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** (*hypertext markup language*) for a web page.

You can think of the web page as being similar to a file. **HTML** is a file format that tells your browse how to render the web page.

HTML 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

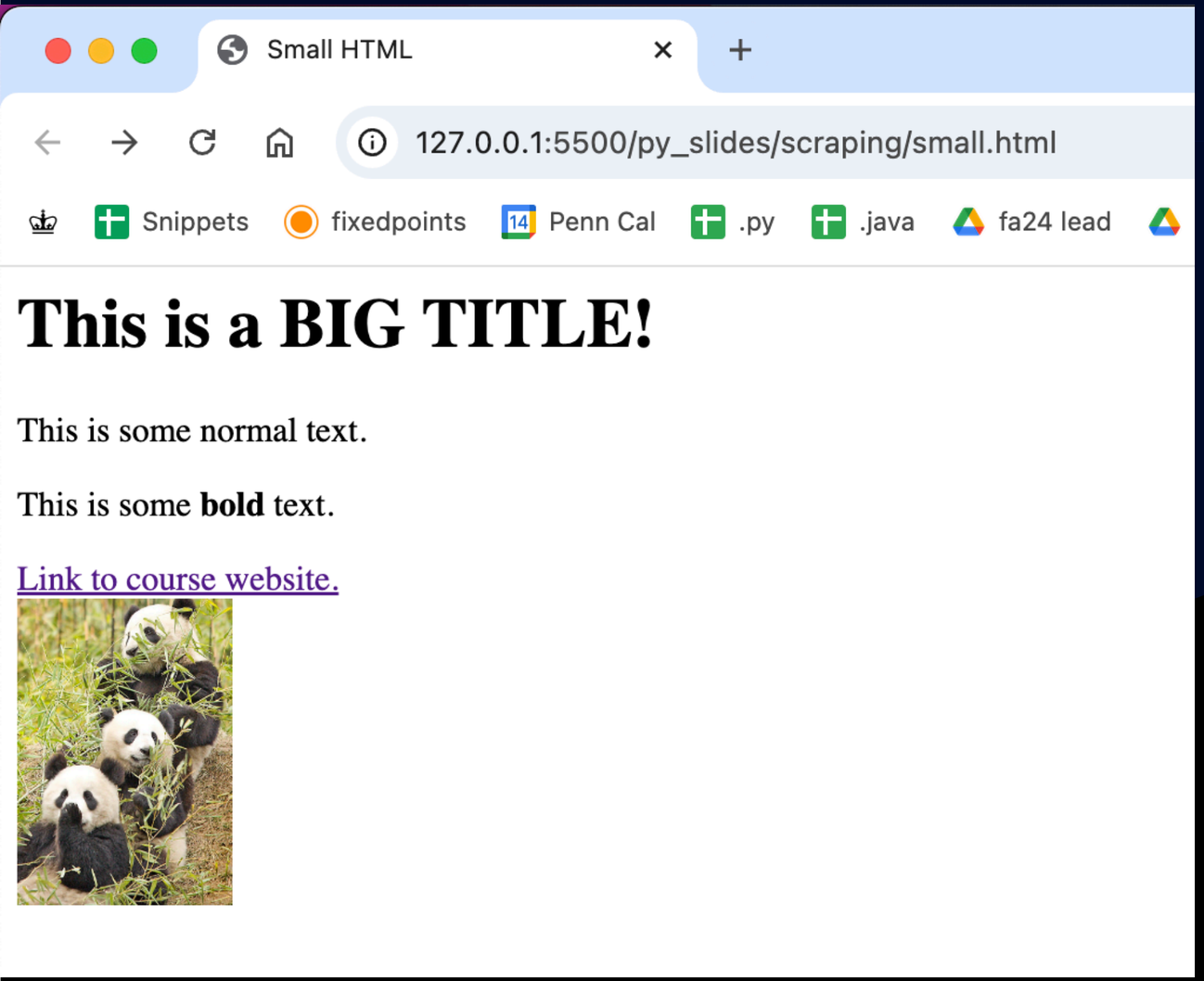
HTML is a language based on **tags**, which convey instructions about how the information inside of them should be handles & displayed.

- 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

```
<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 (Demo)



`example.html`:

```html
<h1>This is a BIG TITLE!</h1>
<!-- This is a comment. This is file is on the course website as example.html -->
<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:

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

Practice: Reading an HTML Site:

```
<html>
  <head><title>My Travel Blog</title></head>
  <body>
    <header>
      <h1>Adventures Around the World</h1>
      <p>Sharing stories from every corner of the globe.</p>
    </header>
    <h2>Recent Destinations</h2>
    <p>Here are a few places I've explored this year:</p>
    <ul>
      <li>Kyoto, Japan</li>
      <li>Lisbon, Portugal</li>
      <li>Marrakech, Morocco</li>
    </ul>
  </body>
</html>
```

(S7) Which text acts as the main heading of the page? What tag is it wrapped in?

(S8) There is a bulleted list on the site, how long is it?

(S9) I want to add a image (`"japan.png"`) before the list; which tag could you add to include one? Where would you put it?

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


```
<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

BeautifulSoup

Python
Fall 2024
University of Pennsylvania

Parsing through 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"
```

Basic Summary.

- `soup.<tag_name>` gives the first tag of that name.
- `<tag>.name` gives you the name of that tag
- `<tag>.string` gives you the contents of that tag

Practice (L11)

Given the site we saw earlier, and assuming we parse it into a variable called `soup`, fill in the blanks:

```
print(soup.                ) # what goes here to make it print "And it's perfect"
print(soup.                ) # what goes here to make it print "It's lightweight and loads fast:"
```

```
<html>
  <head><title>The Perfect Website</title></head>
  <body>
    <header>
      <h1>This is a website.</h1>
      <p>And it's perfect.</p>
    </header>
      <h2>Seriously, what else do you want?</h2>
      <p>Let me describe your perfect website:</p>
      <ul>
         <li>It's lightweight and loads fast</li>
         <li>Fits on all your screens</li>
         <li>Looks the same in all your browsers</li>
         <li>Accessible to every person that visits your site</li>
      </ul>
  </body>
</html>
```

Example: Traversing through HTML

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

`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.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.title.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>"]
```

Summary Extended:

- `soup.<tag_name>` gives the first tag of that name.
- `<tag>.name` gives you the name of that tag
- `<tag>.string` gives you the contents of that tag
- `<tag>["attribute_name"]` Tags can be treated like dictionaries where the attribute names are the keys.
- `soup.find_all("<tag_name>")` Retuns all tags that are of the specified type
  - `soup.find_all("<tag_name>", class_='<class_name>')` Returns all tags that are of the specified type and the specified class

Practice: (C12)

Assume we have parsed the below html into an object `soup`:

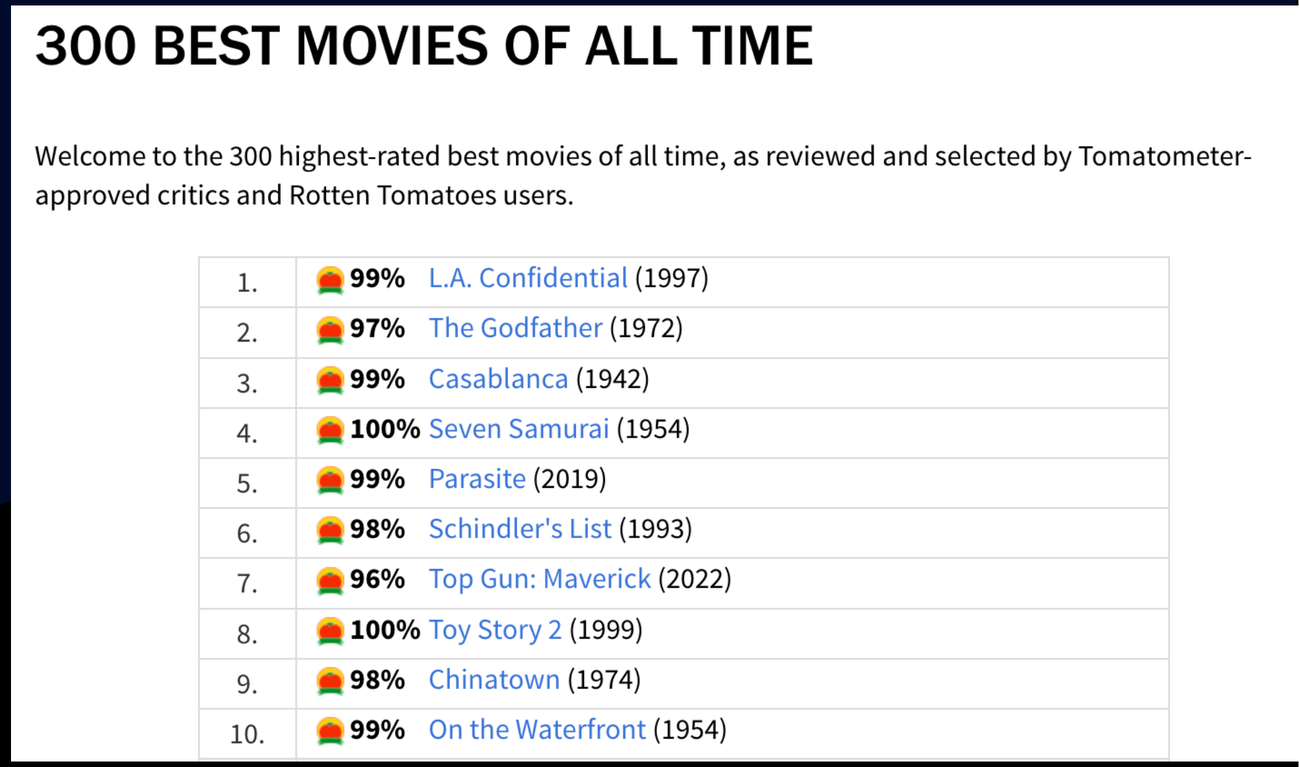
```
<html>
  <head><title>Joel's Book Club</title></head>
  <body>
    <p class="description">These are our featured books this month:</p>
    <p class="books">Check out:
      <a href="http://books.com/gatsby" class="book" id="book1">The Great Gatsby</a>,
      <a href="http://books.com/1984" class="book" id="book2">Slaughterhouse Five</a>,
      and
      <a href="http://books.com/bravenewworld" class="book" id="book3">Brave New World</a>.
      <a href="/" class="empty"/a>
    </p>
    <p class="books">More recommendations coming soon!</p>
  </body>
</html>
```

1. Get a list of all the `<a/>` book tags
2. Get a list of all the book titles
3. Get a list of the links e.g. (`["http://books.com/gatsby", ...]`)

Inspect Element

Scraping the `Rotten Tomatos`'s ranking for well...movies!

Let's go ahead and check out the website here



300 BEST MOVIES OF ALL TIME

Welcome to the 300 highest-rated best movies of all time, as reviewed and selected by Tomatometer-approved critics and Rotten Tomatoes users.

1.	**99%** L.A. Confidential (1997)
2.	**97%** The Godfather (1972)
3.	**99%** Casablanca (1942)
4.	**100%** Seven Samurai (1954)
5.	**99%** Parasite (2019)
6.	**98%** Schindler's List (1993)
7.	**96%** Top Gun: Maverick (2022)
8.	**100%** Toy Story 2 (1999)
9.	**98%** Chinatown (1974)
10.	**99%** On the Waterfront (1954)

credit: https://editorial.rottentomatoes.com/guide/best-movies-of-all-time/

Scraping Goal...

The end goal is to eventually create a csv with the name and ranking of the movies.

Rank	Score	Movie Title	Year
1	99%	L.A. Confidential	1997
2	97%	The Godfather	1972
3	99%	Casablanca	1942
4	100%	Seven Samurai	1954
5	99%	Parasite	2019

(L13) What is type of *tag* that contains the entire content of all movies? Are there multiple?

(L13) What is the type of *tag* that contains the name of a singular movie? What about the score?