



WEB SCRAPING WITH PYTHON

# Web Scraping With Python

Thomas Laetsch  
Data Scientist, NYU



# Business Savvy

## What are businesses looking for?

- Comparing prices
- Satisfaction of customers
- Generating potential leads
- ...and much more!

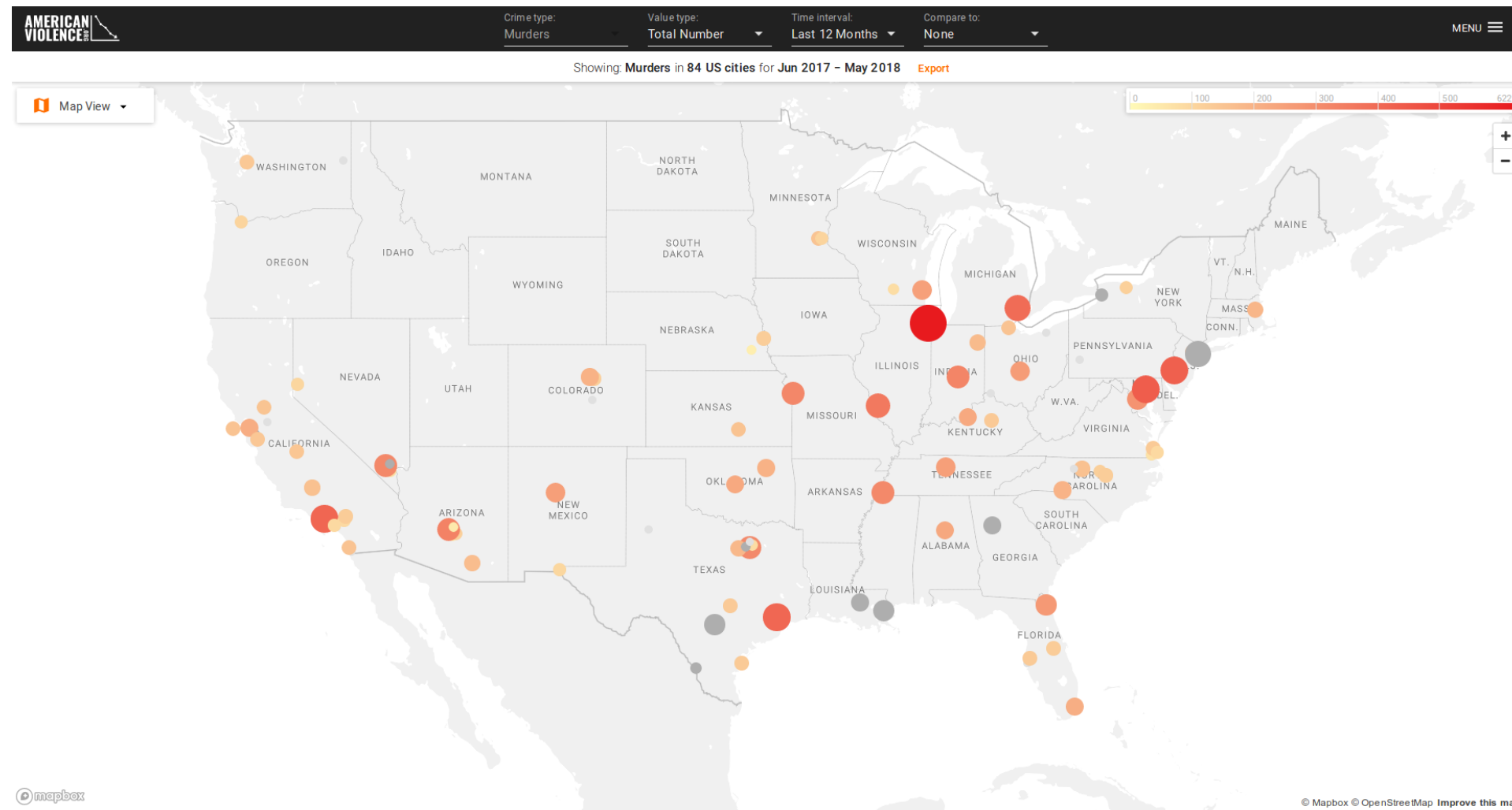


# It's Personal

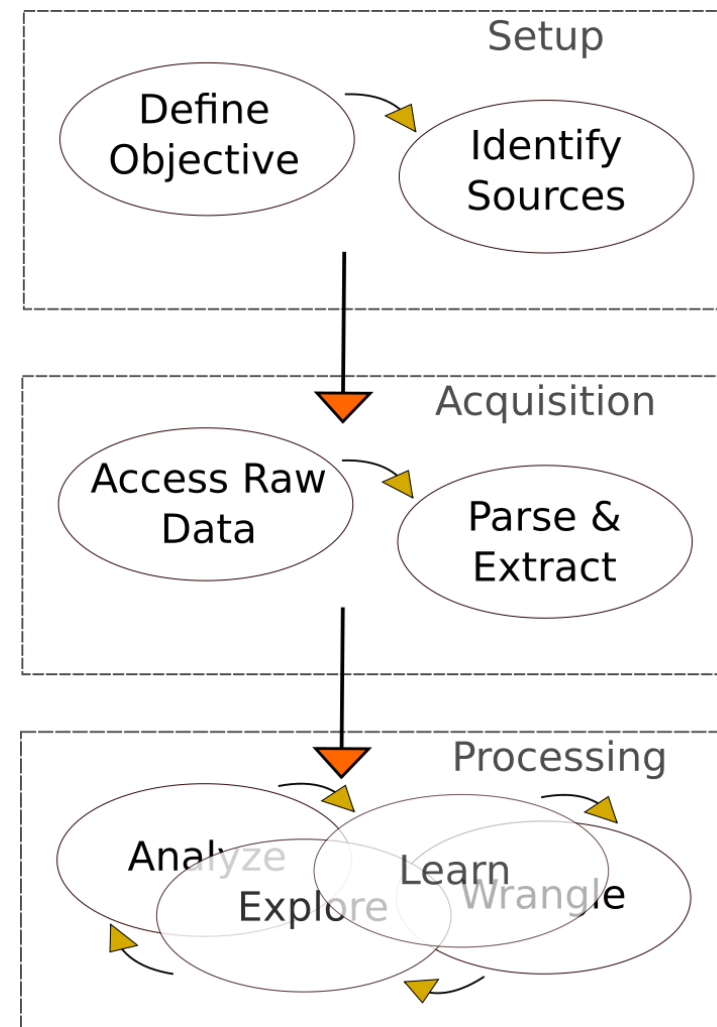
## What could you do?

- Search for your favorite memes on your favorite sites.
- Automatically look through classified ads for your favorite gadgets.
- Scrape social site content looking for hot topics.
- Scrape cooking blogs looking for particular recipes, or recipe reviews.
- ...and much more!

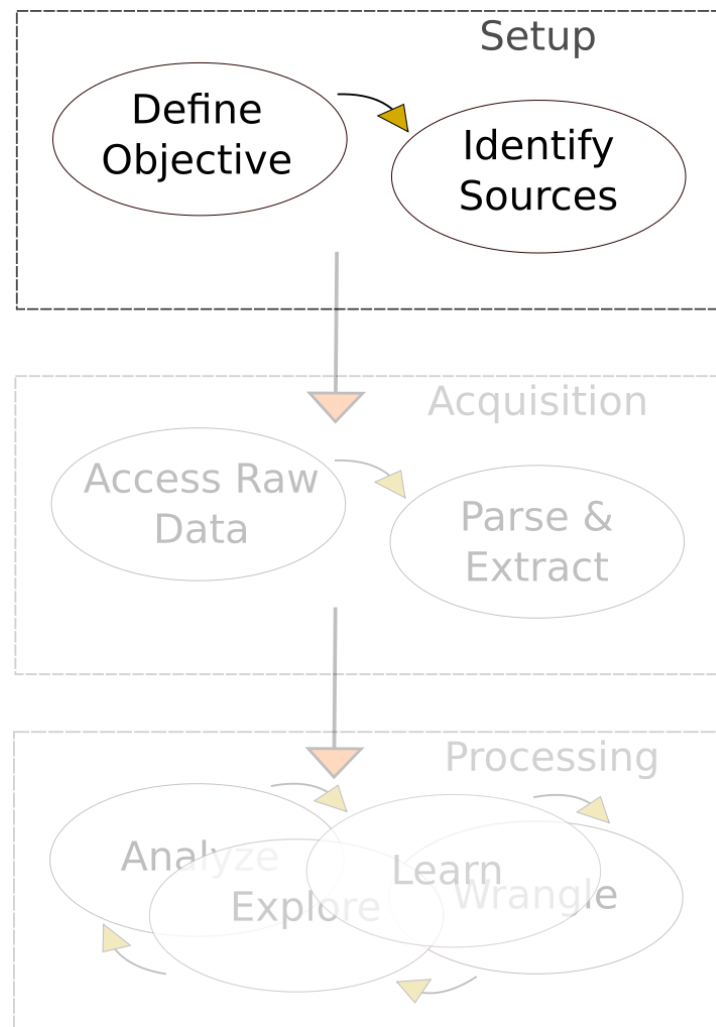
# About My Work



# Pipe Dream



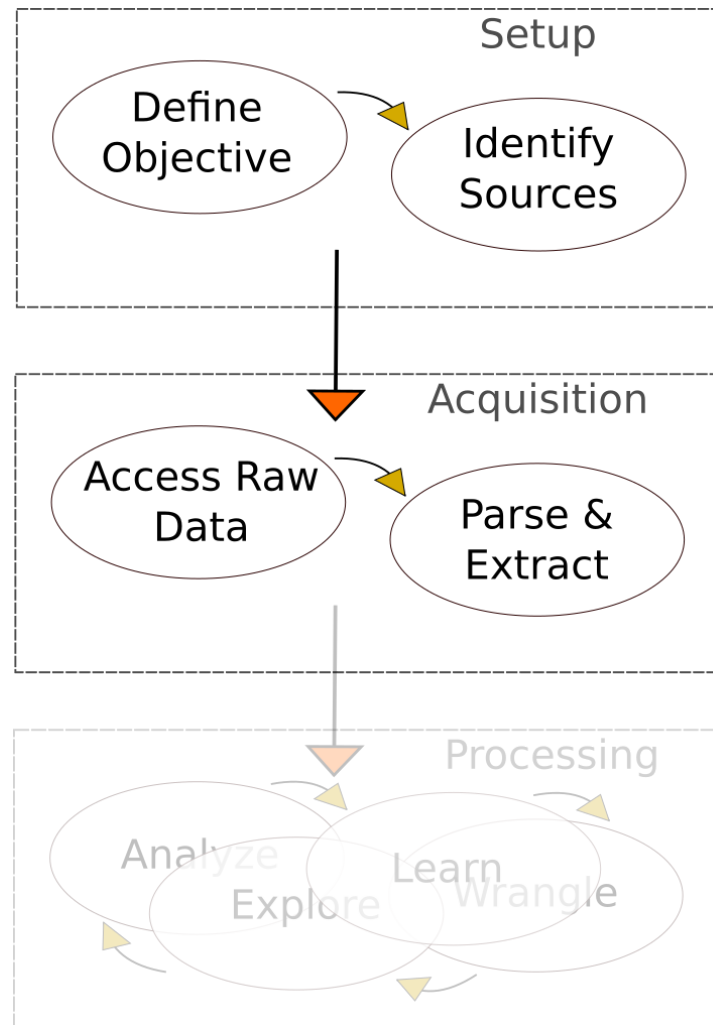
# Pipe Dream: Setup



## Setup

- Understand what we want to do.
- Find sources to help us do it.

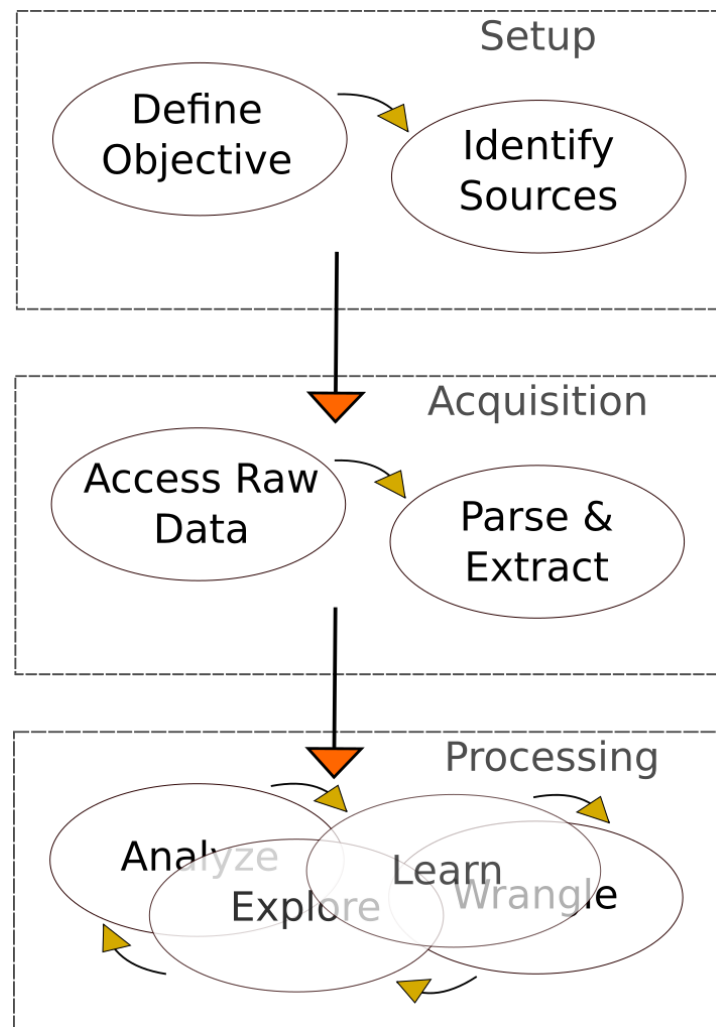
# Pipe Dream: Acquisition



## Acquisition

- Read in the raw data from online.
- Format these data to be usable.

# Pipe Dream: Processing



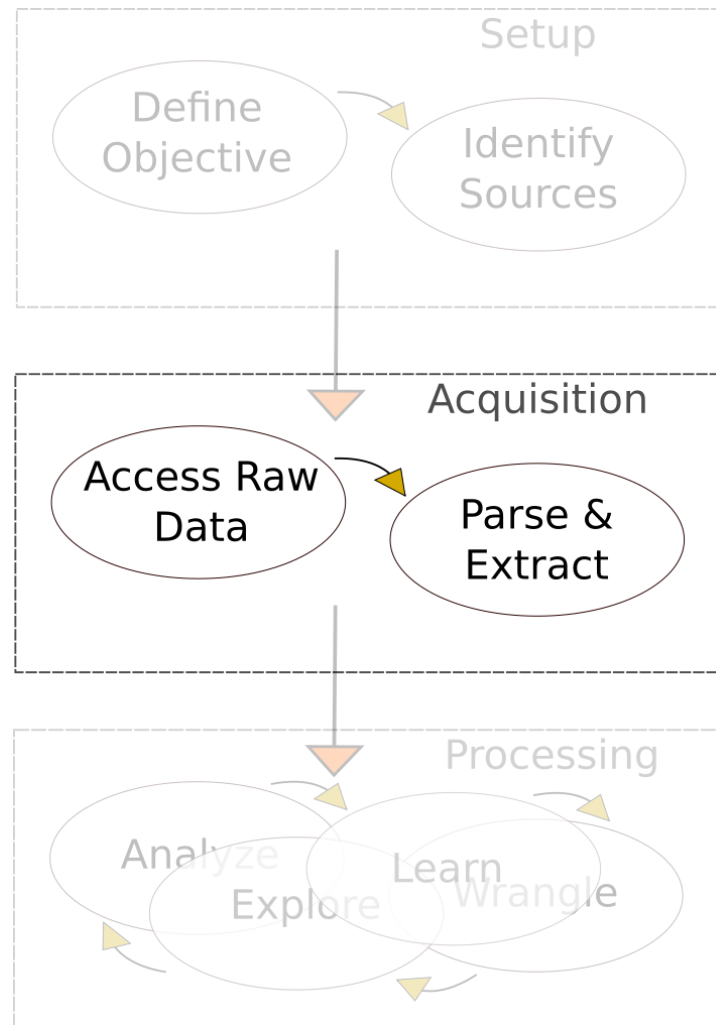
## Processing

- Many options!





# How do you do?



## Our Focus

- Acquisition!
- (Using `scrapy` **via** `python`)



## WEB SCRAPING WITH PYTHON

**Are you in?**



WEB SCRAPING WITH PYTHON

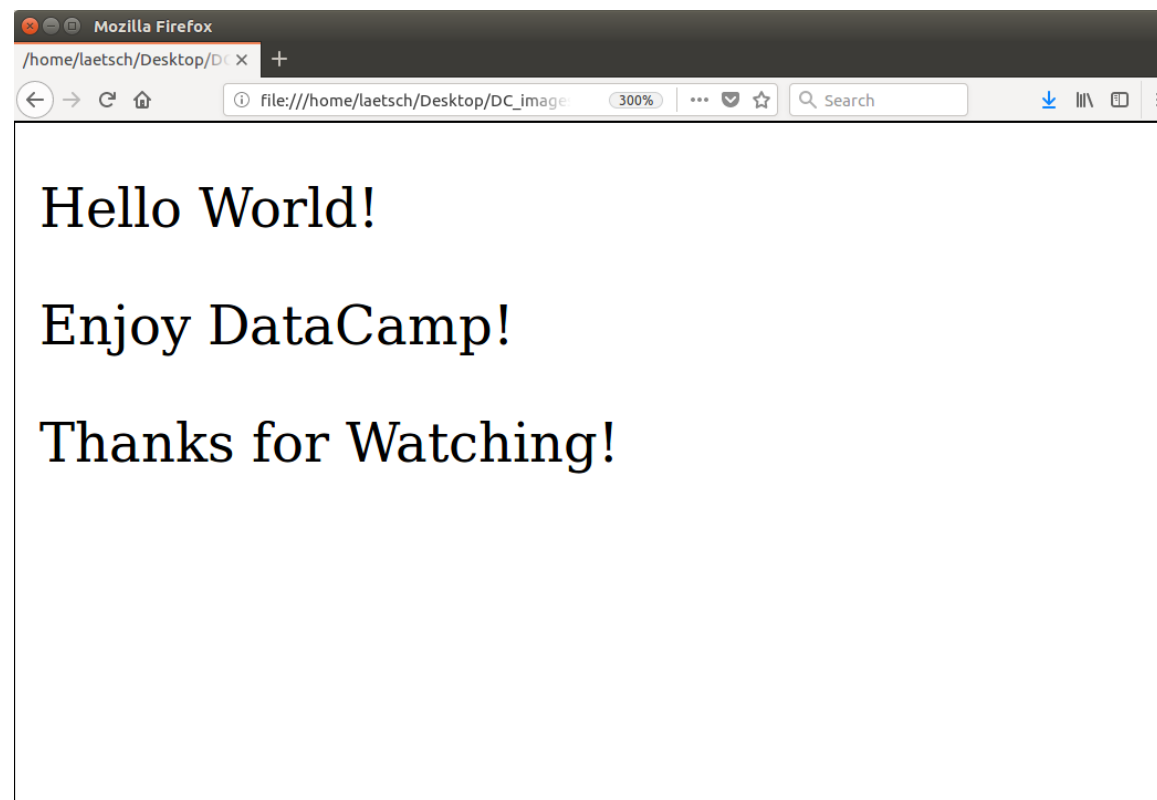
# HyperText Markup Language

Thomas Laetsch  
Data Scientist, NYU



# The main example

```
<html>
  <body>
    <div>
      <p>Hello World!</p>
      <p>Enjoy DataCamp!</p>
    </div>
    <p>Thanks for Watching!</p>
  </body>
</html>
```





# HTML tags

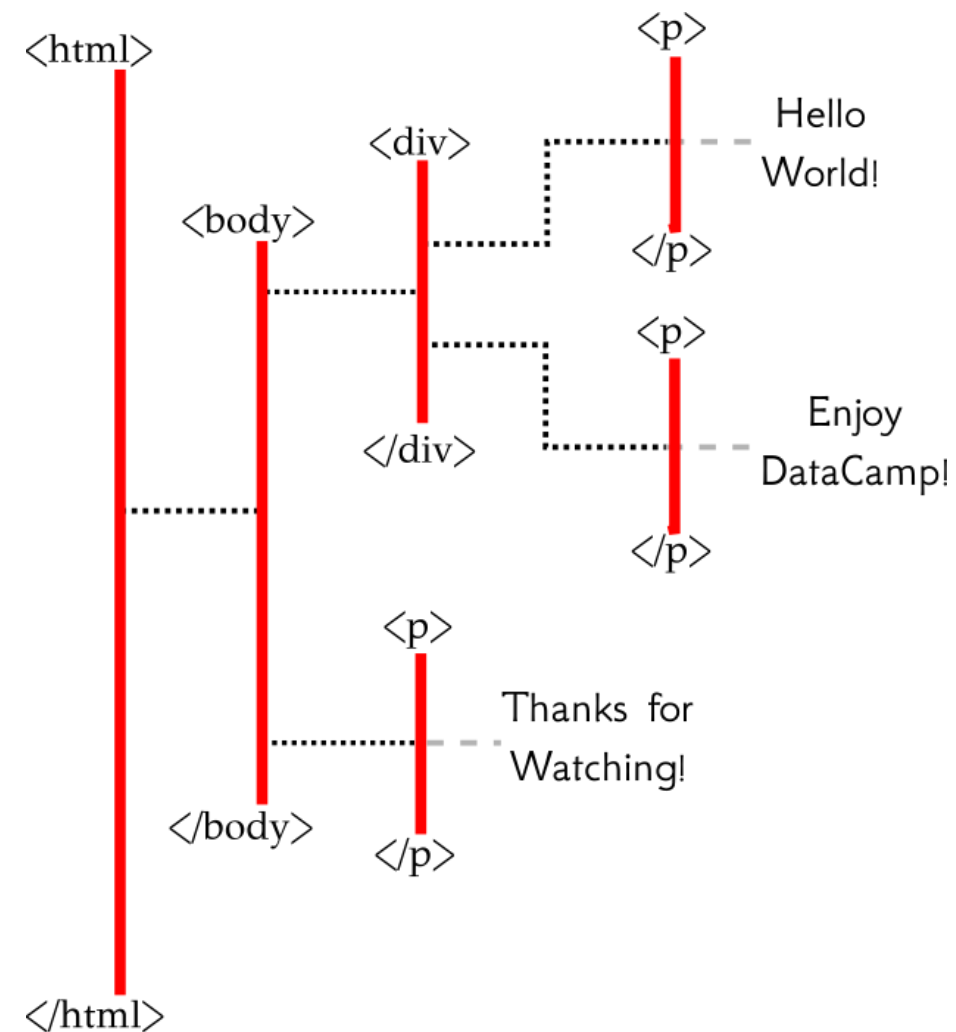
```
<html>
  <body>
    <div>
      <p>Hello World!</p>
      <p>Enjoy DataCamp!</p>
    </div>
    <p>Thanks for Watching!</p>
  </body>
</html>
```

- `<html> ... </html>`
- `<body> ... </body>`
- `<div> ... </div>`
- `<p> ... </p>`



# The HTML tree

```
<html>
  <body>
    <div>
      <p>Hello World!</p>
      <p>Enjoy DataCamp!</p>
    </div>
    <p>Thanks for Watching!</p>
  </body>
</html>
```

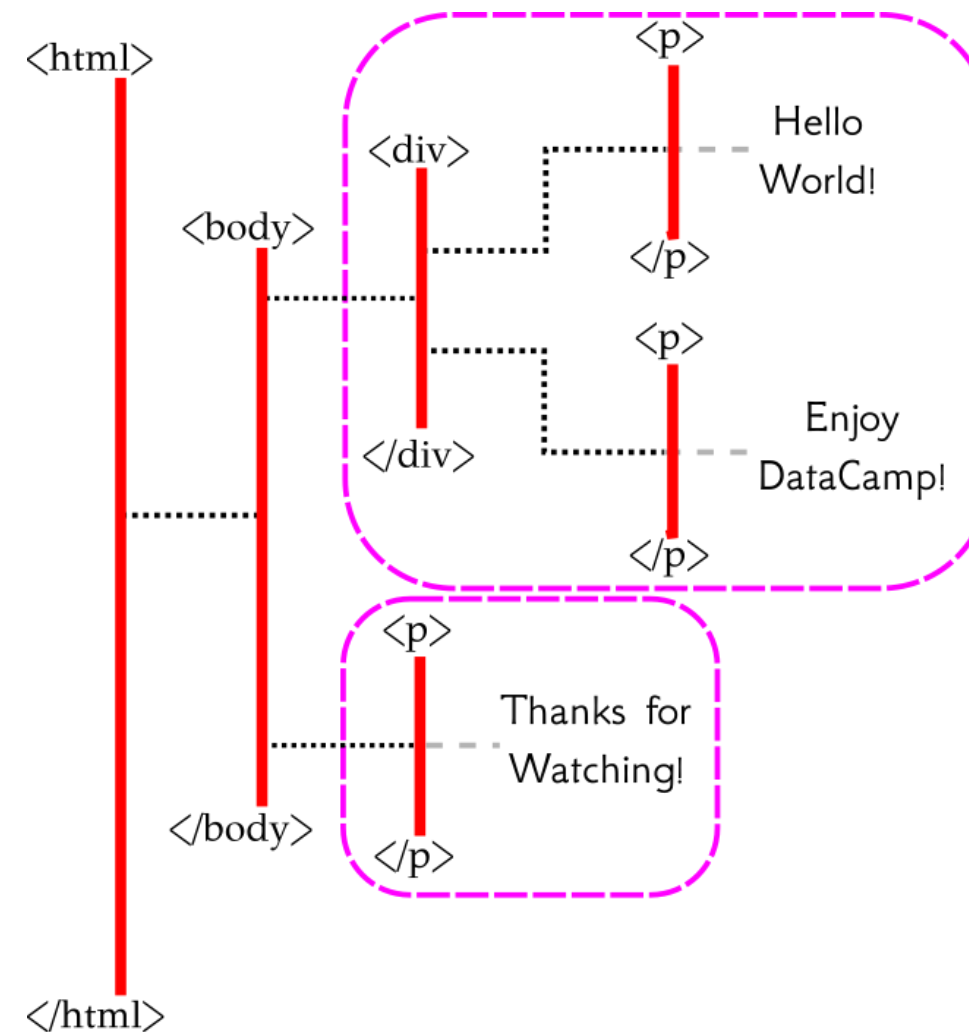


# The HTML tree: Example 1

```

<html>
  <body>
    <div>
      <p>Hello World!</p>
      <p>Enjoy DataCamp!</p>
    </div>
    <p>Thanks for Watching!</p>
  </body>
</html>

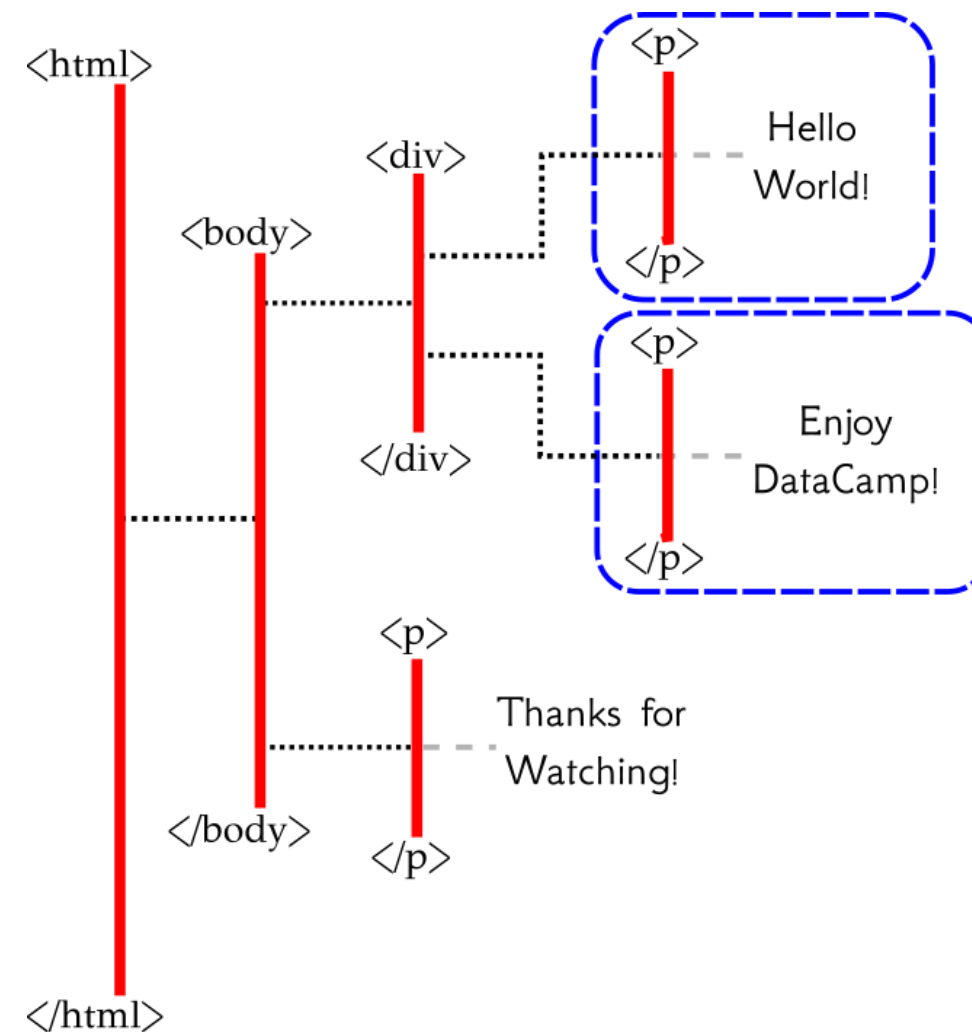
```





# The HTML tree: Example 2

```
<html>
  <body>
    <div>
      <p>Hello World!</p>
      <p>Enjoy DataCamp!</p>
    </div>
    <p>Thanks for Watching!</p>
  </body>
</html>
```







WEB SCRAPING WITH PYTHON

# Introduction to HTML Outro



WEB SCRAPING WITH PYTHON

# HTML Tags and Attributes

Thomas Laetsch  
Data Scientist, NYU



# Do we have to?

- Information within HTML tags can be valuable
- Extract link URLs
- Easier way to select elements



# Tag, you're it!

`<tag-name attrib-name="attrib info">`

`..element contents..`

`</tag-name>`

- We've seen **tag names** such as **html**, **div**, and **p**.
- The **attribute name** is followed by **=** followed by information assigned to that attribute, usually quoted text.



Let's "div"vy up the tag

```
<div id="unique-id" class="some class">
```

..div element contents..

```
</div>
```

- **id** attribute should be unique
- **class** attribute doesn't need to be unique



"a" be linkin'

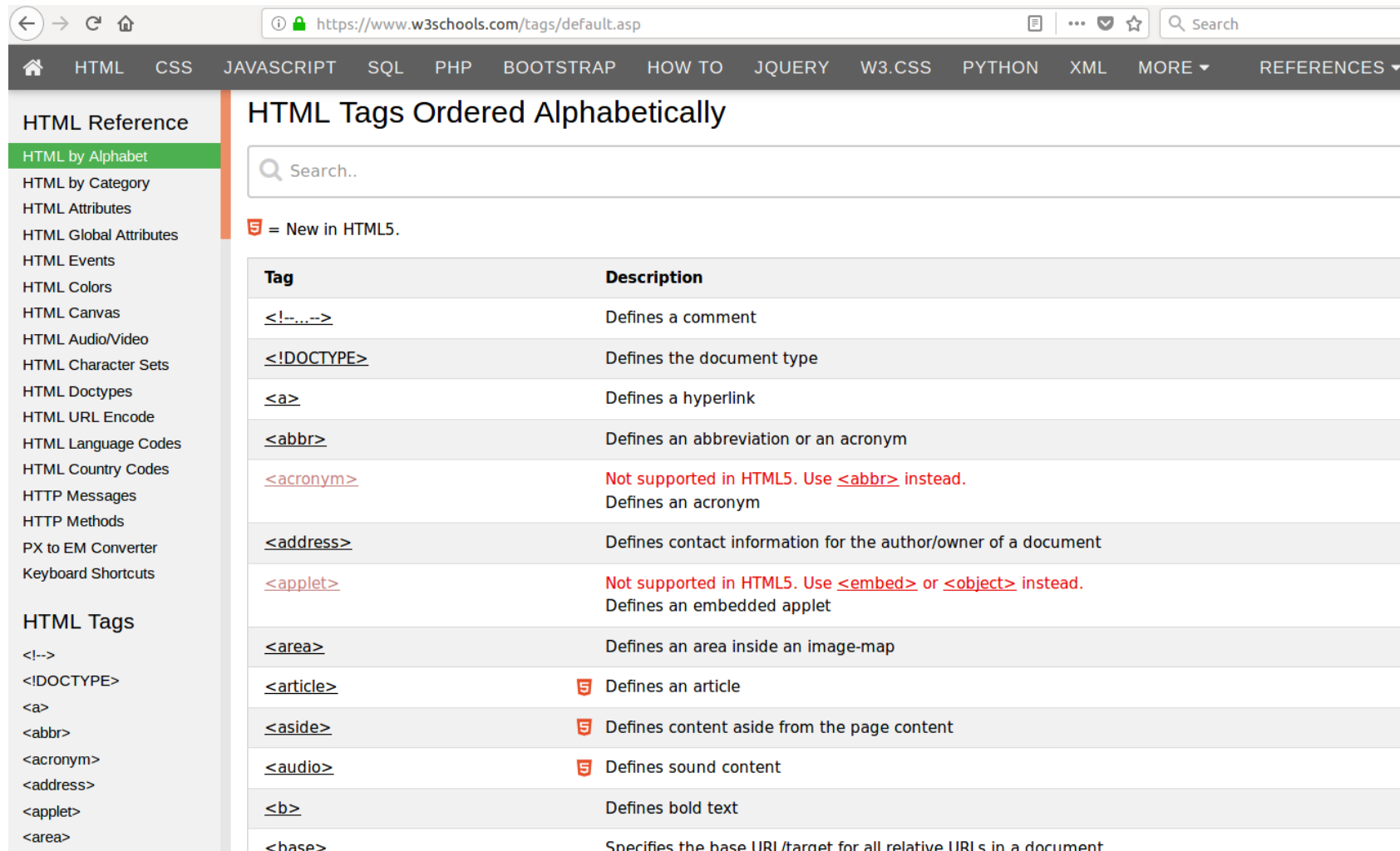
```
<a href="https://www.datacamp.com">
```

This text links to DataCamp!




```
</a>
```

- **a** tags are for **hyperlinks**
- **href** attribute tells what link to go to

# Tag Traction



The screenshot shows the W3Schools website's HTML Tags page. The browser's address bar displays the URL `https://www.w3schools.com/tags/default.asp`. The navigation menu at the top includes links for HTML, CSS, JAVASCRIPT, SQL, PHP, BOOTSTRAP, HOW TO, JQUERY, W3.CSS, PYTHON, XML, MORE, and REFERENCES. On the left sidebar, the 'HTML Reference' section is expanded, with 'HTML by Alphabet' selected. Below this, a list of HTML tags is provided, including `<!-->`, `<!DOCTYPE>`, `<a>`, `<abbr>`, `<acronym>`, `<address>`, `<applet>`, `<area>`, `<article>`, `<aside>`, `<audio>`, `<b>`, and `<base>`. The main content area is titled 'HTML Tags Ordered Alphabetically' and features a search bar. A legend indicates that a red icon represents tags 'New in HTML5'. The table below lists the tags and their descriptions, with some tags marked as not supported in HTML5.

| Tag                           | Description   |
|-------------------------------|---|
| <code>&lt;!--&gt;</code>      | Defines a comment   |
| <code>&lt;!DOCTYPE&gt;</code> | Defines the document type   |
| <code>&lt;a&gt;</code>        | Defines a hyperlink   |
| <code>&lt;abbr&gt;</code>     | Defines an abbreviation or an acronym   |
| <code>&lt;acronym&gt;</code>  | Not supported in HTML5. Use <code>&lt;abbr&gt;</code> instead.<br>Defines an acronym  |
| <code>&lt;address&gt;</code>  | Defines contact information for the author/owner of a document  |
| <code>&lt;applet&gt;</code>   | Not supported in HTML5. Use <code>&lt;embed&gt;</code> or <code>&lt;object&gt;</code> instead.<br>Defines an embedded applet      |
| <code>&lt;area&gt;</code>     | Defines an area inside an image-map   |
| <code>&lt;article&gt;</code>  |  Defines an article                          |
| <code>&lt;aside&gt;</code>    |  Defines content aside from the page content |
| <code>&lt;audio&gt;</code>    |  Defines sound content                       |
| <code>&lt;b&gt;</code>        | Defines bold text   |
| <code>&lt;base&gt;</code>     | Specifies the base URL/target for all relative URLs in a document   |



WEB SCRAPING WITH PYTHON

# Et Tu, Attributes?





WEB SCRAPING WITH PYTHON

# Crash Course X

**Thomas Laetsch**  
Data Scientist, NYU



# Another Slasher Video?

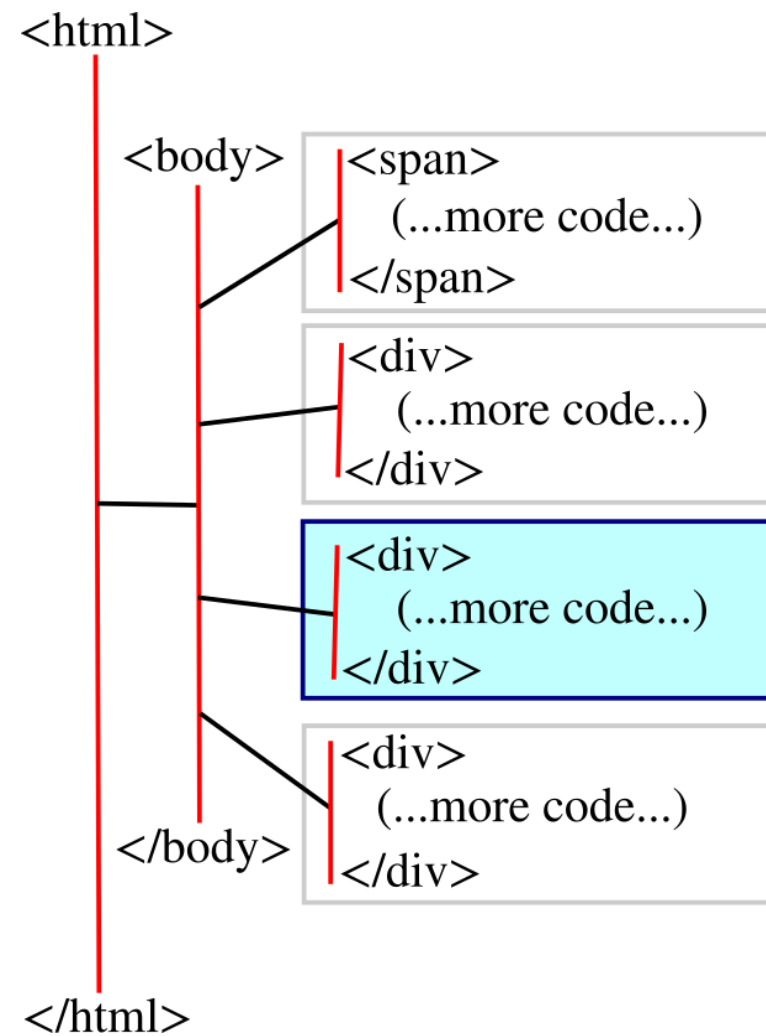
```
xpath = '/html/body/div[2]'
```

## Simple XPath:

- Single forward-slash / used to move forward one generation.
- tag-names between slashes give direction to which element(s).
- Brackets [] after a tag name tell us which of the selected siblings to choose.



# Another Slasher Video?



```
xpath = '/html/body/div[2]'
```



# Slasher Double Feature?

- Direct to all `table` elements within the entire HTML code:

```
xpath = '//table'
```

- Direct to all `table` elements which are descendants of the 2nd `div` child of the `body` element:

```
xpath = '/html/body/div[2]//table'
```



WEB SCRAPING WITH PYTHON

**Ex(path)celent**