

# Web Scraping With Python

WEB SCRAPING IN PYTHON



**Thomas Laetsch**  
Data Scientist, NYU

# Business Savvy

## What are businesses looking for?

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

Before moving to specifics and technicalities, let me convince you that these techniques can be a valuable addition to your data-science know-how, and that this course will be the perfect place to start or strengthen the foundational pieces of this skill set.

### 2. Business Savvy

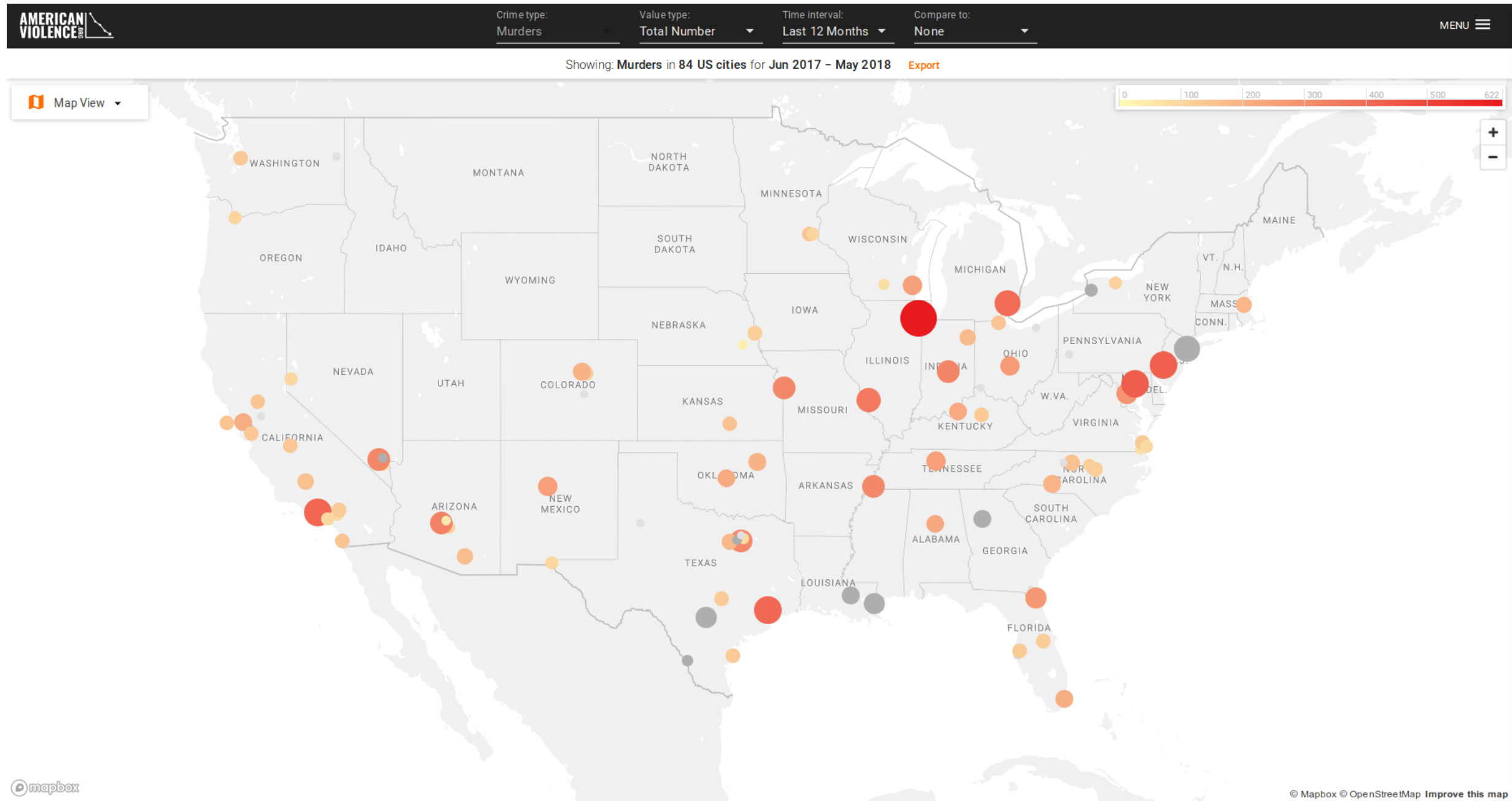
What can businesses gain from web-scraping? Well, they can scrape competitor sites to gather prices for similar products or services to compare and adjust their own price set-points. They can scrape online reviews of their products or services, and gather public opinion around the company in general. They can scrape social media sites, or other public forums for contact or other information of clients or potential clients, to meaningfully direct resources towards this group of possible customers. And this was just a short list!

# 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

## 5. Pipe Dream

To better visualize the focus of what you will learn in these lectures and exercises, let's roughly breakdown the web-scraping pipeline into three pieces.

## 6. Pipe Dream: Setup

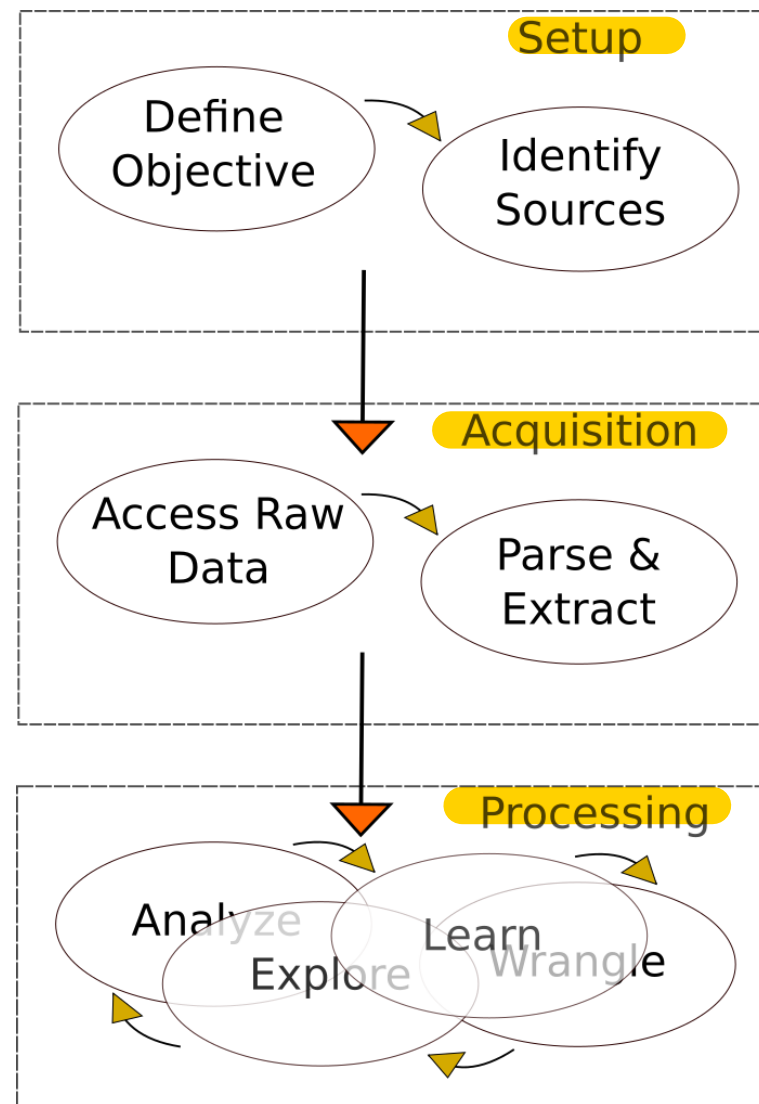
The first piece is the setup, that is, defining the goal or task and identifying the online sources which you believe will help you achieve the desired end result.

## 7. Pipe Dream: Acquisition

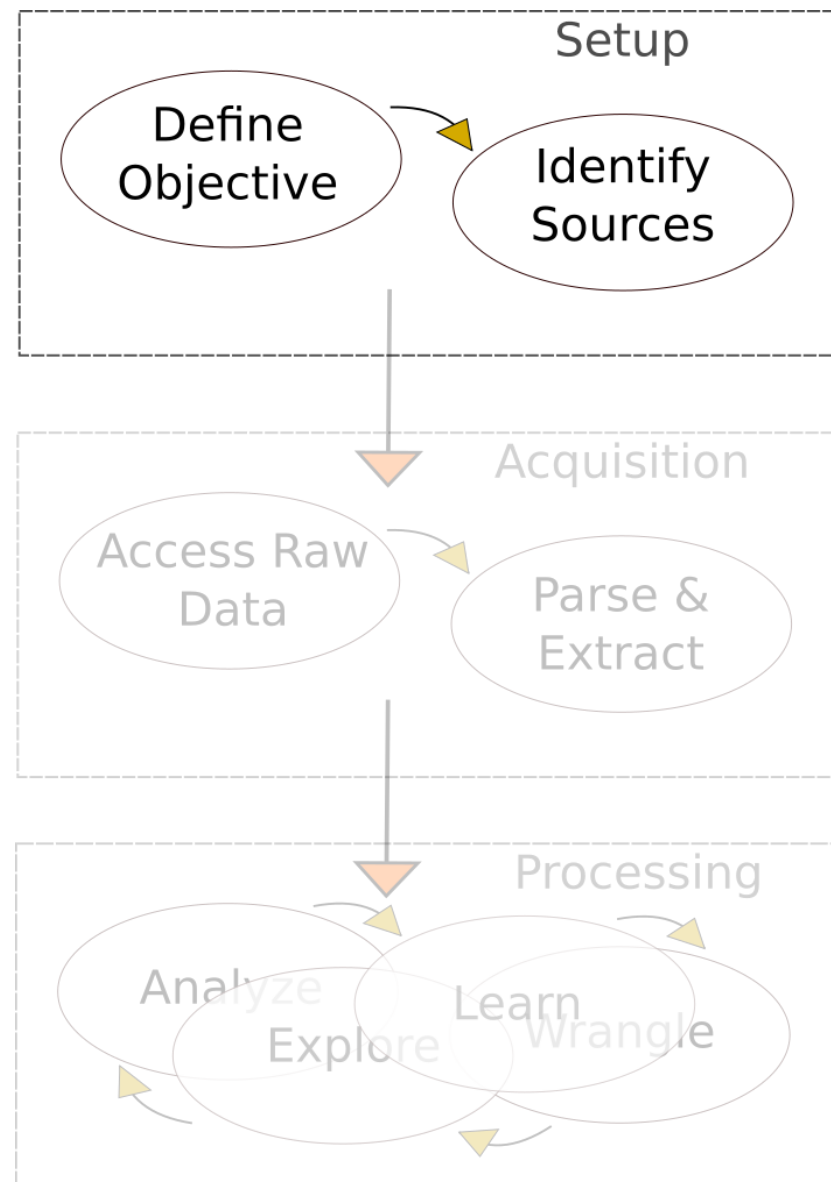
The second is the acquisition of these online data. This includes accessing the data, parsing this information, and extracting these data into meaningful and useful data structures.

## 8. Pipe Dream: Processing

The third is the processing phase, where you run these downloaded data through whatever analyses



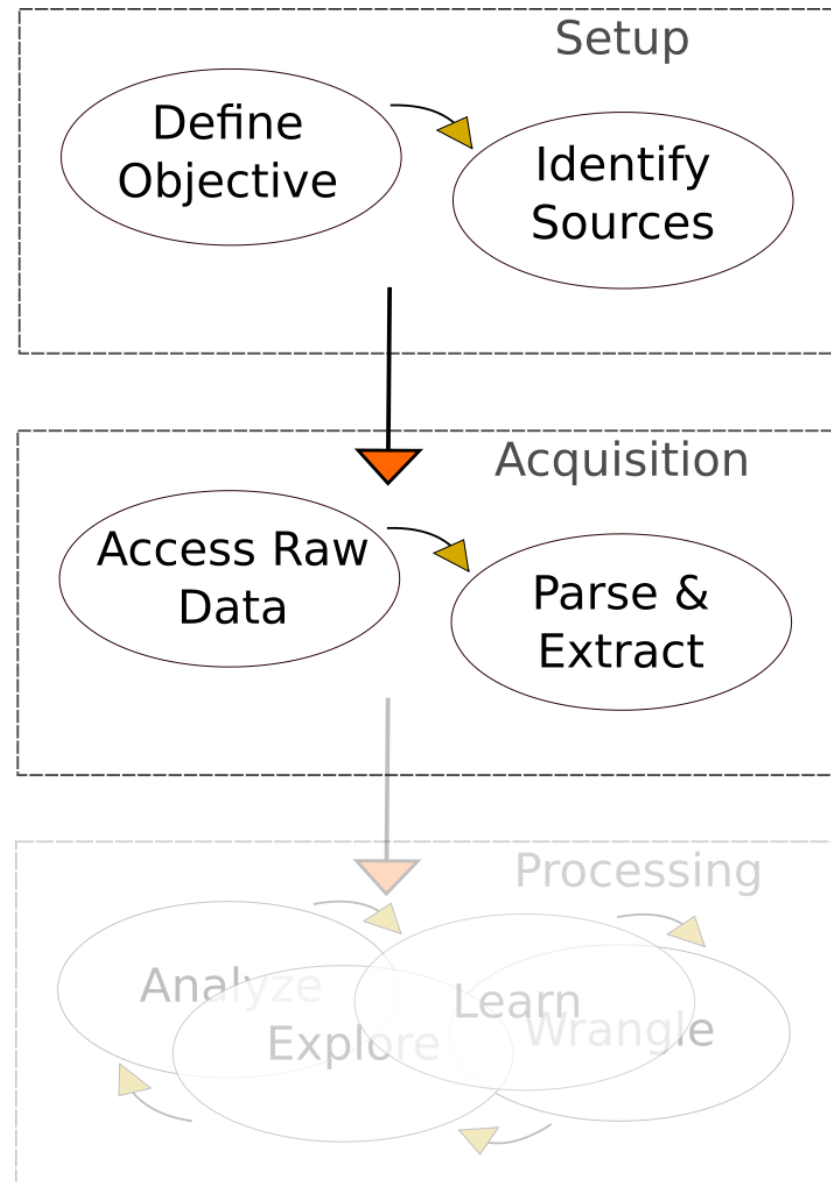
# Pipe Dream: Setup



## Setup

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

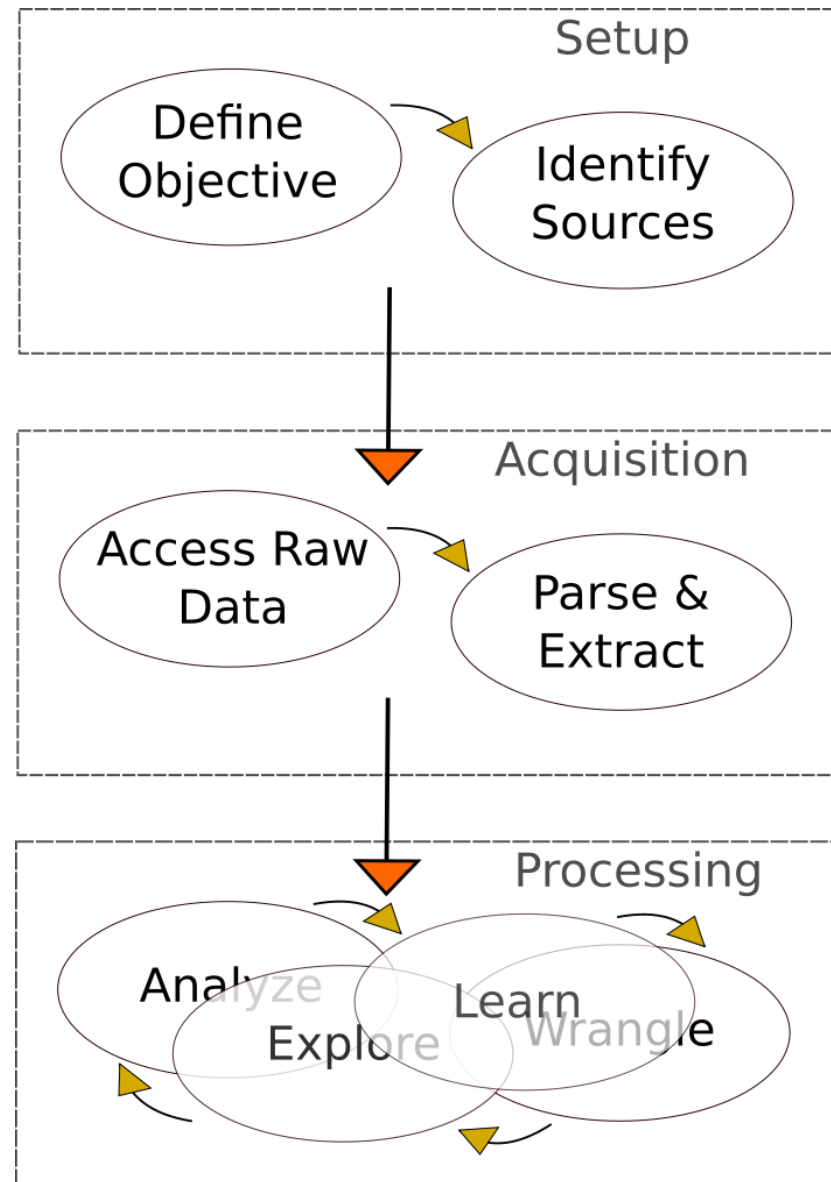
# Pipe Dream: Acquisition



## Acquisition

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

# Pipe Dream: Processing



## Processing

- Many options!



# How do you do?

. How do you do?

This course focuses on the acquisition phase. To accomplish this, we will be using python and the web-crawling framework scrapy. We chose scrapy since we can jump in quickly, and easily scale to large scraping projects. However, even if you aren't sold on using scrapy or python, you will still build techniques and intuition that will be valuable in any computational web-scraping environment you enjoy!

## Our Focus

- Acquisition!
- (Using scrapy via python )

# Are you in?

WEB SCRAPING IN PYTHON

# HyperText Markup Language

WEB SCRAPING IN PYTHON



**Thomas Laetsch**  
Data Scientist, NYU

# The main example

```
<html>
```

```
  <body>
```

```
    <div>
```

```
      <p>Hello World!</p>
```

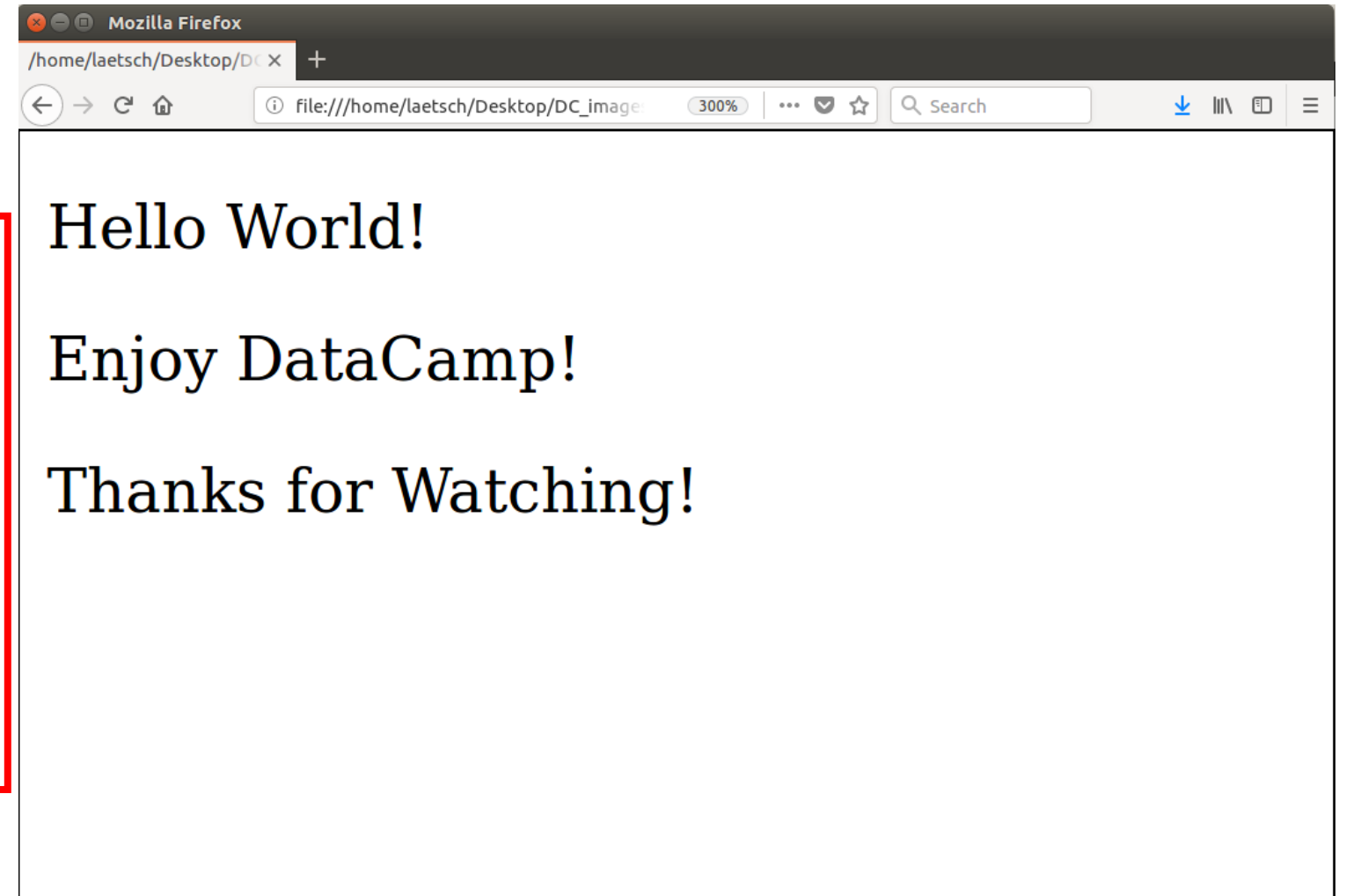
```
      <p>Enjoy DataCamp!</p>
```

```
    </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>

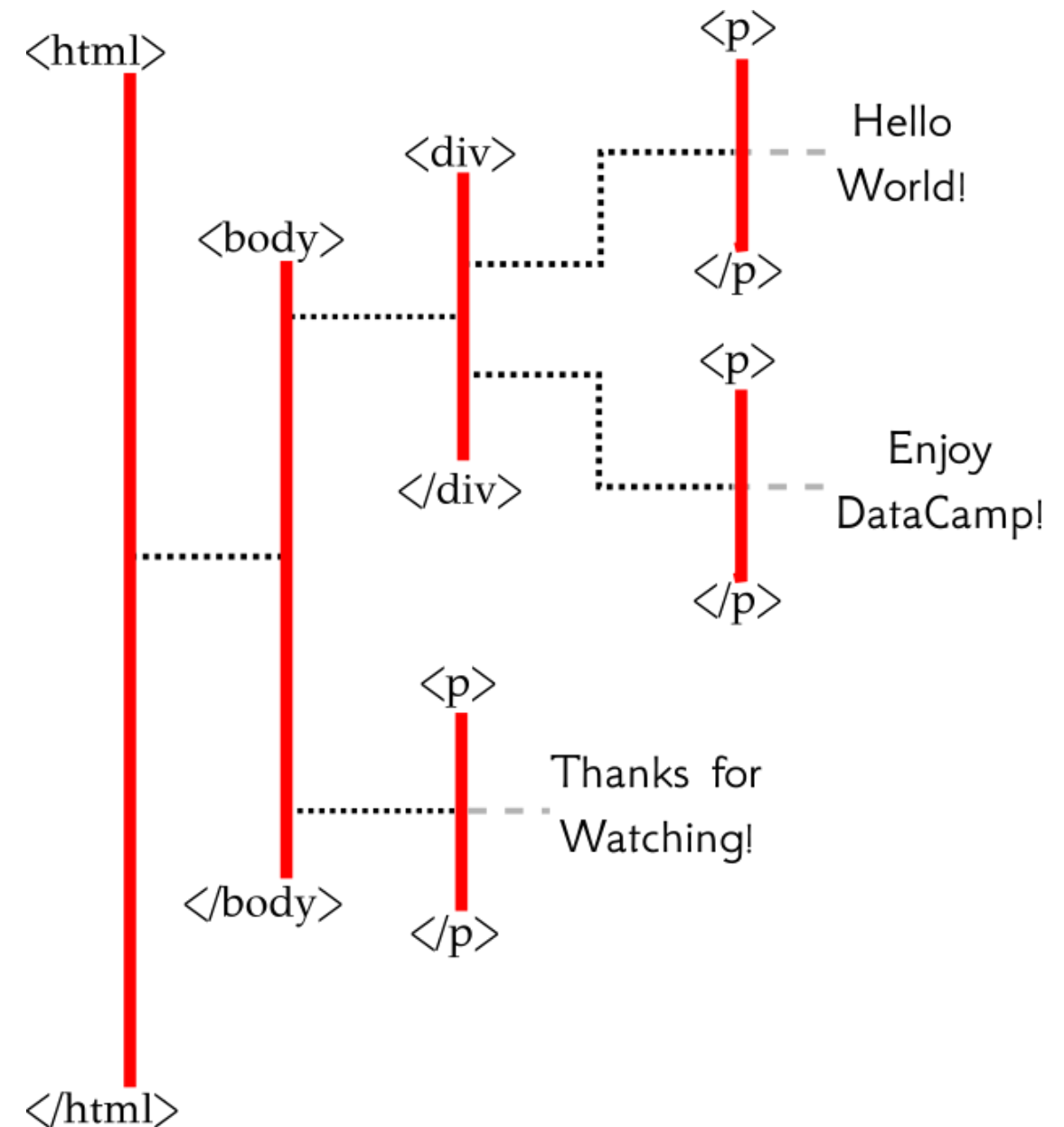
- <html> ... </html>
- <body> ... </body>
- <div> ... </div>
- <p> ... </p>

## 3. HTML tags

The elements contained within the angle brackets are called "HTML tags", which, in well-formatted HTML, usually come in pairs. The pair contains a starting tag without a forward slash, and stopping tag with a forward slash. The root tag containing the main HTML content are those with the text "html" within the brackets. We also have a body tag, defining the body of the html; a div tag defining a section of the body; and several p tags defining paragraphs within the body.

# The HTML tree

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

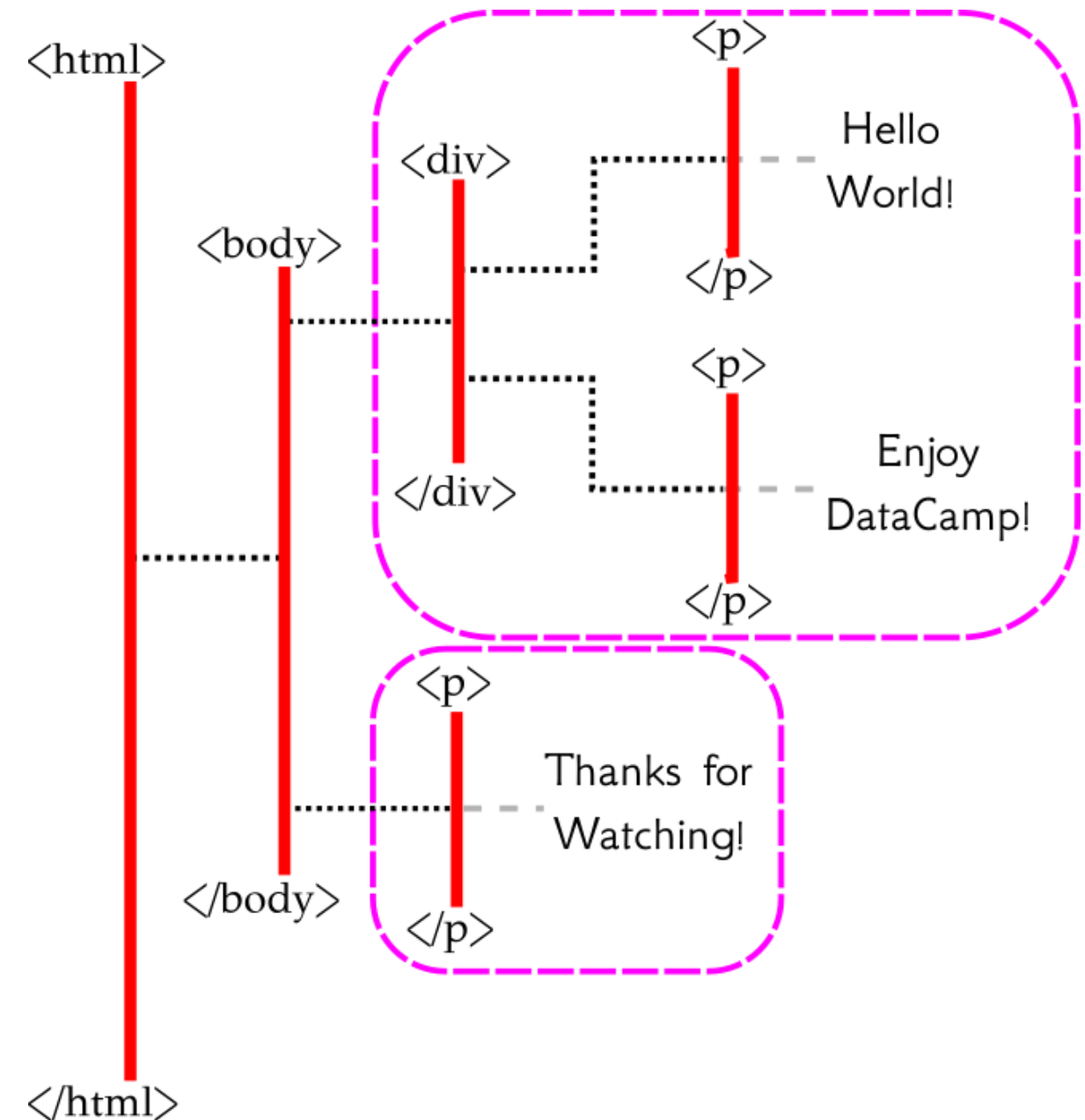


## 4. The HTML tree

We notice that the HTML tags are nested within others, such as the body tag nested within the html root tag; a div tag nested within the body tag; two paragraph tags nested within the div tag; etc.. This nesting gives rise to a hierarchy in the HTML which can be visualized as a tree structure as displayed here. As we move from left to right, we are moving forward generations, as we move top to bottom, we are moving between the same generation, and moving between siblings if the elements come from the

# The HTML tree: Example 1

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

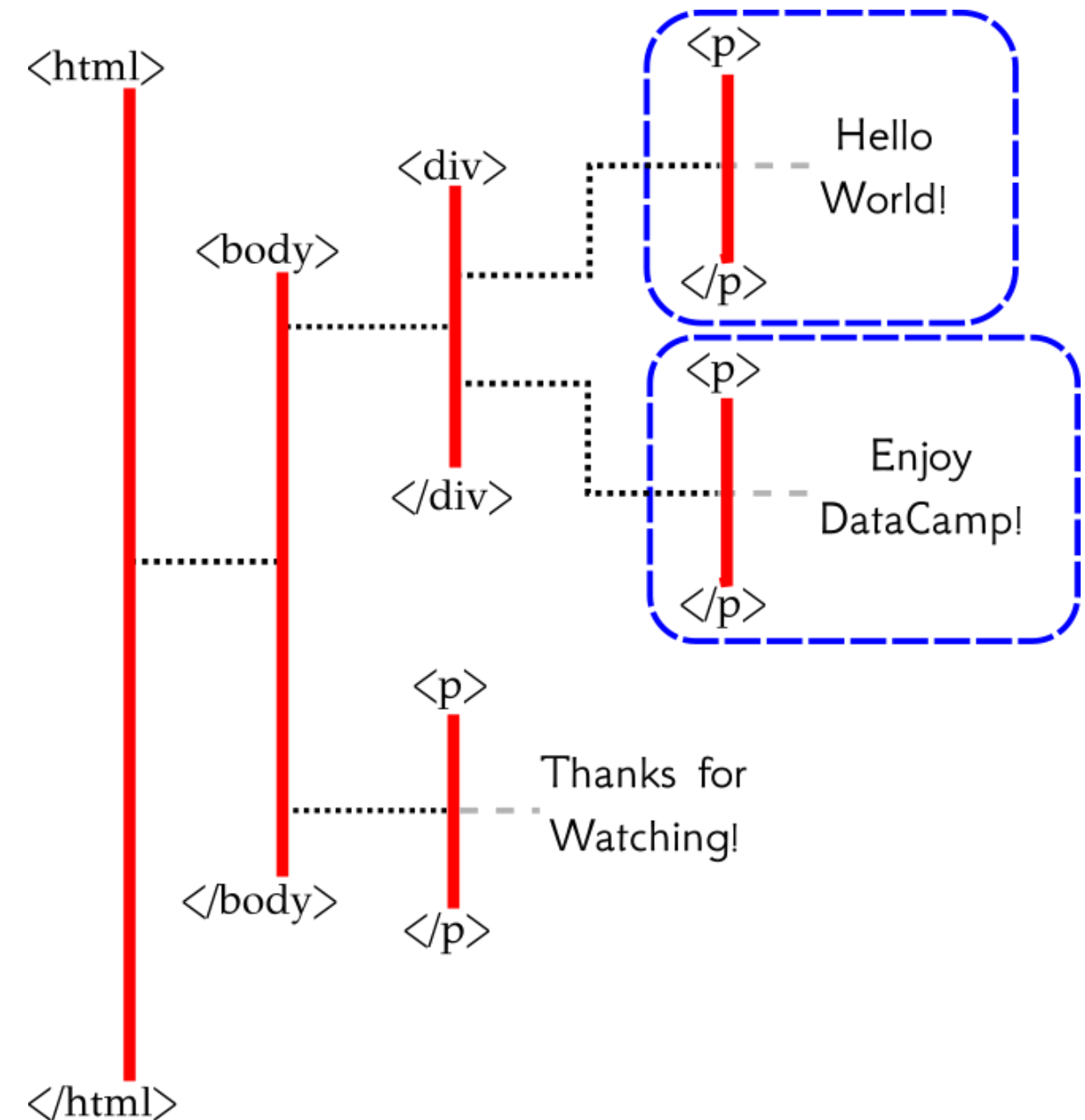


## 5. The HTML tree: Example 1

For example, the children of the body element are the same as the second generation descendants of the root html element; in order from top to bottom, div is the first child, p is the second. These two elements (the div and p elements) are siblings as they are both children of the same parent, the parent being the body element.

# The HTML tree: Example 2

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



## 6. The HTML tree: Example 2

The two third generation descendants of the html element are circled here. Both of these elements are paragraph p elements, and both are children of the same div element parent, and hence siblings. Note that these two circled paragraph elements are not descendants of the third, uncircled paragraph element. This is graphically represented by the fact that, although the two circled



# Introduction to HTML Outro

WEB SCRAPING IN PYTHON

# HTML Tags and Attributes

WEB SCRAPING IN PYTHON



**Thomas Laetsch**  
Data Scientist, NYU

# Do we have to?

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

## 2. Do we have to?

You may ask why we want to get more specific here? It turns out that sometimes we want to access information that is held within the HTML tags themselves -- we often want access this info in order to find the URL pointed to by a specific link on the site, or because it can potentially give us another method to select specific HTML elements with a more friendly syntax than traversing the entire HTML tree.

# 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.

## 3. Tag, you're it!

To start, let's look at an abstract tag formatting. There are many HTML tag types that follow the same formatting; we have already seen three tag names: the html, div, and p tags. These tags can also contain **\*\*attributes\*\*** which provide special instructions for the contents contained within that tag. Specific html attribute names are followed by an equals sign, followed by information which is being passed to that attribute within the tag; in well-formatted HTML the information is in quotes.

Sound confusing? Don't worry!

# 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

## 4. Let's "div"vy up the tag

To look at a specific example, let's consider a div tag with two attributes: id and class. We chose these two attributes here because they arise frequently in practice. In well-formatted HTML, the id attribute can be used as a unique identifier for the tag element; in this case, the id "unique-id" should only belong to this specific div element, giving us a quick way to identify it. The class attribute "some class" can also help us identify this div element, but even in well-formatted HTML, it doesn't need to be unique. No tag needs to have an id nor a class attribute, but all tags can be given an id and a class. A point that will find its way to a future lesson is that a tag can belong to multiple classes, and this is done when the class attribute (that is, the quoted text assigned for that class) has multiple class names separated by spaces; in fact, this div tag would belong to both classes: "some" and "class".

# "a" be linkin'

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

This text links to DataCamp!

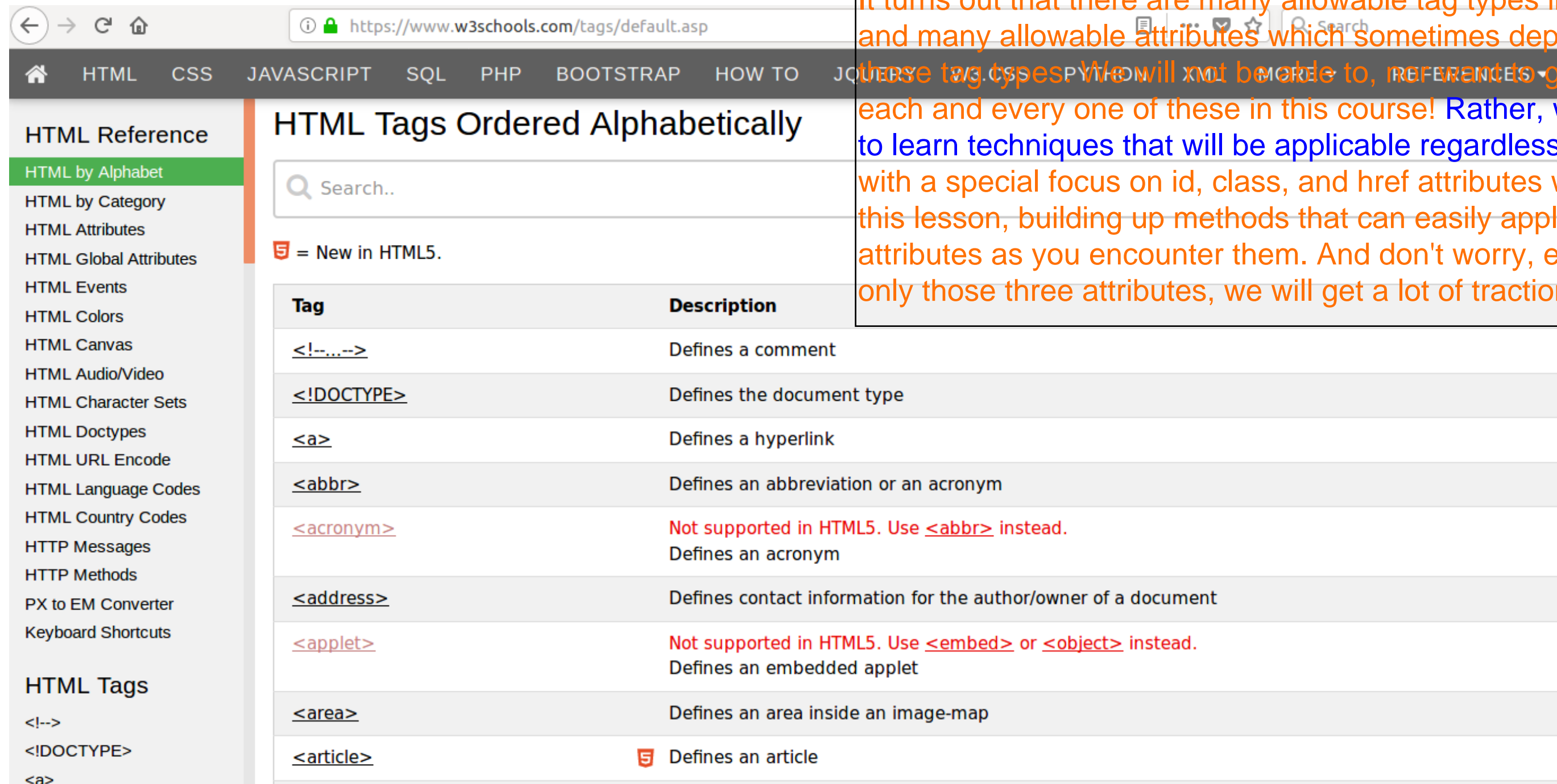
```
</a>
```


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

## 5. "a" be linkin'

Let's look at another example. The "a" tag-name here is the specific tag for hyperlinks, the links we click on within a website to redirect somewhere. The most important attribute within these hyperlink tags is the "href" attribute. This attribute is used to identify the URL where the hyperlink redirects to.

# Tag Traction

A screenshot of the W3Schools website showing the 'HTML Tags Ordered Alphabetically' page. The browser's address bar shows 'https://www.w3schools.com/tags/default.asp'. The navigation bar includes links for HTML, CSS, JAVASCRIPT, SQL, PHP, BOOTSTRAP, HOW TO, JQUERY, XML, CSS, PYTHON, and MORE. The left sidebar has a 'HTML Reference' section with 'HTML by Alphabet' highlighted, and a 'HTML Tags' section listing various tags. The main content area has a search bar and a table of HTML tags. A legend indicates that a red 'S' icon means 'New in HTML5'. The table lists tags from <!--> to <article>, with descriptions and notes on HTML5 support for some tags like <acronym> and <applet>.

Tag	Description
<a href="#"><u>&lt;!--...--&gt;</u></a>	Defines a comment
<a href="#"><u>&lt;!DOCTYPE&gt;</u></a>	Defines the document type
<a href="#"><u>&lt;a&gt;</u></a>	Defines a hyperlink
<a href="#"><u>&lt;abbr&gt;</u></a>	Defines an abbreviation or an acronym
<a href="#"><u>&lt;acronym&gt;</u></a>	Not supported in HTML5. Use <a href="#"><u>&lt;abbr&gt;</u></a> instead. Defines an acronym
<a href="#"><u>&lt;address&gt;</u></a>	Defines contact information for the author/owner of a document
<a href="#"><u>&lt;applet&gt;</u></a>	Not supported in HTML5. Use <a href="#"><u>&lt;embed&gt;</u></a> or <a href="#"><u>&lt;object&gt;</u></a> instead. Defines an embedded applet
<a href="#"><u>&lt;area&gt;</u></a>	Defines an area inside an image-map
<a href="#"><u>&lt;article&gt;</u></a>	 Defines an article

## 6. Tag Traction

It turns out that there are many allowable tag types in HTML, and many allowable attributes which sometimes depend on those tag types. We will not be able to, nor want to go through each and every one of these in this course! Rather, we're going to learn techniques that will be applicable regardless of the tag; with a special focus on id, class, and href attributes we saw in this lesson, building up methods that can easily apply to other attributes as you encounter them. And don't worry, even with only those three attributes, we will get a lot of traction!

# Et Tu, Attributes?

## WEB SCRAPING IN PYTHON

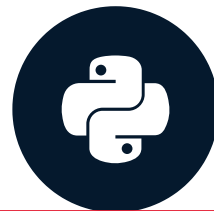
### 7. Et Tu, Attributes?

So, what did we learn in this lesson? We focused on HTML specific syntax, learning the general abstract structure of HTML tags. We saw how to identify the tag-name and attributes within those tags.



# Crash Course X

WEB SCRAPING IN PYTHON



**Thomas Laetsch**  
Data Scientist, NYU

## 1. Crash Course X

if we want to describe where these elements are within our programs (programs made to navigate and scrape HTML), then we need to build up a standard, program-friendly language or syntax to do so. This lesson will give a crash course in some basics of what's called XPath notation, one of two common choices for this purpose. And, in the next chapters, we will go deeper into both syntaxes with many more examples.

# Another Slasher Video?

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

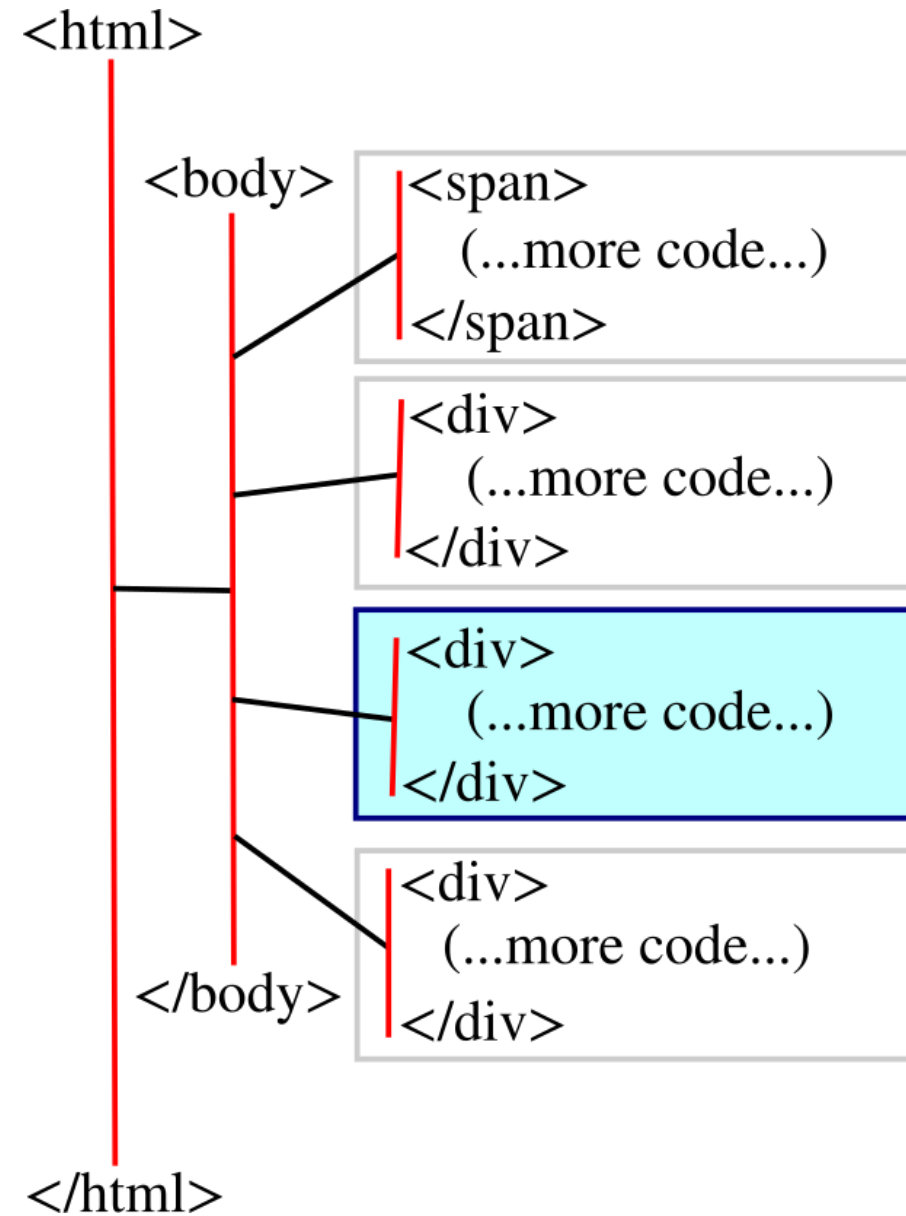
## Simple XPath:

- 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.

## 2. Another Slasher Video?

... The single forward-slash moves us forward one generation. In fact, if we think of the tag-names as the "directory" names, then these simple XPath expressions will look very much like navigating between directories. What might seem unfamiliar are the brackets. These brackets are used to help specify which element or elements we want to direct to. For example, there could be several `div` elements which are children of the `body` element (that is, several `div` siblings), so, we can use the brackets to narrow in on the `div` element we want.

# Another Slasher Video?



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

## 3. Another Slasher Video?

To illustrate the sample XPath string we wrote in the last slide, here we have highlighted the div element which would be selected within a tree representation of some HTML. Notice that the number 2 in the brackets of our XPath expression refers to the second div element of the three div elements (ordered from top to bottom as usual), paying attention to the fact that the first child of the body is a span element, so is not counted when looking at the div elements.

# 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'
```

## 4. Slasher Double Feature?

Using the double-forward slash tells us to "look forward to all future generations" (instead of just one generation like the single forward-slash). So, for example, we could navigate to all table elements within an HTML document by simply typing double forward-slash table. Or, we could want to restrict to a specific div element (say, the one we learned how to navigate to in the last couple slides), and navigate to all table elements which are descendants of that div element.

# Ex(path)celent

## WEB SCRAPING IN PYTHON

### 5. Ex(path)celent

And, that's it for now! We have only just scratched the surface of XPath notation, but we've gone deep enough that you can begin to write some code and get your feet wet navigating HTML computationally. Let me emphasize here that XPath is general, meaning that it is not python specific. So, if you decide to start scraping the web in R, say, most libraries there will also be able to read and interpret your XPath strings.