CIS 1100

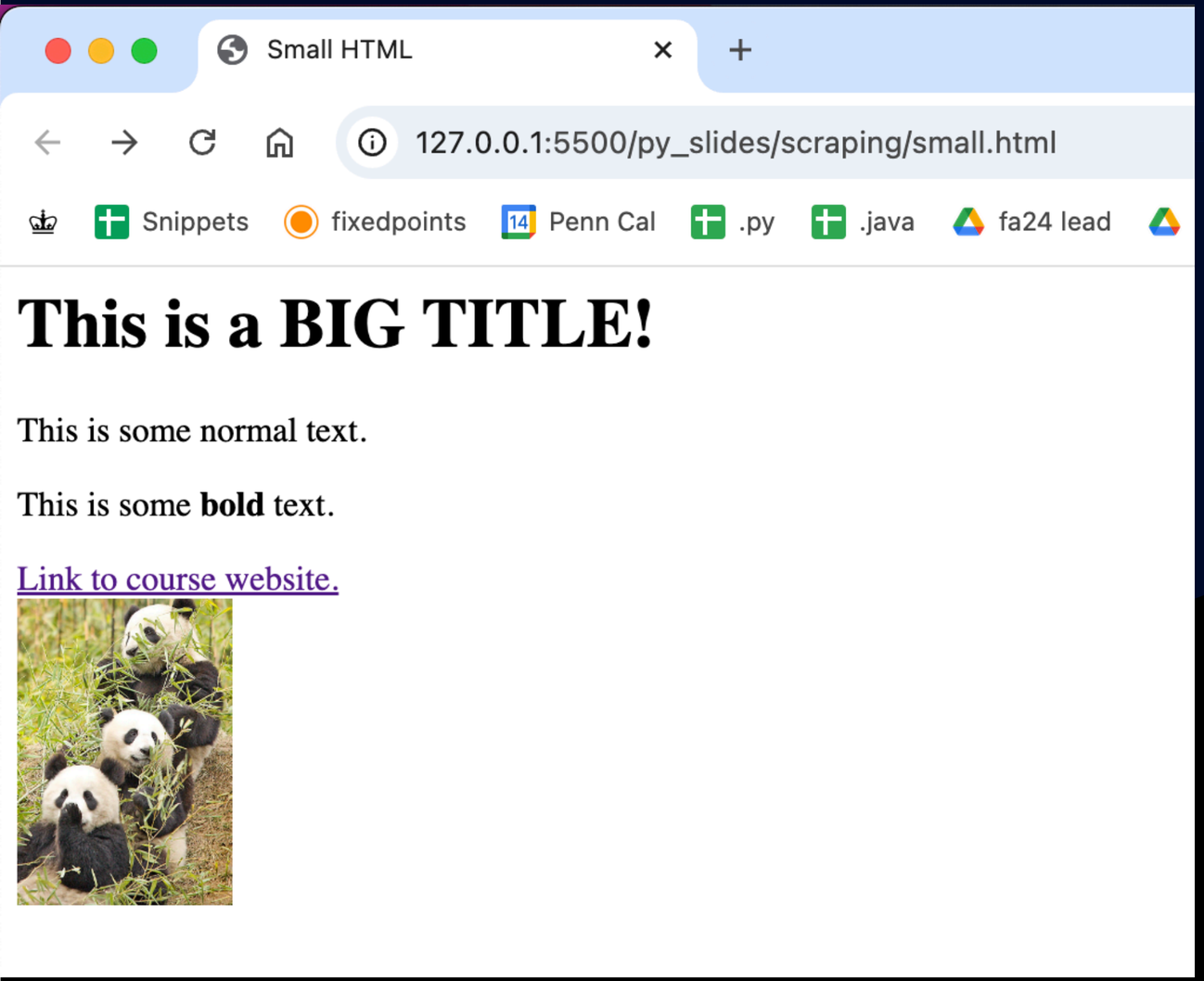
Scraping

Python

Fall 2024

University of Pennsylvania

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>
```

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')
```

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
- `<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: (L11)

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

```
<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>
```

3. Get a list of the links for each sister? e.g. (`["http://example.com/elsie", "http://example.com/lacie", ...]`)

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>
```

Example Pratice (C12)

See `simple_syllabus.html` linked from the course website as an example.

```
<table class="table table-striped">
        <tr><th>Date</th><th>Topics</th><th>Slides</th><th>Example Code</th><th>Due Dates</th></tr>
        <tr class="">
            <td>Wed, Aug 28, 2024</td>
            <td>Introduction</td>
            <td><a target="_blank" href="../intro.pdf">✏️</a></td>
            <td></td><td> </td>
        </tr>
        <tr class="">
            <td>Fri, Aug 30, 2024</td>
            <td>Hello, World!</td>
            <td><a target="_blank" href="../hello_world.pdf">🐭</a>
                <a target="_blank" href="../hello_world_lecture.pdf">✏️</a></td>
            <td><a target="_blank" href="./hello_world.py">hello_world.py</a><br /></td><td> </td>
        </tr>
```

1. get a list of all the row tags `['<tr><th>.....</tr>', '<tr><td>  ... ']`
2. list of all non-header_rows

Practice Part 2 (C14)

```
<table class="table table-striped">
        <tr><th>Date</th><th>Topics</th><th>Slides</th><th>Example Code</th><th>Due Dates</th></tr>
        <tr class="">
            <td>Wed, Aug 28, 2024</td>
            <td>Introduction</td>
            <td><a target="_blank" href="../intro.pdf">✏️</a></td>
            <td></td><td> </td>
        </tr>
        <tr class="">
            <td>Fri, Aug 30, 2024</td>
            <td>Hello, World!</td>
            <td><a target="_blank" href="../hello_world.pdf">🐭</a>
                <a target="_blank" href="../hello_world_lecture.pdf">✏️</a></td>
            <td><a target="_blank" href="./hello_world.py">hello_world.py</a><br /></td><td> </td>
        </tr>
```

From the previous step, get:

1. a list of all dates
2. a list of all lecture topics

Practice Part 3: (C16)

```
<tr class="">
    <td>Mon, Sep 9, 2024</td><td>Variables &amp;amp; Types</td>
    <td><a target="_blank" href="../datatypes.pdf">📽</a>
        <a target="_blank" href="../types_lecture2.pdf">✏️</a></td>
    <td></td><td> </td>
</tr>
<tr class="success">
    <td>Wed, Sep 11, 2024</td><td>Conditionals</td>
    <td><a target="_blank" href="../conditionals.pdf">📽</a>
        <a target="_blank" href="../conditionals_lecture.pdf">✏️</a></td>
    <td></td><td>HW00 @ 11:59pm </td>
</tr>
```

1. Get a list of all row tags, but only when an assignment is due

   (Is there a pattern you notice? One of the two rows above has a hw due)

2. Populate a dictionary that maps the date (string) to the HW due message (e.g. one of the entries should map `"Wed, Sep11, 2024"` to `"HW00 @ 11:59pm"`