Welcome to Web Scraping with Python!

Please log onto the Etherpad:
https://pad.carpentries.org/2026-03-18-bham-web-scraping-Python

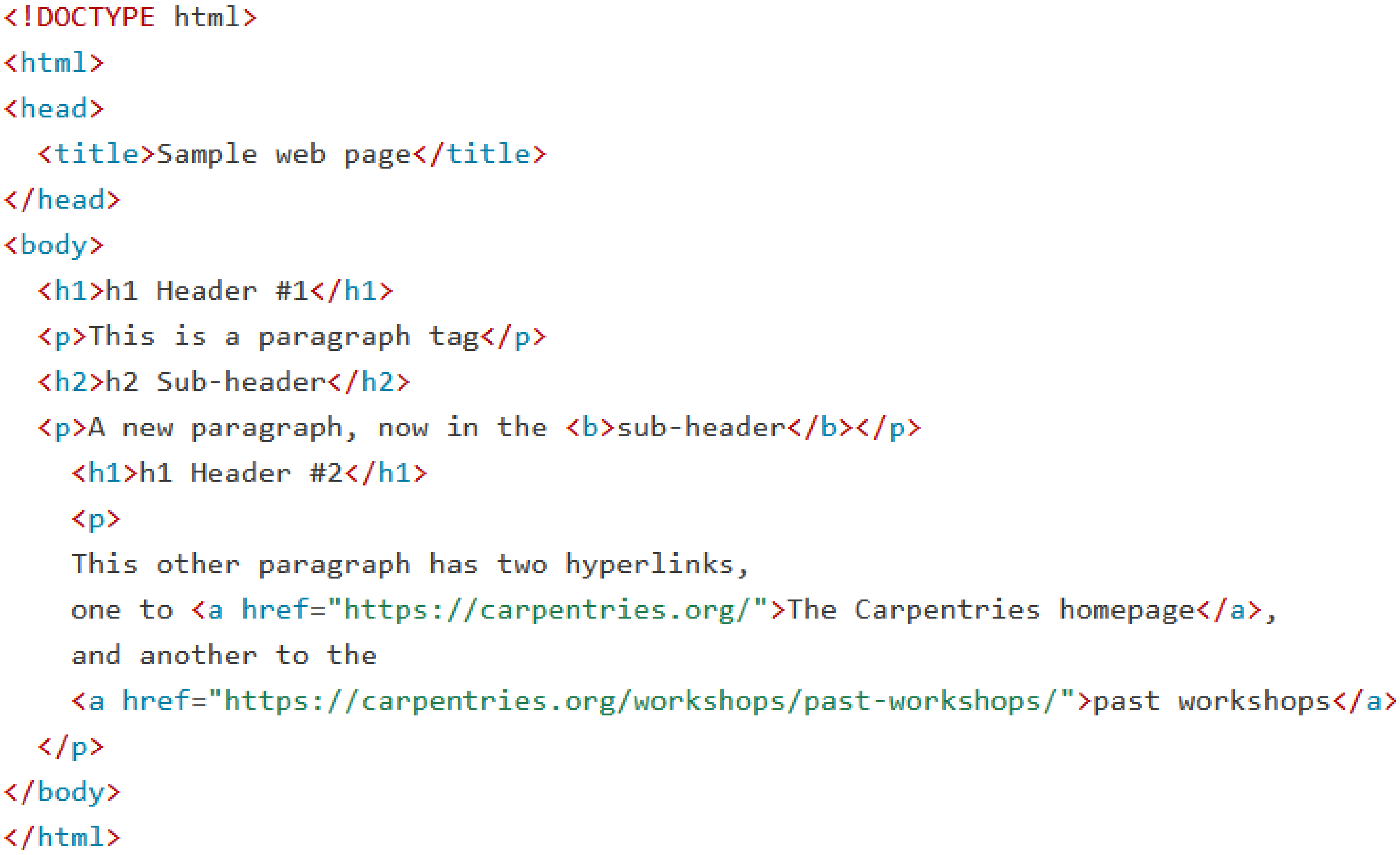
Add your details in the 'Sign in' section

Check if you have completed the Software Setup'

“Web scraping (web harvesting or web data extraction) is a computer software technique of extracting information from websites.”
Wikipedia https://en.wikipedia.org/wiki/Web_scraping

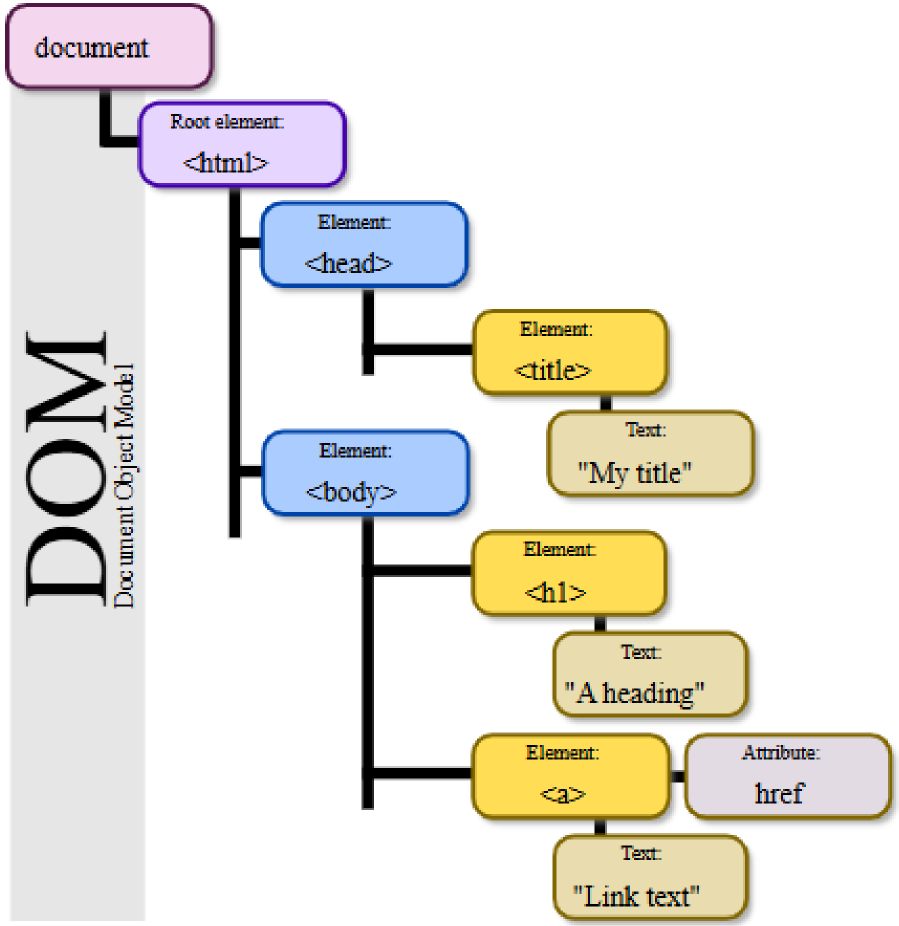
Web scraping typically targets one web site at a time to extract unstructured information and put it in a structured form for reuse.

Hypertext Markup Language – html



```
<!DOCTYPE html>
<html>
<head>
  <title>Sample web page</title>
</head>
<body>
  <h1>h1 Header #1</h1>
  <p>This is a paragraph tag</p>
  <h2>h2 Sub-header</h2>
  <p>A new paragraph, now in the <b>sub-header</b></p>
    <h1>h1 Header #2</h1>
    <p>
    This other paragraph has two hyperlinks,
    one to <a href="https://carpentries.org/">The Carpentries homepage</a>,
    and another to the
    <a href="https://carpentries.org/workshops/past-workshops/">past workshops</a>
  </p>
</body>
</html>
```

html Tree Format



Source: Wikipedia. Author: Birger Eriksson

html tags

- **<hmtl>...</html>** The root element that contains the entire document.
- **<head>...</head>** Contains metadata such as the page title that the browser displays.
- **<body>...</body>** Contains the content that will be shown on the webpage.
- **<h1>...</h1>, <h2>...</h2>** Define headers of levels 1, 2, 3, and so on.
- **<p>...</p>** Represents a paragraph.
- **<a href="">...</a>** Creates a hyperlink; the destination URL is set with the **href** attribute.
- **<img src="" alt="">** Embeds an image, with the image source specified by `src` and alternative text provided by `alt`. It doesn't have an opening tag.
- **<table>...</table>, <th>...</th>, <tr>...</tr>, <td>...</td>** Define a table structure, with headers (`<th>`), rows (`<tr>`), and cells (`<td>`).
- **<div>...</div>** Groups sections of HTML content together.
- **<script>...</script>** Embeds or links to JavaScript code.

Selenium ‘find’ instructions

.find_element() - finds first matching element

- To select first <table> element:

```
driver.find_element(by=By.TAG_NAME, value=“table”)
```

- To find a row with <tr class=“film”>:

```
driver.find_element(by=By.CLASS_NAME, value=“film”)
```

.find_elements() – finds all matching elements

The Scraping Pipeline

1. Understand the structure of the website

2. Determine whether the content is static or dynamic
   - Static content can be accessed directly using the ‘requests’ library and parsed using BeautifulSoup
   - Dynamic content is loaded or updated by JavaScript and typically requires Selenium to render the page before parsing

3. Build your scraping pipeline

4. Clean, format and store the data in a structured format. Pandas DataFrames may be used