The Economics Crime over Time and Space
Theory, Practice and Applications
(A primer in) Webscraping with Python

Arpita Ghosh

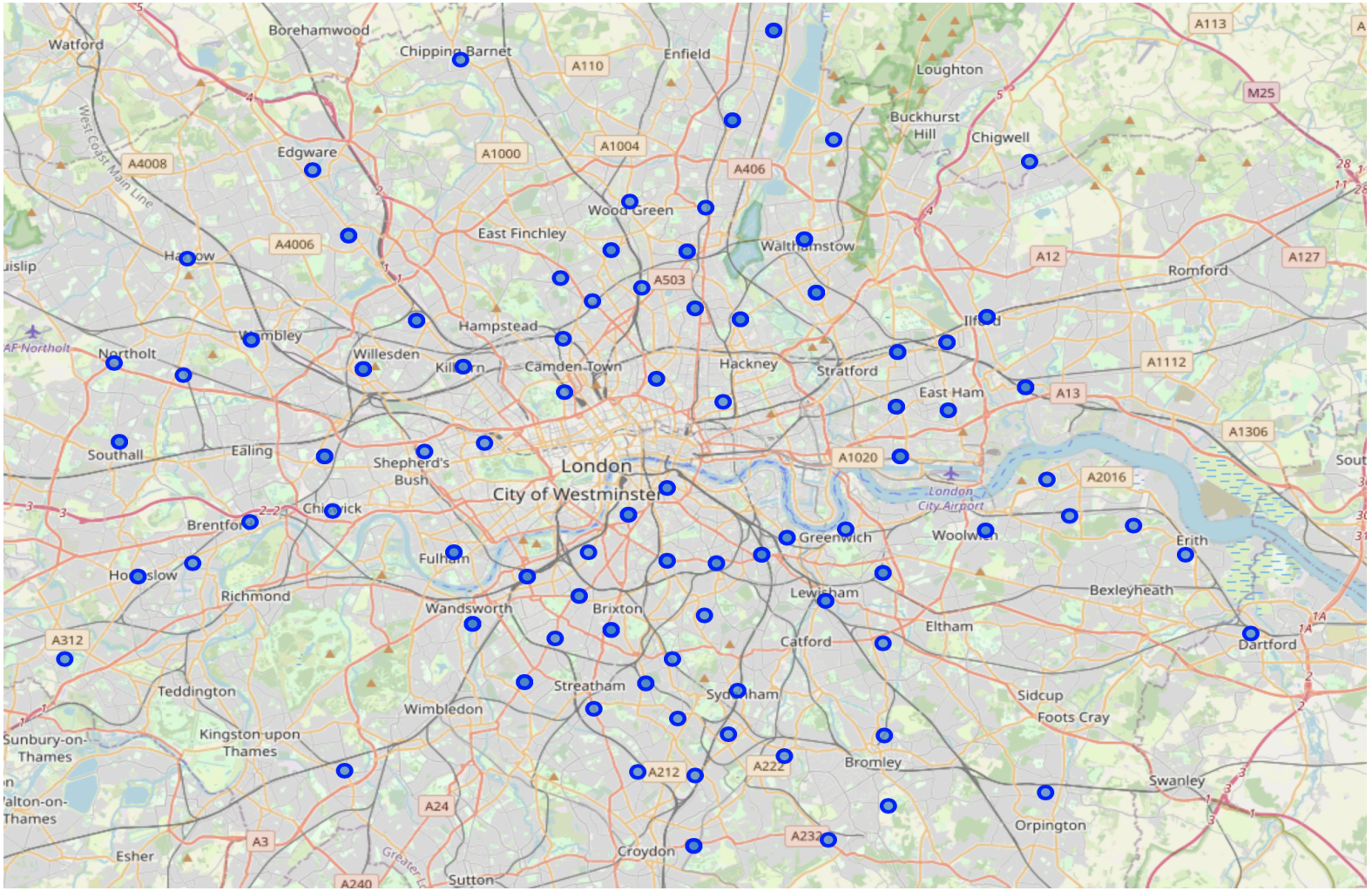
22nd September 2022

Why do we care?

- Vast and unused data online - newspapers, blogs, etc
- Quite rich data that can capture outcomes and in cases even treatments
- External validity - opinions, concepts, or even for robustness
- Computing power has become cheap and tools are available to use
- We like having more data!

At the end of my talk. . .

we will have mapped the unsolved murders in London (during 2005-2021), getting the data scraped from a website.



Aim for this session

1. Look at the website murdermap
    - Explore how to understand HTML (a little)
2. We focus on the unsolved murder cases tab
3. Understand how the website is organised and what do we want to obtain
    - Write a script to get the 1st level of data
4. Write a script to actually obtain the raw data of interest

Assumption/Goal/Declaration

Goal

We are only interested to extract the date and location of these murders

There is other info in the websites like name and age, but we ignore them.

Assumption

We are not looking to perfect the scraper

There are ways to make the scraper work more efficiently.

Declaration

I do cheat a little at places and don't write a beautiful code

The aim is to give the knowhow not have a data for research.

HTML structure

Hyper Text Markup Language

Essentially standard language for building web pages; with a series of elements

Mainly has two parts - HEAD (title of page, HTML version etc) and BODY (everything that the website needs to display)

Two type of HTML in scraping websites:

- Static websites: like Wikipedia or similar - use requests with Scrapy, Beautifulsoup, Selenium etc
- Dynamic websites: newspapers, social media etc. Can be also user specific like need to log in etc. - use Selenium or Splash

Most websites are dynamic, it's a matter of when it changes.

Locators: Xpath and CSS Selector

Web-element locators

Types of locators: element ID, field name, text, link text, CSS class, **Xpath, CSS selectors**

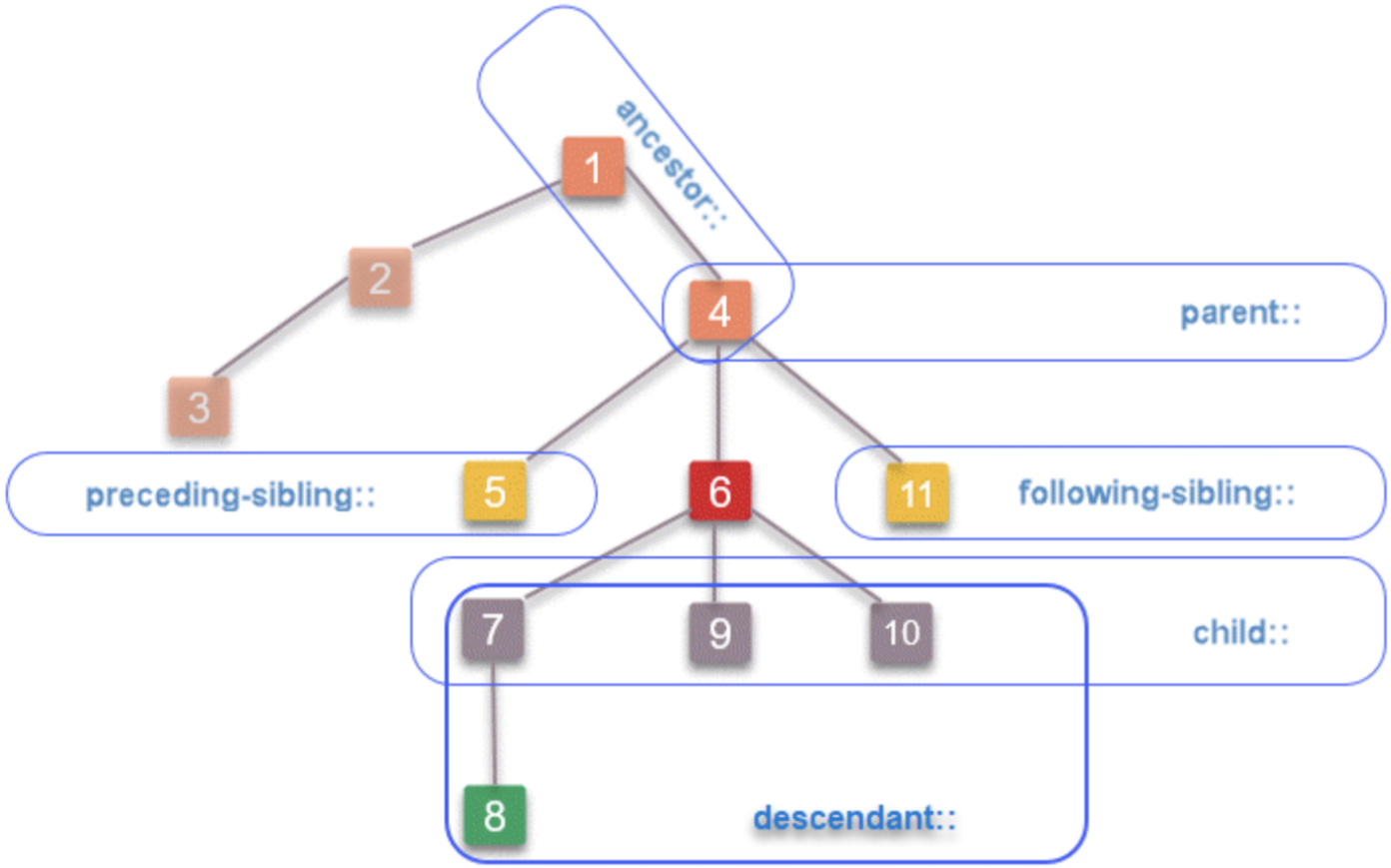
- *XPath:* Used in Selenoum to identify web elements uniquely on a web page. Can be absolute or relative. Always better to use relative Xpath (probability of getting it wrong is less).
- *CSS selectors:* Cascading Style Sheets (CSS) is a declarative language to control how websites look in browser - CSS selectors are used to locate HTML elements we want to style. There are different ones like element, id, class selectors. Prefer id as that is unique in an HTML.

Both of these are most frequently used to locate elements - CSS has better performance and speed than Xpath but Xpath is more versatile (can be used to locate elements up or down). We we will not go in more details in this session.

Xpath hierarchy

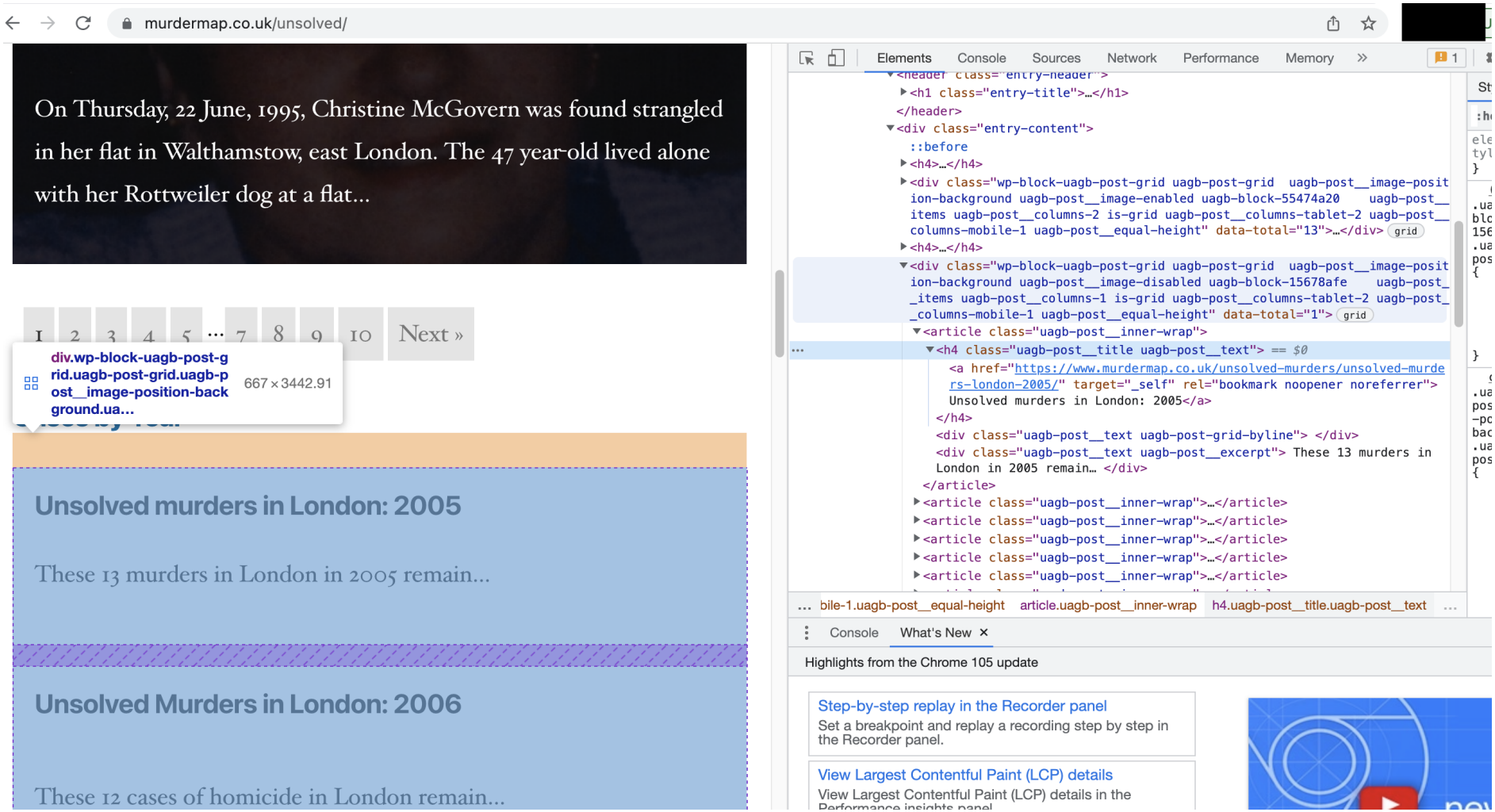
Xpath axes: locating elements relative to known elements

Ancestors, descendant, following, following sibling, preceding, child, parent etc



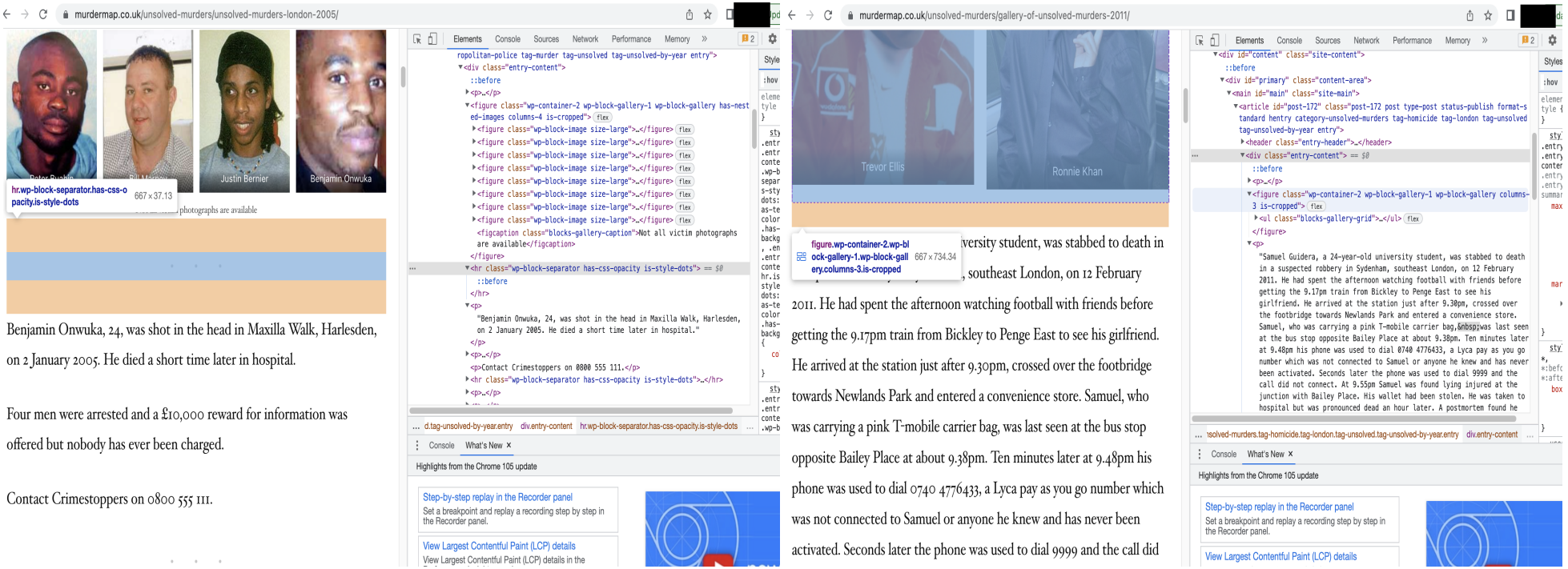
Source : XPath Expression components. http://www.iro.umontreal.ca/~lapalme/ForestInsteadOfTheTrees/HTML/ch04s01.html

Let's look at the website - unsolved tab



Open the murdermap website in Google and scroll down to the cases by year, right click on an element and select inspect.

Website structure - two cases: 2005, 2011



First look at the url links: they differ widely, not just the years. Second look at the way the paragraphs are arranged: sometimes with separators after the picture block and sometimes not! **Messy website!**

Some important things

API vs Scraping

Both gives access to data on websites: if publicly available, use API (Application Programming Interface). But can be expensive, limited, or might not even exist. In that case can build essentially own API by web scraping.

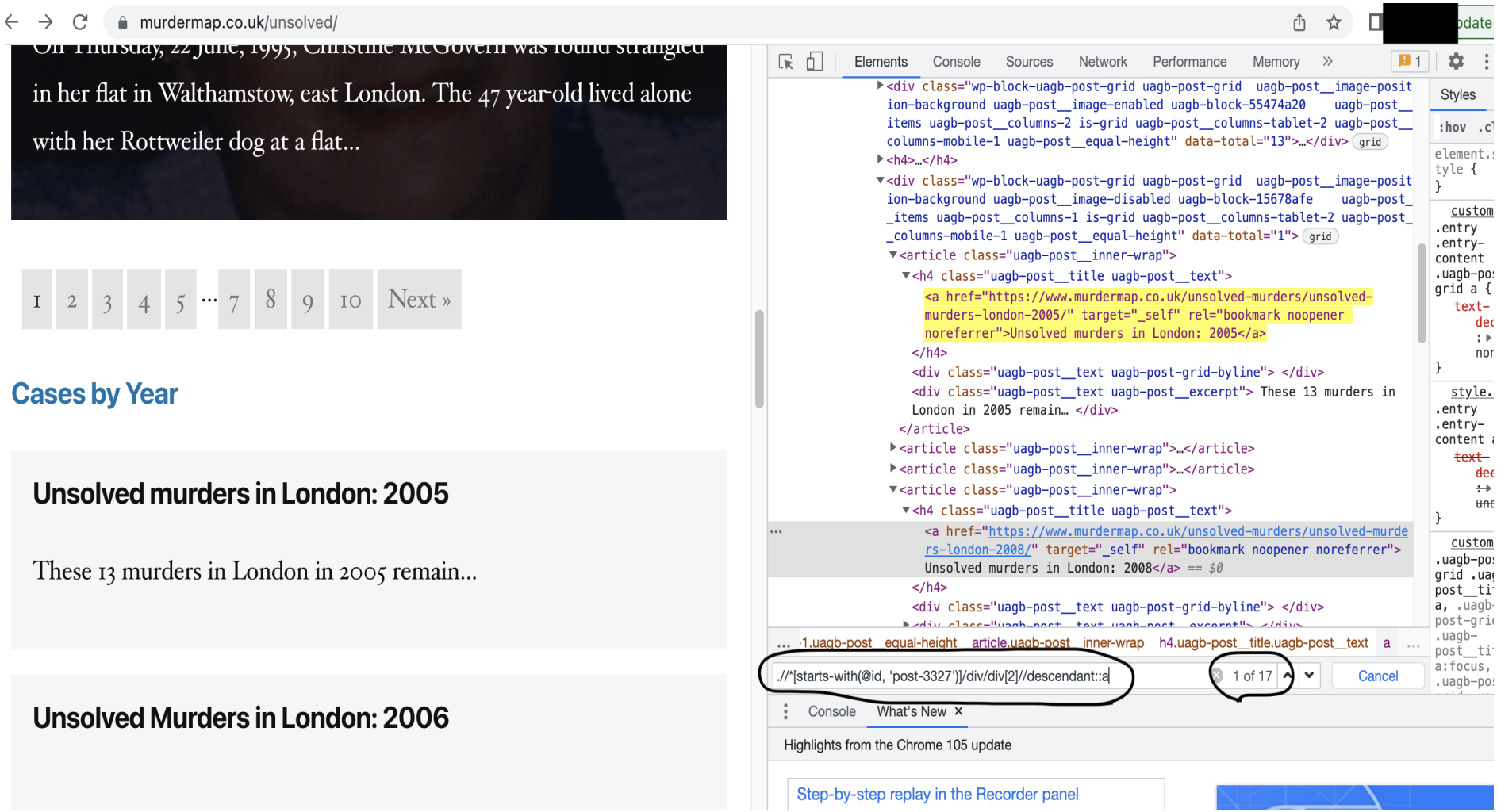
- Ethics: read note of terms of use; create new data, not duplicate it; reasonable queries; providing User Agent string with clear intentions/contact (easier with requests, more complicated in Selenium) etc.
- Use of VPNs/proxy: allows crawling websites with reduced chances of getting blocked
- Imitate human actions: Using random time delays between commands or actions, having user agent etc.

Extracting URLs

- We need to get the target urls first from the unsolved tab in murdermap.
  - https://www.murdermap.co.uk/unsolved-murders/unsolved-homicides-2012/ or
  - https://www.murdermap.co.uk/unsolved-murders/londons-unsolved-murders-2010/ or
  - https://www.murdermap.co.uk/unsolved-murders/unsolved-murders-london-2005/ etc

- Good to know how to do in stages, as it can be used in scraping other sites, newspapers - like BBC, Wayback Machine etc

Extracting URLs - the code

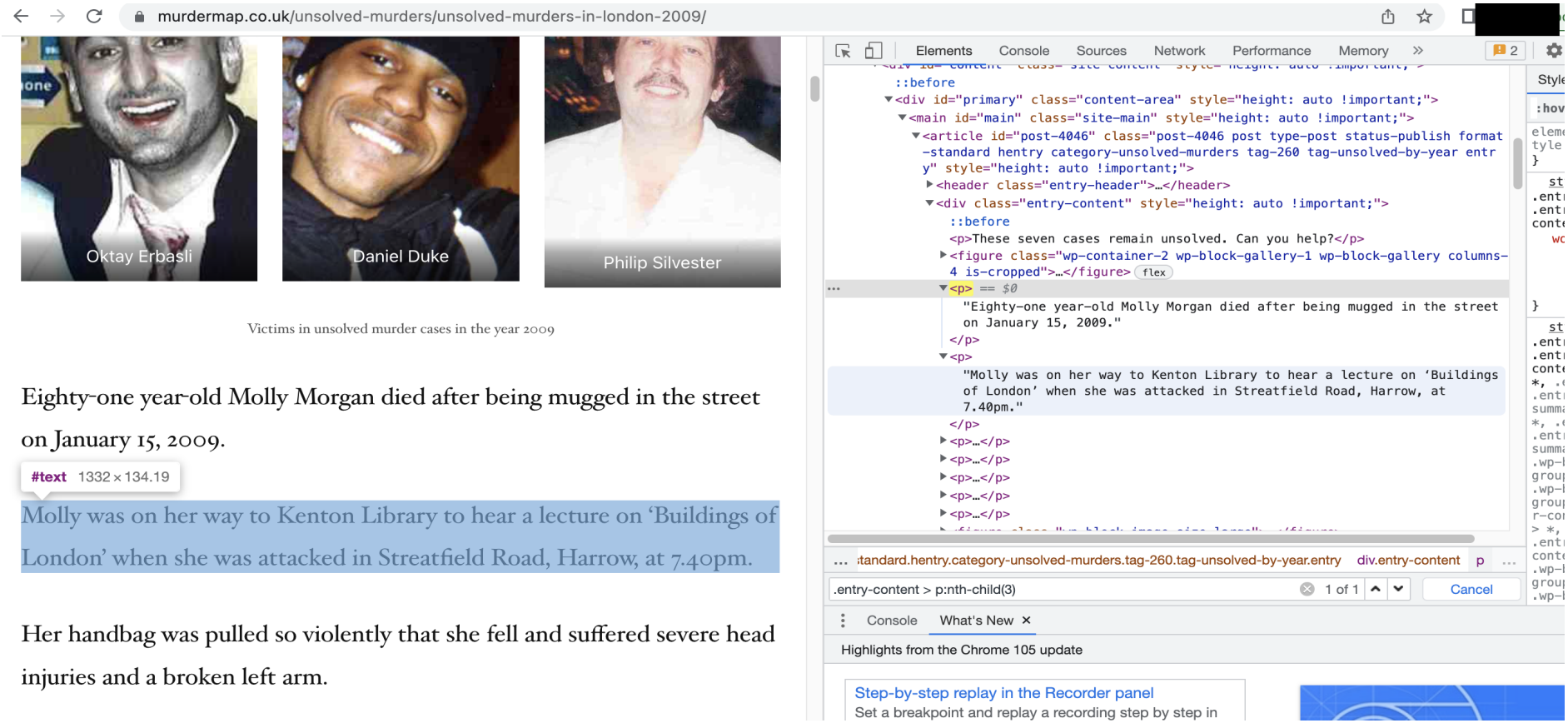


Write a loop to extract the href attributes from the elements with the given Xpath: 17 years - all urls

Extracting Data

- Target: dates and locations of unsolved murders in London.
- Looking at the texts - mainly the first 2 paragraphs are of interest.
- Have the date and location along with name, age, how etc.
- Sometimes the 1st case of a year start after 3 dots following the picture block.
- Sometimes they start without any separator.
- Messy structure $\Rightarrow$ nested loops and exceptions handling

Extracting Data - the code



Split in types: when there is a separator in the beginning (Type 1) extract 2 'p' sibling texts, if there are separators in the page but not before the 1st case (Type 2) extract the 2 sibling where separators are and the 1st case as well, if there is no separators at all (Type 3) extract paragraph texts as cases.

Wrapping up

- Observed how to extract some text data from a messy website.
- Words of caution:
  - must be careful with number of attempts
  - Scraper.py can be improved to count how many paragraphs are there for each case and extract 2 or 1 from it
  - entire code is specific to this case - NOT a generic scraper

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Back