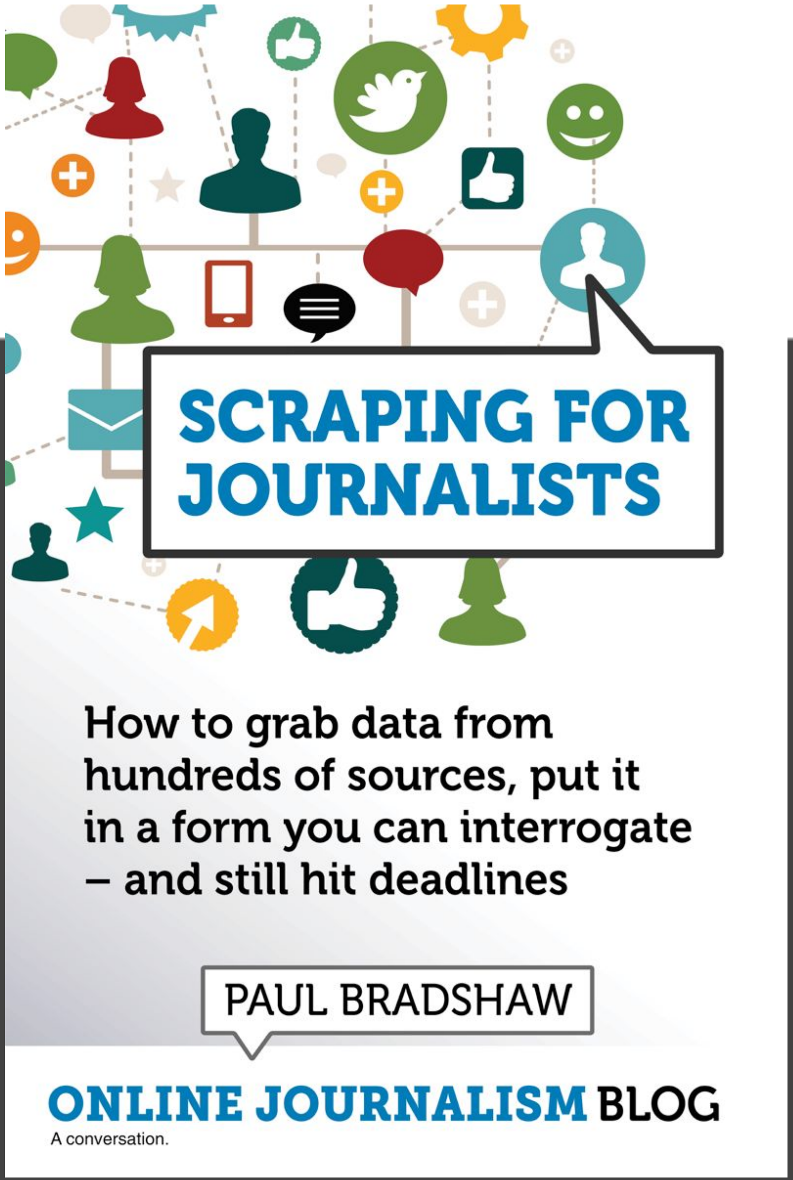
Lists redux: storing the data


SCRAPING FOR JOURNALISTS
How to grab data from hundreds of sources, put it in a form you can interrogate – and still hit deadlines
PAUL BRADSHAW
ONLINE JOURNALISM BLOG
A conversation.

Paul Bradshaw

Leanpub.com/scrapingforjournalists

What we’ll cover

- Using **lists** to extract the data you want
- Using **pandas** data frames to store it

The story so far

- We’ve extracted 50 <div> tags
- (You might have grabbed 50 dates too)
- How do we store them?

The data needs to line up

We need 50 of each, so check:

- You have the right number
- The first and last items match what you expect

If not, try:

- Change the selector to be more specific
- Add an index/slice to select the right range of items
- Select every other item
- Google solutions to your problem/ask ChatGPT

Tip: slicing a list

Slicing a list involves specifying a start and end index like so:

```
  first10 = mylist[0:10]
```

If you don’t specify a start or end point, it will default to the start or end of the list:

```
  first10 = mylist[:10]
  from10on = mylist[9:]
```

Don’t forget negative indices too:

```
  last10 = mylist[-10:]
```

We want to extract text

- `.select( )` grabs the tags-and-text and produces a **list** of matches
- We can also drill down further, into just the text of **each item**
- Add `.get_text()` to a single item to do just that

```
item1text = mylist[0].get_text
```

- Create a loop to do it to each item in turn

```
#grab the first item from the list
'divswewant'

#apply the .get_text() method to it to
grab the text


divswewant[0].get_text()
```

```
#loop through the divswewant list
for i in divswewant:
    #extract the text
    casename = i.get_text()
    print(casename)
```

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


#loop through the divswewant list
for i in divswewant:
   casename = i.get_text()
   #add the text and link to the
previously empty lists
   casetitles.append(casename)
```

Introducing pandas!

We need to store the data

The pandas library has functions to create a data frame (table) and add to it

- The **`pandas.DataFrame()`** function creates a data frame with specified columns
- If you imported `pandas as pd`, then it's **`pd.DataFrame()`**
- Lists can be used as columns.

```
#create a dataframe which uses
two lists as its two columns
casedataframe = pd.DataFrame(
{"case name" : casetitles, "date"
: datelist} )
```

Introducing dictionaries!

The dictionary variable

- Uses **curly brackets**
- Contains a list of **pairs**, separated by a colon
- `{“name” : “Paul”, “age” : 21}`
- The first part of the pair is the **key**
- The second part is the **value**
- ...So they’re called **key-value pairs**
- The key is always a string; the value can be a string, number, True/False, or anything else
- Multiple dictionaries can be used to create rows in a table, e.g. row 2 might be:
  `{“name” : “Xian”, “age” : 31}`

Creating a dictionary

```
#create a dictionary
#with 2 key-value pairs
mydictionary = {“name” : “Paul”,
“age” : 21}
```

Expanding a dictionary

- `#create an empty dictionary`
  `mydictionary = {}`
- `#create a key and store a value`
  `mydictionary[‘name’] = “Paul”`
  `mydictionary[‘age’] = 21`
- `#print the dictionary`
  `print(mydictionary)`

```
#create a dataframe which uses
two lists as its two columns

casedataframe = pd.DataFrame(
{"case name" : casetitles, "date"
: datelist} )
```

Curly brackets = the dictionary

We need to export the data

The pandas library has functions to import and export data to and from CSV

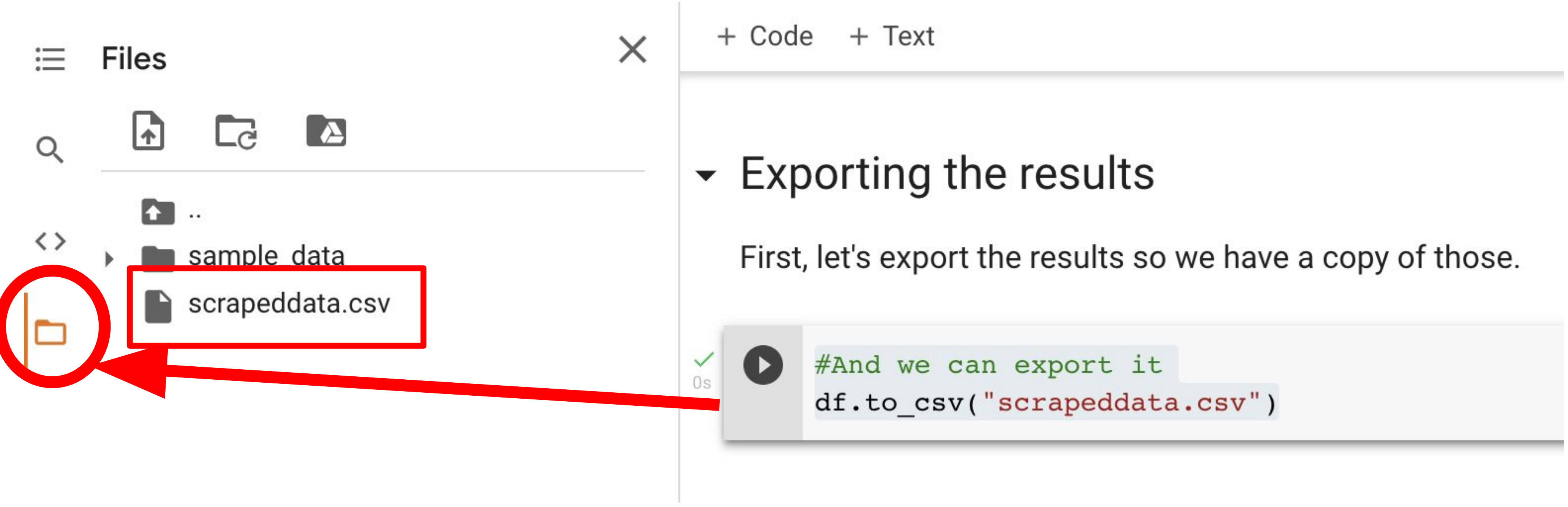
- The **`.to_csv()`** function creates a CSV with a specified name, using the data frame it's attached to
  `mydataframe.to_csv("mycsv.csv")`

- The CSV file will be in the Files area in the left hand navigation in Colab

```
casedataframe.to_csv(“scrapeddata
.csv”)
```

Get it out!

```
#export it
df.to_csv("scrapeddata.csv")
```



Exporting the results

First, let's export the results so we have a copy of those.

```
#And we can export it
df.to_csv("scrapeddata.csv")
```

Try it now:

- In your notebook scrape the page and extract the contents of:
  - <div> tags with class="gem-c-document-list__item-title"
  - <time> tags

- Loop through those tags and append the results of `.get_text()` to a new list
- Store the two lists in a dataframe
- Export the dataframe as a CSV

Recap

- Use loops and **.append( )** to create new lists based on old lists (e.g. getting the text of each item)

- Use pandas to create a data frame to store data

We want to extract attributes

- **`.select( )`** grabs the tags-and-text and produces a **list** of matches
- We can also drill down further, into attributes of **each item**
- Add **`[‘href’]`** to a single item to grab the href=“” attribute
  `item1text = mylist[0][‘href’]`

- Create a loop to do it to each item in turn

Going into child tags

- **`.select( )`** grabs the tags-and-text and produces a **list** of matches
- We can also drill down further, into tags within **each item**
- Add **`.select( )`** again, to a single item to grab a specified tag
  `item1text = mylist[0].select(‘a’)`
- Remember this will create another list, so you’ll need to drill down to a specific item
  `item1text =`
  `mylist[0].select(‘a’)[0]`

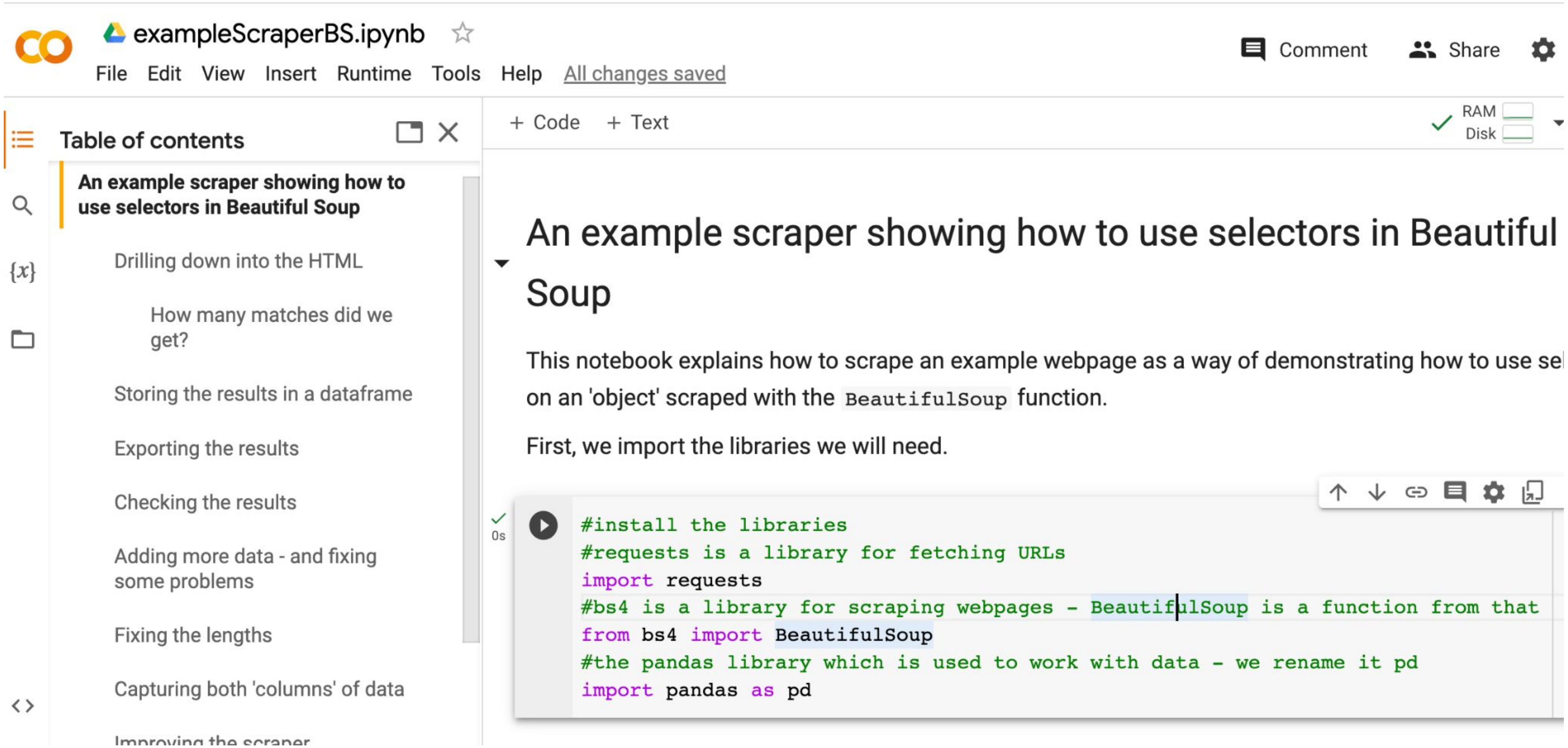


Table of contents
An example scraper showing how to use selectors in Beautiful Soup
Drilling down into the HTML
How many matches did we get?
Storing the results in a dataframe
Exporting the results
Checking the results
Adding more data - and fixing some problems
Fixing the lengths
Capturing both 'columns' of data
Improving the scraper

An example scraper showing how to use selectors in Beautiful Soup

This notebook explains how to scrape an example webpage as a way of demonstrating how to use sel
on an 'object' scraped with the `BeautifulSoup` function.

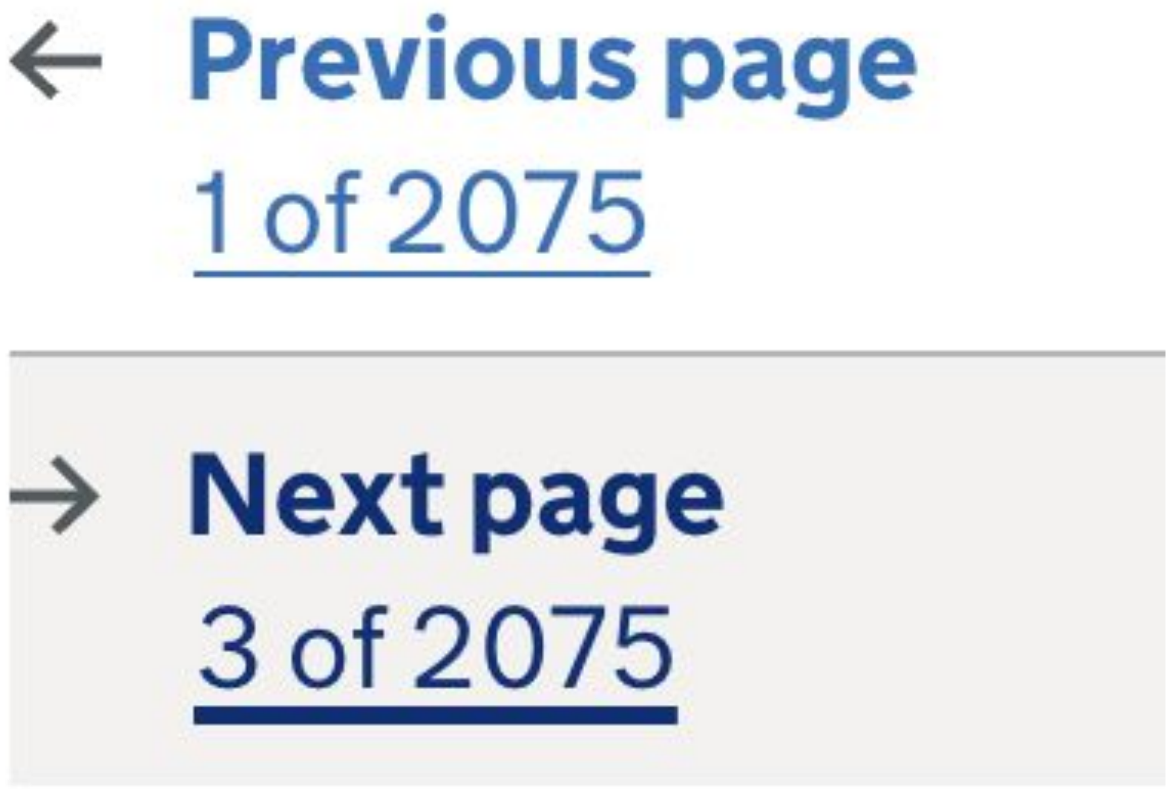
First, we import the libraries we will need.

```
#install the libraries
#requests is a library for fetching URLs
import requests
#bs4 is a library for scraping webpages - BeautifulSoup is a function from that
from bs4 import BeautifulSoup
#the pandas library which is used to work with data - we rename it pd
import pandas as pd
```

https://colab.research.google.com/drive/1UuFhIQYB7K6cjONPNOaGfQeQbqTv-FkE?usp=sharing

The next 7 days:

- Make a copy of the two Colab notebooks shared with you this week
- Work through them, running the code - try adapting it and see what happens
- Try to use the knowledge from this week to adapt some code suggested by ChatGPT/Gemini
- Try to apply it to a webpage that interests you - what errors do you get? What new challenges do you face? Share your notebook with Paul!



← Previous page

1 of 2075

→ Next page

3 of 2075

https://www.gov.uk/employment-tribunal-decisions**?page=2**