https://web.archive.org/web/20121007172955/http://www.nga.gov/collection/anZ1.htm

http://www.imdb.com/

https://www.metacritic.com/

pip install bs4

or pip install beautifulsoup4

pip install requests

**The components of a web page**

When we visit a web page, our web browser makes a request to a web server. This request is called a GET request, since we’re getting files from the server. The server then sends back files that tell our browser how to render the page for us. The files fall into a few main types:

* HTML — contain the main content of the page.
* CSS — add styling to make the page look nicer.
* JS — Javascript files add interactivity to web pages.

## The requests library

The first thing we’ll need to do to scrape a web page is to download the page. We can download pages using the Python requests library. The requests library will make a GETrequest to a web server, which will download the HTML contents of a given web page for us. There are several different types of requests we can make using requests, of which GET is just one.

import requests

page = requests.get("url")

page

page.status\_code : 200

page.content

## Parsing a page with BeautifulSoup

As you can see above, we now have downloaded an HTML document.

We can use the BeautifulSoup library to parse this document, and extract the text from the p tag. We first have to import the library, and create an instance of the BeautifulSoup class to parse our document:

from bs4

import BeautifulSoup

soup = BeautifulSoup(page.content, 'html.parser')

soup.find\_all(‘p’)

print(soup.prettify())

As all the tags are nested, we can move through the structure one level at a time. We can first select all the elements at the top level of the page using the children property of soup. Note that children returns a list generator, so we need to call the list function on it:

list(soup.children)

p = list(body.children)[1]

p.get\_text()

## Finding all instances of a tag at once

What we did above was useful for figuring out how to navigate a page, but it took a lot of commands to do something fairly simple. If we want to extract a single tag, we can instead use the find\_all method, which will find all the instances of a tag on a page.

soup = BeautifulSoup(page.content, 'html.parser')

soup.find\_all('p')

soup.find\_all('p')[0].get\_text()

## Searching for tags by class and id

We introduced classes and ids earlier, but it probably wasn’t clear why they were useful. Classes and ids are used by CSS to determine which HTML elements to apply certain styles to. We can also use them when scraping to specify specific elements we want to scrape. To illustrate this principle, we’ll work with the following page:

soup.find\_all('p', class\_='outer-text')

soup.find\_all(class\_="outer-text")

soup.find\_all(id="first")

Downloading weather data

page = requests.get("http://forecast.weather.gov/MapClick.php?lat=37.7772&lon=-122.4168")

soup = BeautifulSoup(page.content, 'html.parser')

seven\_day = soup.find(id="seven-day-forecast")

forecast\_items = seven\_day.find\_all(class\_="tombstone-container")

tonight = forecast\_items[0]

print(tonight.prettify())

**Extracting information from the page**

As you can see, inside the forecast item tonight is all the information we want. There are 4pieces of information we can extract:

* The name of the forecast item — in this case, Tonight.
* The description of the conditions — this is stored in the title property of img.
* A short description of the conditions — in this case, Mostly Clear.
* The temperature low — in this case, 49 degrees.

We’ll extract the name of the forecast item, the short description, and the temperature first, since they’re all similar:

page = requests.get("http://forecast.weather.gov/MapClick.php?lat=37.7772&lon=-122.4168")

soup = BeautifulSoup(page.content, 'html.parser')

seven\_day = soup.find(id="seven-day-forecast")

forecast\_items = seven\_day.find\_all(class\_="tombstone-container")

tonight = forecast\_items[0]

print(tonight.prettify())

img = tonight.find("img")

desc = img['title']

print(desc)

**Extracting all the information from the page**

Now that we know how to extract each individual piece of information, we can combine our knowledge with css selectors and list comprehensions to extract everything at once.

In the below code, we:

* Select all items with the class period-name inside an item with the class tombstone-container in seven\_day.
* Use a list comprehension to call the get\_text method on each BeautifulSoup object.

period\_tags = seven\_day.select(".tombstone-container .period-name")

periods = [pt.get\_text() for pt in period\_tags]

periods

short\_descs = [sd.get\_text() for sd in seven\_day.select(".tombstone-container .short-desc")]

temps = [t.get\_text() for t in seven\_day.select(".tombstone-container .temp")]

descs = [d["title"] for d in seven\_day.select(".tombstone-container img")]print(short\_descs)print(temps)print(descs)

## Combining our data into a Pandas Dataframe

We can now combine the data into a Pandas DataFrame and analyze it. A DataFrame is an object that can store tabular data, making data analysis easy. If you want to learn more about Pandas, check out our free to start course here.

In order to do this, we’ll call the DataFrame class, and pass in each list of items that we have. We pass them in as part of a dictionary. Each dictionary key will become a column in the DataFrame, and each list will become the values in the column:

import pandas as pd

weather = pd.DataFrame({

"period": periods,

"short\_desc": short\_descs,

"temp": temps,

"desc":descs

})

Weather

Example 2: Read data from website and write to file

import requests

import csv

from bs4 import BeautifulSoup

page = requests.get('https://web.archive.org/web/20121007172955/http://www.nga.gov/collection/anZ1.htm')

soup = BeautifulSoup(page.text, 'html.parser')

last\_links = soup.find(class\_='AlphaNav')

last\_links.decompose()

# Create a file to write to, add headers row

f = csv.writer(open('z-artist-names.csv', 'w'))

f.writerow(['Name', 'Link'])

artist\_name\_list = soup.find(class\_='BodyText')

artist\_name\_list\_items = artist\_name\_list.find\_all('a')

for artist\_name in artist\_name\_list\_items:

names = artist\_name.contents[0]

links = 'https://web.archive.org' + artist\_name.get('href')

# Add each artist’s name and associated link to a row

f.writerow([names, links])

Example 3: Get movie data

## Scraping data for over 2000 movies

We want to analyze the distributions of IMDB and Metacritic movie ratings to see if we find anything interesting. To do this, we’ll first scrape data for over 2000 movies.

It’s essential to identify the goal of our scraping right from the beginning. Writing a scraping script can take a lot of time, especially if we want to scrape more than one web page. We want to avoid spending hours writing a script which scrapes data we won’t actually need.

from requests import get

url = 'http://www.imdb.com/search/title?release\_date=2017&sort=num\_votes,desc&page=1'

response = get(url)

print(response.text[:500])

from bs4 import BeautifulSoup

html\_soup = BeautifulSoup(response.text, 'html.parser')

type(html\_soup)

movie\_containers = html\_soup.find\_all('div', class\_ = 'lister-item mode-advanced')

print(type(movie\_containers))

print(len(movie\_containers))

## Extracting the data for a single movie

We can access the first container, which contains information about a single movie, by using list notation on movie\_containers.

# Lists to store the scraped data in

names = []

years = []

imdb\_ratings = []

metascores = []

votes = []

# Extract data from individual movie container

for container in movie\_containers:

# If the movie has Metascore, then extract:

if container.find('div', class\_ = 'ratings-metascore') is not None:

# The name

name = container.h3.a.text

names.append(name)

# The year

year = container.h3.find('span', class\_ = 'lister-item-year').text

years.append(year)

# The IMDB rating

imdb = float(container.strong.text)

imdb\_ratings.append(imdb)

# The Metascore

m\_score = container.find('span', class\_ = 'metascore').text

metascores.append(int(m\_score))

# The number of votes

vote = container.find('span', attrs = {'name':'nv'})['data-value']

votes.append(int(vote))

import pandas as pd

test\_df = pd.DataFrame({'movie': names,

'year': years,

'imdb': imdb\_ratings,

'metascore': metascores,

'votes': votes

})

print(test\_df.info())

test\_df