CIS 1100

Requesting Websites

Python

Fall 2024

University of Pennsylvania

Review: From Last Time

Rotten Tomato Top 300 Movies:

```
<table class="aligncenter" style="width: 75%; padding: 8px">
    <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">
            <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>
```

Last lecture, we saw how to parse a table in HTML if we had the file downloaded on our computer. Today, we'll go ahead fetch and parse tables like this directly from the web instead of from a local file by using the `requests` library!

`requests`

`pip install requests` to get access to a library that allows you to:

- programmatically "visit" websites
- get responses (HTML) within your program
- do all kinds of advanced stuff like *upload information to servers* or *communicate with APIs*

The Very Very Very Basics

- `get("my.url.com")` queries the website at that URL and returns a `Response`
- A `Response` is a dense object that contains
  information about what the remote server "said"
  - response code: a number that indicates whether your request was processed properly
  - information about the data encoding
  - the text of the response, i.e. some/all the HTML (or JSON...)

A Minimal Request

```
import requests

url = "https://www.cis.upenn.edu/~cis110/current/py/homework/homework.html"
r = requests.get(url)
print(r)
```

🖨️ ▼

```
<Response [200]>
```

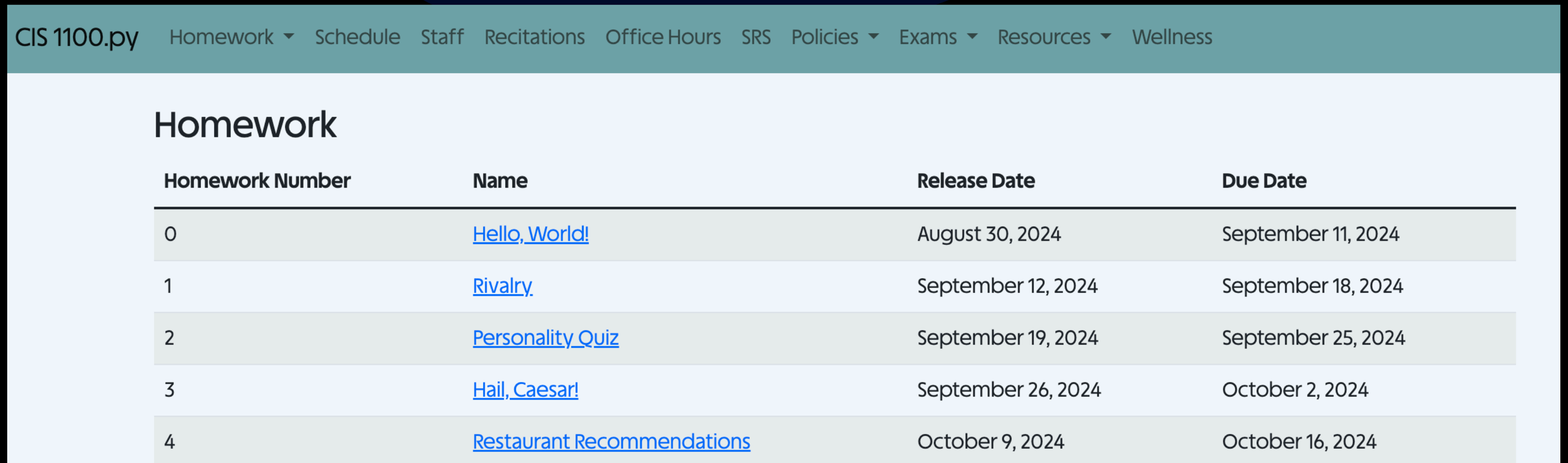
✅

A Minimal Request

```
import requests

url = "https://www.cis.upenn.edu/~cis110/current/py/homework/homework.html"
r = requests.get(url)
print(r.text)
```

`r.text` is just a string containing HTML, though. We know what to do with that...



A Minimal Request

```
import requests
from bs4 import BeautifulSoup

url = "https://www.cis.upenn.edu/~cis110/current/py/homework/homework.html"
r = requests.get(url)
soup = BeautifulSoup(r.text, 'html.parser')
links = soup.table.find_all('a')
print([link.text for link in links])
```

```
['Hello, World!', 'Rivalry', 'Personality Quiz', 'Hail, Caesar!', 'Restaurant Recommendations']
```

Practice: (L11)

Complete the snippet of code so that we download the HTML for the URL below and parse its contents into a BeautifulSoup `soup` object.

```
import requests
from bs4 import BeautifulSoup

url = "https://editorial.rottentomatoes.com/guide/best-movies-of-all-time/"


soup = BeautifulSoup(......)
```

Code from Last Time

Code snippet where we parsed a *single table*.

```
soup = BeautifulSoup(file, "html.parser")
rows = soup.find_all("tr")

movies = []
for elem in rows:
    movie = dict()
    movie["title"] = elem.a.string
    movie["year"] = elem.find("span", class_ = "year").string.strip()[1:5]
    movie["score"] = elem.strong.string
    movie["link"] = elem.a["href"]
    movies.append(movie)
#saving list of movie dictionaries as csv!
df = pd.DataFrame(movies)
df.to_csv("movies.csv", index = False)
```

Last time, we used this code to loop through a single movie table and save the data to a CSV file called `movies.csv`. If you're unclear about this code, ask a TA!

Practice (C12): Full HTML Document

```
<html lang="en-US" class="hitim">
<html>
    <head></head>
    <body>
        <table class="aligncenter" style="width: 75%; padding: 8px">
            <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"></td>
            </tr>
            <!-- More Movie Rows -->
        </table>
        <!-- More Movie Tables -->
    </body>
</html>
```

- Given the full HTML document:
  - Put all the tables that contain movies into a list. *Hint: do they have a specific attribute?*
  - Modify last lecture's code (see next slide), loop through all the movies on the site and save them to a CSV file.

Extra: Why request every time?

If we need to get HTML from hundreds of websites, making a new request each time can be slow and inefficient.

Wouldn't it be easier if we could just save the HTML once and reuse it?

```
url = "www.someurlhere.com"
response = requests.get(url)
html_text = response.text

file = open("myhtml.html", "w")
file.write(html_text)
file.close()
```

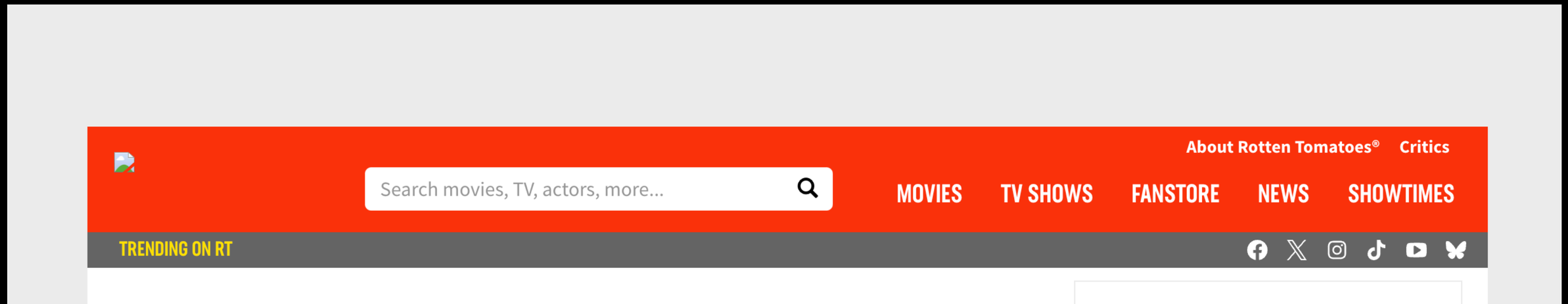
- Instead of requesting the HTML again, next time, you can just open and read the contents of the file `myhtml.html`.

Extra: Why does my HTML look different?

Sometimes the HTML you get from `requests` looks very different from what you see in your browser. Why is that?

Your browser does a lot more than just load HTML — it also
- Runs `JavaScript`, which can change the page after it's loaded.
- It also pulls in extra content from other sources (like ads or pop-ups).

When you use `requests`, you're only getting the raw HTML; and that's why there's no ads in the HTML even if you open it using a browser.



Extra: What do we do if the website is bare?

Sometimes a website's HTML looks almost empty
because most of its content is loaded dynamically.

In these cases, `requests` and BeautifulSoup won’t be enough.

The solution is to use the `Selenium` library, which gives you a *webdriver*—a tool that can simulate everything a real browser does.

With Selenium, you can simulate things like:

- Page navigation
- Clicking buttons and links
- Filling out forms
- Waiting for elements to load
- Running JavaScript and seeing the updated page