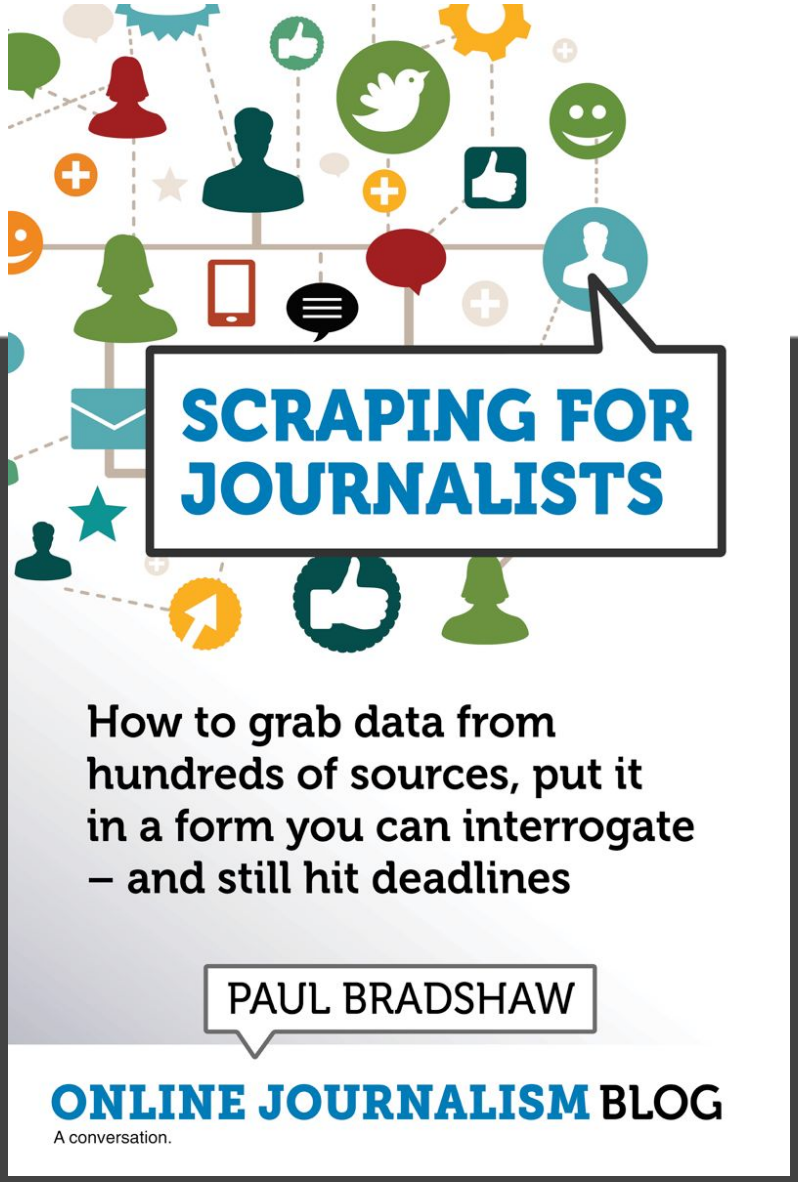
The 8-line scraper



Paul Bradshaw
Leanpub.com/scrapingforjournalists

What we'll cover

- Recap: selectors (stripping it back to 8 key lines)
- Expanding the scraper: more columns, more pages
- Potential obstacles

```
#import the 3 libraries
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

```
#fetch the page from the URL
response =
requests.get("https://theferret.scot/articles/")


#turn into a beautiful soup object
soup =
BeautifulSoup(response.content,'html.parser')
```

```
#select and store our target tags
alistofmatches = soup.select("h2 a")


#use that list as a column in a data frame
df = pd.DataFrame( { "column 1" : alistofmatches } )
#export that as a CSV file
df.to_csv("scrapeddata.csv", index=False)
```

Tag-and-text items

- Each item in the list extracted by `soup.select( )` is a combination of the HTML tag and any text (and other tags) it contains
- You can drill into these with special soup ‘methods’. The most common are:
  `.get_text()`
  `[‘href’]`
- These fetch just the text, and just the href= value (the link) respectively
- Apply them to all items within the list by looping through it.

```
#create an empty list
justhref = []


#loop through our scraped tags
for i in alistofmatches:
  #extract href= value and append to the list
  justhref.append(i['href'])


#use *that* list as a column in a data frame
df = pd.DataFrame( { "column 1" : justhref } )
```

Indented lines of code indicate what code runs 'inside the loop'

Expanding the data frame

- The data frame is created with a **dictionary** that looks like this `{ "column 1" : alistofmatches }`
- The first part is just a string that names the column
- The second part (after the colon) is the name of a list which will fill that column
- To add more columns, just repeat, with a different column name and a different list, with a comma between each pair:
  ```
  { "column 1" : justtext,
  "column 2" : justhref}
  ```
- This goes within the code that creates the data frame:
  `df = pd.DataFrame( )`

```
#create a data frame where those matches fill
two columns, each of which is named in quotation
marks

df = pd.DataFrame( { "column 1" : justtext,
                     "column 2" : justhref} )
```

Two lists - they have to be the same length or you will get an error

Expanding to other pages

- Check if the ‘next’ page has a page number in its URL
- Create a list of numbers to loop through to create those URLs using `range( )`
- Loop through that list of numbers and add it to the URL - but you’ll need to convert to a string first.
- Create an empty list before looping through the list, and add each URL to that list as you loop through and create each one

```
#create an empty list to store the URLs we are about to
generate
url_list = []

#loop through the numbers 1 to 100
for i in range(1,100):
 #convert to a string, add to the end of a base URL
 pagedurl =
"https://www.gov.uk/employment-tribunal-decisions?page="
+str(i)
 #append to the list
 url_list.append(pagedurl)
```

Adaptations

You have the basic code, it’s all about adapting that to the specific page(s) you want to scrape

- Change the URL
- Change the selector
- Create new lists with other selectors
- Add those as extra columns in the data frame
- Generate lists for paginated URLs
- Scrape lists for linked URLs
- Loop through the list and apply scraping code
- Store the results in a list of data frames
- Concatenate those into one