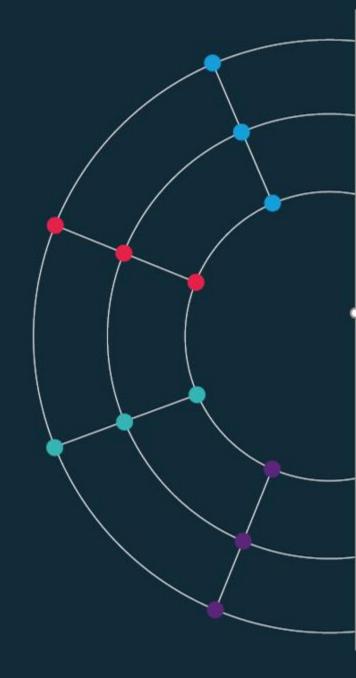


Data scraping



Data scraping

Scraping the HTML source

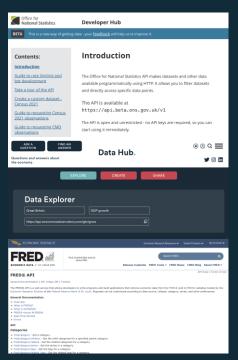


Data, so far...

Today we have used data from



Structured Files (e.g. Excel, CSV, JSON)

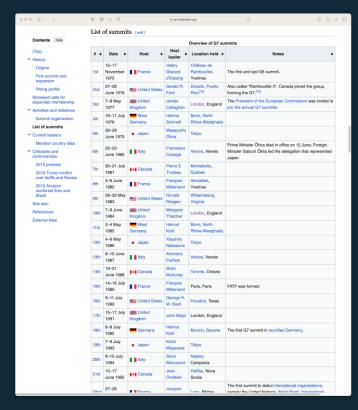


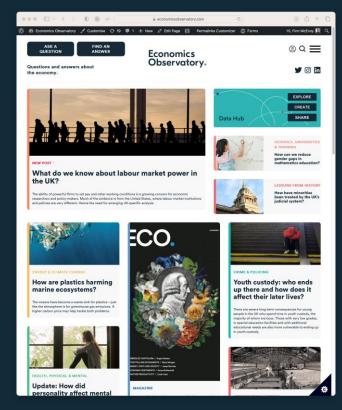
APIs (e.g. ONS, ECO, FRED)

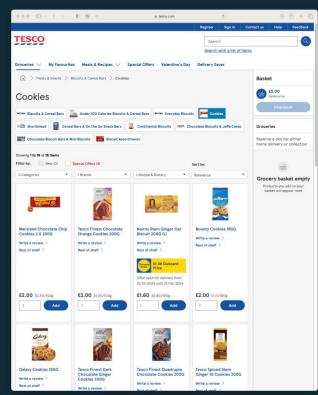
But what do we do when the data we want isn't available?

Data, so far...

What if we want







Data from Wikipedia

News and Media

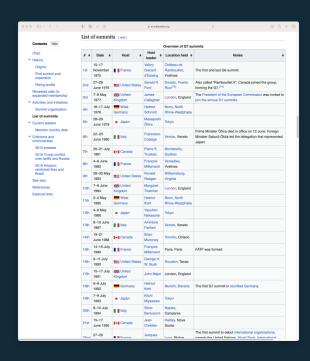
Prices from Supermarkets

Scraping.

- The automated extraction of data from websites
- Scraping the HTML source. Easy. Automated.
- Scraping static HTML pages.
 - A bit more difficult. Can be automated.
 - CAPTCHA. Impersonating a human user. Zombie browser.
- Scraping HTML pages generated on-the-fly with JavaScript.
 - Hard. Only zombie browser works, and only in some cases.

Your first scraper.

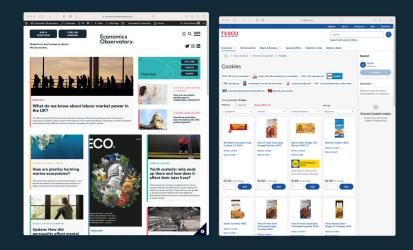
1. Extract data from Wikipedia





Your first scraper.

2. Take a quick look at more complicated scraping



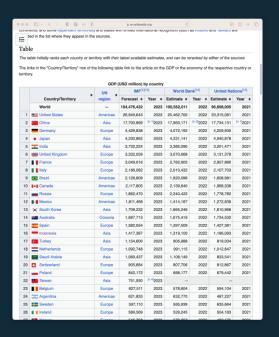
3. Share the tools to learn more

Overview.

- 1. Identify the data needed
- 2. Look Around do you need to scrape?
- 3. Scrape the data
- 4. Cleaning and Visualising

Identify the data needed.

- You can scrape almost anything
- ... but scraping is most useful for hard-to-find data



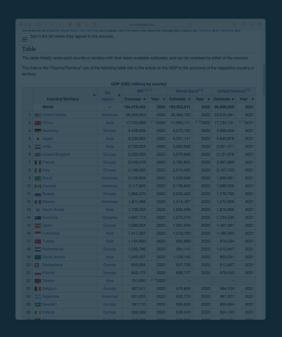
You could scrape

Wikipedia's list of

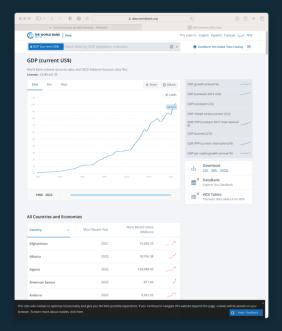
countries by GDP...

Identify the data needed.

- You can scrape almost anything
- ... but scraping is most useful for hard-to-find data



You could scrape
Wikipedia's list of
countries by GDP...

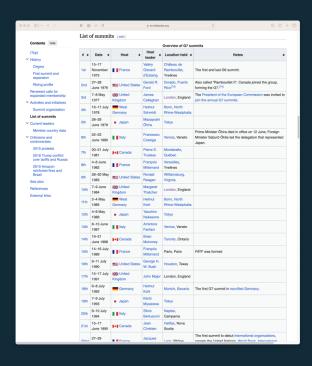


... but there's no point

If you can just download
the data

Look around.

- Search the web for exactly the data you want
- Try to find a download first



Our Example

Table of G7 Meetings

- √ Excel/CSV Unavailable
- √ Table Available

Scrape the data.

- Scraping data from tables on webpages is easy with Python
- We can use Pandas, which we have already seen today

pd.read_html(url)

(Loads every table from a webpage)

Scrape the data.

• To read all the tables, we point url to our example page

```
url = "https://en.wikipedia.org/wiki/G7" tables_from_webpage = pd.read_html(url)
```

Which makes a list of every table.

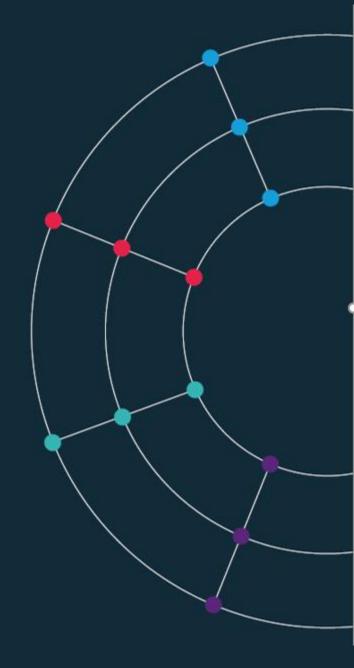
Data scraping

Scraping the HTML source



Data scraping

https://github.com/EconomicsObservatory/courses/blob/main/README.md



Code-along.

Your first scraper

In this fifth practical session, we will use Google Colab to use Python to scape data from Wikipedia (and another example if there is time). We will also embed a chart displaying the scraped data into your website, using VS Code and GitHub.

- We will run you through the following guided notebook:
 "Session_5_Scraping_basic.ipynb" (open in Google Colab)
- For a further advanced examples, go to:
 "Session_5_Scraping_advanced.ipynb" (open in Google Colab)

Scrape the data.

The list of G7 meetings, our target, is the 3rd table on the webpage:

tables_from_webpage[2]

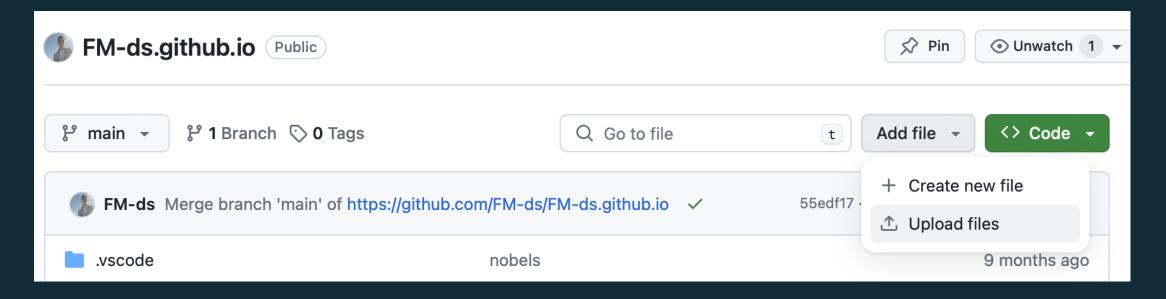
	#	Date	Host	Host leader	Location held	Notes
0	1st	15–17 November 1975	France	Valéry Giscard d'Estaing	Château de Rambouillet, Yvelines	The first and last G6 summit.
1	2nd	27–28 June 1976	United States	Gerald R. Ford	Dorado, Puerto Rico[74]	Also called "Rambouillet II". Canada joined th
2	3rd	7–8 May 1977	United Kingdom	James Callaghan	London, England	The President of the European Commission was i
3	4th	16-17 July 1978	West Germany	Helmut Schmidt	Bonn, North Rhine- Westphalia	NaN
4	5th	28-29 June 1979	Japan	Masayoshi Ōhira	Tokyo	NaN
5	6th	22–23 June 1980	Italy	Francesco Cossiga	Venice, Veneto	Prime Minister Ōhira died in office on 12 June
6	7th	20-21 July 1981	Canada	Pierre E. Trudeau	Montebello, Québec	NaN
7	8th	4–6 June 1982	France	François Mitterrand	Versailles, Yvelines	NaN
8	9th	28-30 May 1983	United States	Ronald Reagan	Williamsburg, Virginia	NaN
9	10th	7–9 June 1984	United Kingdom	Margaret Thatcher	London, England	NaN

- We have a messy table of data
- Let's clean it up to answer the following question:
 - 'What's the most popular location for G7 meetings?'

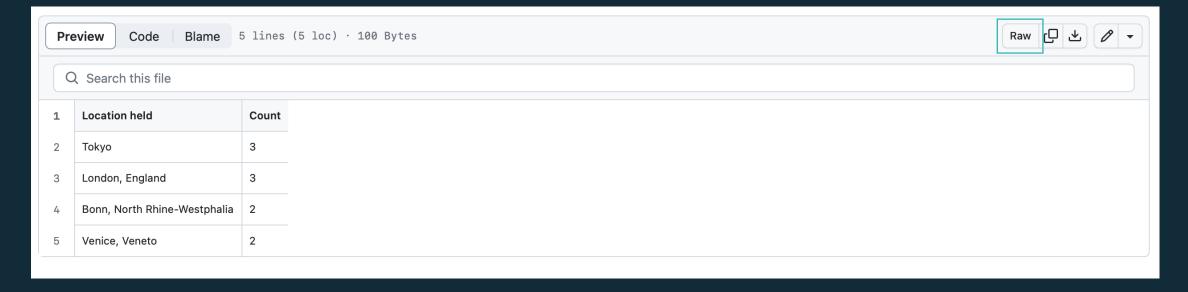
	#	Date	Host	Host leader	Location held
0	1st	15–17 November 1975	France	Valéry Giscard d'Estaing	Château de Rambouillet, Yvelines
1	2nd	27-28 June 1976	United States	Gerald R. Ford	Dorado, Puerto Rico[74]
2	3rd	7-8 May 1977	United Kingdom	James Callaghan	London, England
3	4th	16-17 July 1978	West Germany	Helmut Schmidt	Bonn, North Rhine- Westphalia
4	5th	28-29 June 1979	Japan	Masayoshi Ōhira	Tokyo
5	6th	22-23 June 1980	Italy	Francesco Cossiga	Venice, Veneto
6	7th	20-21 July 1981	Canada	Pierre E. Trudeau	Montebello, Québec
7	8th	4-6 June 1982	France	François Mitterrand	Versailles, Yvelines
8	9th	28-30 May 1983	United States	Ronald Reagan	Williamsburg, Virginia
9	10th	7–9 June 1984	United Kingdom	Margaret Thatcher	London, England

(full code in Notebook)

After saving our table and uploading to GitHub, we can use it in Vega-Lite



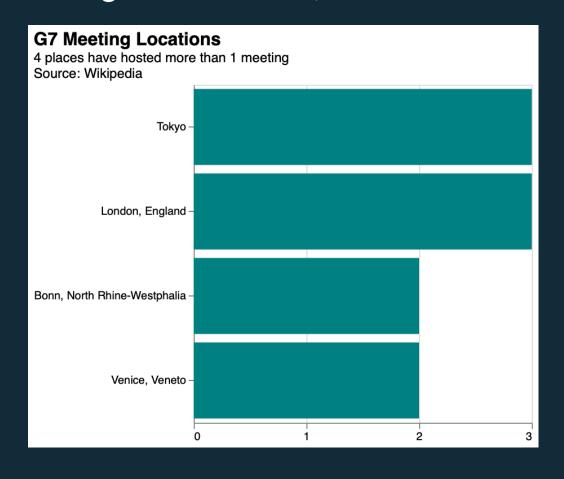
After saving our table and uploading to GitHub, we can use it in Vega-Lite



After saving our table and uploading to GitHub, we can use it in Vega-Lite

```
Location held, Count Tokyo, 3
"London, England", 3
"Bonn, North Rhine-Westphalia", 2
"Venice, Veneto", 2
```

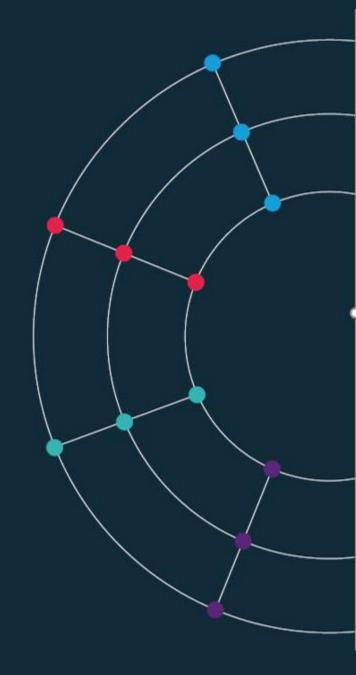
Linking to our data, we can use it in a chart:



(see "chart_g7_meeting_hosts.json")

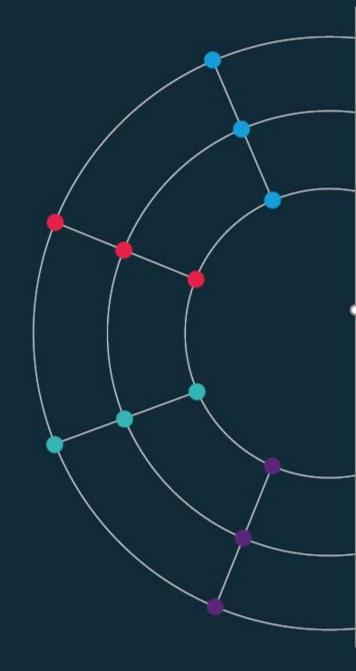
Data scraping

Scraping the HTML source (advanced)



Data scraping

https://github.com/EconomicsObservatory/courses/blob/main/README.md

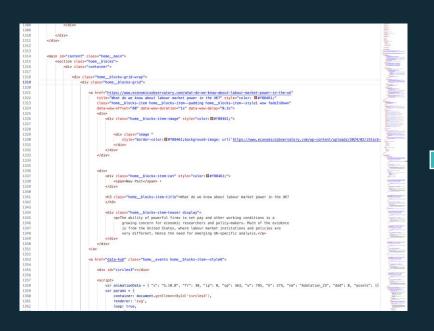


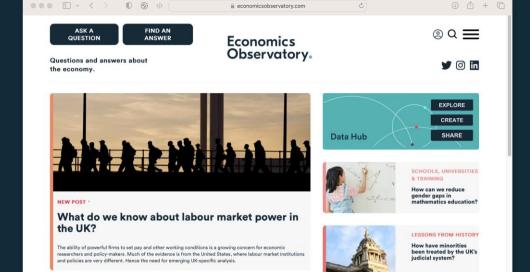
Scraping HTML Source.

- Scraping tables is easy but sometimes we want data that isn't nicely formatted
- Instead, we can extract data by searching the HTML
- Everything is defined in the HTML, we just have to find it

Scraping HTML Source.

For example, we can scrape ECO headlines and tag-lines by 'parsing' the HTML

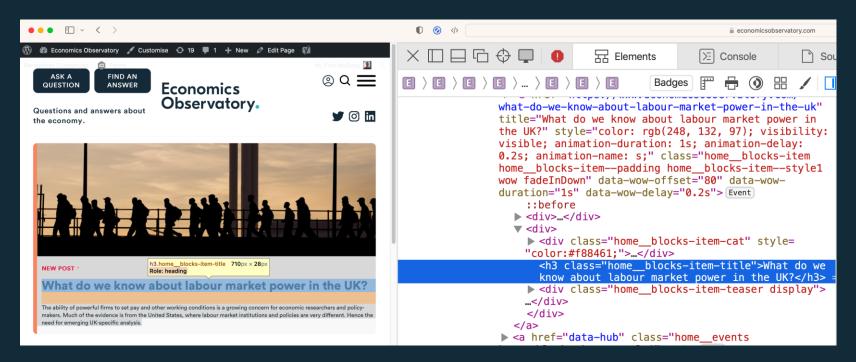




meconomicsobservatory.com

Scraping HTML Source.

We determine how the target data is defined using 'inspect-element'



We see titles have a class "home_blocks-item-title"

Code-along.

A more advanced scraper

In this bonus practical session, we will use Google Colab to use scrape data from the Economics Observatory website using Python. Again, we can also embed a chart displaying the scraped data into your website, using VS Code and GitHub.

Work through the following guided notebook:
 "Session_5_Scraping_advanced.ipynb" (open in Google Colab)

Learn more.

- In this session we have tried basic data scraping with Pandas and seen advanced scraping with BeautifulSoup
- There's still much more to learn
 - Choose your own projects (how can you make your job easier?)
 - Try bigger projects (scrape 100 pages, not just 1)
 - Try advanced tools (e.g. Selenium)

Learn more, responsibly.

- Rate-limiting: avoid making too many requests at once
- Ethics: Ensure your scraping activities do not harm the website's operation
- Data Privacy: Be mindful of personal data collection. Comply with relevant data protection laws (like GDPR).

