CIS 1100

More BeautifulSoup and HTML

Python

Fall 2024

University of Pennsylvania

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"`, `width`, `height`
`ul`	Unordered (bulleted) list	
`ol`	Ordered (numbered) list	

Tag Name	Purpose	Attributes
`li`	List item (used inside `ul` or `ol`)	
`div`	Generic container for grouping	`class`, `id`, etc.
`span`	Inline container (small text blocks)	`class`, `id`, etc.
`table`	Table structure	
`tr`	Table row	
`td`	Table data	
`th`	Table header	
`strong`	Bold text	
`em`	Italicized (emphasized) text	

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>
```

- Tags can have **more than one class**, separated by spaces:

```
<p class="fancy highlighted">This is fancy and highlighted 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 cells 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>
```

BeautifulSoup + 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')
```

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

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>")` Returns 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

Summary Extended II:

- `tag.find_next_siblings(name, attrs)`
  Returns **all matching siblings** that appear after the current tag.
- `tag.find_previous_siblings(name, attrs)`
  Returns **all matching siblings** that appear before the current tag.
- `tag.find_all_next(name, attrs)`
  Searches **forward through the document** and returns **all matches** after this tag.
- `tag.find_all_previous(name, attrs)`
  Searches **backward through the document**, returning **all matching tags**.

Note: many of these methods have singular versions (e.g. `find_next()`)

Erata: what happened with `Reduce`?

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>
```

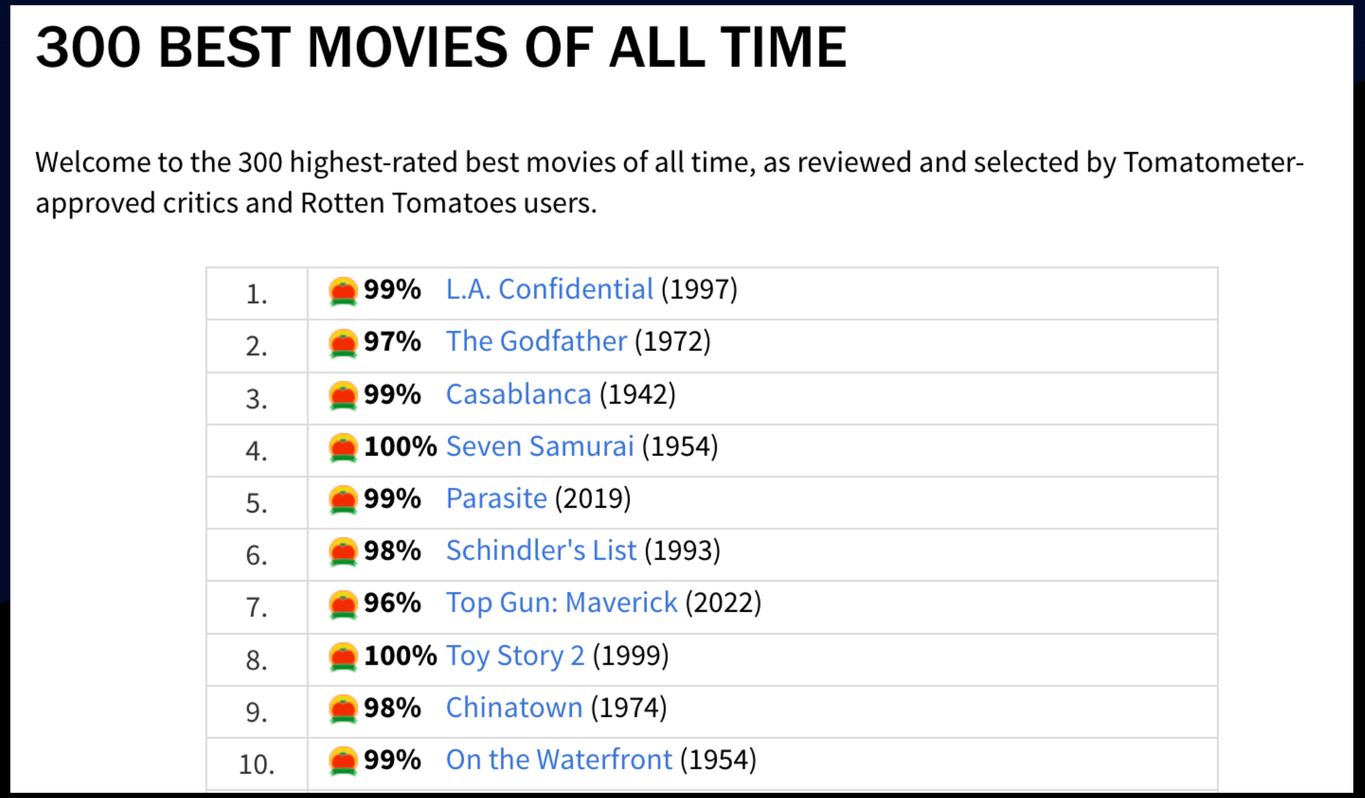
2. Get a list of all the book titles

```
# THIS IS WRONG!
reduce(lambda a, b: a.append(b.string), soup.find_all("a", class_="book"), [])
```

Inspect Element

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

Let's go ahead and check out the website.



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)

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

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

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

Example Pratice (L11)

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

```
<table class="aligncenter" style="width: 75%; padding: 8px">
  <tbody>
    <tr style="height: 23px; border: 1px solid #dddddd">
      <td style="width: 10%; height: 23px; text-align: center">1.</td>
      <td style="width: 500px; height: 23px; border: 1px solid #dddddd">
        <p class="apple-news-link-wrap movie">
          <span class="score-wrap">
            <img
              class="apple-critic-score-icon"
              src="https://images.fandango.com/cms/assets/c6672520-d359-11ea-a15f-bdf29fa24277--certified-fresh.png"
              width="16"
              height="16"
            />
            <span class="score"><strong>99%</strong></span>
          </span>

          <span class="details">
            <a class="title" href="https://www.rottentomatoes.com/m/la_confidential">L.A. Confidential</a>
            <span class="year">(1997)</span>
          </span>
        </p>
      </td>
    </tr>
```

1. Get a list of all `<tr>` tags
2. Get a list of all the first `<td>` in every row

Practice Part 2 (C12)

```
<table class="aligncenter" style="width: 75%; padding: 8px">
  <tbody>
    <tr style="height: 23px; border: 1px solid #dddddd">
      <td style="width: 10%; height: 23px; text-align: center">1.</td>
      <td style="width: 500px; height: 23px; border: 1px solid #dddddd">
        <p class="apple-news-link-wrap movie">
          <span class="score-wrap">
            <img
              class="apple-critic-score-icon"
              src="https://images.fandango.com/cms/assets/c6672520-d359-11ea-a15f-bdf29fa24277--certified-fresh.png"
              width="16"
              height="16"
            />
            <span class="score"><strong>99%</strong></span>
          </span>

          <span class="details">
            <a class="title" href="https://www.rottentomatoes.com/m/la_confidential">L.A. Confidential</a>
            <span class="year">(1997)</span>
          </span>
        </p>
      </td>
    </tr>
```

From the previous step, get:

1. Get a list of all movie titles from a single table
2. Get a list of all release years without parentheses (e.g. "1997")

Practice Part 3: (C14)

```
<table class="aligncenter" style="width: 75%; padding: 8px">
  <tbody>
    <tr style="height: 23px; border: 1px solid #dddddd">
      <td style="width: 10%; height: 23px; text-align: center">1.</td>
      <td style="width: 500px; height: 23px; border: 1px solid #dddddd">
        <p class="apple-news-link-wrap movie">
          <span class="score-wrap">
            <img
              class="apple-critic-score-icon"
              src="https://images.fandango.com/cms/assets/c6672520-d359-11ea-a15f-bdf29fa24277--certified-fresh.png"
              width="16"
              height="16"
            />
            <span class="score"><strong>99%</strong></span>
          </span>

          <span class="details">
            <a class="title" href="https://www.rottentomatoes.com/m/la_confidential">L.A. Confidential</a>
            <span class="year">(1997)</span>
          </span>
        </p>
      </td>
    </tr>
```

1. Get a list of all `<tr>` tags that have a `100%` rating
2. Extract the link (i.e. the `href` attribute) for each of those matching rows

Practice Part 4: (C16)

```
movies = [
    {
        "title": "L.A. Confidential",
        "year": 1997,
        "score": "99%",
        "link": "https://www.rottentomatoes.com/m/la_confidential"
    },
    # More movies here...
]
```

Save all the information you've scraped into a list of dictionaries,

where each dictionary represents a movie. Include all movies!

Saving Scraped Data

After organizing your scraped data into a list of dictionaries matching the format in the previous slide, you can save your newly scrapped information into a CSV to use later.

```
import pandas as pd

df = pd.DataFrame(movies)
df.to_csv("movies.csv", index=False)
```