b.a.miller@lse.ac.uk](mailto:b.a.miller@lse.ac.uk))

# Plan for today

- Introduction
- Some key features of the internet
- HTML and CSS
- Fundamentals of web scraping
- Coding

# Examples

- An increasing amount of data is available on the web
  - Speeches, biographical information ...
  - Social media data, articles, press releases ...
  - Geographic information, conflict data ...
- These datasets are often provided in an **unstructured format**
- **Web scraping** is the process of extracting this information automatically and transforming it into a **structured dataset**

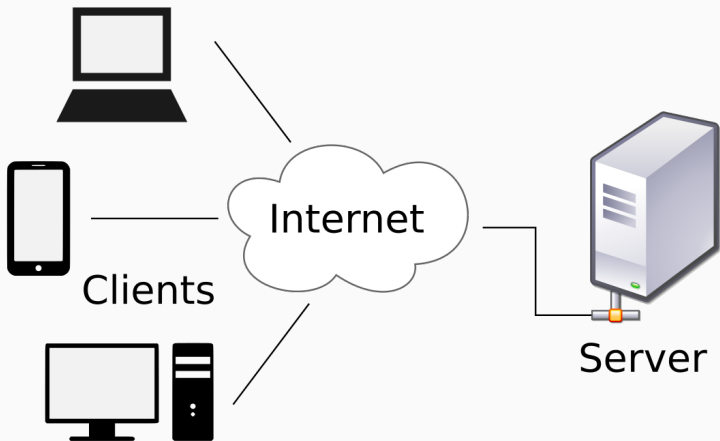
# Why automate?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical or infeasible
- **In contrast, automated web scraping:**
  1. Scales well for large datasets
  2. Allows for dynamic data collection
  3. Is (mostly) reproducible
  4. Involves adaptable techniques
  5. Facilitates detecting and fixing errors
- **When to scrape?**
  1. Trade-off between your time today and your time in the future. Invest in your future self!
  2. Computer time is often cheap; human time more expensive

# Two different approaches

1. **Screen scraping:** Extract data from source code of website, with html parser and/or regular expressions
  - *requests* (today) and *Selenium* packages (not covered)
2. **Web APIs:** A set of structured http requests that return JSON or XML data.

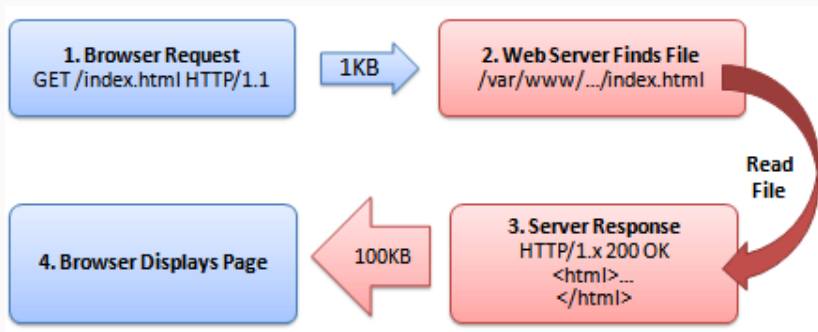
## Some key features of the internet: Client-server model



# Client-server model

- Client: User computer, tablet, phone, software application, etc.
- Server: Web server, mail server, file server, etc.
- Client makes request to the server
  - Depending on what you want to get, the request might be
    - HTTP: Hypertext Transfer Protocol
    - HTTPS: Hypertext Transfer Protocol Secure
    - SMTP: Simple Mail Transfer Protocol
    - FTP: File Transfer Protocol
- Server returns response

# Request and response in the case of HTTP

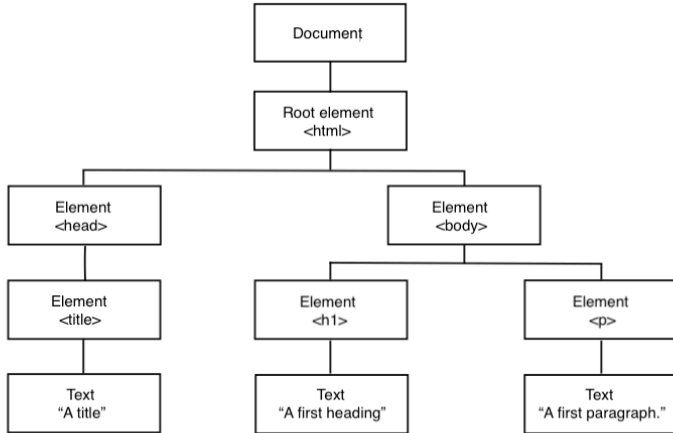




## HTML: Hypertext Markup Language

- HTML displays mostly **static** content
- Many contents of dynamic webpages cannot be found in HTML
  - Example: Google Maps
- Understanding what is static and dynamic in a webpage is a crucial first step for web scraping

# HTML tree structure



# A very simple HTML file

```
<!DOCTYPE html>
<html>
  <head>
    <title>A title</title>
  </head>
  <body>
    <h1>A first heading</h1>
    <p>A first paragraph.</p>
  </body>
</html>
```

From: W3Schools

## Slightly more features

```
<!DOCTYPE html>
<html>
  <head>
    <title>A title</title>
  </head>
  <body>
    <h1>A first heading</h1>
    <p>A first paragraph.</p>
    <p>A second paragraph with some <b>
      formatted</b> text.</p>
    <p>A third paragraph with a <a href="http
      ://www.lse.ac.uk">hyperlink</a>.</p>
  </body>
</html>
```

# With some content divisions

```
<!DOCTYPE html>
<html>
  <head>
    <title>A title</title>
  </head>
  <body>
    <div>
      <h1>Heading of the first division</h1>
      <p>A first paragraph.</p>
      <p>A second paragraph with some <b>formatted</b> text.</p>
      <p>A third paragraph with a <a href="http://www.lse.ac.uk">hyperlink</a>.</p>
    </div>
    <div>
      <h1>Heading of the second division</h1>
      <p>Another paragraph with some text.</p>
    </div>
  </body>
</html>
```

- **Cascading Style Sheets (CSS):** Style sheet language which describes formatting of HTML components, useful for us because of selectors
- **Javascript:** Adds functionalities to the websites, e.g. change content/structure after website has been loaded

## Adding Simple CSS Example 1 <head>

```
<!DOCTYPE html>
<html>
  <head>
    <!-- CSS start -->
    <style>
      p { color: green; }
    </style>
    <!-- CSS end -->
    <title>A title</title>
  </head>
  ...
```

## Adding Simple CSS Example 1 <body>

```
...
<body>
  <div>
    <h1>Heading of the first division</h1>
    <p>A first paragraph.</p>
    <p>A second paragraph with some <b>formatted</b>
      <b> text.</p>
    <p>A third paragraph with a <a href="http://
      www.lse.ac.uk">hyperlink</a>.</p>
  </div>
  <div>
    <h1>Heading of the second division</h1>
    <p>Another paragraph with some text.</p>
  </div>
</body>
</html>
```



## Adding Simple CSS Example 2 <head>

```
<!DOCTYPE html>
<html>
  <head>
    <!-- CSS start -->
    <style>
      .text-about-web-scraping {
        color: orange;
      }
      .division-two h1 {
        color: green;
      }
    </style>
    <!-- CSS end -->
    <title>A title</title>
  </head>
  ...
```

## Adding Simple CSS Example 2 <body>


```
...  
<body>  
  <div>  
    <h1>Heading of the first division</h1>  
    <p>A first paragraph.</p>  
    <p>A second paragraph with some <b>formatted</b>  
      <b> text.</p>  
    <p class="text-about-web-scraping">A third  
      paragraph now with text about web  
      scraping.</p>  
  </div>  
  <div class="division-two">  
    <h1>Heading of the second division</h1>  
    <p>Another paragraph with some text.</p>  
    <p class="text-about-web-scraping">A last  
      paragraph about web scraping.</p>  
  </div>  
</body>  
</html>
```

# Scenario 1: Data in table format

List of international courts [\[ edit \]](#)

Name	Scope	Years active	Subject matter
<a href="#">International Court of Justice</a>	Global	1945–present	General disputes
<a href="#">International Criminal Court</a>	Global	2002–present	Criminal prosecutions
<a href="#">Permanent Court of International Justice</a>	Global	1922–1946	General disputes
<a href="#">Appellate Body</a>	Global	1995–present	Trade disputes within the <a href="#">WTO</a>
<a href="#">International Tribunal for the Law of the Sea</a>	Global	1994–present	Maritime disputes
<a href="#">African Court of Justice</a>	Africa	2009–present	Interpretation of <a href="#">AU</a> treaties
<a href="#">African Court on Human and Peoples' Rights</a>	Africa	2006–present	Human rights
<a href="#">COMESA Court of Justice</a>	Africa	1998–present	Trade disputes within <a href="#">COMESA</a>
<a href="#">ECOWAS Community Court of Justice</a>	Africa	1996–present	Interpretation of <a href="#">ECOWAS</a> treaties
<a href="#">East African Court of Justice</a>	Africa	2001–present	Interpretation of <a href="#">EAC</a> treaties
<a href="#">SADC Tribunal</a>	Africa	2005–2012	Interpretation of <a href="#">SADC</a> treaties

# Scenario 2: Data in unstructured format



India English Register for updates Search 11,072,800 Visitors

I PAID A BRIBE

I DID NOT PAY A BRIBE

I MET AN HONEST OFFICER

BRIBE HOTLINE

ALL REPORTS

NEWS

REPORT A BRIBE

All Reports > I Paid A Bribe

ALL / I PAID A BRIBE / BRIBE FIGHTER / HONEST OFFICER / BRIBE HOTLINE

I PAID A BRIBE

1 day ago

76 views

**POLICE NILO GHUSS (bribe)**  
Passport | Police Verification for Passport | Paid INR 5,000  
Reported on January 17, 2016 from Bankura, West Bengal | Report #89544  
What will happen to this country.. police mamu's govt income: 30,000 per month. Per day GHUSS income 5000 (per passport verification). Imagine they t...[Read more](#)  
[How to Get a Passport Verified in Ghaziabad](#)

I PAID A BRIBE

1 day ago

104 views

**Corruption due to vague rules**  
Police | Traffic Violations | Paid INR 500  
Reported on January 16, 2016 from Mumbai, Maharashtra | Report #89509  
At Chembur near Eastern Expressway traffic cop stopped me and started checking docs..all was fine buy puc expired..then he pointed out film.. He took...[Read more](#)  
[Things to Know on Traffic Offences and Respective Penalties](#)

I PAID A BRIBE

2 days ago

105 views

**Bribe collected by Staff of Enrollment agency**  
Municipal Services | Aadhaar or UID Related | Paid INR 120  
Reported on January 16, 2016 from Mysore, Karnataka | Report #89467  
UIDAI has to take a stand on fees to be paid to enrolment agencies for processing Aadhaar

**FILTER REPORTS**

Which city?  
All cities

Department  
All departments

Bribe Amount  
All Amount

SUBMIT

**INSPIRE OTHERS  
WITH YOUR STORY**

Manik Taneja, a sports enthusiast, wrote against a custom official on [Ipaidabribe.com](#), for cough up a hefty bribe by a Customs official at Bengaluru airport.

SEE HIS STORY

Ever Paid A Bribe?


Report your Bribe Story!


See action taken.


www.ipaidabribe.com/reports/paid


19


# Scenario 3: Hidden behind web forms


 MONITOR  
LEGISLATIVO


 INICIO


 PERFIL IDEAL


 NOTICIAS

 CANDIDATOS



 ASAMBLEA NACIONAL

 ABUSOS

 CONTÁCTENOS







RESULTADOS DE LA CONSULTA



Seleccione  Partido 



BUSCAR



DIPUTADOS ENCONTRADOS



  
Unidad   
Julio Ygarza  
Estado: Amazonas

  
Unidad   
Mauligmer Baloa  
Estado: Amazonas

  
Unidad   
Nirma Guarulla  
Estado: Amazonas

  
Unidad   
José Brito  
Estado: Anzoátegui

  
Unidad   
Chaim Bucarán  
Estado: Anzoátegui

  
Unidad   
Richard Arteaga  
Estado: Anzoátegui

# Three main scenarios

## 1. Data in *table* format

- Automatic extraction with *requests* or select specific table with *inspect element* in browser

## 2. Data in *unstructured* format

- Element identification key in this case
  - *Inspect element* in browser
- Identify the target e.g. with *CSS* or *XPath* selector

## 3. Data hidden *behind web forms*

- Element identification to e.g. find text boxes, buttons, and results
- Automation of web browser with *Selenium*

## Identifying elements via CSS selector (1/2)

- Selecting by tag-name (e.g., *h3*)

```
<h3>This is the main item</h3>
```

- Selecting by class (e.g., *.itemdisplay*)

```
<div class='itemdisplay'>This is the main item</div>
```

- Selecting by id (e.g., *#maintitle*)

```
<div id='maintitle'>my main title</div>
```

## Identifying elements via CSS selector (2/2)

- Selecting by tag structure (e.g., hyperlink tag *a* inside *div* tag):

```
<div><a href='https://www.google.com'>Google  
Link</a></div>
```

- Selecting by nth child of a parent element (e.g., body > p:nth-child(2))

```
<body><p>First paragraph</p><p>Second paragraph  
.</p></body>
```

Reference and further examples: CSS Selectors



# The rules of the game

1. Respect the hosting site's wishes
  - Check if an API exists or if data are available for download
  - Respect copyright and ethics; what are you allowed to do?
  - Keep in mind where data comes from and give credit
  - Some websites disallow scrapers via *robots.txt* file
2. Limit your bandwidth use
  - Wait some time after each hit
  - Scrape only what you need, and just once
3. When using APIs, read documentation
  - Is there a batch download option?
  - Are there any rate limits?
  - Can you share the data?

# requests and BeautifulSoup

---

# Using the Requests Library

- **Requests** is a Python library for sending all kinds of HTTP requests.
- It is used to send requests to a server to access web resources.
- Basic usage:

```
import requests  
response = requests.get('https://example.com')
```

- Always check the response status to ensure the request was successful.

# Understanding HTTP Responses

- The **get** method returns a response object with several methods and properties.
- Commonly used properties:

```
response.status_code  # The status code of the  
                        response  
response.content      # The content of the  
                        response, in bytes  
response.text         # The content of the  
                        response, in Unicode
```

# Introduction to BeautifulSoup

- **BeautifulSoup** is a library for pulling data out of HTML and XML files.
- It works with parsers like *html.parser* or *lxml* to provide idiomatic ways of navigating and searching the parse tree.
- Basic usage:

```
from bs4 import BeautifulSoup  
soup = BeautifulSoup(html_content, 'html.  
parser')
```

- Where *html\_content* can be loaded from a website via the Requests library.

# Finding Elements with BeautifulSoup

- Use BeautifulSoup methods to find and extract data from HTML:

```
titles = soup.find_all('h1')  
for title in titles:  
    print(title.text)
```