DATA2001

Data Science, Big Data and Data Variety

Week 6

Web Scraping

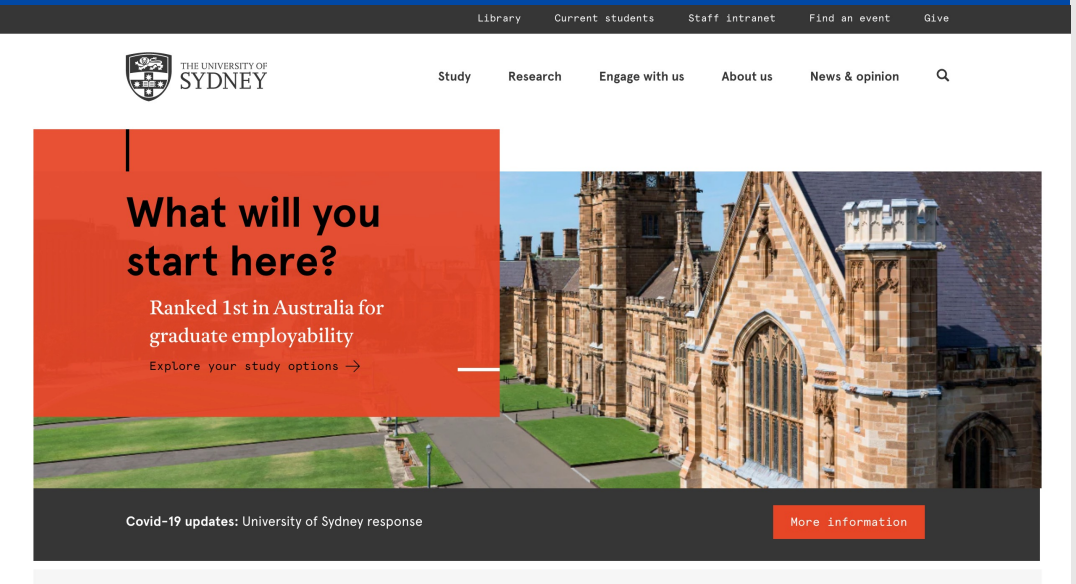
THE UNIVERSITY OF SYDNEY

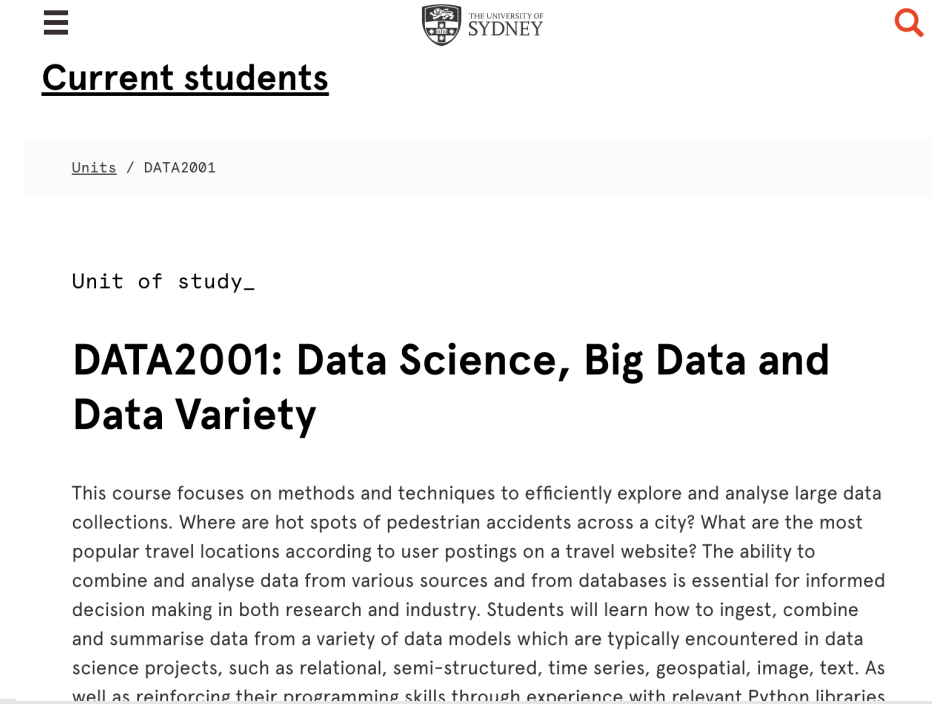
Outline

- Introduction – 10 mins
- Scraping Data Exploration – 50 mins
- Web Traversal, Storage and Querying – 40 mins

Fair warning: today's tutorial is quite Python heavy, and may get quite detailed. Don't stress about remembering all the syntactic semantics – our focus today is ***exploring a semi-structured data source*** *and* ***exercising problem solving****.*
(even if the example can be a little tricky!)

Dataset





Not all data is presented as neatly as a structured dataframe of rows and columns, like the datasets we've used so far. Often, meaningful information exists in unstructured or semi-structured formats.

Today we'll explore how to extract data from **webpages**, by focussing firstly on a familiar example – the online **UoS outline** for this subject – and then investigating how this can be scaled up in application.

Webpages

- Documents intended for web browsers are written in **HTML** (HyperText Markup Language)
  - These are tree-like structures, comprising multiple elements that start and end with tags
  - e.g.

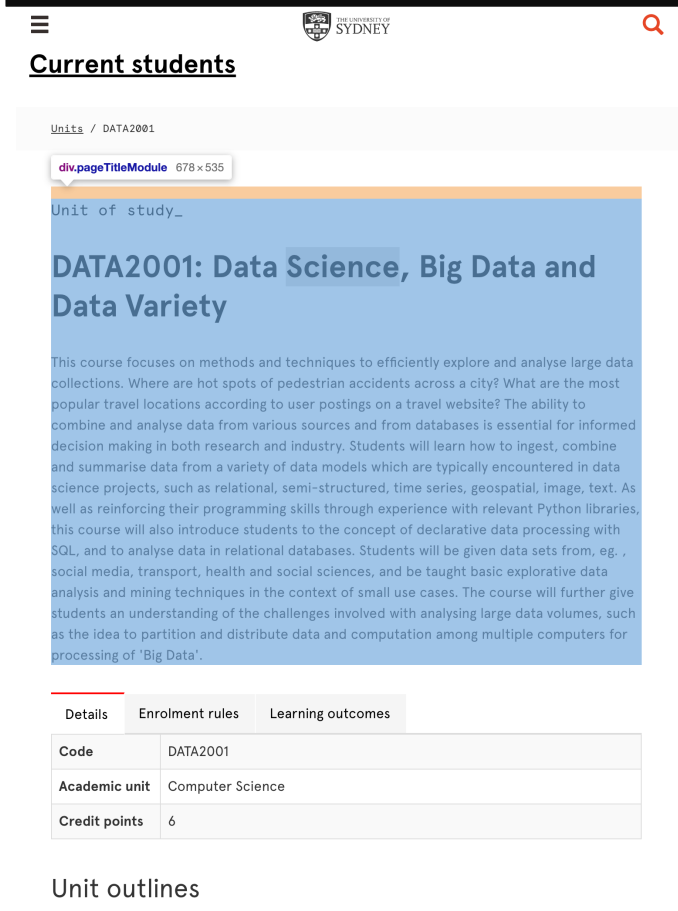
```
<div class="firstSection">
   <h1>This is a heading!</h1>
   <p>This is a paragraph containing text</p>
   <a href="https://bit.ly/3JX9nLM">This is a link</a>
</div>
```

→ this would have a heading element ('h1' tag), a paragraph ('p' tag) and a hyperlink ('a' tag), all contained within a 'div'

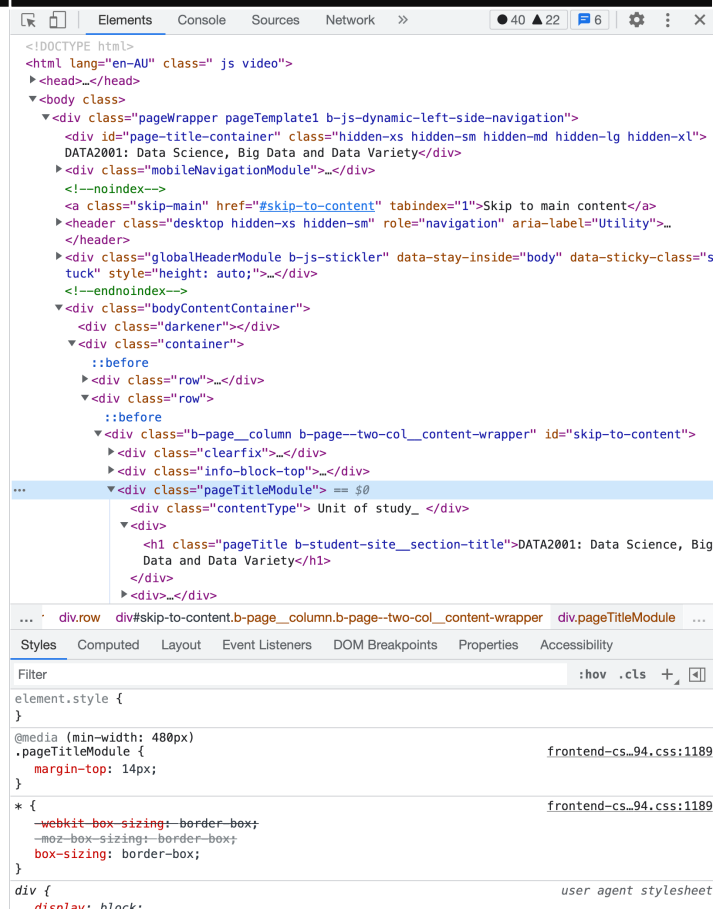
→ elements can have classes (not unique), or a single id (unique), which is particularly useful for setting styles with **CSS**

Inspect Element

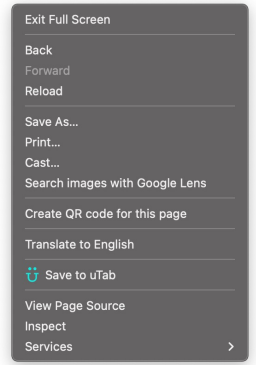
Webpage



Content



*Right clicking anywhere on a webpage should give an "**Inspect**" or "Inspect Element" option, which reveals, and allows interaction with, the underlying source code.*



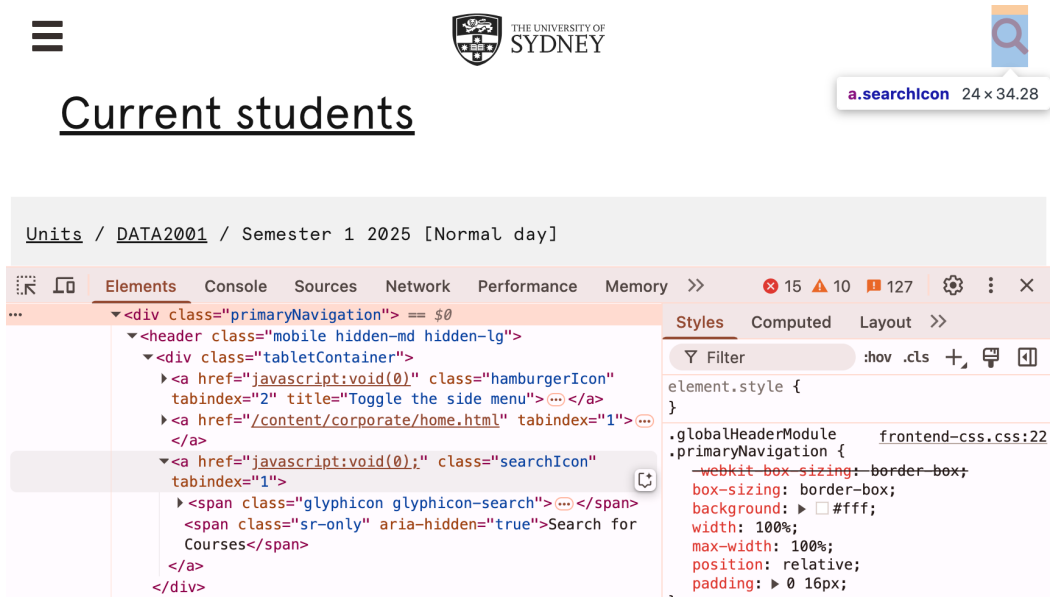
***Hovering** over an element in the HTML code will reveal where it exists on the webpage.*

*The **CSS styling** of each element can also be viewed when a HTML element is clicked. Here for example, we see an element with an upper margin of 14 pixels.*

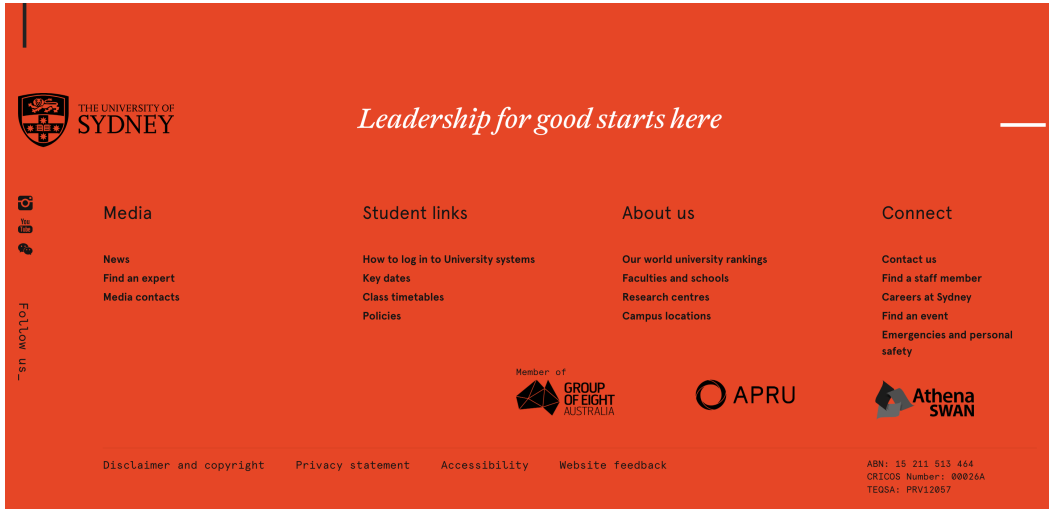
Webpage Parsing

Jump to the Jupyter Notebook for this week and begin exploring the webpage.

(return and continue with the slides after completing Section 1)



Guiding screenshot for the demonstration in Section 1.3
Can we extract some attributes of the header links?



Guiding screenshot for the task in Section 1.3
Can we extract the text and links for the social media platforms?

Thought Questions

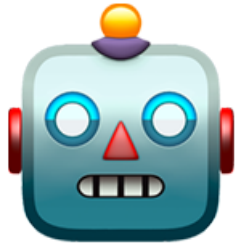
What applications/benefits could web scraping have?

What challenges are faced when attempting to web scrape?

How do we determine what is legal, and remain a good internet citizen?

Robots.txt

The location for websites to specify what can and can't be scraped is **robots.txt**.
For any given web domain, simply put '/robots.txt' on the end. Below is Sydney University's:



```
User-agent: FunnelBack
Disallow: /education_social_work/bulletin/

User-agent: *
Allow: /
Sitemap:https://www.sydney.edu.au/sitemap.xml
Allow: /muni-content/
Allow: /medicine-health/schools/sydney-school-of-health-sciences/academic-staff/
Allow: /science/about/our-people/academic-staff/
```

User-agent: the group the rules apply to (e.g. 'AdsBot-Google' may be subject to different terms of use, '*' indicates all other users)

It will generally then detail what is allowed/disallowed. Here, the root directory ('/') is listed as "**allow**", so our use here is permissible.

Some may even specify a "**crawl-delay**", which specifies the minimum time that must be left in between requests. Even if this is not specified, **always be sure to add delays** between requests!

```
Disallow: /library/images/
Disallow: /library/scripts/
Disallow: /library/styles/
Disallow: /library/test/
Disallow: /library/templates/
Disallow: /library/stream/
Disallow: /library/screens/
Disallow: /library/cgi-bin/
Disallow: /library/unified-search/
Disallow: /library/contacts/email-campaigns/

Disallow: /styleguide/
Disallow: /agents/

Disallow: /errors/
Disallow: /architecture/about/our-people/academic-staff/staff-profile.html
Disallow: /law/about/our-people/academic-staff/staff-profile.html
Disallow: /music/about/our-people/academic-staff/staff-profile.html
Disallow: /engineering/about/our-people/academic-staff/staff-profile.html
Disallow: /medicine-health/about/our-people/academic-staff/staff-profile.html
Disallow: /medicine-health/schools/faculty-of-health-sciences/academic-staff/staff-profile.html
Disallow: /arts/about/our-people/academic-staff/staff-profile.html
```

There will also often be sites listed as "**disallow**" that should not be visited programmatically. For USYD, this entails the pages of academic staff profiles, for example.

Tasks

Summarising the tasks remaining, to be completed in Jupyter Notebook

1.4) Extract the details of all **assessments** in the webpage.

2.2) Create a **function** that receives a URL + returns assessment data.

3.3) After ingestion, develop an SQL **query** that reports the first session, last session, and avg weight of each assessment type.

4) BONUS OPTIONAL TASK*: Extract the list of **all OLEs**, and for each, extract the assessments for their most recent UoS outline. Load this into a database and determine, according to your own metrics, the 'ideal' OLE.

*Pursue this final task with caution, and follow the prompts provided in the notebook.

Done! Enjoy your week :)