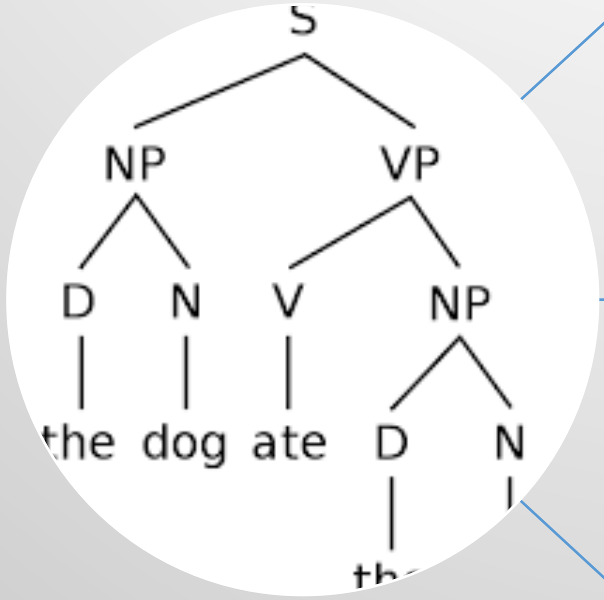
Natural Language Processing

Dr. Karen Mazidi

Part Three: Sentences



Topics

- Finding a corpus
- Getting a corpus from the web

Quizzes

- Q: later

Homework

- Project: Web Crawler

Finding a corpus

- A corpus (plural, corpora) is a collection of linguistic data such as written texts
- Corpora are used by linguists to study language
- Corpora are used by NLP practitioners to build a model of language, or machine learning tasks

Finding a corpus

Some resources:

- https://nlpforhackers.io/corpora/
- https://github.com/jojonki/NLP-Corpora
- https://en.wikipedia.org/wiki/List_of_text_corpora
- https://ai.googleblog.com/2019/09/announcing-two-new-natural-language.html

Building a corpus

- WoZ – wizard of Oz technique for dialogue systems involves having humans interact with people, modeling best practices
- Annotating data from the web can be expensive and time-consuming
  - Stanford used stars to annotate movie reviews
  - An alternative is using Amazon's Mechanical Turk

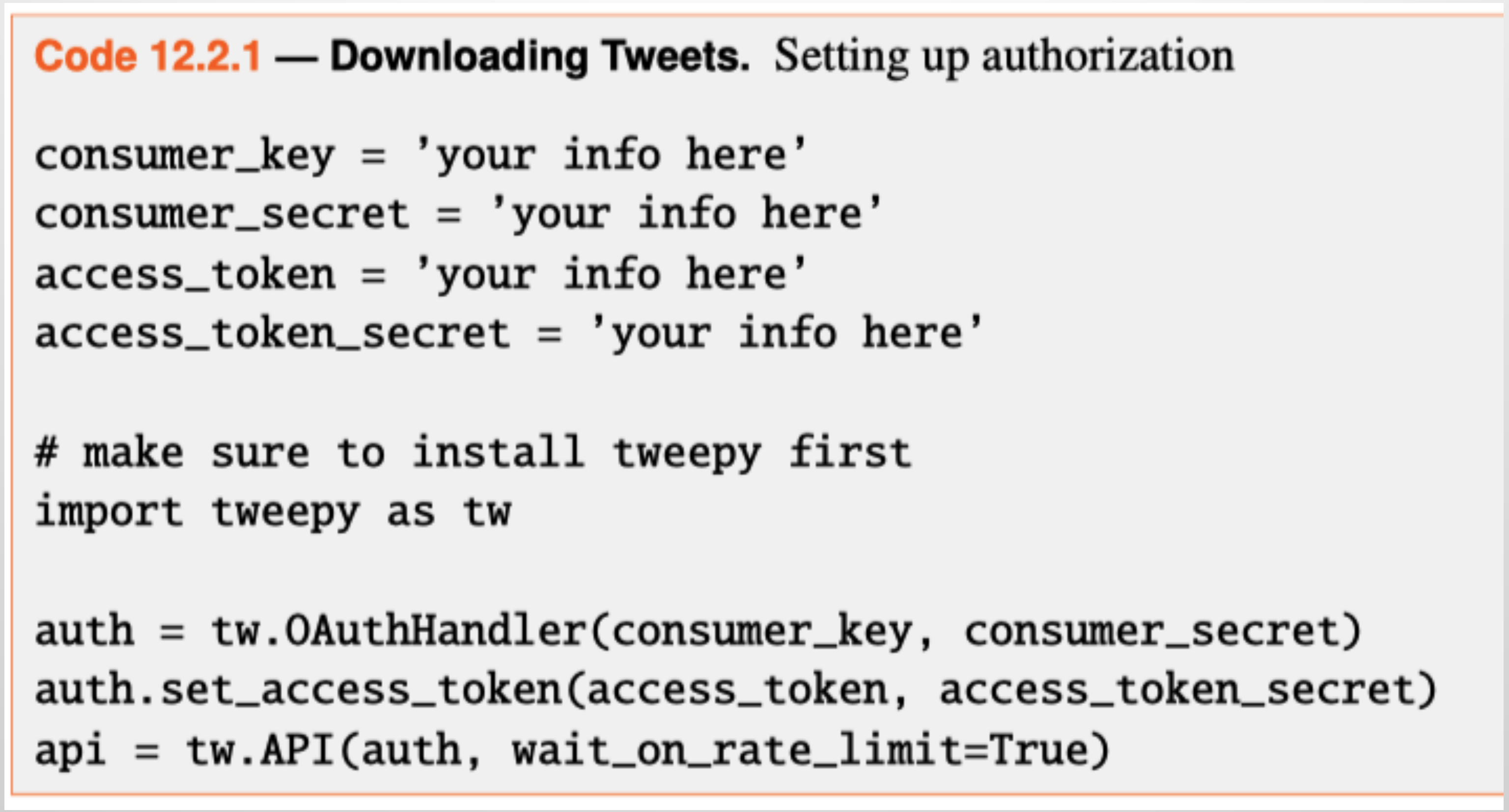
Building a corpus

- A gold standard data set has been reliably annotated with labels, typically by multiple experts
- Kappa is a measure of inter-annotator agreement
- Using 4 or more MTurk workers (with a few caveats) gets quality comparable to annotation by experts

Get Twitter data

- The tweepy library
- Must have user account and developer account: https://developer.twitter.com/en/apply-for-access
- Set up a free personal account with standard access
- Get personal keys from the Developer pages

Step 1: Set up authorization

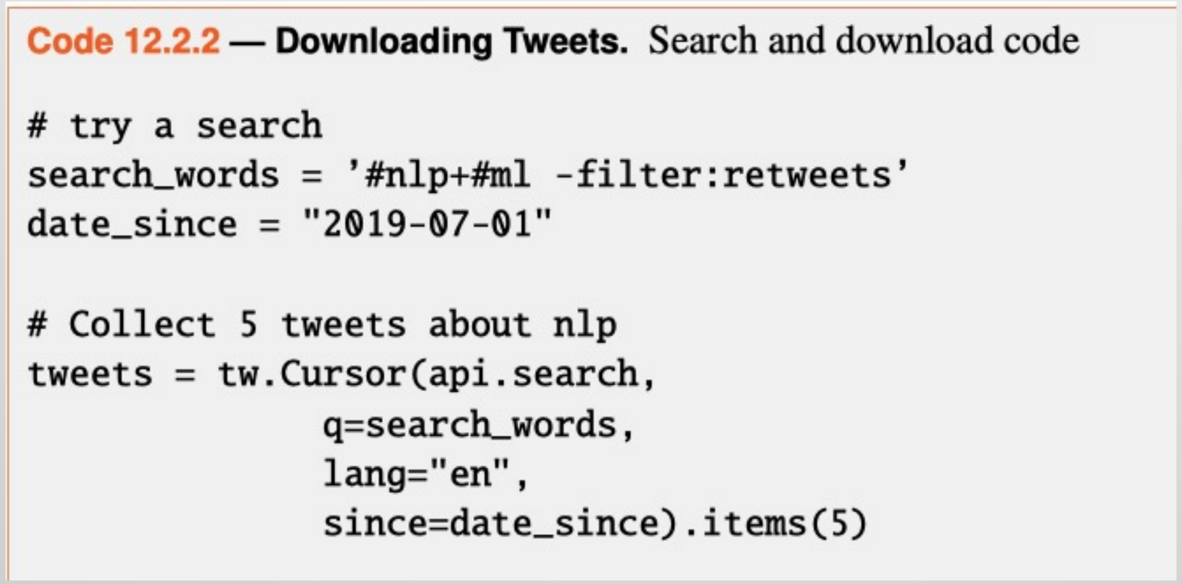


Code 12.2.1 — Downloading Tweets. Setting up authorization

```
consumer_key = 'your info here'
consumer_secret = 'your info here'
access_token = 'your info here'
access_token_secret = 'your info here'

# make sure to install tweepy first
import tweepy as tw

auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tw.API(auth, wait_on_rate_limit=True)
```

Search

- tw.Cursor() method does the search
- Returns an iterable ‘tweets’ object

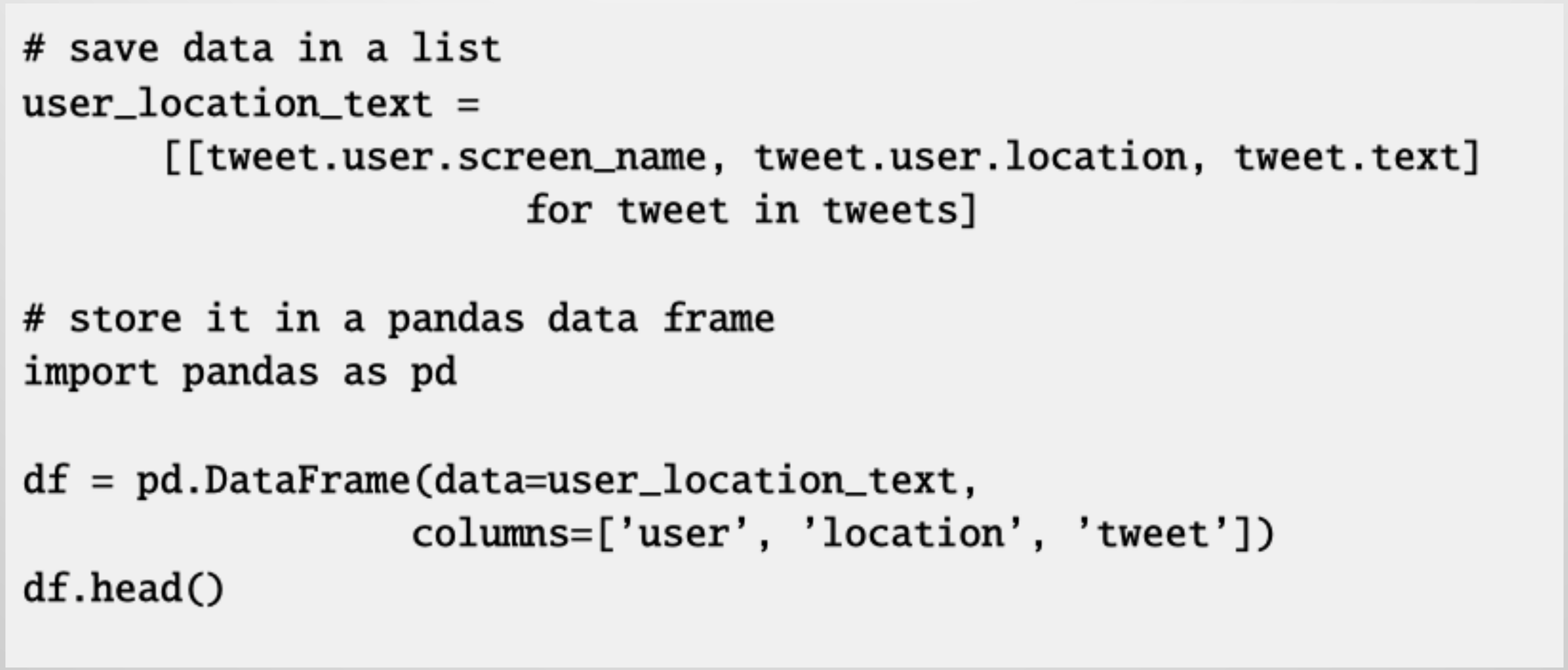


Code 12.2.2 — Downloading Tweets. Search and download code

```
# try a search
search_words = '#nlp+#ml -filter:retweets'
date_since = "2019-07-01"

# Collect 5 tweets about nlp
tweets = tw.Cursor(api.search,
              q=search_words,
              lang="en",
              since=date_since).items(5)
```

Extract info from tweets

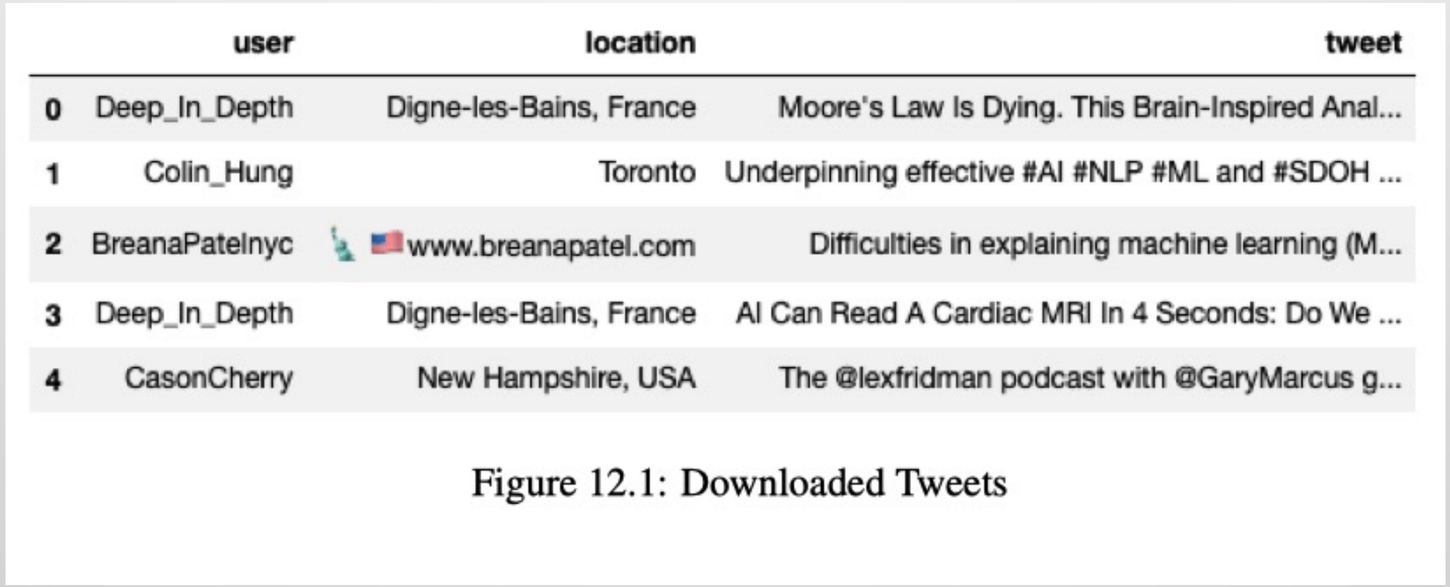
- Iterate over the tweets



```
# save data in a list
user_location_text =
      [[tweet.user.screen_name, tweet.user.location, tweet.text]
                      for tweet in tweets]

# store it in a pandas data frame
import pandas as pd

df = pd.DataFrame(data=user_location_text,
                 columns=['user', 'location', 'tweet'])
df.head()
```

output

- information extracted:

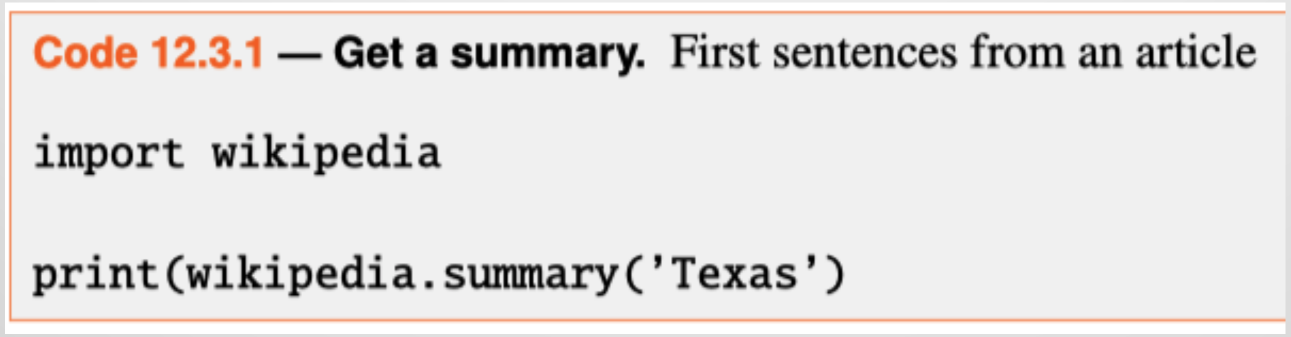


	user	location	tweet
0	Deep_In_Depth	Digne-les-Bains, France	Moore's Law Is Dying. This Brain-Inspired Anal...
1	Colin_Hung	Toronto	Underpinning effective #AI #NLP #ML and #SDOH ...
2	BreanaPatelnyc	🗽 🇺🇸 www.breanapatel.com	Difficulties in explaining machine learning (M...
3	Deep_In_Depth	Digne-les-Bains, France	AI Can Read A Cardiac MRI In 4 Seconds: Do We ...
4	CasonCherry	New Hampshire, USA	The @lexfridman podcast with @GaryMarcus g...

Figure 12.1: Downloaded Tweets

Getting Wikipedia data

- Library wikipedia; other methods suggested for heavy downloads
- Get a summary:

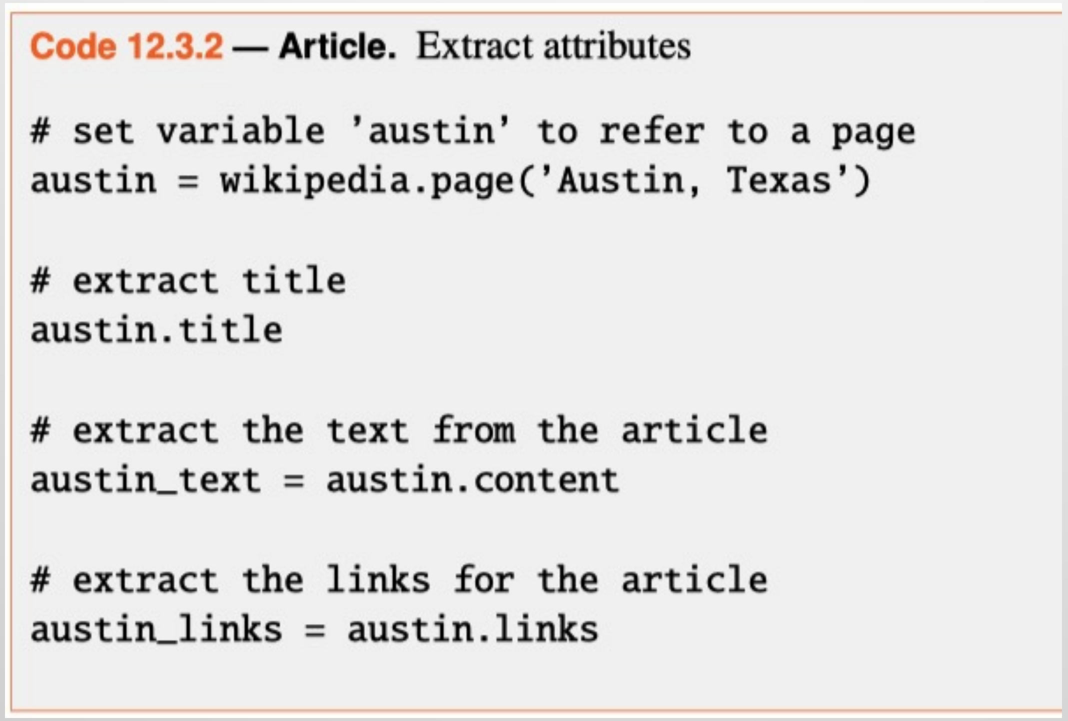


Code 12.3.1 — Get a summary. First sentences from an article

```
import wikipedia

print(wikipedia.summary('Texas'))
```

Extract attributes from a page

- After the page is downloaded



Code 12.3.2 — Article. Extract attributes

```
# set variable 'austin' to refer to a page
austin = wikipedia.page('Austin, Texas')

# extract title
austin.title

# extract the text from the article
austin_text = austin.content

# extract the links for the article
austin_links = austin.links
```

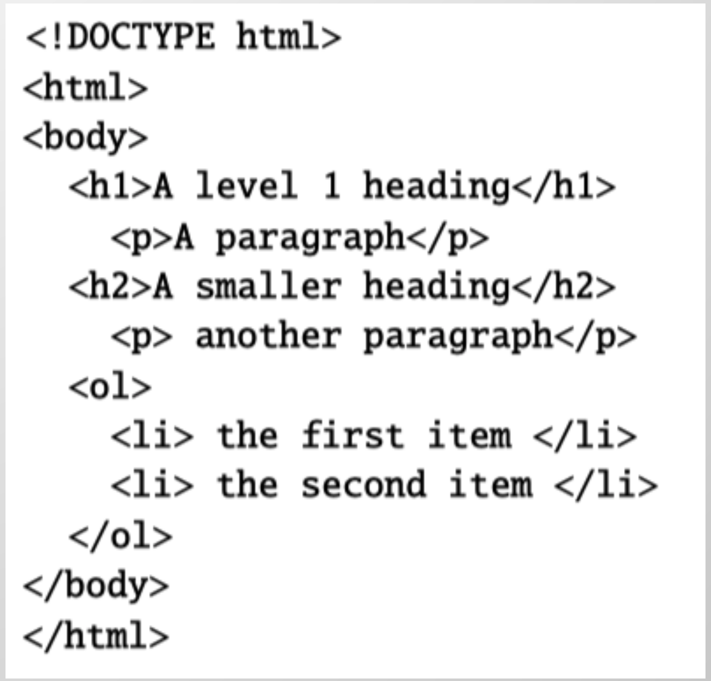
- See online notebooks for more examples

Web scraping

- Web scraping – extracting information from web pages
- Web crawler – program that crawls web links to extract information
- Two helpful libraries:
  - urllib – handles urls
  - Beautiful Soup – extracts data

HTML

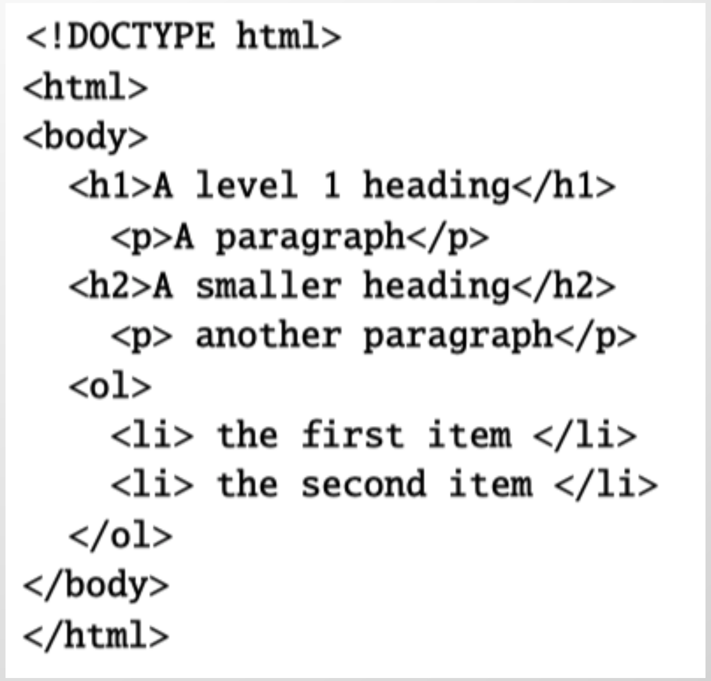
- HTML (Hyper Text Markup Language) to form the structure of the web page
- Simple HTML example:
- Tags are in start/stop pairs
- <x> </x>
- <! ...> comment
- Tags not case sensitive
- First line DOCTYPE required
- Content within body tags is displayed



```
<!DOCTYPE html>
<html>
<body>
  <h1>A level 1 heading</h1>
    <p>A paragraph</p>
  <h2>A smaller heading</h2>
    <p> another paragraph</p>
  <ol>
    <li> the first item </li>
    <li> the second item </li>
  </ol>
</body>
</html>
```

HTML

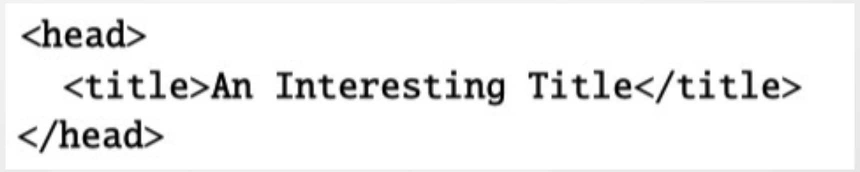
- Headings range h1 to h6
- White space ignored
- <p> for text chunks
- Lists can be ordered
- or unordered (bullets)



```
<!DOCTYPE html>
<html>
<body>
  <h1>A level 1 heading</h1>
    <p>A paragraph</p>
  <h2>A smaller heading</h2>
    <p> another paragraph</p>
  <ol>
    <li> the first item </li>
    <li> the second item </li>
  </ol>
</body>
</html>
```

Head tags

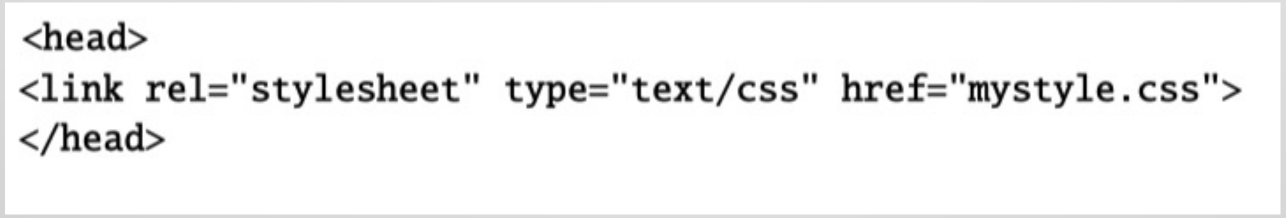
- Before body tags
- Title
- Meta data



```
<head>
  <title>An Interesting Title</title>
</head>
```

CSS

- Formatting is possible within tags for font, color, etc
- Best practice: Separate content and formatting
- Cascading style sheets usually imported from another file within the head tags
- Advantage: format once, applied everywhere



```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Image tags and links

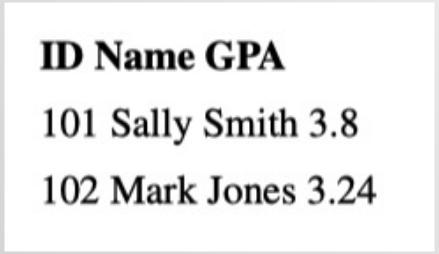
- image tags

```
<img src="myfolder/myimage.png" width="500" height="400"
                  alt="Description here">
```

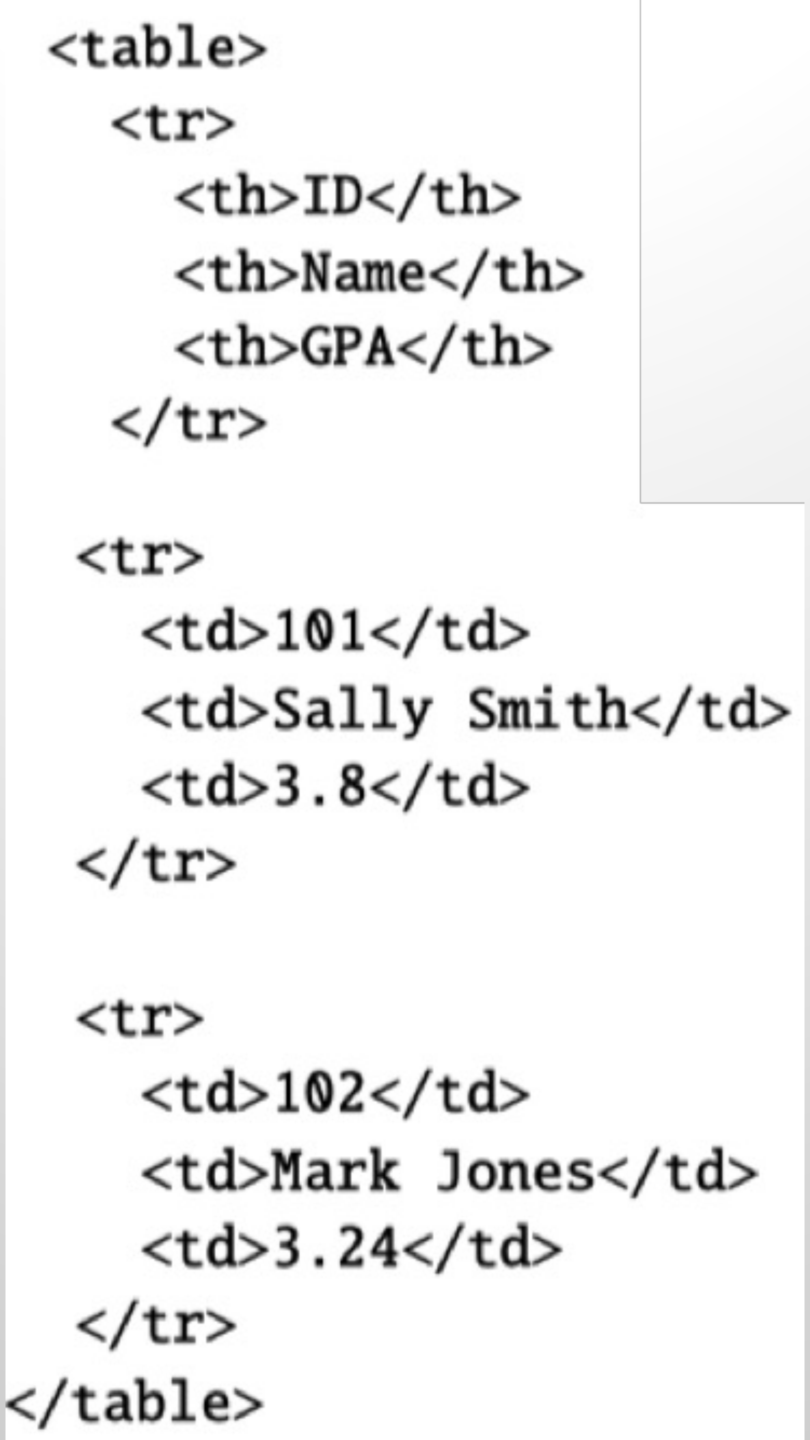
- link tags <a>
  - use / at end

```
<a href="https://www.somepage.com/index.html/" > link text </a>
```

Tables



```
ID Name GPA
101 Sally Smith 3.8
102 Mark Jones 3.24
```



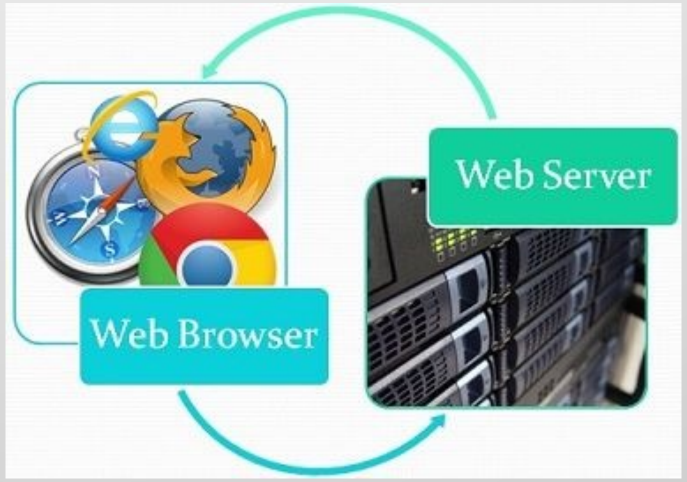
```
<table>
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>GPA</th>
  </tr>

  <tr>
    <td>101</td>
    <td>Sally Smith</td>
    <td>3.8</td>
  </tr>

  <tr>
    <td>102</td>
    <td>Mark Jones</td>
    <td>3.24</td>
  </tr>
</table>
```

Other tags

- Script tags for javascript
- Meta tags for keywords and other info used by web crawlers
- <div> just groups items in a container
- Right-click on a web page, View Source to see more tags

How websites work

- A website is just a collection of pages and other files that share a common domain name like amazon.com
- A web server is a computer that receives requests for a web page from a browser and returns the content of the page
- Client-server relationship

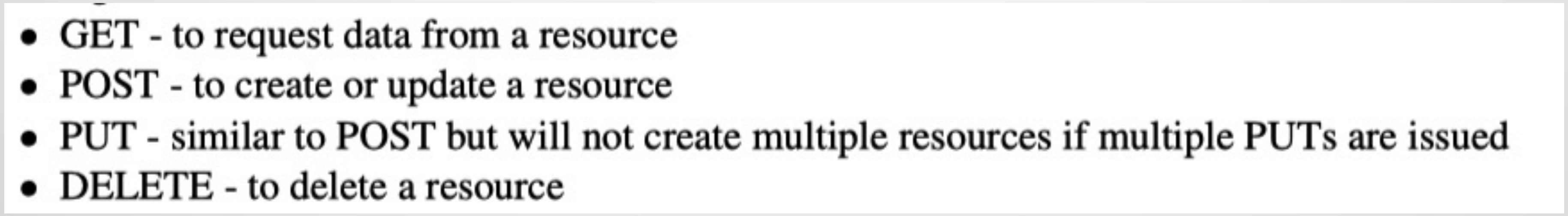


protocols

- HTTP hypertext transfer protocol is the protocol used by web servers and browsers
- Transmission is in packets
- Packets are handled with TCP/IP transmission control protocol and Internet protocol
- The web browser collects the packets and puts together the page in displayable form

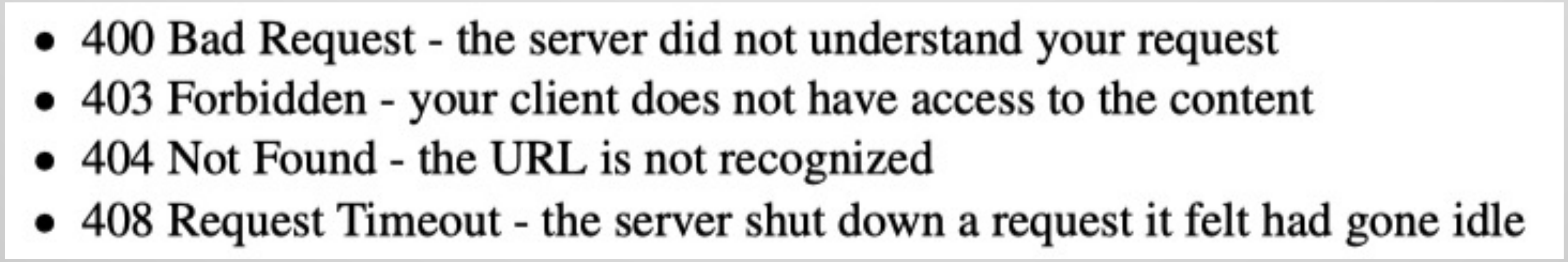
HTTP protocol

- Types of requests and responses



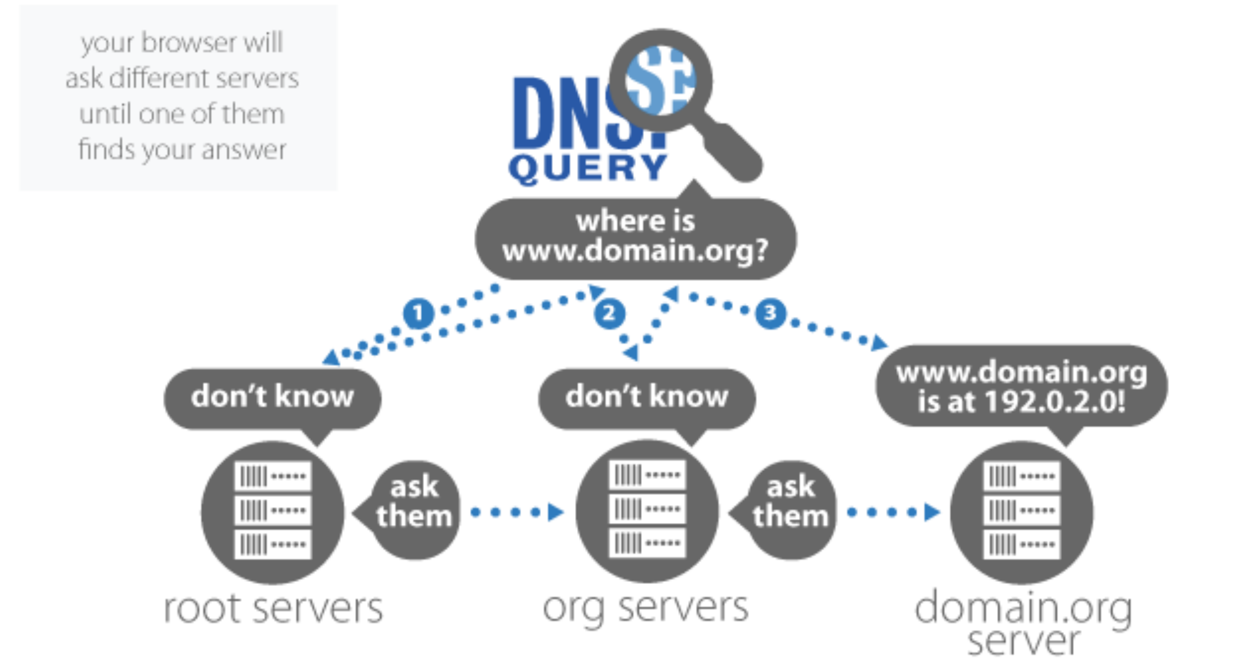
- GET - to request data from a resource
- POST - to create or update a resource
- PUT - similar to POST but will not create multiple resources if multiple PUTs are issued
- DELETE - to delete a resource

- Status codes



- 400 Bad Request - the server did not understand your request
- 403 Forbidden - your client does not have access to the content
- 404 Not Found - the URL is not recognized
- 408 Request Timeout - the server shut down a request it felt had gone idle

Domain names



DNS query what happens when you enter a domain name into your browser?

- Domain names like www.amazon.com get translated to an IP address
- DNS servers provide lookup of IP address

IP addresses

- IPv4 addresses are 32 bits (4 billion addresses)
- Four numbers separated by dots

```
72.14.255.255
```

- IPv6 addresses are 128 bits
- Written in hex with groups separated by colons

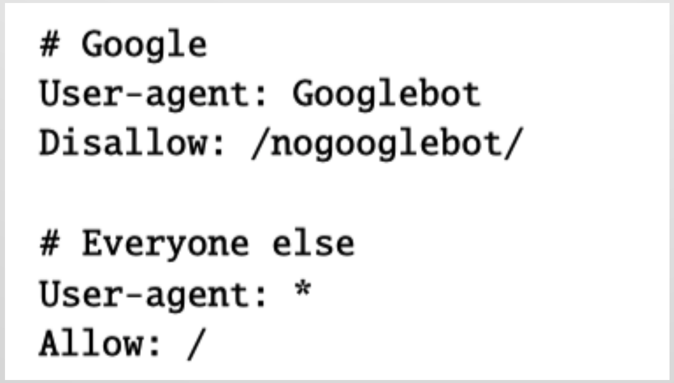
```
2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

Question

True or false. If it’s on the web, it’s free.

The robots.txt file

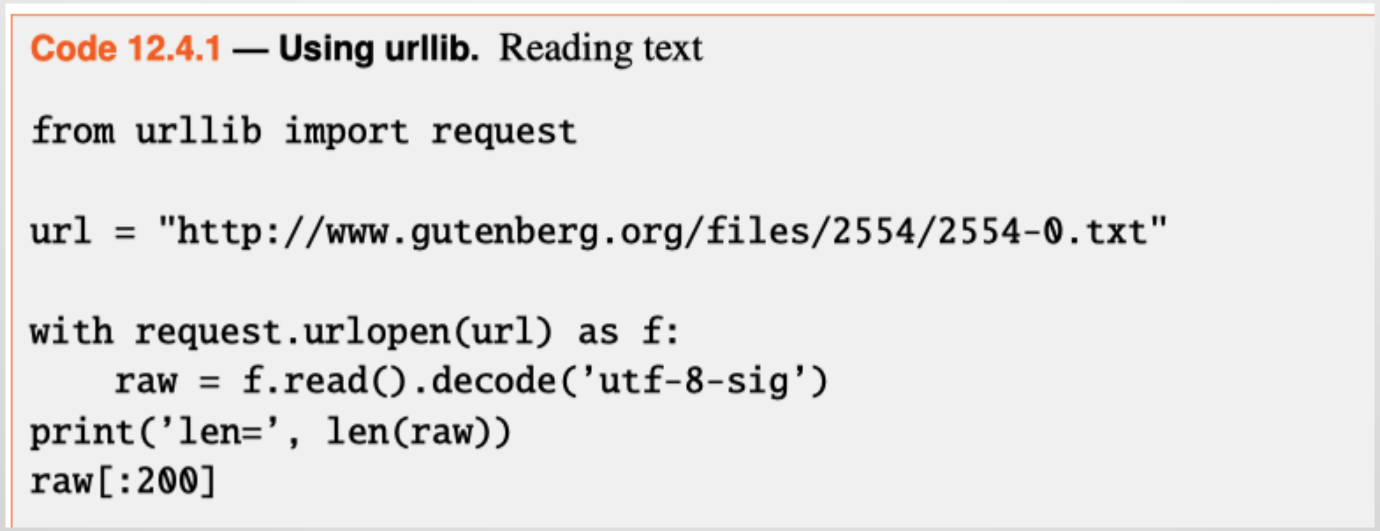
- Specifies how to crawl a site
- Typically on root directory
- First item below is for Google’s web crawler
- Second is for everyone else



```
# Google
User-agent: Googlebot
Disallow: /nogooglebot/

# Everyone else
User-agent: *
Allow: /
```

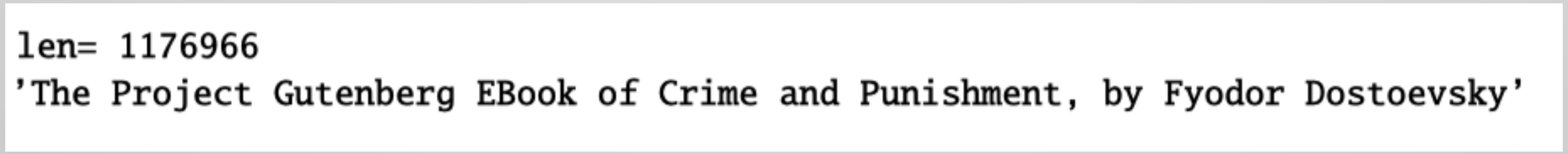
The urllib library

- To download some text:



Code 12.4.1 — Using urllib. Reading text

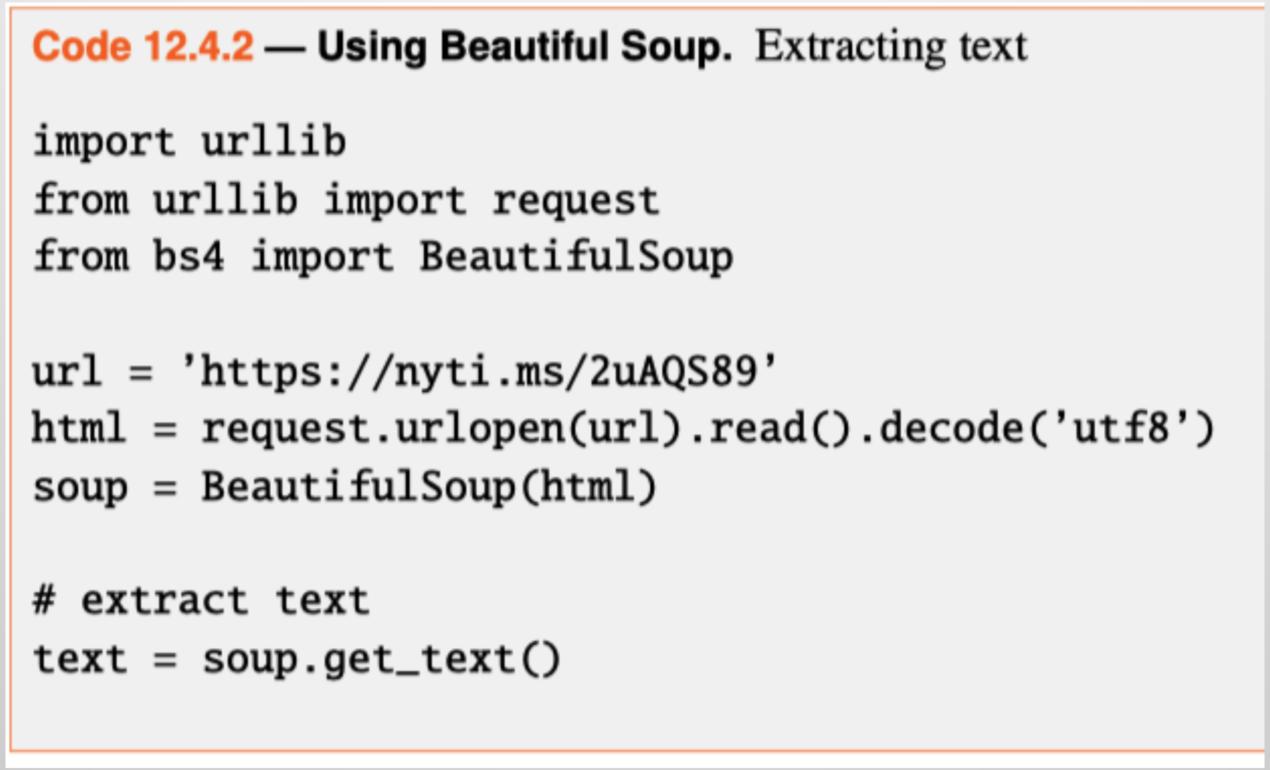
```
from urllib import request

url = "http://www.gutenberg.org/files/2554/2554-0.txt"

with request.urlopen(url) as f:
    raw = f.read().decode('utf-8-sig')
print('len=', len(raw))
raw[:200]
```



```
len= 1176966
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky'
```

Beautiful Soup

- Install with pip/pip3 install BeautifulSoup4
- First, create a soup object

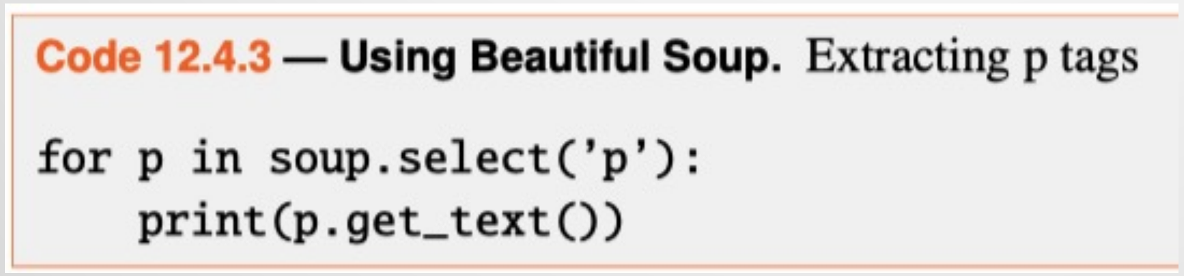


Code 12.4.2 — Using Beautiful Soup. Extracting text

```
import urllib
from urllib import request
from bs4 import BeautifulSoup

url = 'https://nyti.ms/2uAQS89'
html = request.urlopen(url).read().decode('utf8')
soup = BeautifulSoup(html)

# extract text
text = soup.get_text()
```

Extract by tag type

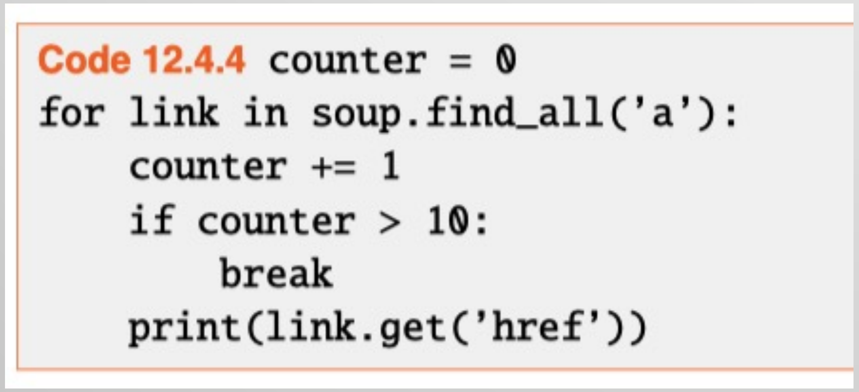
- Extract paragraphs



Code 12.4.3 — Using Beautiful Soup. Extracting p tags

```
for p in soup.select('p'):
    print(p.get_text())
```

- Extract links



Code 12.4.4

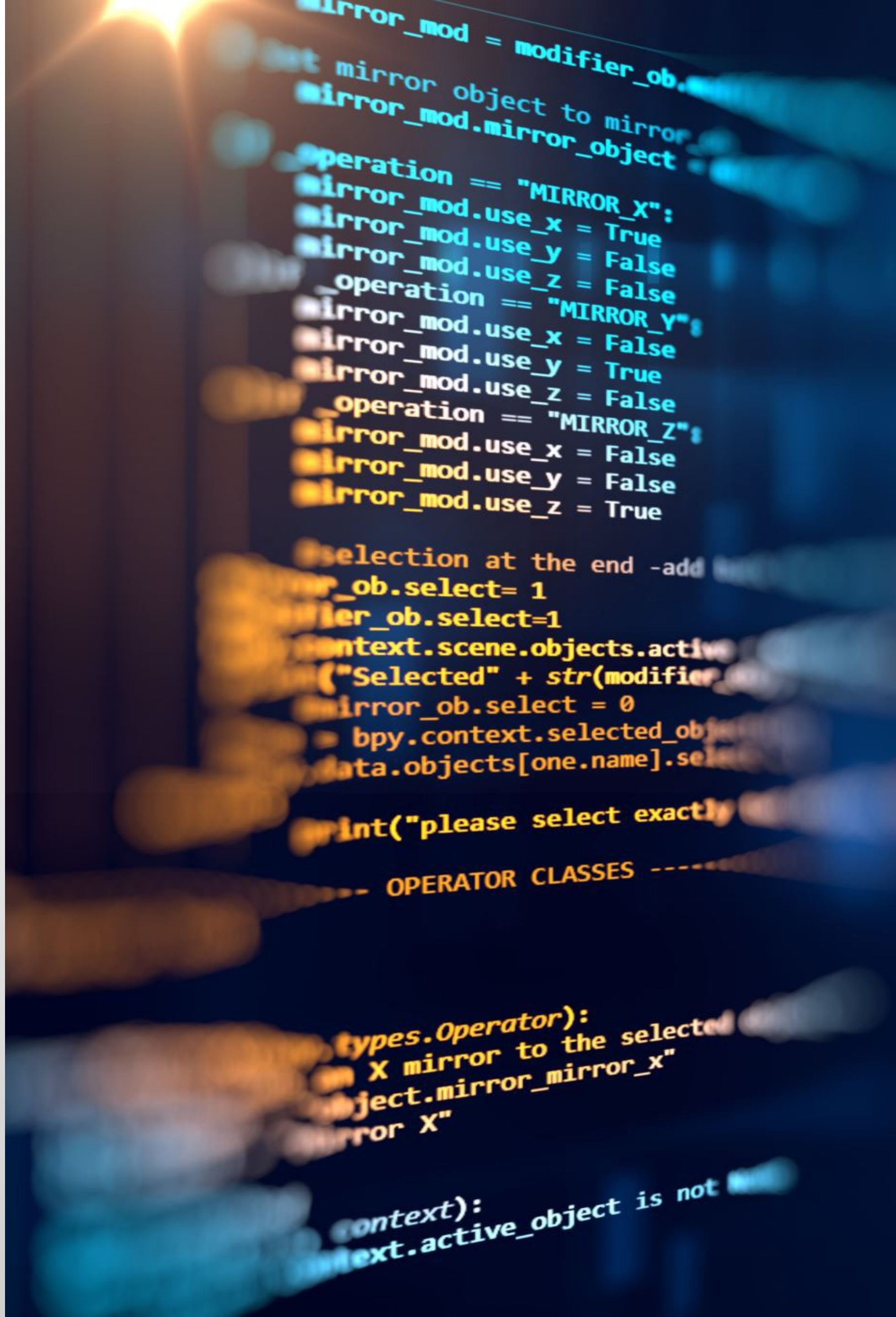
```
counter = 0
for link in soup.find_all('a'):
    counter += 1
    if counter > 10:
        break
    print(link.get('href'))
```

Web crawler

- See this notebook: 3 - Web crawler – almost
- https://github.com/kjmazidi/NLP/tree/master/Xtra_Python_Material/Web_Scraping

Project: Web Crawler

- Build a corpus that could be used for a chatbot



```
irror_mod = modifier_ob.
 mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
_operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
_operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

election at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.active
("Selected" + str(modifier
irror_ob.select = 0
 bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactly

 OPERATOR CLASSES ----

types.Operator):
 X mirror to the selected
ject.mirror_mirror_x"
rror X"

ontext):
ext.active_object is not
```

Python Code Examples

- https://github.com/kjmazidi/NLP/tree/master/Xtra_Python_Material/Web_Scraping

Essential points to note

- Many major sites provide an API for crawling
- Respect the rules of a site when you scrape data