CIS 1100

Scraping

Python

Fall 2024

University of Pennsylvania

Disclaimer

This is a module that deals with advanced topics in a cursory manner. Adjust your expectations correspondingly.

- Perfect understanding? ❌
- Neat & practical techniques? ✅

Scraping

Web Scraping is the process of:

1. traversing the internet to find web pages that contain interesting information
2. extracting that information from each web page
3. storing the extracted information in a useful format

A Scraper's Guide to the Internet

The **internet** is a set of interconnected data servers (other computers).

To browse the internet, you ask your computer to connect to another computer—this is called a **request.**

Requests are answered with **responses** that contain:

- the data you asked for, or
- an explanation for why you're not getting the data you asked for

CIS 1100

HTML

Python

Fall 2024

University of Pennsylvania

A Scraper's Guide to Responses

The response's "data that you asked for" can come in any shape.
But for a typical user, it comes in the form of **HTML** for a web page.

HTML, or *hypertext markup language*, is a system of arranging the contents of a website. It can include:

- text!
- tables!
- links!
- images!
- groups!
- code!

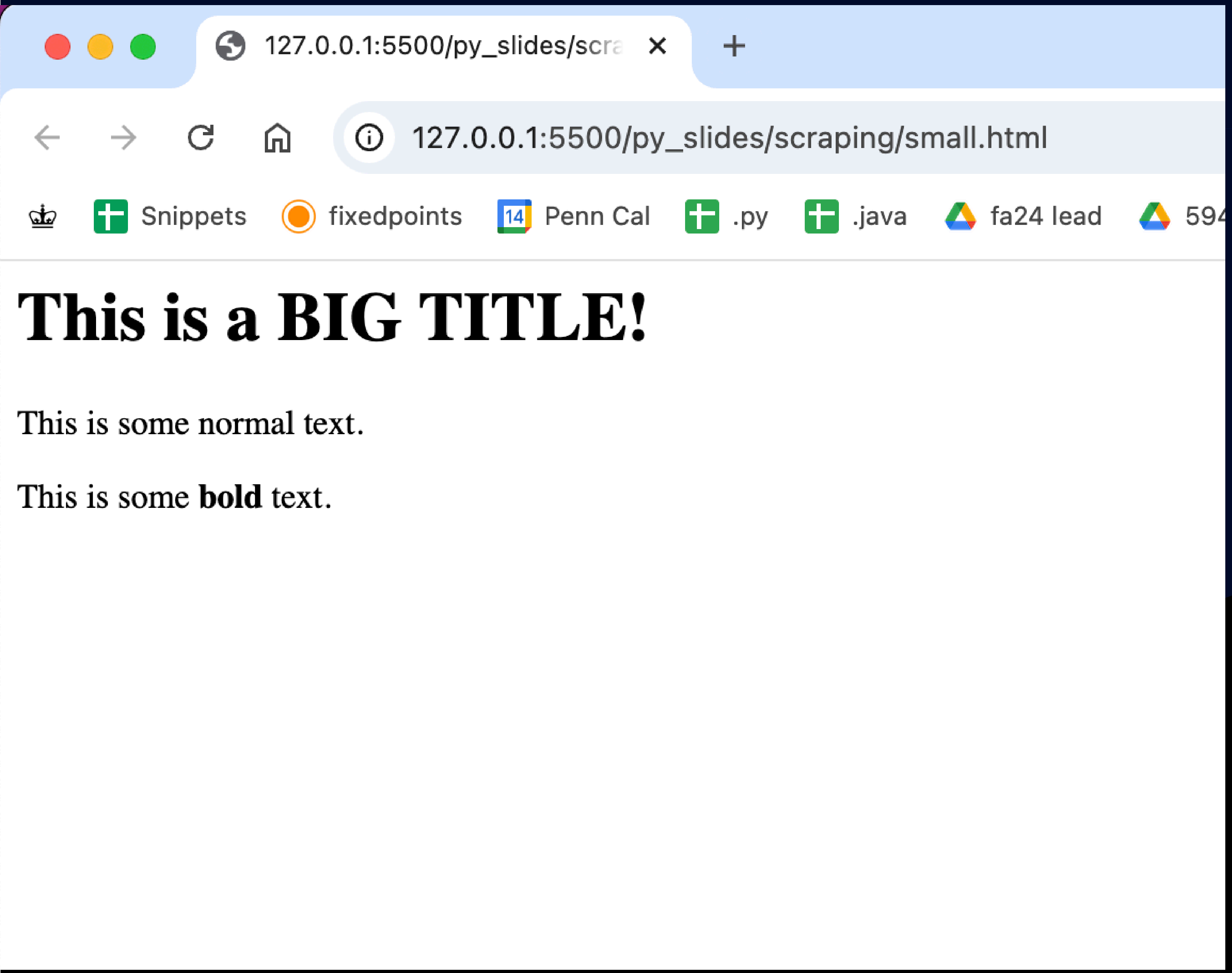
The Very Very Very Basics of HTML

HTML is a language based on **tags**, which convey instructions about how the information inside of them should be handles & displayed.

- Tags contain data, including text and other tags
- Tags that contain data are *opened and closed*
- Tags can have **attributes**, which are key-value pairs that describe some feature of this tag

```html
<h1>This is a BIG TITLE!</h1>

<p>This is some normal text.</p>
<p>This is some <strong>bold</strong> text.</p>
```

HTML Describes a Web Page


127.0.0.1:5500/py_slides/scra
127.0.0.1:5500/py_slides/scraping/small.html
Snippets
fixedpoints
14
Penn Cal
.py
.java
fa24 lead
594

This is a BIG TITLE!

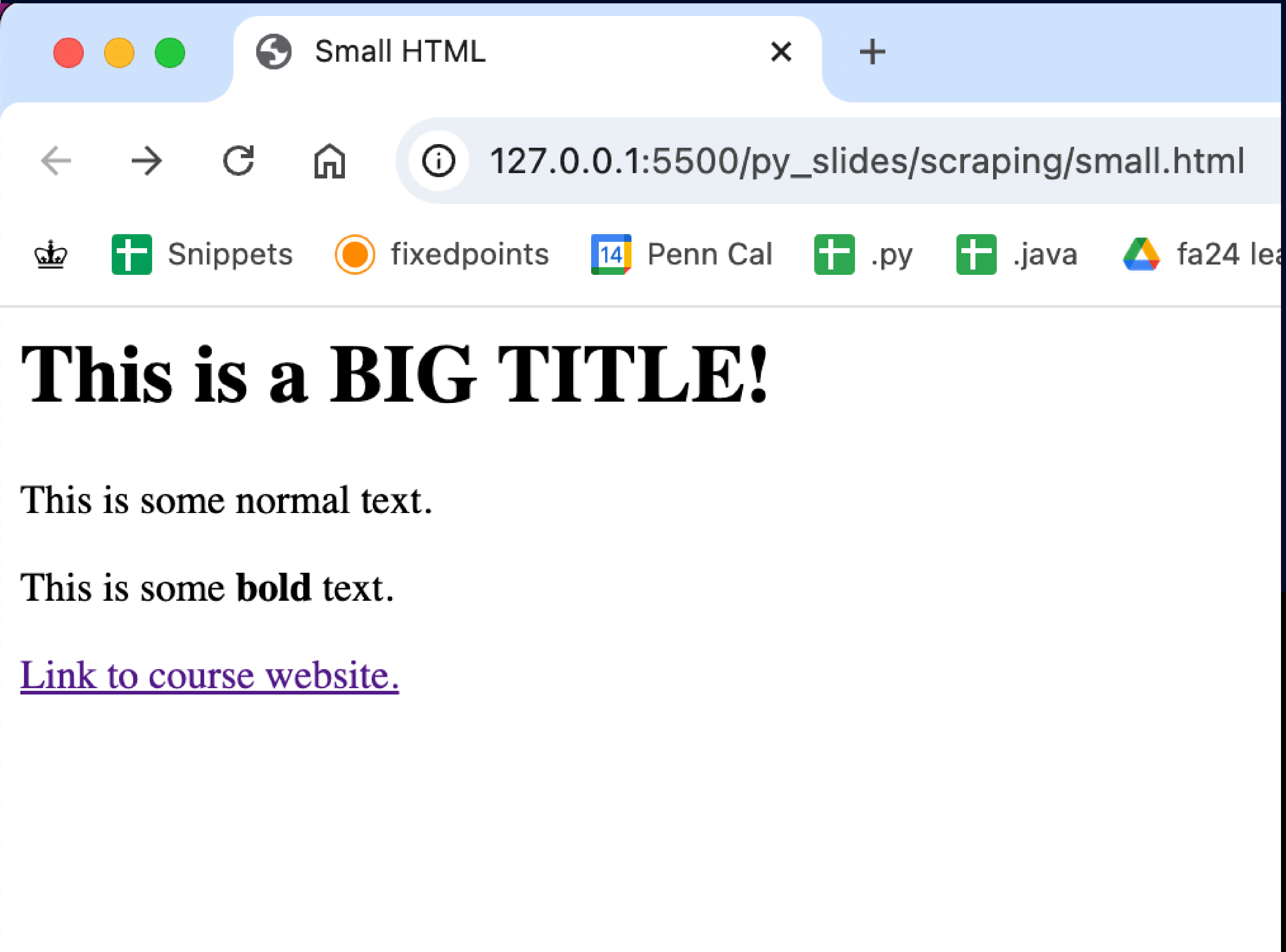
This is some normal text.

This is some **bold** text.

`small.html`:

```
<h1>This is a BIG TITLE!</h1>

<p>This is some normal text.</p>
<p>This is some <strong>bold</strong> text.</p>
```

HTML Describes a Web Page



This is a BIG TITLE!

This is some normal text.

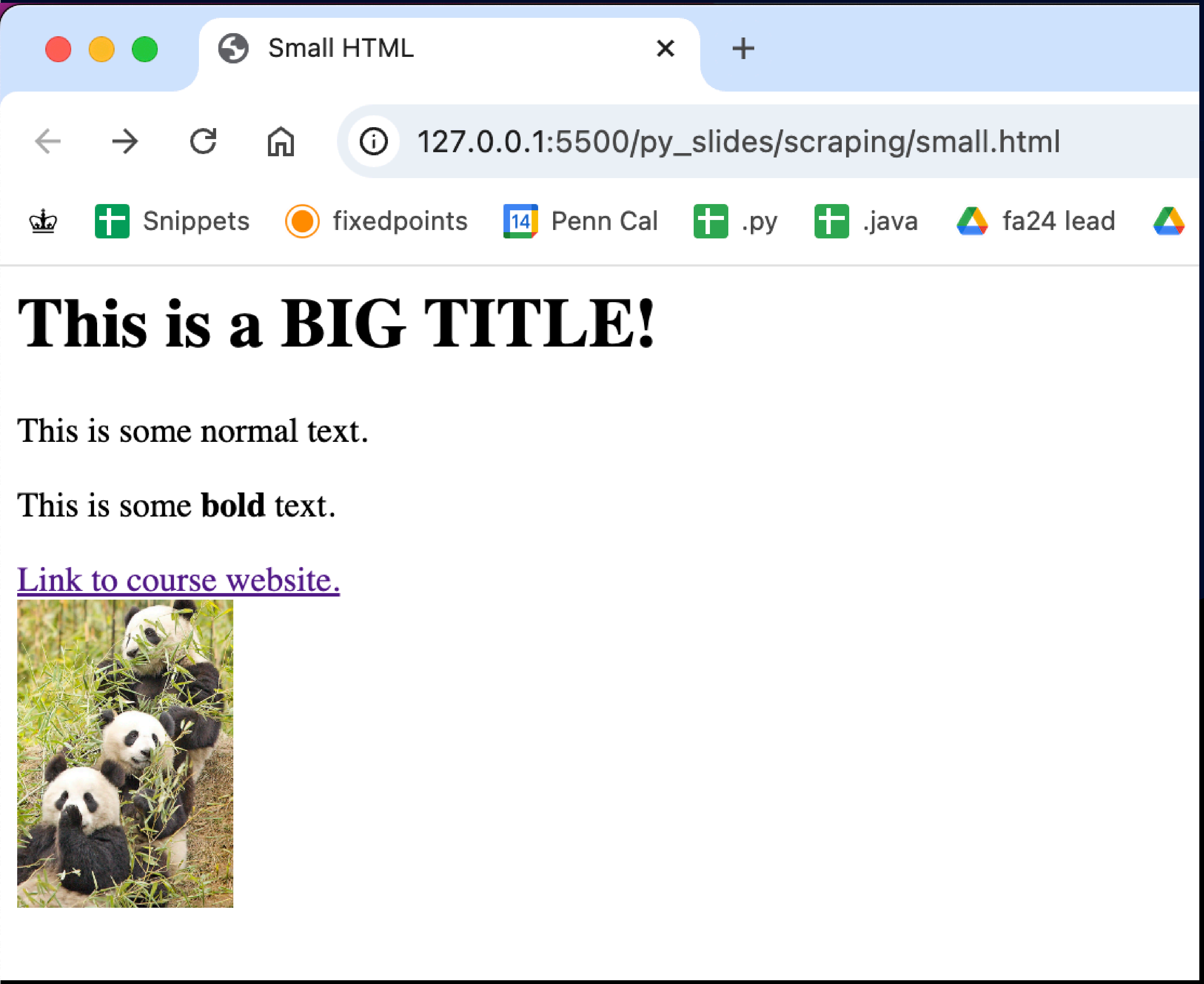
This is some **bold** text.

Link to course website.

`small.html`:

```
<h1>This is a BIG TITLE!</h1>

<p>This is some normal text.</p>
<p>This is some <strong>bold</strong> text.</p>

<a href="https://cis1100.com">Link to course website.</a>
```

HTML Describes a Web Page



This is a BIG TITLE!

This is some normal text.

This is some **bold** text.

Link to course website.

`small.html`:

```
<h1>This is a BIG TITLE!</h1>

<p>This is some normal text.</p>
<p>This is some <strong>bold</strong> text.</p>

<a href="https://cis1100.com">Link to course website.</a>
<br />
<img src="image-59.png" width="100px" />
```

Basic HTML Tag Summary

Tag Name	Purpose	Attributes
`h1`	Big header for titles	
`h2`, `h3`, `h4`	Slightly smaller headers for subtitles	
`p`	Basic paragraph text	
`a`	Links	`href="link-to-thing.com"`
`br`	Line Break	
`img`	Image	`src="picture.png"`, optional things like `width` or `height`

Classes: Categories for Tags

HTML tags can belong to categories called **classes**.

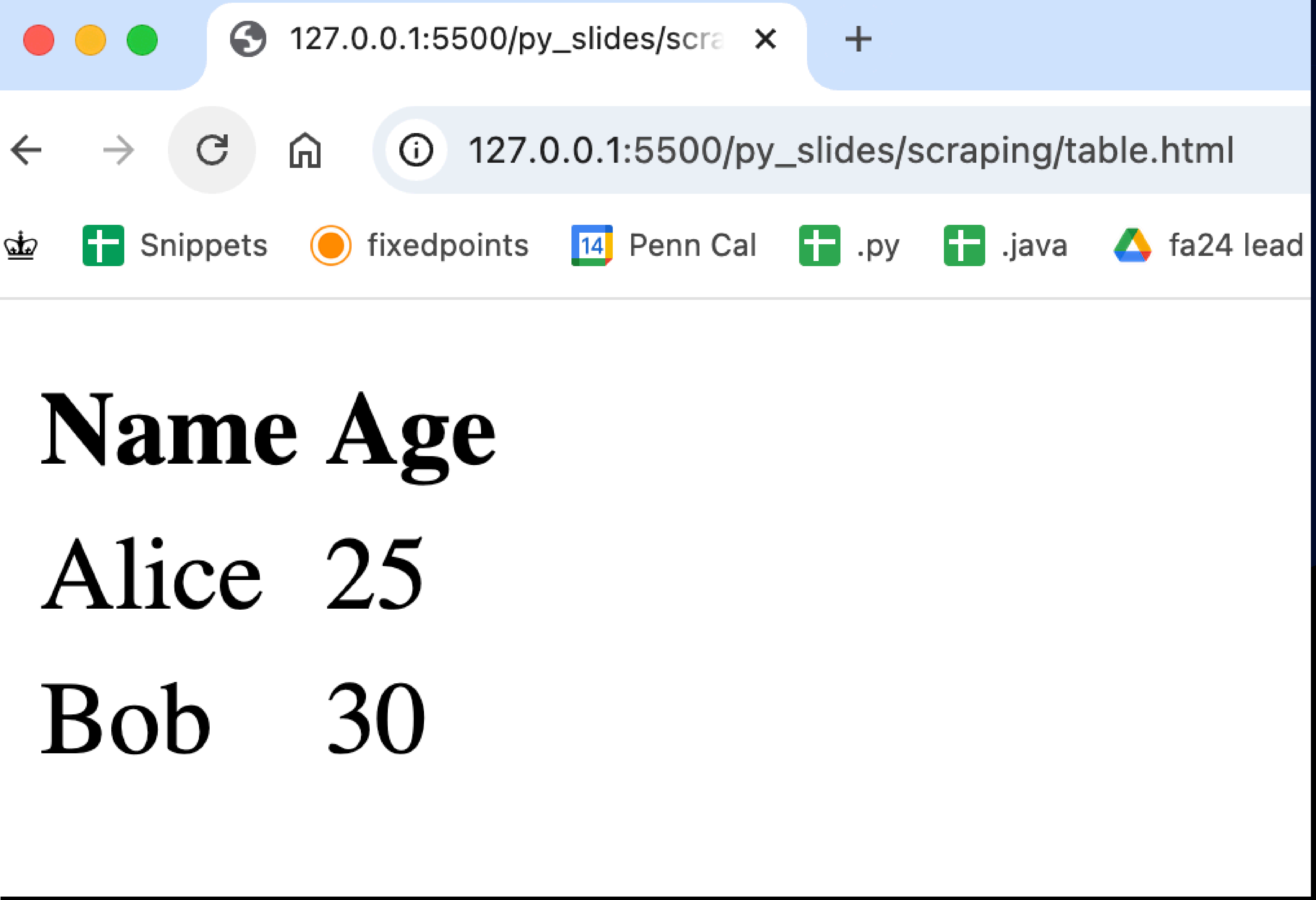
- Classes are usually used for styling purposes
- Help differentiate between tags of the same type that have different meanings on a page
- classes are just attributes:

```html
<p class="fancy">This is fancy text...</p>
<p class="normal">This is normal text...</p>
```

Other Structural Tags

- `div` tags
  - don't have any visible structure of their own by default
  - represent a "section" of the page
  - used to apply organization or style rules to all other tags they contain
- `table` tags represent tables
  - tables consist of rows
    - rows are represented using `tr` tags
    - rows consist of cells
      - header cells are represented with `th` tags
      - data dells are represented with `td` tags

Basics of a Table


127.0.0.1:5500/py_slides/scra
127.0.0.1:5500/py_slides/scraping/table.html
Snippets
fixedpoints
Penn Cal
.py
.java
fa24 lead

Name	Age
Alice	25
Bob	30

```
<table>
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Alice</td>
    <td>25</td>
  </tr>
  <tr>
    <td>Bob</td>
    <td>30</td>
  </tr>
</table>
```

CIS 1100

HTML to Data

Python

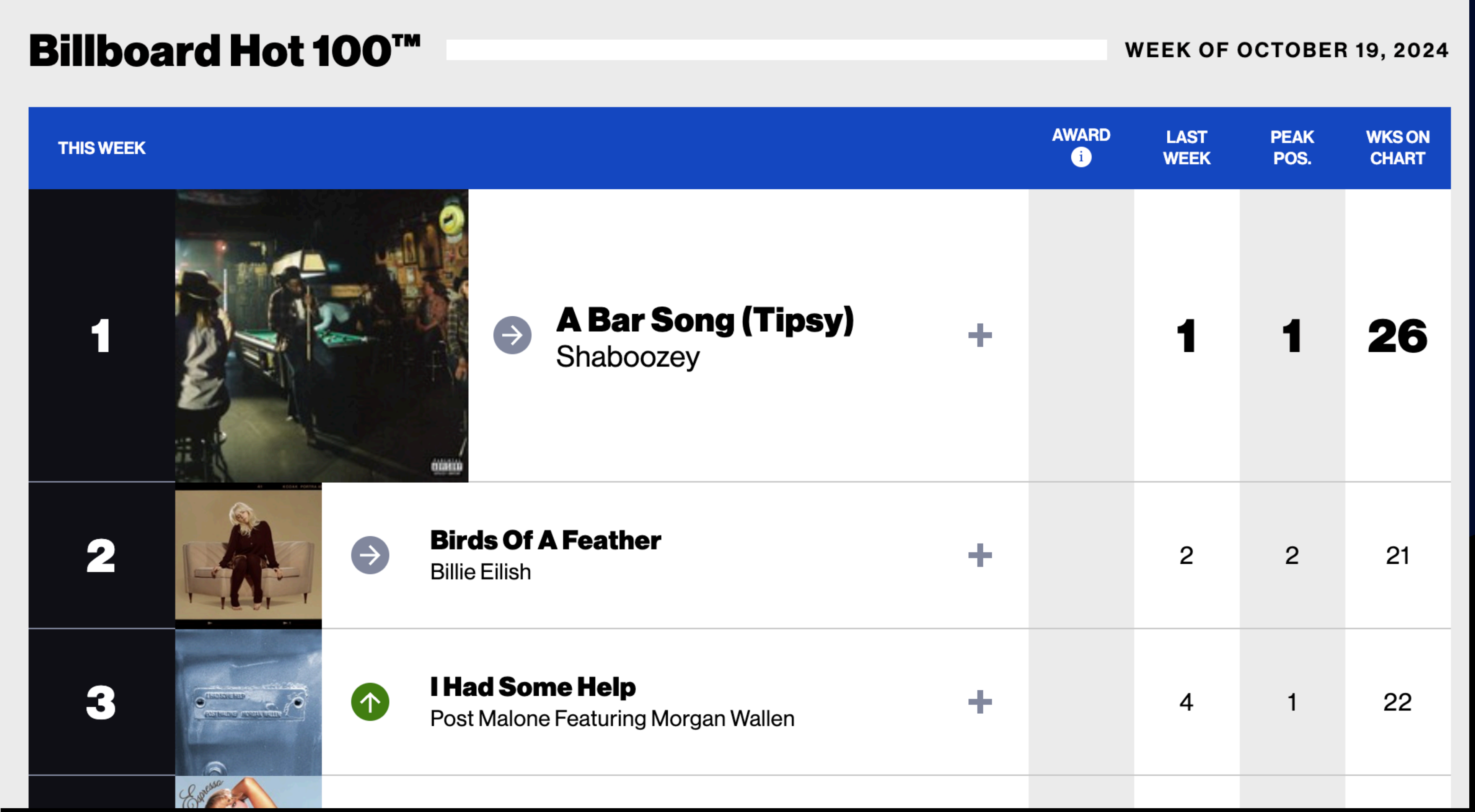
Fall 2024

University of Pennsylvania

Credit to Jonathan Soma

Adapted from his excellent guide on scraping with Python.

Scraping Popular Songs



Billboard Hot 100™ WEEK OF OCTOBER 19, 2024

THIS WEEK		AWARD	LAST WEEK	PEAK POS.	WKS ON CHART
1	**A Bar Song (Tipsy)** Shaboozey		1	1	26
2	**Birds Of A Feather** Billie Eilish		2	2	21
3	**I Had Some Help** Post Malone Featuring Morgan Wallen		4	1	22

Suppose you want to keep track of the most popular songs week after week. You can find this information out from billboard.com.

Scoping Out the Data

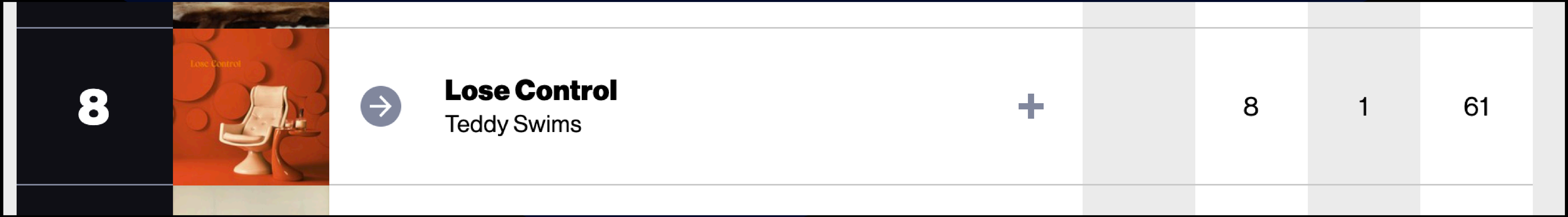
Before we can get the data off the page, we need to understand our problem.

We want to build a structured dataset, but **what data are we trying to get?**

1. Inspect the page to figure out what one entry in your dataset would look like.
2. Use the browser's web inspector to find what the HTML looks like for each entry.
3. Find the common tags/classes used for each entry and write this down.

Find Your Entries

If we want to track weekly performance of songs on the Billboard Hot 100™, then we would want at least the title,artist(s), & current position



8

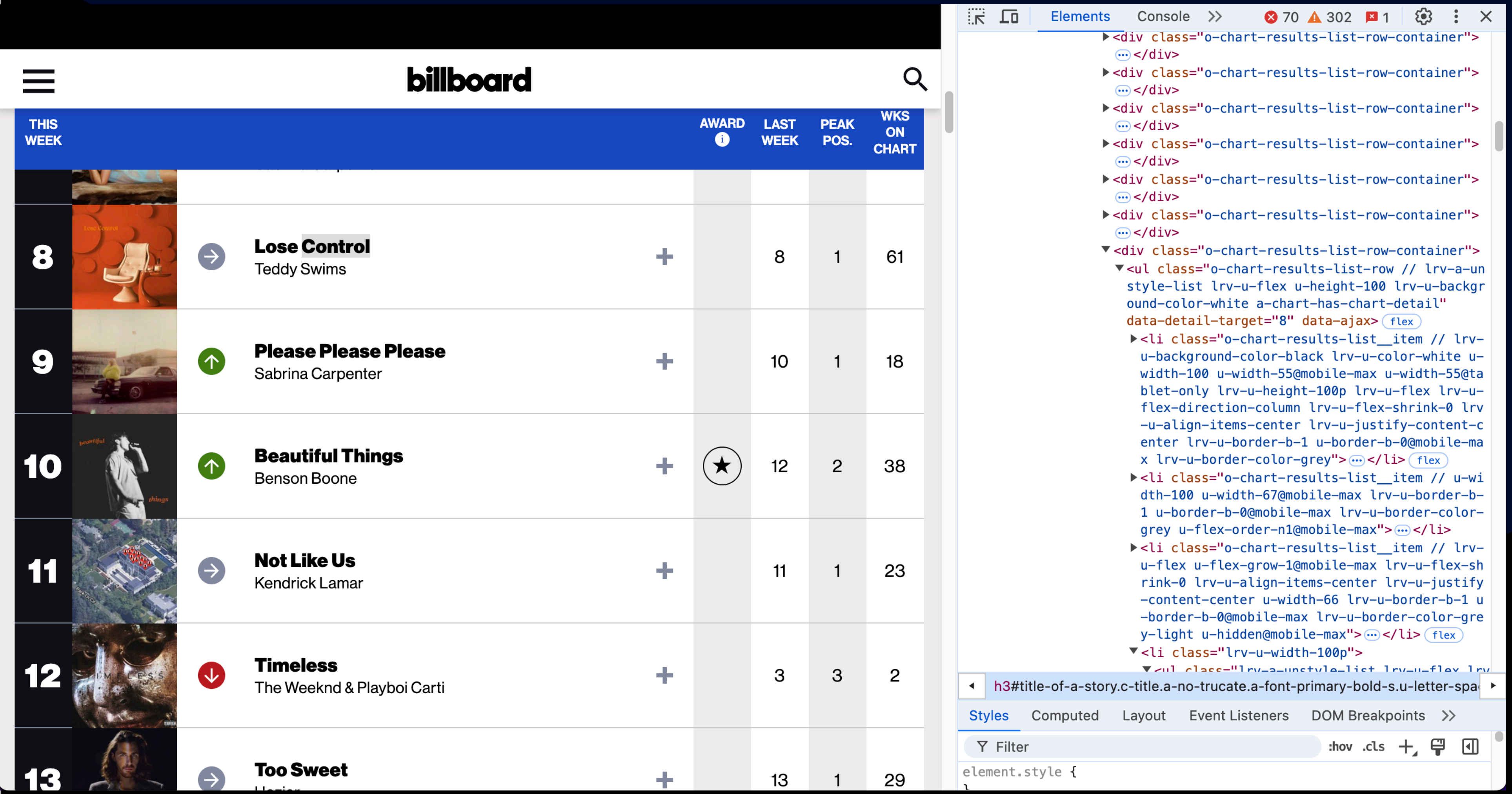
Lose Control
Teddy Swims

+

8 1 61

Each row of the website's table seems to have all this information and more.

Identify the Encapsulating HTML



THIS WEEK		AWARD	LAST WEEK	PEAK POS.	WKS ON CHART
8	**Lose Control** Teddy Swims		8	1	61
9	**Please Please Please** Sabrina Carpenter		10	1	18
10	**Beautiful Things** Benson Boone	★	12	2	38
11	**Not Like Us** Kendrick Lamar		11	1	23
12	**Timeless** The Weeknd & Playboi Carti		3	3	2
13	**Too Sweet** Hozier		13	1	29

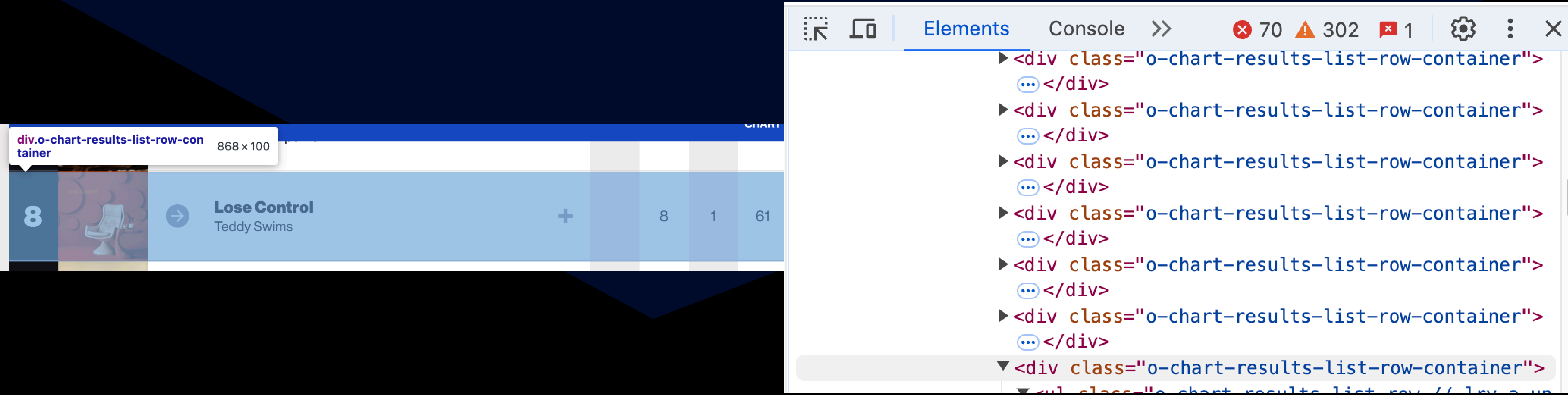
```
▶<div class="o-chart-results-list-row-container">
 ⋯ </div>
▶<div class="o-chart-results-list-row-container">
 ⋯ </div>
▶<div class="o-chart-results-list-row-container">
 ⋯ </div>
▶<div class="o-chart-results-list-row-container">
 ⋯ </div>
▶<div class="o-chart-results-list-row-container">
 ⋯ </div>
▶<div class="o-chart-results-list-row-container">
 ⋯ </div>
▼<div class="o-chart-results-list-row-container">
  ▼<ul class="o-chart-results-list-row // lrv-a-un
   style-list lrv-u-flex u-height-100 lrv-u-backgr
   ound-color-white a-chart-has-chart-detail"
   data-detail-target="8" data-ajax> flex
    ▶<li class="o-chart-results-list__item // lrv-
     u-background-color-black lrv-u-color-white u-
     width-100 u-width-55@mobile-max u-width-55@ta
     blet-only lrv-u-height-100p lrv-u-flex lrv-u-
     flex-direction-column lrv-u-flex-shrink-0 lrv
     -u-align-items-center lrv-u-justify-content-c
     enter lrv-u-border-b-1 u-border-b-0@mobile-ma
     x lrv-u-border-color-grey"> ⋯ </li> flex
    ▶<li class="o-chart-results-list__item // u-wi
     dth-100 u-width-67@mobile-max lrv-u-border-b-
     1 u-border-b-0@mobile-max lrv-u-border-color-
     grey u-flex-order-n1@mobile-max"> ⋯ </li>
    ▶<li class="o-chart-results-list__item // lrv-
     u-flex u-flex-grow-1@mobile-max lrv-u-flex-sh
     rink-0 lrv-u-align-items-center lrv-u-justify
     -content-center u-width-66 lrv-u-border-b-1 u
     -border-b-0@mobile-max lrv-u-border-color-gre
     y-light u-hidden@mobile-max"> ⋯ </li> flex
    ▼<li class="lrv-u-width-100p">
```

```
h3#title-of-a-story.c-title.a-no-trucate.a-font-primary-bold-s.u-letter-spa
```

```
element.style {
```

Use your favorite browser's **inspector** to look at the HTML underlying the page.

- Right click --> inspect element
- pressing F12 usually works too



```
▶<div class="o-chart-results-list-row-container">
 ••• </div>
▶<div class="o-chart-results-list-row-container">
 ••• </div>
▶<div class="o-chart-results-list-row-container">
 ••• </div>
▶<div class="o-chart-results-list-row-container">
 ••• </div>
▶<div class="o-chart-results-list-row-container">
 ••• </div>
▶<div class="o-chart-results-list-row-container">
 ••• </div>
▼<div class="o-chart-results-list-row-container">
  ▼<ul class="o-chart-results-list-row // lrv-a-un
```

Find the Common Tag

Looks like each table row is stored in a `div` with the class `o-chart-results-list-row-container`

- Even more specifically, that `div` stores a `ul` with the class `o-chart-results-list-row`
- Either will work—just want a tag that contains all of the important information for each entry in your data set. You can narrow down with Python later.

Demo!

CIS 1100

BeautifulSoup

Python

Fall 2024

University of Pennsylvania

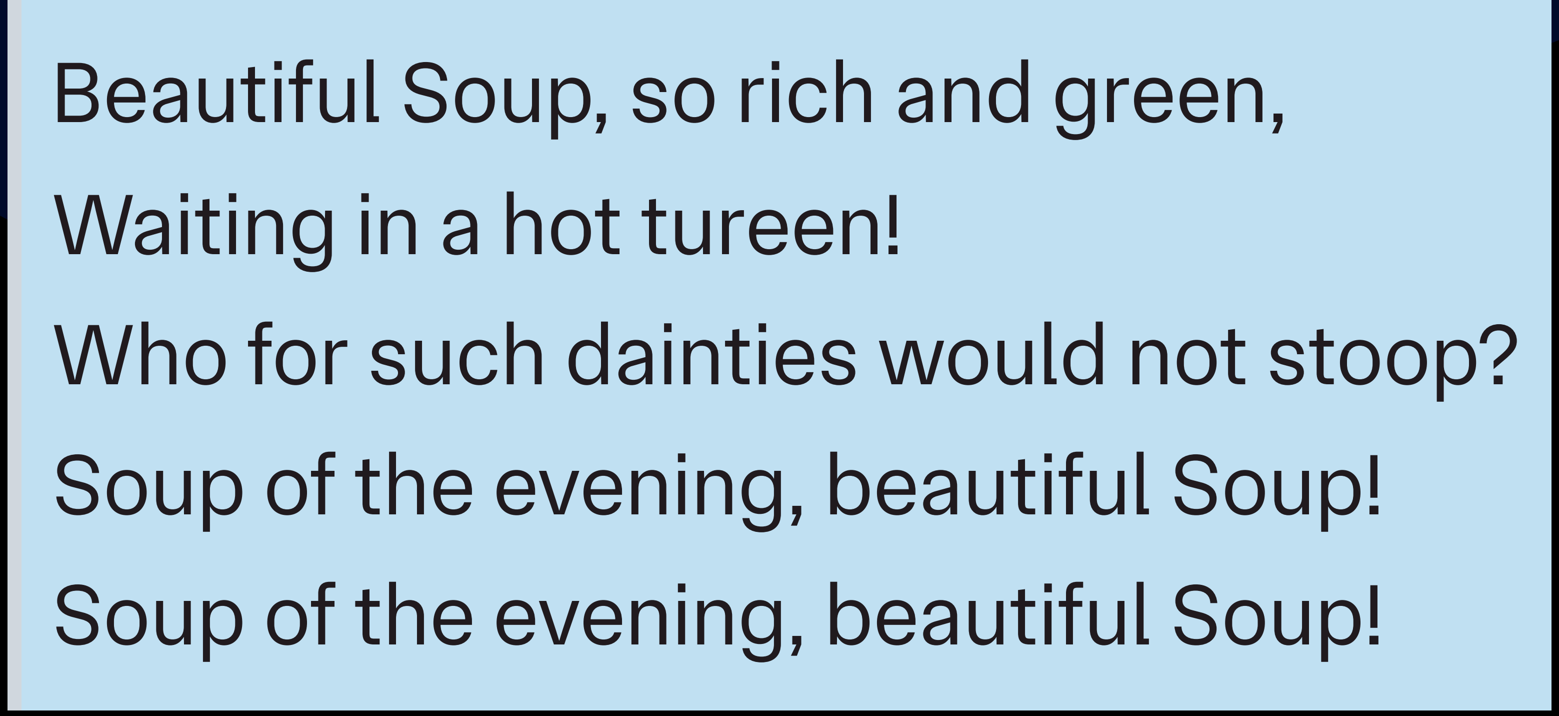
Parsing through HTML

Now that we know how to identify our entities of interest in the HTML, how do we write code that pulls it out of the HTML for us?

The answer: **BeautifulSoup**

BeautifulSoup

- Python library used to parse, traverse, and search HTML
- Load the HTML into a Python object, then use methods & attributes to find tags and their matching data.



Beautiful Soup, so rich and green,
Waiting in a hot tureen!
Who for such dainties would not stoop?
Soup of the evening, beautiful Soup!
Soup of the evening, beautiful Soup!

Parsing HTML

This example assumes that you have downloaded webpage somehow into a file called `index.html`.

```
from bs4 import BeautifulSoup
html_file = open('index.html', 'r')
html_doc = html_file.read()
soup = BeautifulSoup(html_doc, 'html.parser')
```

Example: Getting Info from a Tag

Copied from official documentation.

`index.html`:

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.title ➡️ "<title>The Dormouse's story</title>"
```

Example: Getting Info from a Tag

`.name` gives the type of tag you have

`index.html`:

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.title.name ➡ "title"
```

Example: Getting Info from a Tag

`.string` gives the text inside of the tag you have

`index.html`:

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.title.string ➡ "The Dormouse's story"
```

Example: Traversing through HTML

`.parent` refers to the tag this one is contained inside of

`index.html`:

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.title.parent.name ➡ "head"
```

Example: Traversing through HTML

`.tag_name` always gives the *first matching tag.*

`index.html`:

```html
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.p.string ➡️ "The Dormouse's story"
```

Example: Reading Tag Attributes

Tags can be treated like dictionaries where the attribute names are the keys.

`index.html`:

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.p["class"] ➡ "title"
```

Example: Getting All Matching Tags

`.find_all("tag_name")` finds all tags with a matching name.

`index.html`:

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.find_all('a') ➡
['<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>',
 '<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>',
 '<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>']
```

Example: Getting All Matching Tags

`.find_all("tag_name", class_="c_name")`

finds all tags with a matching name and class.

`index.html`:

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
```

```
soup.find_all('p', class_='title') ➡
["<p class='title'><b>The Dormouse's story</b></p>"]
```

CIS 1100

Sifting Through Soup

Python

Fall 2024

University of Pennsylvania

Returning to Popular Music...

Prerequisites:

- Found the tag name & class that contains one "future row" of our dataset
  - `ul` with the class `o-chart-results-list-row`
- Saved the HTML of the webpage to a file (manually in browser, or by `requests`)

Starting out

```python
from bs4 import BeautifulSoup

html_file = open('index.html', 'r')
html_doc = html_file.read()
soup = BeautifulSoup(html_doc, 'html.parser')

rows = soup.find_all('ul', class_='o-chart-results-list-row')
print(len(rows))
```

🖨️ 🔽

```
100
```

✅

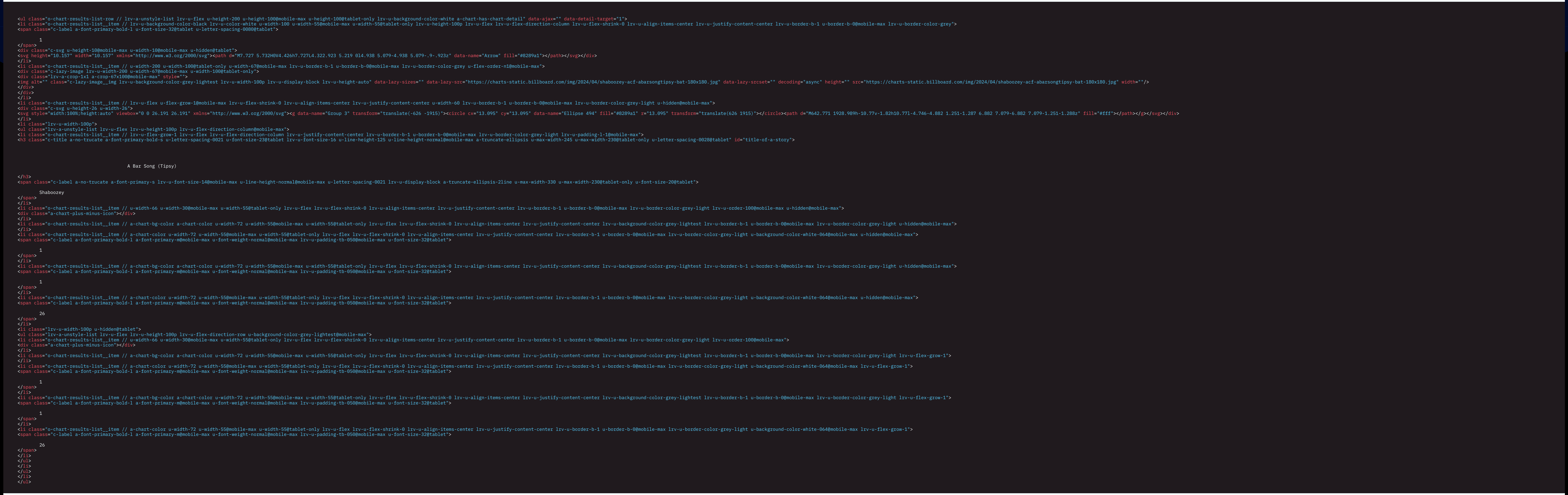
Starting out

```
from bs4 import BeautifulSoup

html_file = open('index.html', 'r')
html_doc = html_file.read()
soup = BeautifulSoup(html_doc, 'html.parser')

rows = soup.find_all('ul', class_='o-chart-results-list-row')
print(rows[0])
```

🖨 ▼

…

Uhhh.....



```
<ul class="o-chart-results-list-row // lrv-a-unstyle-list lrv-u-flex u-height-200 u-height-100@mobile-max u-height-100@tablet-only lrv-u-background-color-white a-chart-has-chart-detail" data-ajax="" data-detail-target="1">
<li class="o-chart-results-list__item // lrv-u-background-color-black lrv-u-color-white u-width-100 u-width-55@mobile-max u-width-55@tablet-only lrv-u-height-100p lrv-u-flex lrv-u-flex-direction-column lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey">
<span class="c-label a-font-primary-bold-l u-font-size-32@tablet u-letter-spacing-0080@tablet">

        1
</span>
<div class="c-svg u-height-10@mobile-max u-width-10@mobile-max u-hidden@tablet">
<svg height="10.157" width="10.157" xmlns="http://www.w3.org/2000/svg"><path d="M7.727 5.732H0V4.426h7.727L4.322.923 5.219 0l4.938 5.079-4.938 5.079-.9-.923z" data-name="Arrow" fill="#8289a1"></path></svg></div>
</li>
<li class="o-chart-results-list__item // u-width-200 u-width-100@tablet-only u-width-67@mobile-max lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey u-flex-order-n1@mobile-max">
<div class="c-lazy-image lrv-u-width-200 u-width-67@mobile-max u-width-100@tablet-only">
<div class="lrv-a-crop-1x1 a-crop-67x100@mobile-max" style="">
<img alt="" class="c-lazy-image__img lrv-u-background-color-grey-lightest lrv-u-width-100p lrv-u-display-block lrv-u-height-auto" data-lazy-sizes="" data-lazy-src="https://charts-static.billboard.com/img/2024/04/shaboozey-acf-abarsongtipsy-bat-180x180.jpg" data-lazy-srcset="" decoding="async" height="" src="https://charts-static.billboard.com/img/2024/04/shaboozey-acf-abarsongtipsy-bat-180x180.jpg" width=""/>
</div>
</div>
</li>
<li class="o-chart-results-list__item // lrv-u-flex u-flex-grow-1@mobile-max lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center u-width-60 lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light u-hidden@mobile-max">
<div class="c-svg u-height-26 u-width-26">
<svg style="width:100%;height:auto" viewbox="0 0 26.191 26.191" xmlns="http://www.w3.org/2000/svg"><g data-name="Group 3" transform="translate(-626 -1915)"><circle cx="13.095" cy="13.095" data-name="Ellipse 494" fill="#8289a1" r="13.095" transform="translate(626 1915)"></circle><path d="M642.771 1928.989h-10.77v-1.82h10.77l-4.746-4.882 1.251-1.287 6.882 7.079-6.882 7.079-1.251-1.288z" fill="#fff"></path></g></svg></div>
</li>
<li class="lrv-u-width-100p">
<ul class="lrv-a-unstyle-list lrv-u-flex lrv-u-height-100p lrv-u-flex-direction-column@mobile-max">
<li class="o-chart-results-list__item // lrv-u-flex-grow-1 lrv-u-flex lrv-u-flex-direction-column lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light lrv-u-padding-l-1@mobile-max">
<h3 class="c-title a-no-trucate a-font-primary-bold-s u-letter-spacing-0021 u-font-size-23@tablet lrv-u-font-size-16 u-line-height-125 u-line-height-normal@mobile-max a-truncate-ellipsis u-max-width-245 u-max-width-230@tablet-only u-letter-spacing-0028@tablet" id="title-of-a-story">



                                        A Bar Song (Tipsy)

</h3>
<span class="c-label a-no-trucate a-font-primary-s lrv-u-font-size-14@mobile-max u-line-height-normal@mobile-max u-letter-spacing-0021 lrv-u-display-block a-truncate-ellipsis-2line u-max-width-330 u-max-width-230@tablet-only u-font-size-20@tablet">

        Shaboozey
</span>
</li>
<li class="o-chart-results-list__item // u-width-66 u-width-30@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light lrv-u-order-100@mobile-max u-hidden@mobile-max">
<div class="a-chart-plus-minus-icon"></div>
</li>
<li class="o-chart-results-list__item // a-chart-bg-color a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-background-color-grey-lightest lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light u-hidden@mobile-max">
</li>
<li class="o-chart-results-list__item // a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light u-background-color-white-064@mobile-max u-hidden@mobile-max">
<span class="c-label a-font-primary-bold-l a-font-primary-m@mobile-max u-font-weight-normal@mobile-max lrv-u-padding-tb-050@mobile-max u-font-size-32@tablet">

        1
</span>
</li>
<li class="o-chart-results-list__item // a-chart-bg-color a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-background-color-grey-lightest lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light u-hidden@mobile-max">
<span class="c-label a-font-primary-bold-l a-font-primary-m@mobile-max u-font-weight-normal@mobile-max lrv-u-padding-tb-050@mobile-max u-font-size-32@tablet">

        1
</span>
</li>
<li class="o-chart-results-list__item // a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light u-background-color-white-064@mobile-max u-hidden@mobile-max">
<span class="c-label a-font-primary-bold-l a-font-primary-m@mobile-max u-font-weight-normal@mobile-max lrv-u-padding-tb-050@mobile-max u-font-size-32@tablet">

        26
</span>
</li>
<li class="lrv-u-width-100p u-hidden@tablet">
<ul class="lrv-a-unstyle-list lrv-u-flex lrv-u-height-100p lrv-u-flex-direction-row u-background-color-grey-lightest@mobile-max">
<li class="o-chart-results-list__item // u-width-66 u-width-30@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light lrv-u-order-100@mobile-max">
<div class="a-chart-plus-minus-icon"></div>
</li>
<li class="o-chart-results-list__item // a-chart-bg-color a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-background-color-grey-lightest lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light lrv-u-flex-grow-1">
</li>
<li class="o-chart-results-list__item // a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light u-background-color-white-064@mobile-max lrv-u-flex-grow-1">
<span class="c-label a-font-primary-bold-l a-font-primary-m@mobile-max u-font-weight-normal@mobile-max lrv-u-padding-tb-050@mobile-max u-font-size-32@tablet">

        1
</span>
</li>
<li class="o-chart-results-list__item // a-chart-bg-color a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-background-color-grey-lightest lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light lrv-u-flex-grow-1">
<span class="c-label a-font-primary-bold-l a-font-primary-m@mobile-max u-font-weight-normal@mobile-max lrv-u-padding-tb-050@mobile-max u-font-size-32@tablet">

        1
</span>
</li>
<li class="o-chart-results-list__item // a-chart-color u-width-72 u-width-55@mobile-max u-width-55@tablet-only lrv-u-flex lrv-u-flex-shrink-0 lrv-u-align-items-center lrv-u-justify-content-center lrv-u-border-b-1 u-border-b-0@mobile-max lrv-u-border-color-grey-light u-background-color-white-064@mobile-max lrv-u-flex-grow-1">
<span class="c-label a-font-primary-bold-l a-font-primary-m@mobile-max u-font-weight-normal@mobile-max lrv-u-padding-tb-050@mobile-max u-font-size-32@tablet">

        26
</span>
</li>
</ul>
</li>
</ul>
</li>
</ul>
```

If we squint...

```
<h3 ...>
A Bar Song (Tipsy)
</h3>
```

and

```
<span ...>
Shaboozey
</span>
```

OK! That's a song name and an artist.

Back to the inspector for more.

CIS 1100

Finding the Columns (DEMO)

Python

Fall 2024

University of Pennsylvania

CIS 1100

Finding the Columns

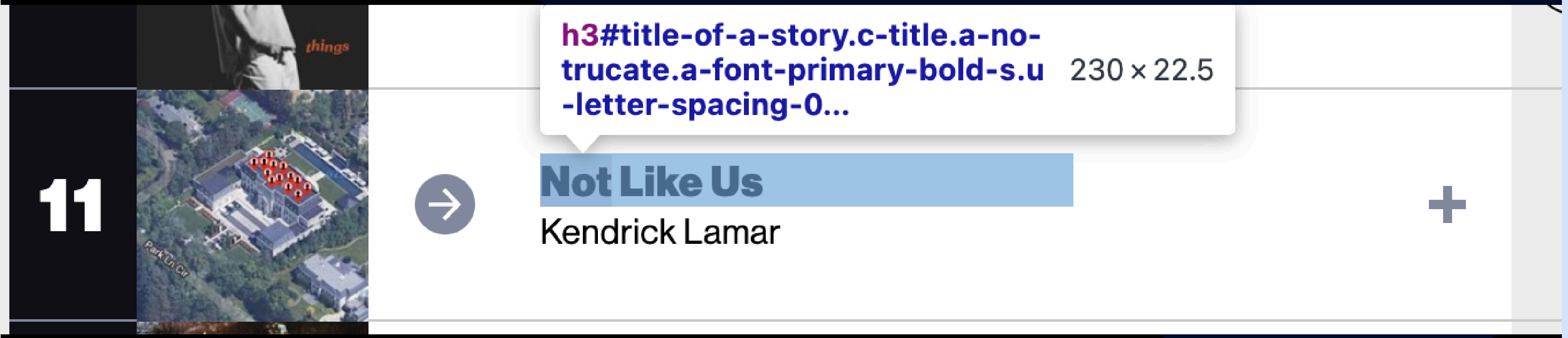
Python

Fall 2024

University of Pennsylvania

Finding the Columns in the Row

Looks like the title is found inside of an `h3` tag with the `id` of `"title-of-a-story"`

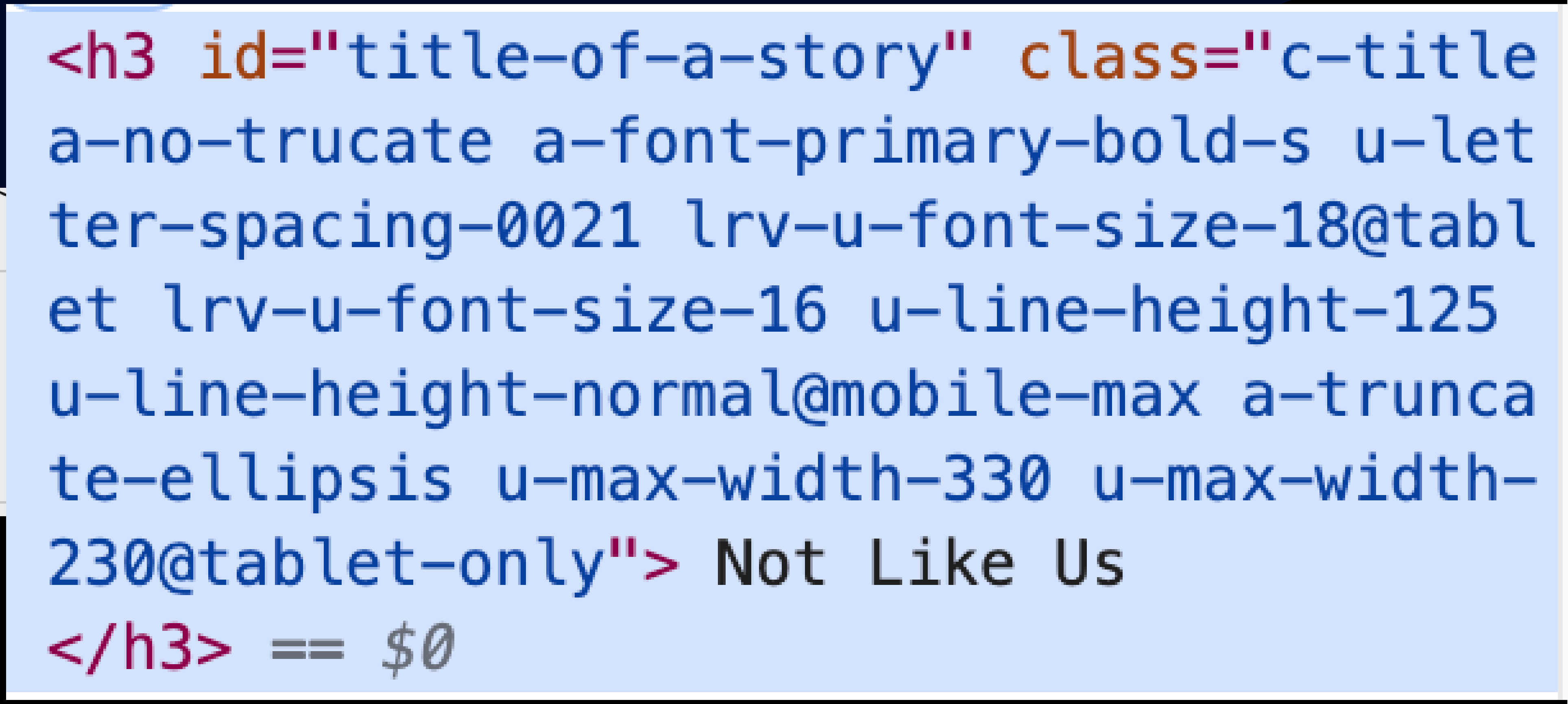


h3#title-of-a-story.c-title.a-no-trucate.a-font-primary-bold-s.u-letter-spacing-0... 230 × 22.5

11

Not Like Us

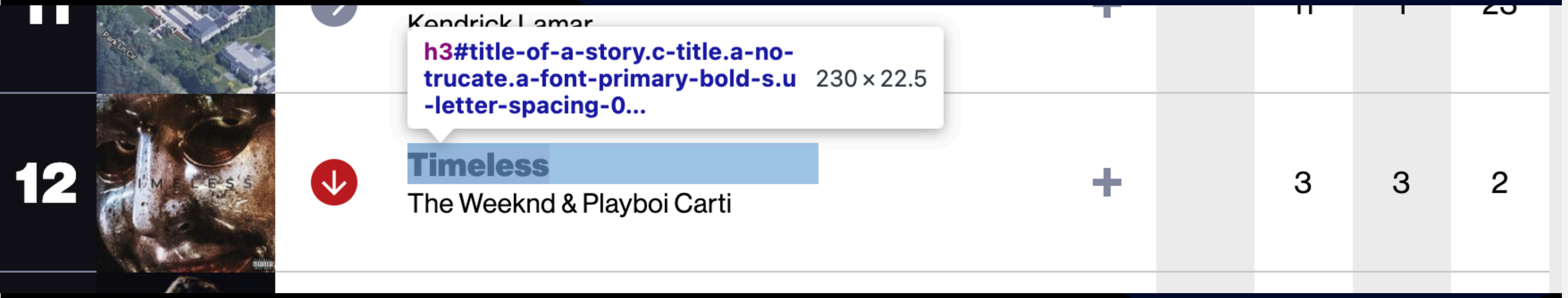
Kendrick Lamar



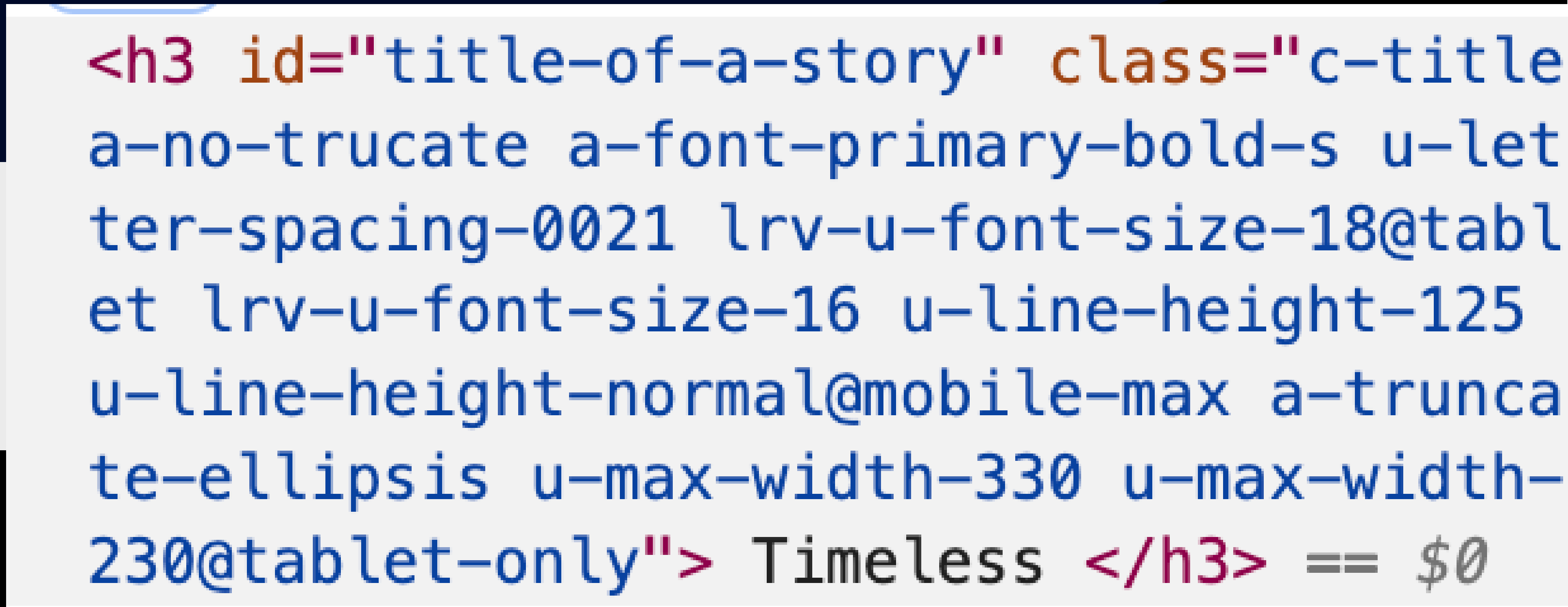
```
<h3 id="title-of-a-story" class="c-title a-no-trucate a-font-primary-bold-s u-letter-spacing-0021 lrv-u-font-size-18@tablet lrv-u-font-size-16 u-line-height-125 u-line-height-normal@mobile-max a-truncate-ellipsis u-max-width-330 u-max-width-230@tablet-only"> Not Like Us
</h3> == $0
```

Finding the Columns in the Row

Confirmed!



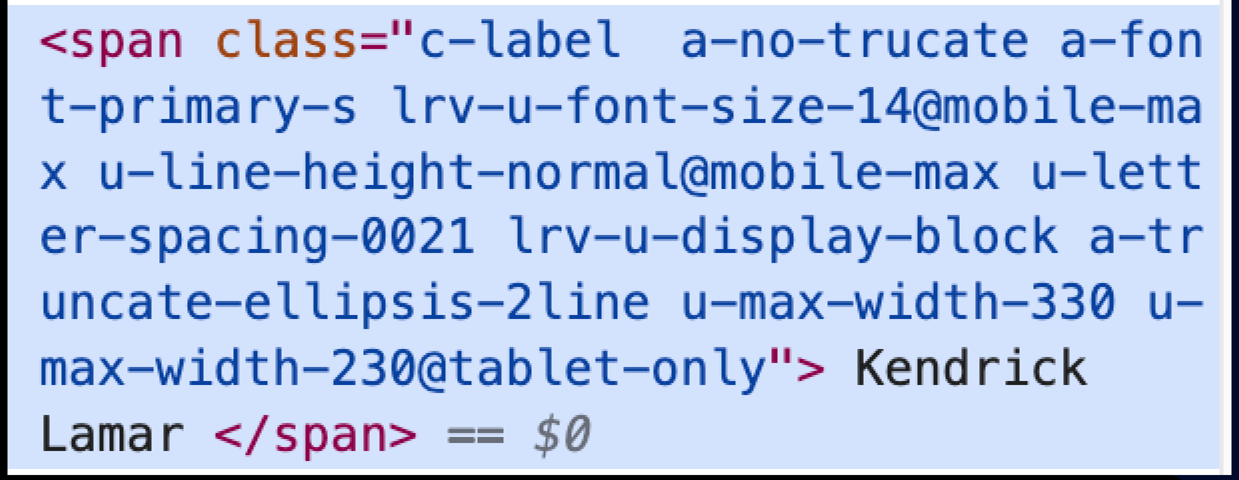
h3#title-of-a-story.c-title.a-no-trucate.a-font-primary-bold-s.u-letter-spacing-0... 230 × 22.5



```
<h3 id="title-of-a-story" class="c-title
a-no-trucate a-font-primary-bold-s u-let
ter-spacing-0021 lrv-u-font-size-18@tabl
et lrv-u-font-size-16 u-line-height-125
u-line-height-normal@mobile-max a-trunca
te-ellipsis u-max-width-330 u-max-width-
230@tablet-only"> Timeless </h3> == $0
```

Finding the Columns in the Row

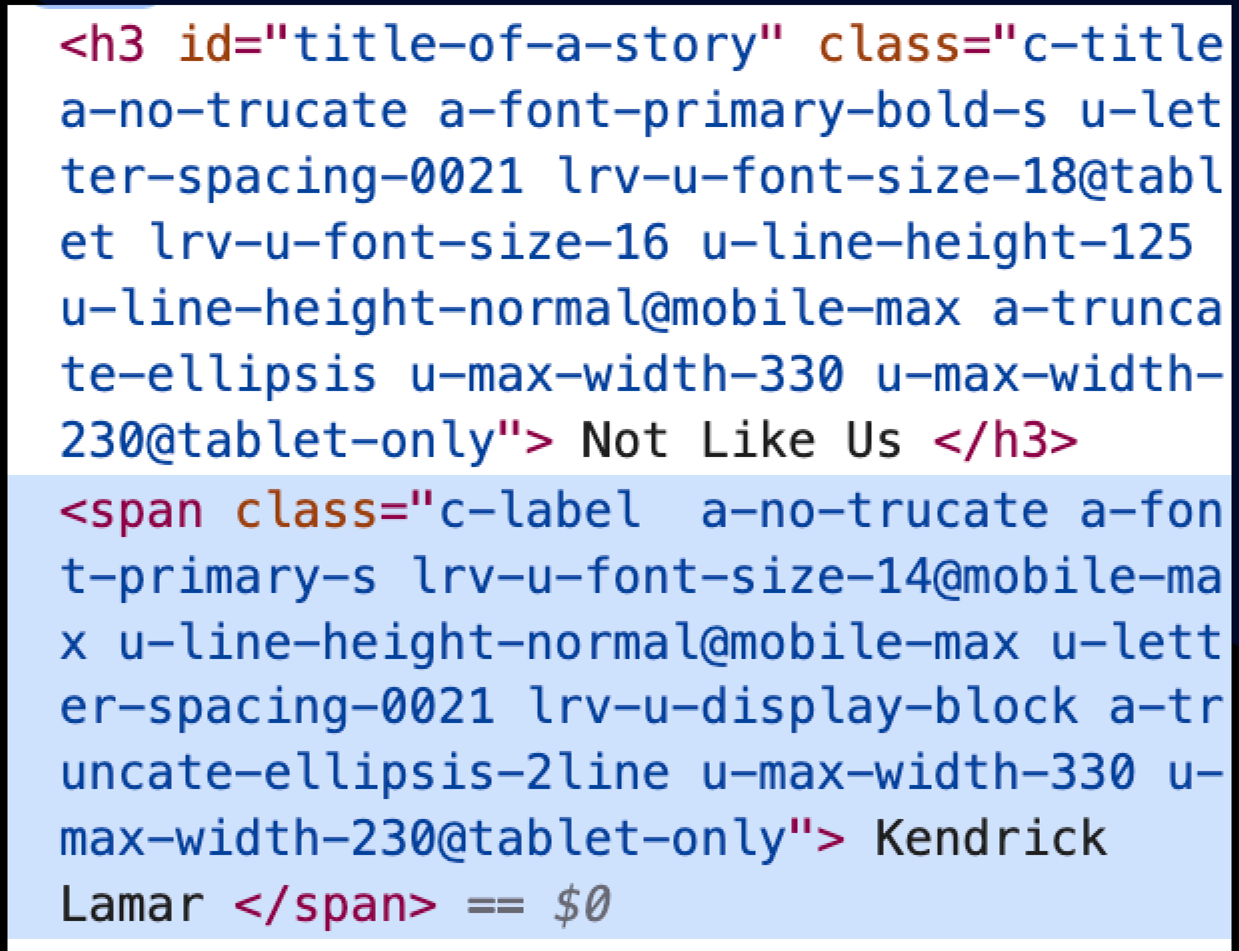
The artist name is a little harder. Nothing immediately jumps out as a unique identifying class name.



```
<span class="c-label  a-no-trucate a-fon
t-primary-s lrv-u-font-size-14@mobile-ma
x u-line-height-normal@mobile-max u-lett
er-spacing-0021 lrv-u-display-block a-tr
uncate-ellipsis-2line u-max-width-330 u-
max-width-230@tablet-only"> Kendrick
Lamar </span> == $0
```

Finding the Columns in the Row

But this info does appear right next to the song name that we do know how to find!

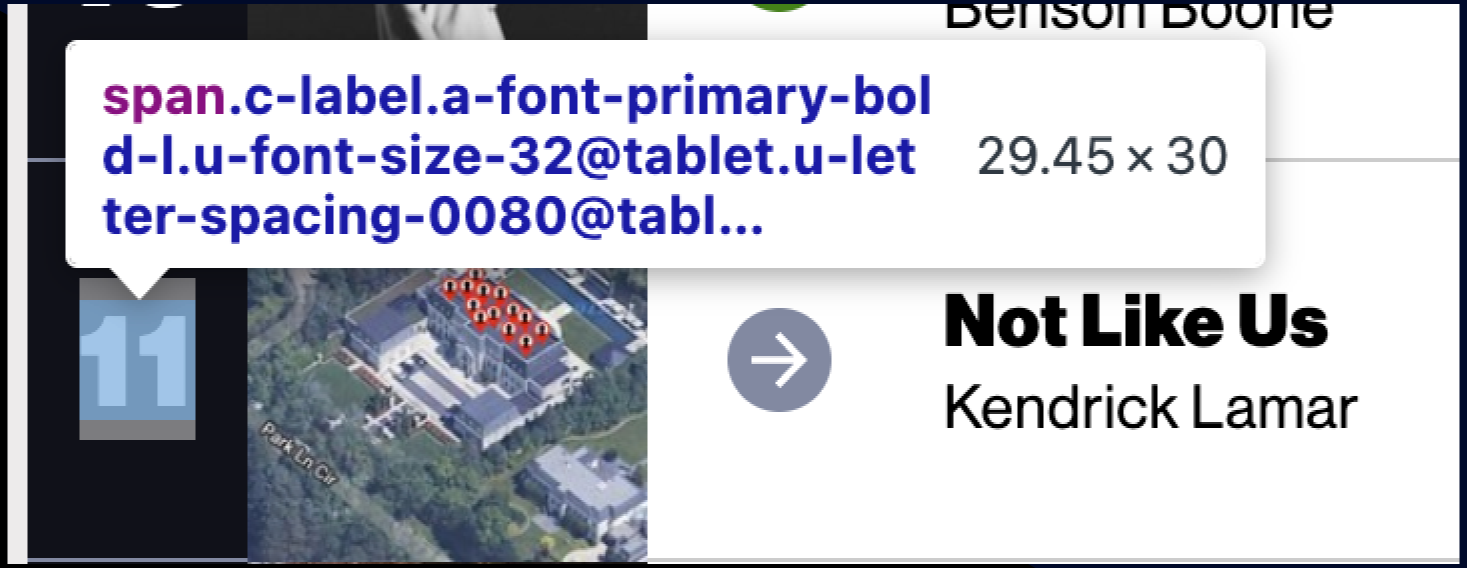


```
<h3 id="title-of-a-story" class="c-title
a-no-trucate a-font-primary-bold-s u-let
ter-spacing-0021 lrv-u-font-size-18@tabl
et lrv-u-font-size-16 u-line-height-125
u-line-height-normal@mobile-max a-trunca
te-ellipsis u-max-width-330 u-max-width-
230@tablet-only"> Not Like Us </h3>
<span class="c-label  a-no-trucate a-fon
t-primary-s lrv-u-font-size-14@mobile-ma
x u-line-height-normal@mobile-max u-lett
er-spacing-0021 lrv-u-display-block a-tr
uncate-ellipsis-2line u-max-width-330 u-
max-width-230@tablet-only"> Kendrick
Lamar </span> == $0
```

It's the text of the next sibling of the next sibling* of the `h3` containing the artist!

Finding the Columns in the Row

To find the chart position, we can look at the big number.



span.c-label.a-font-primary-bold-l.u-font-size-32@tablet.u-letter-spacing-0080@tabl... 29.45 × 30

A `span` with a `c-label` class is not unique, but it does happen to be the first `span` child of the row, so we can just rely on that, conveniently.

Printing the Information

```python
from bs4 import BeautifulSoup

html_file = open('index.html', 'r')
html_doc = html_file.read()
soup = BeautifulSoup(html_doc, 'html.parser')

rows = soup.find_all('ul', class_='o-chart-results-list-row')

print(rows[0].find('span', class_='c-label').text.strip()) # position on chart
print(rows[0].find('h3', id='title-of-a-story').text.strip()) # title
print(rows[0].find(
    'h3', id='title-of-a-story').next_sibling.next_sibling.text.strip()) # artist
```

```
1
A Bar Song (Tipsy)
Shaboozey
```

Printing the Information

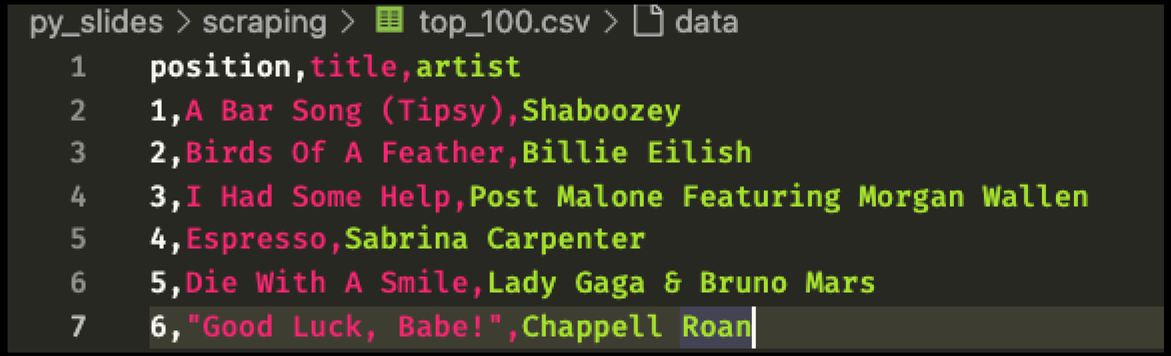
```python
for row in rows:
    position = row.find('span', class_='c-label').text.strip()
    title = row.find('h3', id='title-of-a-story').text.strip()
    artist = row.find(
        'h3', id='title-of-a-story').next_sibling.next_sibling.text.strip()
    print(f'{position}: {title} - {artist}')
```

🖨 ▼

```
1: A Bar Song (Tipsy) - Shaboozey
2: Birds Of A Feather - Billie Eilish
3: I Had Some Help - Post Malone Featuring Morgan Wallen
4: Espresso - Sabrina Carpenter
5: Die With A Smile - Lady Gaga & Bruno Mars
...
```

Saving in a DataFrame / to CSV

```python
dicts = []
for row in rows:
    position = row.find('span', class_='c-label').text.strip()
    title = row.find('h3', id='title-of-a-story').text.strip()
    artist = row.find(
        'h3', id='title-of-a-story').next_sibling.next_sibling.text.strip()
    dicts.append({"position": position, "title": title, "artist": artist})
pd.DataFrame(dicts).to_csv('top_100.csv', index=False)
```



```
position,title,artist
1,A Bar Song (Tipsy),Shaboozey
2,Birds Of A Feather,Billie Eilish
3,I Had Some Help,Post Malone Featuring Morgan Wallen
4,Espresso,Sabrina Carpenter
5,Die With A Smile,Lady Gaga & Bruno Mars
6,"Good Luck, Babe!",Chappell Roan
```

CIS 1100

Requests

Python

Fall 2024

University of Pennsylvania

Taking the Human Out of the Loop

We defined the scraping process like so:

1. traversing the internet to find web pages that contain interesting information
2. extracting that information from each web page
3. storing the extracted information in a useful format

But we've only addressed the latter two points so far!

`requests`

`pip install requests` to get access to a library that allows you to:

- programmatically "visit" websites
- get responses (HTML) within your program
- do all kinds of advanced stuff like *upload information to servers* or *communicate with APIs*

The Very Very Very Basics

- `get("my.url.com")` queries the website at that URL and returns a `Response`
- A `Response` is a dense object that contains information about what the remote server "said"
  - response code: a number that indicates whether your request was processed properly
  - information about the data encoding
  - the text of the response, i.e. all the HTML (or JSON...)

A Minimal Request

```python
import requests

url = "https://www.cis.upenn.edu/~cis110/current/py/homework/homework.html"
r = requests.get(url)
print(r)
```

🖨 🔽

```
<Response [200]>
```

✅

A Minimal Request

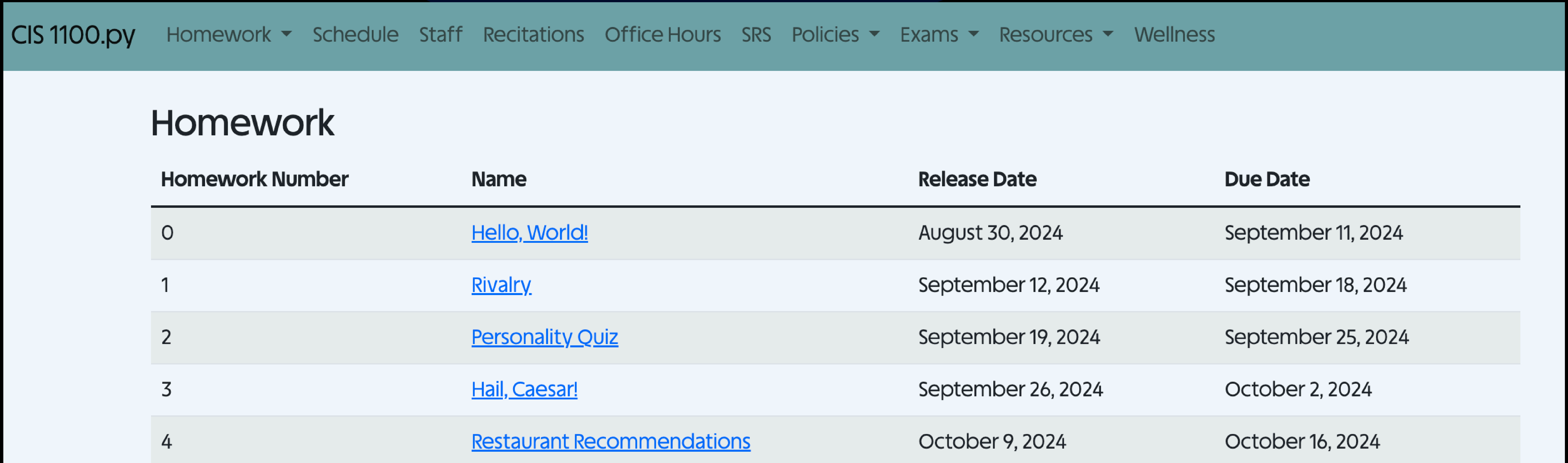
```python
import requests

url = "https://www.cis.upenn.edu/~cis110/current/py/homework/homework.html"
r = requests.get(url)
print(r.text)
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>
      CIS 1100 Homework
    </title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <link
      href="/~cis110/current/assets/css/py_style.css"
      rel="stylesheet"
    />

    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/11.7.0/styles/github.min.css"
    />
    <script src="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/11.7.0/highlight.min.js"></script>
    <script>
      MathJax = {
        tex: {
          inlineMath: [
            ['$', '$'],
            ['\\(', '\\)'],
          ],
        },
        svg: {
          fontCache: 'global',
        },
      }
    </script>
    <script
      type="text/javascript"
      id="MathJax-script"
      async
      src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-svg.js"
    ></script>
  </head>
  <body>


<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a
      class="navbar-brand"
      href="/~cis110/current/py/index.html"
      >CIS 1100.py </a
```

A Minimal Request

```python
import requests

url = "https://www.cis.upenn.edu/~cis110/current/py/homework/homework.html"
r = requests.get(url)
print(r.text)
```

`r.text` is just a string containing HTML, though. We know what to do with that...



Homework

Homework Number	Name	Release Date	Due Date
0	Hello, World!	August 30, 2024	September 11, 2024
1	Rivalry	September 12, 2024	September 18, 2024
2	Personality Quiz	September 19, 2024	September 25, 2024
3	Hail, Caesar!	September 26, 2024	October 2, 2024
4	Restaurant Recommendations	October 9, 2024	October 16, 2024

A Minimal Request

```python
import requests
from bs4 import BeautifulSoup

url = "https://www.cis.upenn.edu/~cis110/current/py/homework/homework.html"
r = requests.get(url)
soup = BeautifulSoup(r.text, 'html.parser')
links = soup.table.find_all('a')
print([link.text for link in links])
```

```
['Hello, World!', 'Rivalry', 'Personality Quiz', 'Hail, Caesar!', 'Restaurant Recommendations']
```