Scraping the Web

kgold for 11/3

HTML

- "HyperText Markup Language": **markup** as opposed to **programming** (instructions for rendering text, but not general programs)
- Instructions that aren't supposed to be rendered appear in **tags** - often these surround text to indicate which text is affected
    - <b>Bold</b>
    - <i>Italics</i>
    - <a href="http://www.google.com">Anchor tags for links</a>

Structure of a classic minimal web page

```
<html>
<head>
<title> My amazing webpage!</title>
</head>
<body>
<p>Some text for the page; p stands for paragraph.</p>

<p>Next paragraph, let's link <a href="http://www.google.com">somewhere.</a>

</body>
</html>
```

Moving ahead in time: CSS

- Rather than attempt to do everything in HTML, web designers delineate logical containers for content, and give rules for rendering it elsewhere
    - The containers are often <div> tags with named classes; the style rules affect the named classes
    - The style rules are kept in a separate .css file that says how to render everything
- This makes it difficult for an automated parser to reason about where text is throughout the page, without actually rendering the page

Sample page snippet using CSS

```html
<body>

  <!-- Primary Page Layout
  –––––––––––––––––––––––––––––––––––––––––––––––––– -->
  <div class="container">
    <div class="row">
      <div class="main-container column">
        <div class="logo-wrapper">
          <a href="index.html"><img src="images/logo.png" /></a>
        </div>
        <nav>
          <ul>
            <li><a href="about.html">About</a></li>
            <li><a href="publications.html">Publications</a></li>
            <li><a href="games.html">Games</a></li>
          </ul>
        </nav>
```

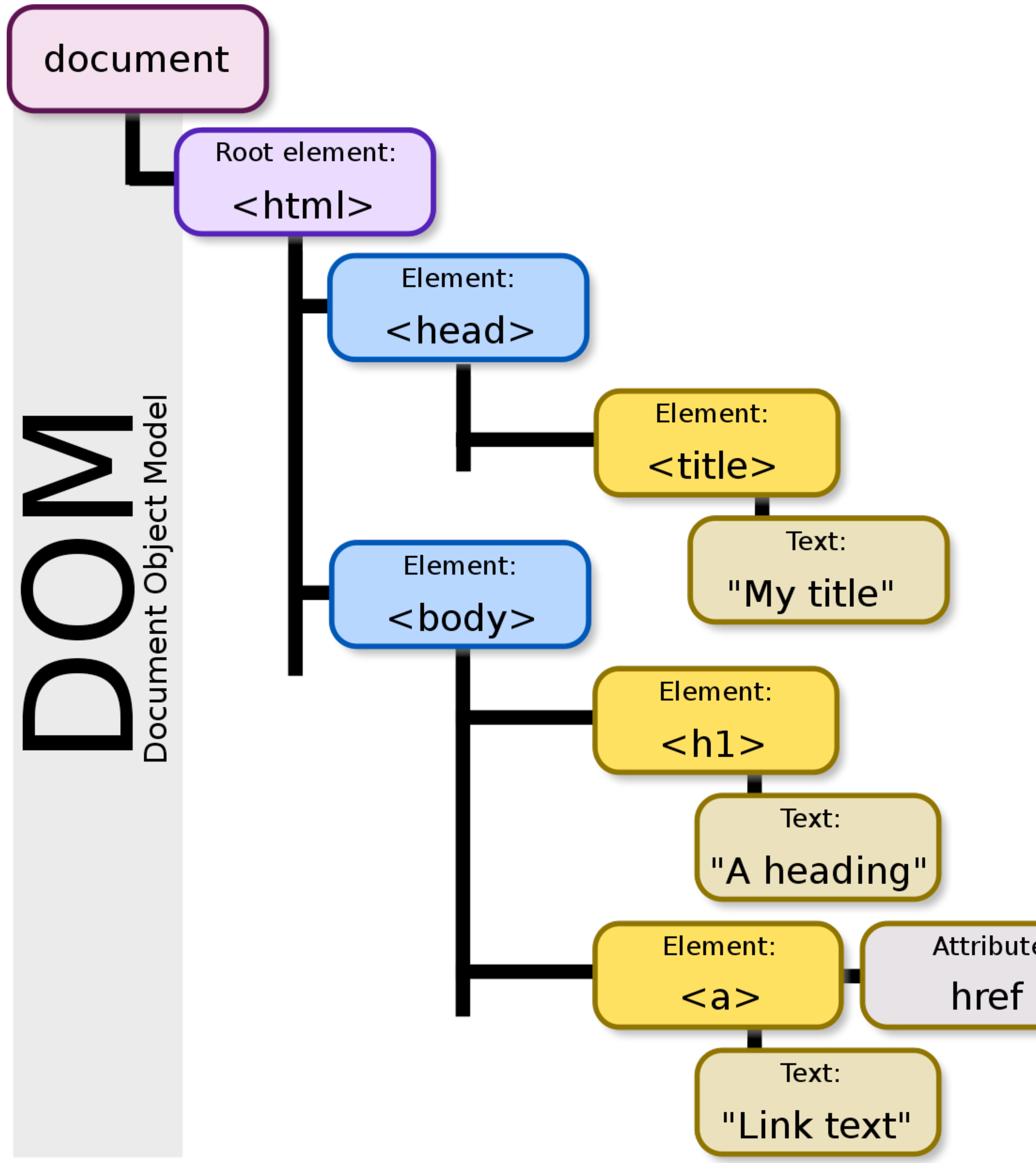
HTML fragment

```css
nav {
  background: rgba(131, 120, 108, 0.7);
}
nav ul {
  list-style: none;
  text-align: center;
  padding: 0;
  margin: 0;
}
nav li {
  display: inline-block;
  margin: 0;
}
nav a {
  color: #fff;
  text-decoration: none;
  padding: 1rem;
  display: block;
}
```

CSS fragment

JavaScript and the DOM

- The more interactive webpages became, the more they used stronger programming languages to be interactive - JavaScript is most common
- These languages can require execution to build out the webpage - the page could **no longer be readable by a scraping bot**
- JavaScript interacts with a tree-structured model of the page called the DOM: Document Object Model
    - Parts of the tree could be dynamically generated or deleted in response to user actions



Getting a Webpage

- Unlike proprietary APIs, webpages are served to everyone without needing permission first
    - Although some have policies against "robots" visiting them, and possibly even safeguards against them
- Pages are served on port 80, where the server listens for GET requests and hands over the appropriate content
- The HTML and any accompanying CSS and JavaScript are served as text for the browser to interpret
    - Images, audio, and video could also be served as binary files

Getting a Webpage with requests

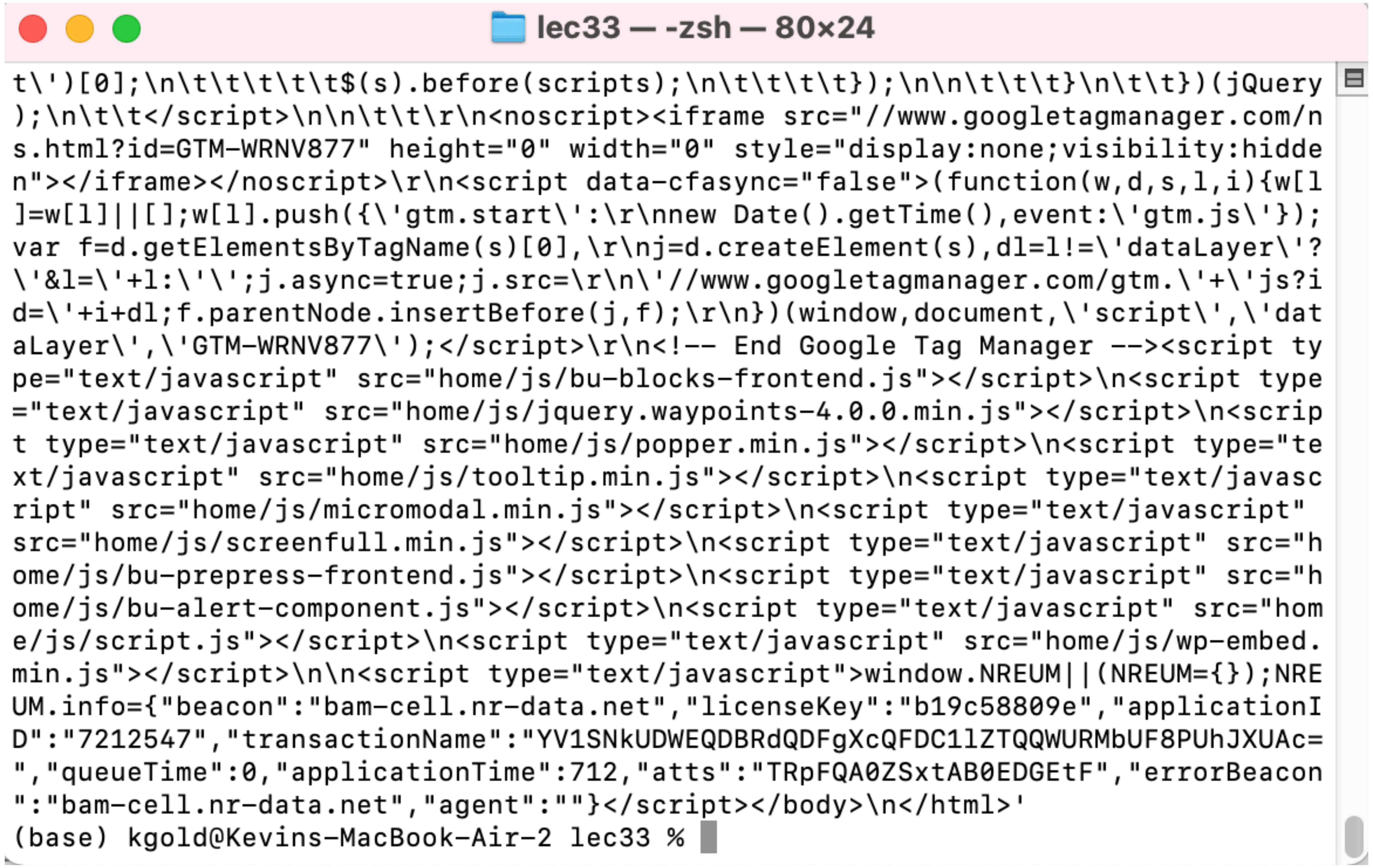
- requests: An easy module for getting the text of webpages

Included in Colab

```
import requests
page = requests.get('http://www.bu.edu')
contents = page.content
```

contents is HTML as string

What you get in 2021 (www.bu.edu)



lec33 — -zsh — 80×24

```
t\')[0];\n\t\t\t\t\t$(s).before(scripts);\n\t\t\t\t});\n\n\t\t\t}\n\t\t})(jQuery
);\n\t\t</script>\n\n\t\t\r\n<noscript><iframe src="//www.googletagmanager.com/n
s.html?id=GTM-WRNV877" height="0" width="0" style="display:none;visibility:hidde
n"></iframe></noscript>\r\n<script data-cfasync="false">(function(w,d,s,l,i){w[l
]=w[l]||[];w[l].push({\'gtm.start\':\r\nnew Date().getTime(),event:\'gtm.js\'});
var f=d.getElementsByTagName(s)[0],\r\nj=d.createElement(s),dl=l!=\'dataLayer\'?
\'&l=\'+l:\'\';j.async=true;j.src=\r\n\'//www.googletagmanager.com/gtm.\'+\'js?i
d=\'+i+dl;f.parentNode.insertBefore(j,f);\r\n})(window,document,\'script\',\'dat
aLayer\',\'GTM-WRNV877\');</script>\r\n<!-- End Google Tag Manager --><script ty
pe="text/javascript" src="home/js/bu-blocks-frontend.js"></script>\n<script type
="text/javascript" src="home/js/jquery.waypoints-4.0.0.min.js"></script>\n<scrip
t type="text/javascript" src="home/js/popper.min.js"></script>\n<script type="te
xt/javascript" src="home/js/tooltip.min.js"></script>\n<script type="text/javasc
ript" src="home/js/micromodal.min.js"></script>\n<script type="text/javascript"
src="home/js/screenfull.min.js"></script>\n<script type="text/javascript" src="h
ome/js/bu-prepress-frontend.js"></script>\n<script type="text/javascript" src="h
ome/js/bu-alert-component.js"></script>\n<script type="text/javascript" src="hom
e/js/script.js"></script>\n<script type="text/javascript" src="home/js/wp-embed.
min.js"></script>\n\n<script type="text/javascript">window.NREUM||(NREUM={});NRE
UM.info={"beacon":"bam-cell.nr-data.net","licenseKey":"b19c58809e","applicationI
D":"7212547","transactionName":"YV1SNkUDWEQDBRdQDFgXcQFDC1lZTQQWURMbUF8PUhJXUAc=
","queueTime":0,"applicationTime":712,"atts":"TRpFQA0ZSxtAB0EDGEtF","errorBeacon
":"bam-cell.nr-data.net","agent":""}</script></body>\n</html>'
(base) kgold@Kevins-MacBook-Air-2 lec33 %
```

Other features of the requests module

- Access sites needing authentication (if you have access)

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
```

- Pass in arguments to website (like search query) as a dict

```
>>> payload = {'key1': 'value1', 'key2': ['value2', 'value3']}

>>> r = requests.get('https://httpbin.org/get', params=payload)
>>> print(r.url)
https://httpbin.org/get?key1=value1&key2=value2&key2=value3
```

- Retrieve binary data, not just webpages

```
from PIL import Image
from io import BytesIO

i = Image.open(BytesIO(r.content))
```

Beautiful Soup 4

- This module is for parsing HTML, retrieving what text it can find and reconstructing the tree
- Two basic use cases are:
    - Retrieving all outgoing links
    - Retrieving all the text meant for human consumption

Creating the BeautifulSoup (tree) with a parser

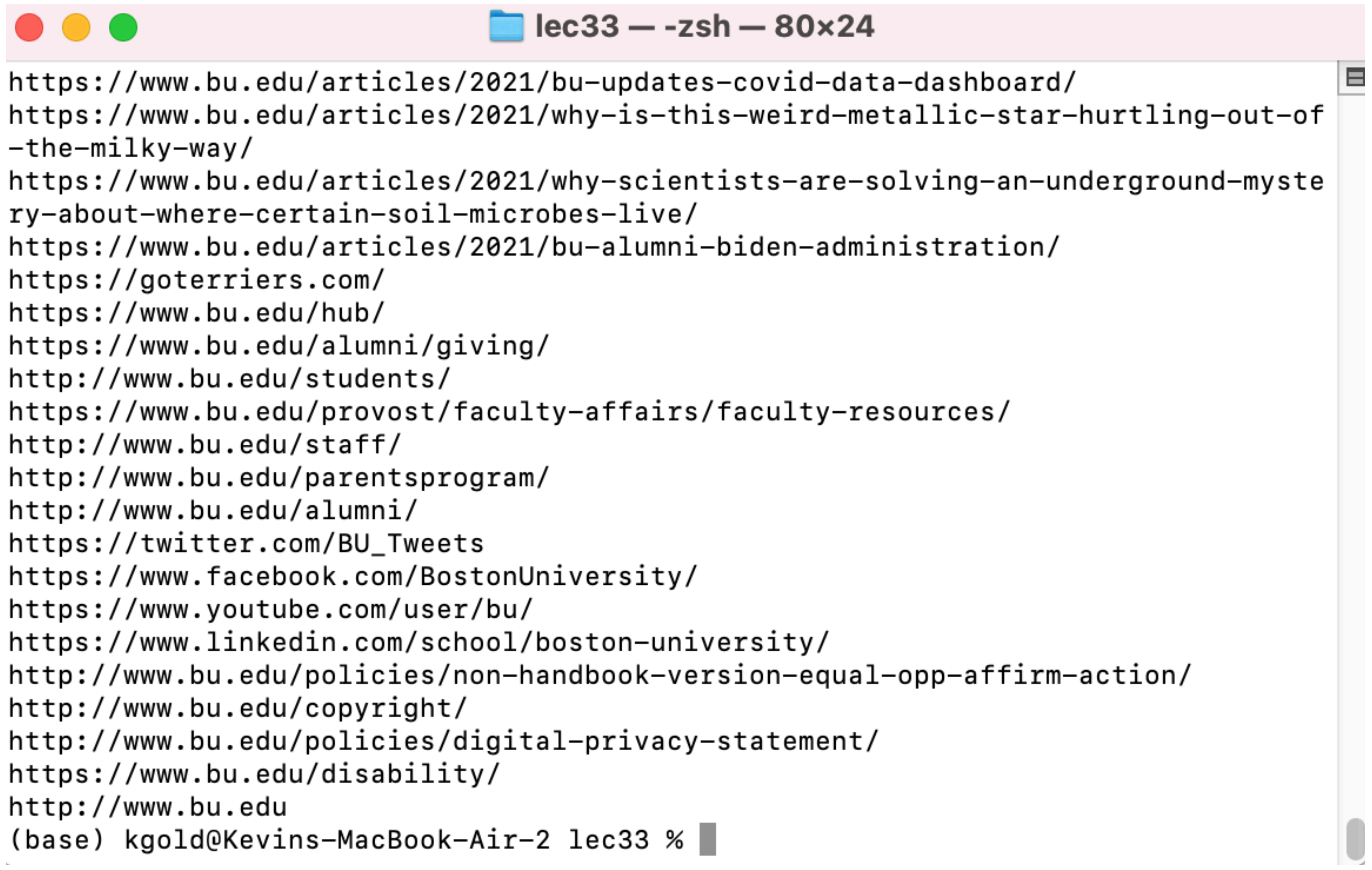
Included in colab

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')

print(soup.prettify())
# <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 class="sister" href="http://example.com/elsie" id="link1">
#     Elsie
#    </a>
#
```

Finding all the links

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

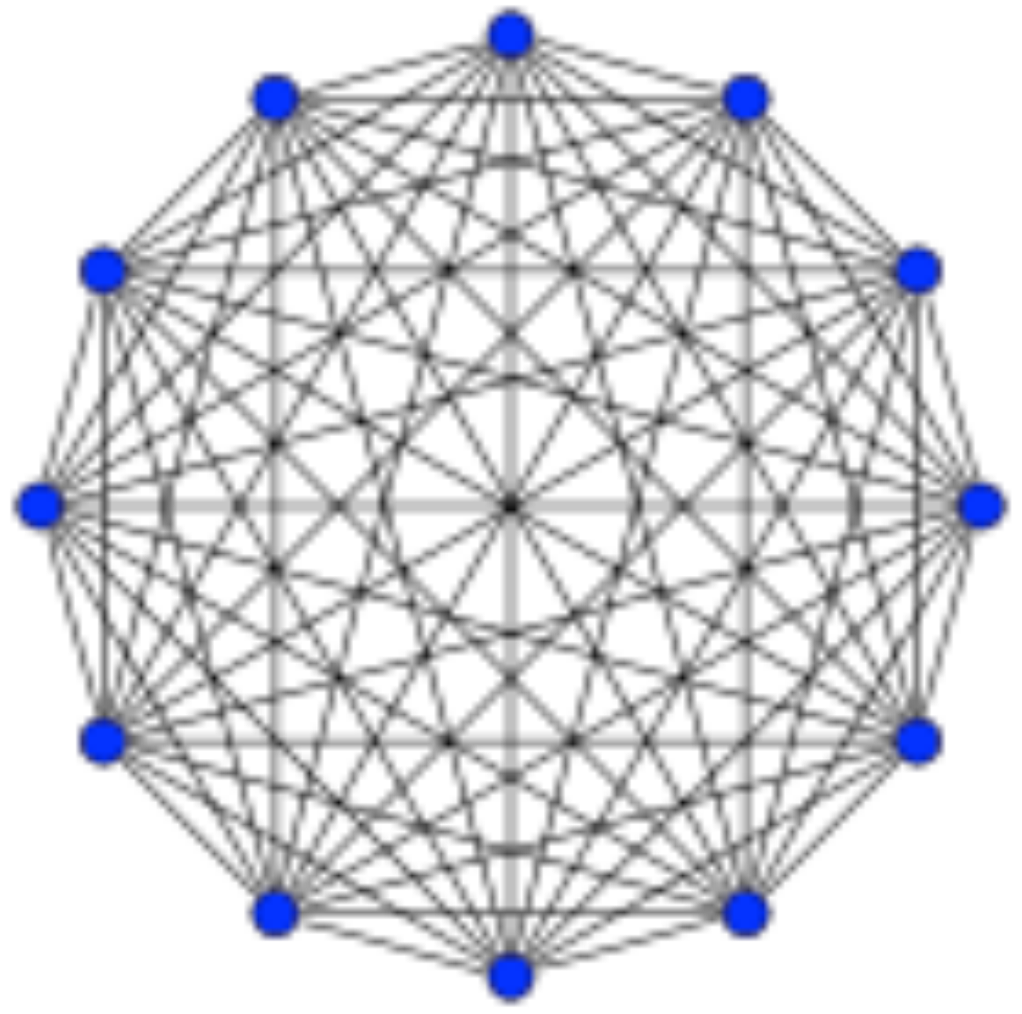


```
https://www.bu.edu/articles/2021/bu-updates-covid-data-dashboard/
https://www.bu.edu/articles/2021/why-is-this-weird-metallic-star-hurtling-out-of
-the-milky-way/
https://www.bu.edu/articles/2021/why-scientists-are-solving-an-underground-myste
ry-about-where-certain-soil-microbes-live/
https://www.bu.edu/articles/2021/bu-alumni-biden-administration/
https://goterriers.com/
https://www.bu.edu/hub/
https://www.bu.edu/alumni/giving/
http://www.bu.edu/students/
https://www.bu.edu/provost/faculty-affairs/faculty-resources/
http://www.bu.edu/staff/
http://www.bu.edu/parentsprogram/
http://www.bu.edu/alumni/
https://twitter.com/BU_Tweets
https://www.facebook.com/BostonUniversity/
https://www.youtube.com/user/bu/
https://www.linkedin.com/school/boston-university/
http://www.bu.edu/policies/non-handbook-version-equal-opp-affirm-action/
http://www.bu.edu/copyright/
http://www.bu.edu/policies/digital-privacy-statement/
https://www.bu.edu/disability/
http://www.bu.edu
(base) kgold@Kevins-MacBook-Air-2 lec33 %
```

Finding all the links

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

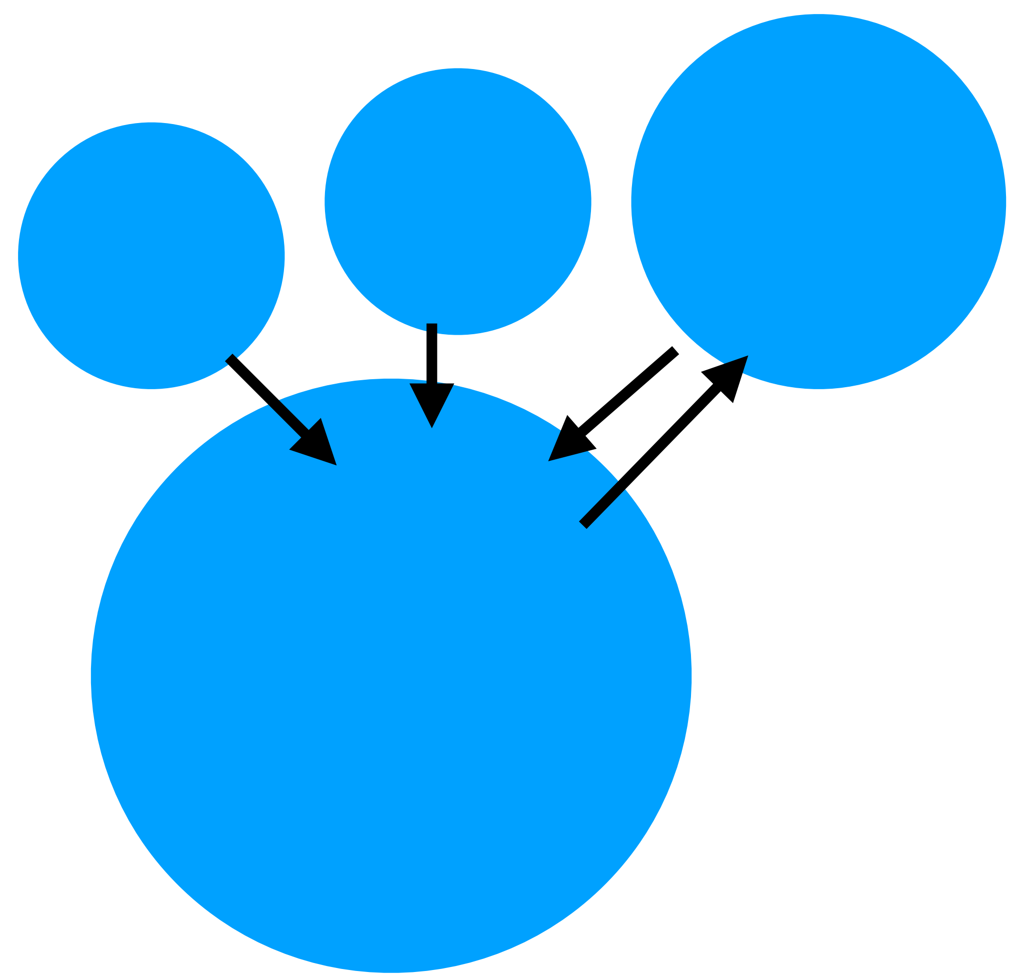
- Sample applications:
  - Find highly interconnected networks of pages that form "communities" to understand a page's ecosystem



Finding all the links

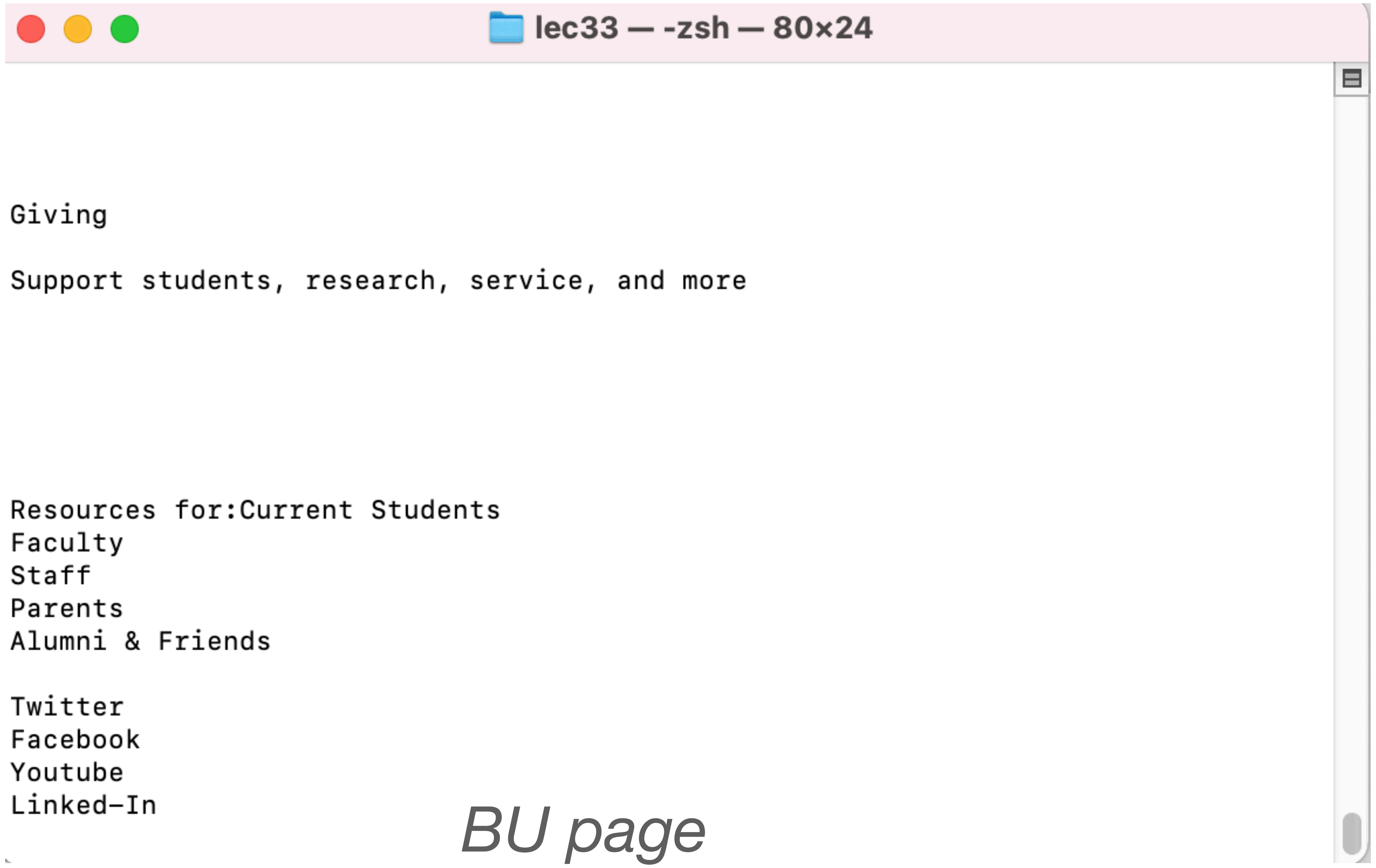
```
for link in soup.find_all('a'):
    print(link.get('href'))
```

- Sample applications:
  - Find highly interconnected networks of pages that form "communities" to understand a pages ecosystem
  - Run PageRank, Google's algorithm that determines importance of page based on who links to you



Finding the readable text

```
print(soup.get_text())
```



```
Giving

Support students, research, service, and more





Resources for:Current Students
Faculty
Staff
Parents
Alumni & Friends

Twitter
Facebook
Youtube
Linked-In
```

BU page

Finding the readable text

```
print(soup.get_text())
```

- Sample applications:
  - Train machine learning to classify page topics based on their text (for use in retrieval later, for example)

Do you drive a Toyota Corolla,
but wonder whether a Honda Fit would be better?
We test-drove 2022 models of both cars...

Finding the readable text

```
print(soup.get_text())
```

- Sample applications:
  - Train machine learning to classify page topics based on their text
  - Determine with "sentiment analysis" whether web content is positive or negative

I **love** this Dyson vacuum

I **hate** Dysons so much

Finding the readable text

```
print(soup.get_text())
```

- Sample applications:
  - Train machine learning to classify page topics based on their text
  - Determine with "sentiment analysis" whether web content is positive or negative
  - Scan the text for facts to extract, like who starred in what film

Kevin Smith's latest film, Clerks 3, is a stunning achievement

Finding other things in the HTML besides links

- Find the page title: `soup.title.string`
- Find bigger headings on a page (<h1> through <h3>):

```
for heading in soup.find_all(['h1', 'h2', 'h3']):
    print(heading.text)
```

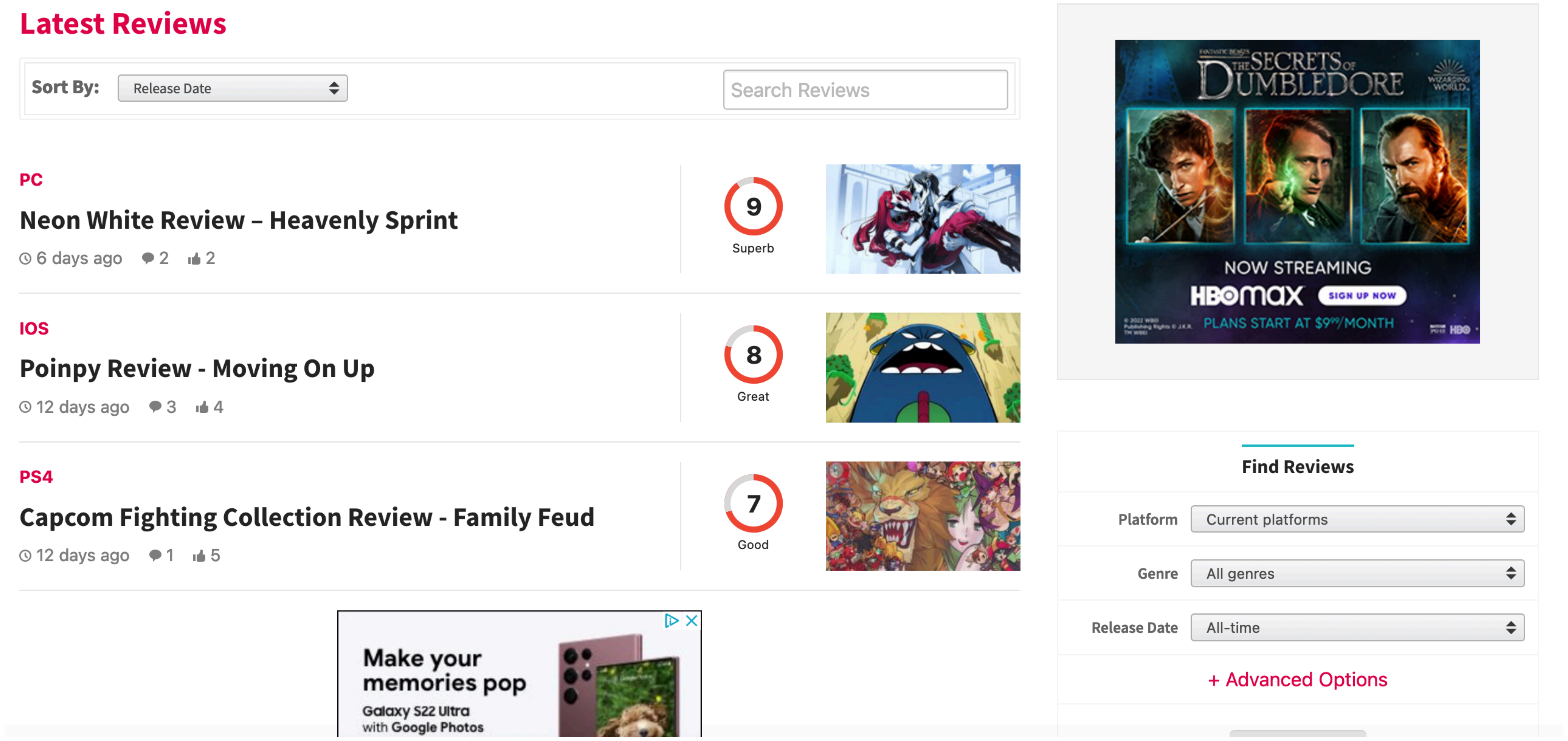
- Search for strings containing a particular regular expression:

```
import re
soup.find_all(string=re.compile("Dormouse"))
```

Finding div classes within a page

- Div tags are often used to divide up a page logically
- The different logical pieces of the page often have different **class** names that the CSS refers to in styling the page
- We can take advantage of this to extract just text from particular parts of the page we care about
-
  ```
  divs = soup.find_all('div', class_='your_classname_here')
  for div in divs:
      print(div.get_text())
  ```

Example: Gamespot review titles



Latest Reviews

Sort By: Release Date

Search Reviews

PC

Neon White Review – Heavenly Sprint

6 days ago 2 2

9 Superb

IOS

Poinpy Review - Moving On Up

12 days ago 3 4

8 Great

PS4

Capcom Fighting Collection Review - Family Feud

12 days ago 1 5

7 Good

Find Reviews

Platform: Current platforms

Genre: All genres

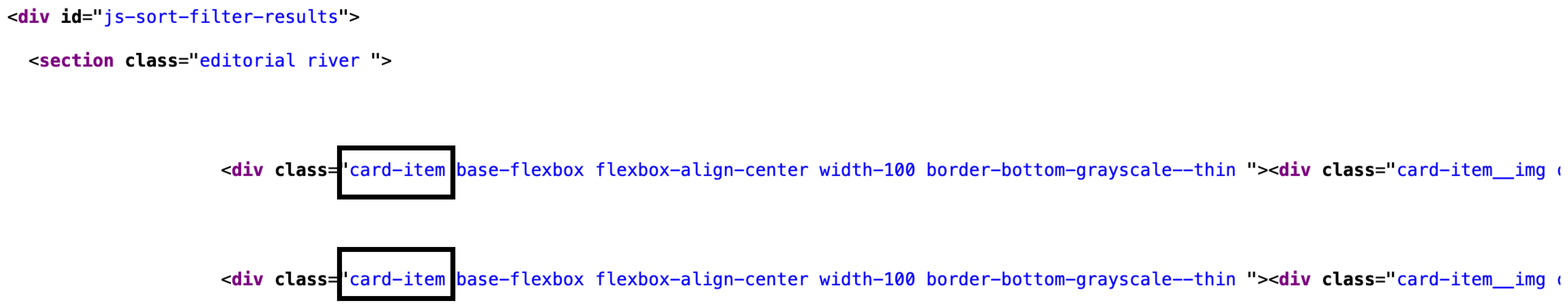
Release Date: All-time

+ Advanced Options

THE SECRETS OF DUMBLEDORE
NOW STREAMING
HBOmax SIGN UP NOW
PLANS START AT $9.99/MONTH
Make your memories pop
Galaxy S22 Ultra with Google Photos

- Site looks like the above
- We want the text in the review cards

Example: Inspecting the page HTML



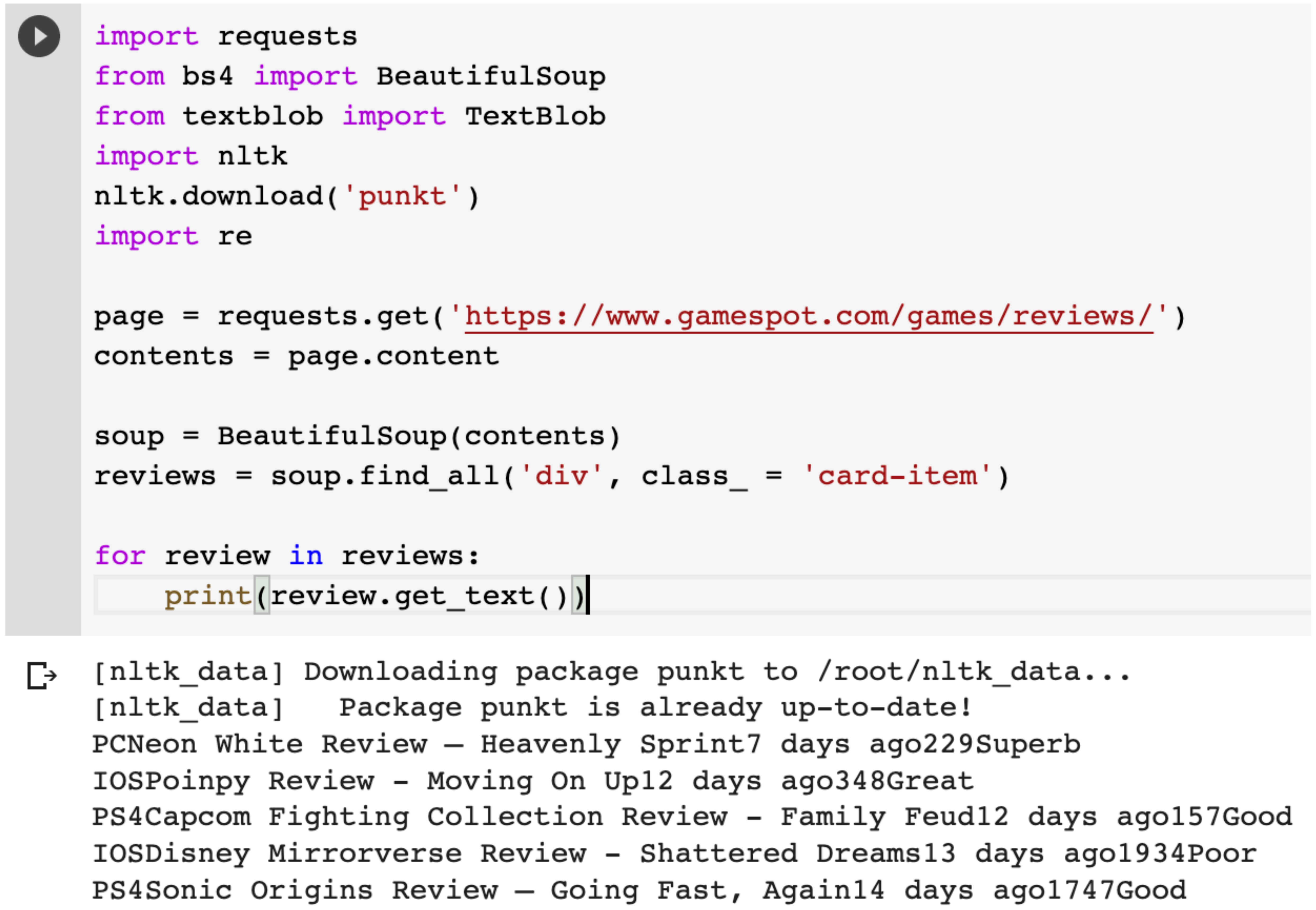
```
<div id="js-sort-filter-results">
  <section class="editorial river ">


                   <div class="card-item base-flexbox flexbox-align-center width-100 border-bottom-grayscale--thin "><div class="card-item__img


                   <div class="card-item base-flexbox flexbox-align-center width-100 border-bottom-grayscale--thin "><div class="card-item__img
```

- Inspection of the HTML and a little experimentation reveals that "card-item" is a class shared by each boxed review

Example: Code to extract the review card text



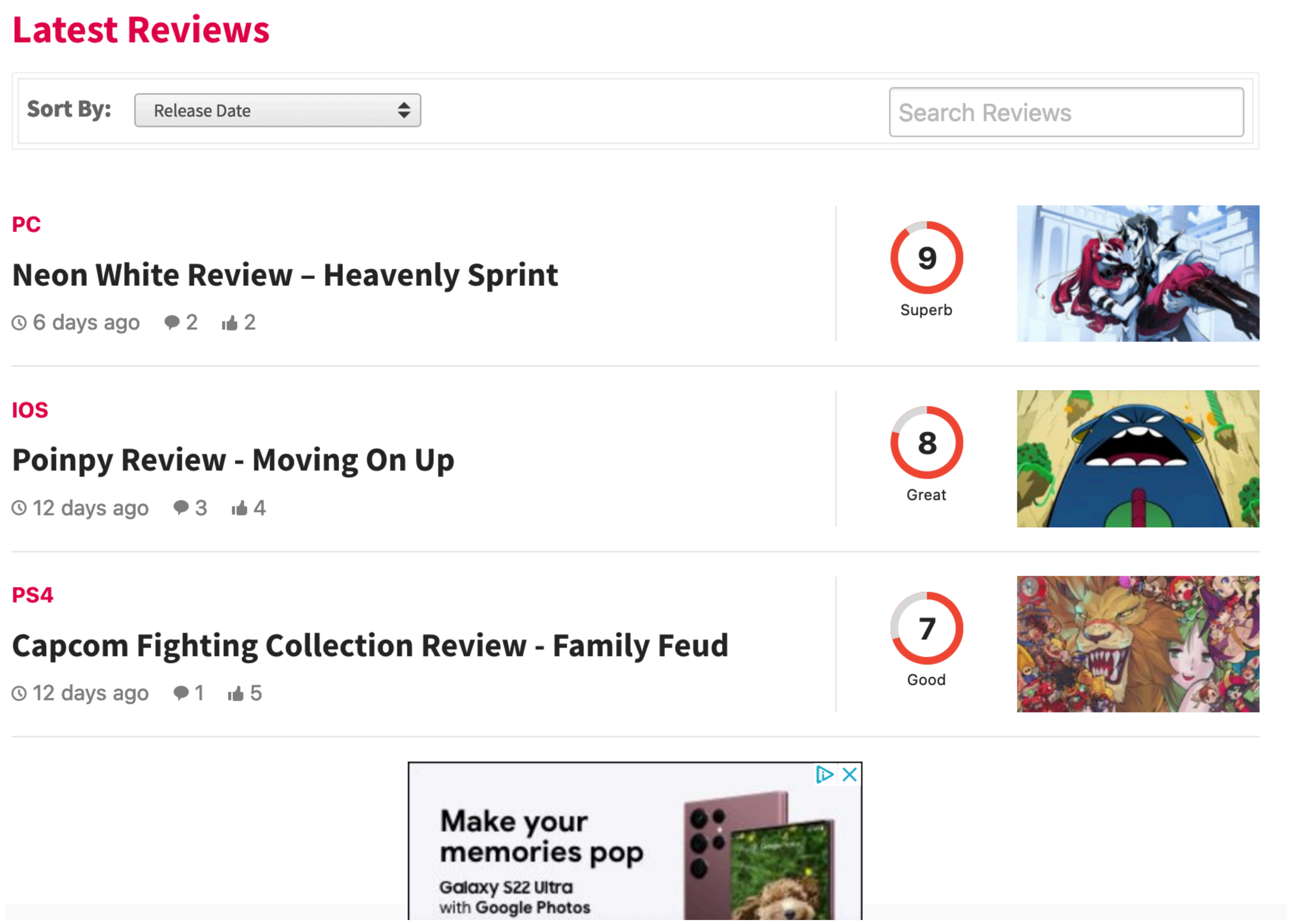
```
import requests
from bs4 import BeautifulSoup
from textblob import TextBlob
import nltk
nltk.download('punkt')
import re

page = requests.get('https://www.gamespot.com/games/reviews/')
contents = page.content

soup = BeautifulSoup(contents)
reviews = soup.find_all('div', class_ = 'card-item')

for review in reviews:
    print(review.get_text())
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
PCNeon White Review – Heavenly Sprint7 days ago229Superb
IOSPoinpy Review - Moving On Up12 days ago348Great
PS4Capcom Fighting Collection Review - Family Feud12 days ago157Good
IOSDisney Mirrorverse Review - Shattered Dreams13 days ago1934Poor
PS4Sonic Origins Review – Going Fast, Again14 days ago1747Good
```



Latest Reviews

Sort By: Release Date
Search Reviews

PC
Neon White Review – Heavenly Sprint
6 days ago 2 2
9 Superb

IOS
Poinpy Review - Moving On Up
12 days ago 3 4
8 Great

PS4
Capcom Fighting Collection Review - Family Feud
12 days ago 1 5
7 Good

Make your memories pop
Galaxy S22 Ultra with Google Photos

Example: Refining the search

- We could look for a div or other tag that more precisely captures the information we want

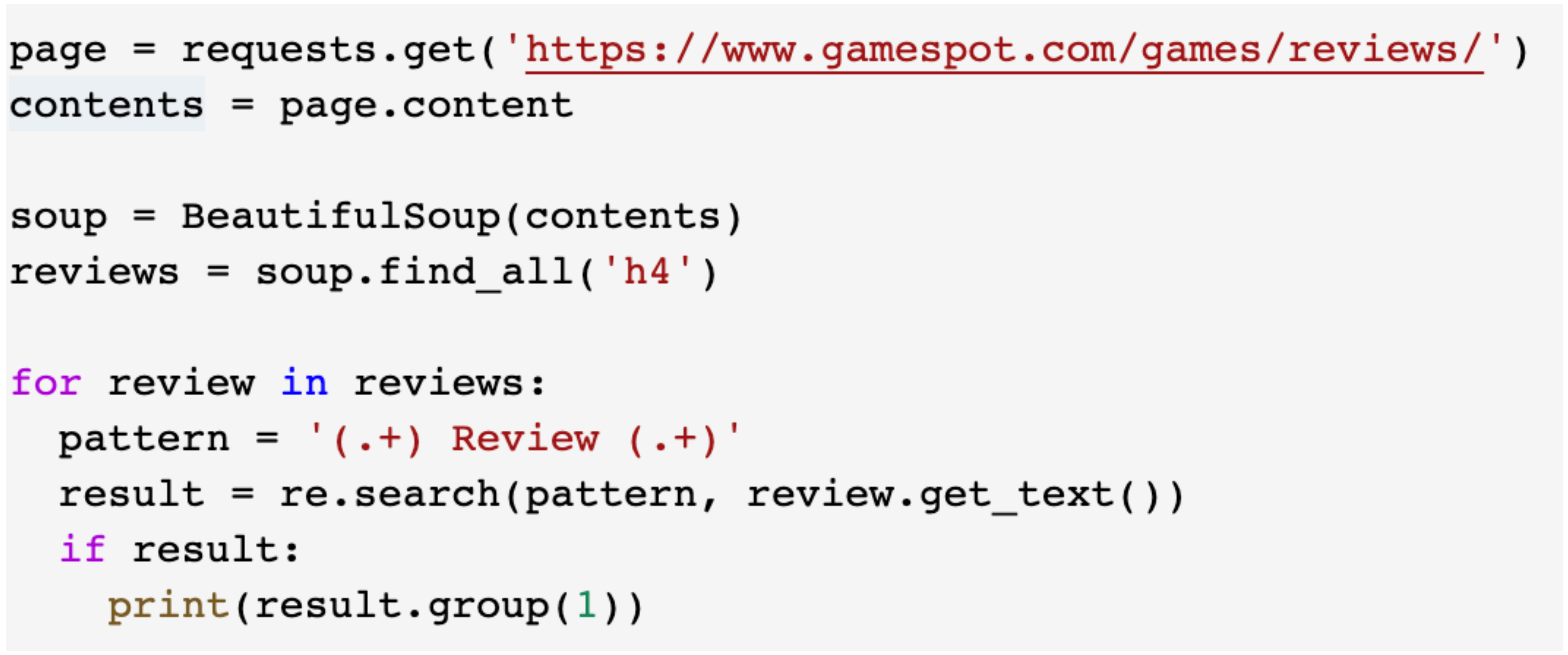
```
<h4 class="card-item__title ">Neon White Review – Heavenly Sprint</h4>
```

```
soup = BeautifulSoup(contents)
reviews = soup.find_all('h4')
```

```
*Neon White Review – Heavenly Sprint
*Poinpy Review - Moving On Up
*Capcom Fighting Collection Review - Family Feud
*Disney Mirrorverse Review - Shattered Dreams
*Sonic Origins Review – Going Fast, Again
*Diablo Immortal Review - Evil On The Go
```

Example: Using regular expressions

- We could alternately, or in addition, use regular expressions to clean and parse the information we got



```
page = requests.get('https://www.gamespot.com/games/reviews/')
contents = page.content

soup = BeautifulSoup(contents)
reviews = soup.find_all('h4')

for review in reviews:
  pattern = '(.+) Review (.+)'
  result = re.search(pattern, review.get_text())
  if result:
    print(result.group(1))
```

```
Neon White
Poinpy
Capcom Fighting Collection
Disney Mirrorverse
Sonic Origins
Diablo Immortal
Fire Emblem Warriors: Three Hopes
Teenage Mutant Ninja Turtles: Shredder's Revenge
The Quarry
Roller Champions
Halo Infinite Multiplayer
Soundfall
Apex Legends Mobile
Hatsune Miku Project Diva Megamix+
Sniper Elite 5
```

Summary

- We can pull text from either webpages to power machine learning
- We can be interested in the text, the graph, or both
- requests plus BeautifulSoup easily pulls text from websites
- It can take some trial and error to find the right patterns to search for