

# DATA 520

## Lecture 16

Python and Web scraping (text)

<https://www.infoworld.com/article/3233250/python/5-essential-python-tools-for-data-sciencenow-improved.html>

**THE SINGULARITY, Simulation, Multiverse**

# The Singularity

At some point during our lifetimes, computers will be much smarter than people at nearly everything, due to hardware and software (AI and ML) advances.

**Examples:** Computers that can play Chess (Deep Blue in 1997) and Go (AlphaGo in 2016) have defeated Grand Masters in each.

**Oct 18, 2017:** The latest Go machine (AlphaGo Zero) learned how to win on its own by playing 3.2 million games in three days. **It defeated the original AlphaGo 100 games to 0!**

# Are you part of a simulation?

In the future, given the advances in computing power, if humans survive, some of them will want to generate very convincing and sophisticated simulated ancestors.

The number of simulated ancestors will exceed the number of actual ancestors.

How can you be sure you are not in a simulation? (with implanted memories)

Bostrom (2003)

# The Multiverse

A virtually infinite number of parallel/alternate universes exist.

Some researchers claim they are more probable than a single unified universe.

Stephen Hawking believes in them.

# Projects

## **You need to write SOME PROGRAM in Python**

- it can be something doable by the OS through many steps (robocopy)  
(get a list of the largest files in your system)
- it can be a web programming challenge
- we can discuss, you can propose

## **You can work alone, or with 1 or 2 others**

- depending on the complexity of the project

## **You must test it somehow**

- compare to independent results, test cases

# Projects

## **You don't need to reinvent the wheel (google)**

- it can come from another language and use elements of it
- it can be an improvement on existing Python code you find
- Dr. Musa has Python event analysis programs
- **BUT you must give credit where it is due**
- provide original code
- document sources and steps

# Why lists[], tuples(), sets{}, and dictionaries{:}?

flexibility ↑	<b>string</b>	stores one text item	speed ↓
	<b>list</b>	stores (vector) data in sequence - can search using index and edit	
	<b>tuple</b>	stores (vector) data in sequence - can search using index	
	<b>set</b>	stores memberships - no duplicates - for faster multiple searches (to compare two 100 item <b>lists/tuples</b> would require 10,000 iterations)	
	<b>dictionary</b>	stores key-value paired values (n x 2) for search using text	
	<b>table (mySQL)</b>	stores data in rows and formatted columns (string, integer, float, etc.)	

# Web Scraping

## From The Programming Historian

<http://programminghistorian.org>

<http://programminghistorian.org/lessons/working-with-web-pages>

We will work with:

<http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33>

**We will use a tool called RegEx, in the re module, that will help us parse text.**



# The Proceedings of the OLD BAILEY London's Central Criminal Court, 1674 to 1913

[Home](#) | [Search](#) | [About The Proceedings](#) | [Historical Background](#) | [API](#) | [The Project](#) | [Contact](#)

In this Section...

[Search Home](#)

[Personal Details](#)

[Ordinary's Accounts, 1676-1772](#)

[Proceedings by date](#)

[Ordinary's Accounts by date](#)

[Statistics](#)

[Custom Search](#)

[Associated Records, 1674-1834](#)

[Research and Study Guides](#)

## BENJAMIN BOWSEY, Breaking Peace > riot, 28th June 1780.

**Reference Number:** t17800628-33

**Offence:** [Breaking Peace](#) > [riot](#)

**Verdict:** [Guilty](#)

**Punishment:** [Death](#)

**Navigation:** [< Previous text \(trial account\)](#) | [Next text \(trial account\) >](#)

**Related Material:** [Associated Records](#)

**Actions:** [Cite this text](#) | [Print-friendly version](#)

[See original](#)



324. BENJAMIN BOWSEY (a blackmoor ) was indicted for that he together with five hundred other persons and more, did, unlawfully, riotously, and tumultuously assemble on the 6th of June to the disturbance of the public peace and did begin to demolish and pull down the dwelling house of Richard Akerman , against the form of the statute, &c.

ROSE JENNINGS , Esq. sworn.

Had you any occasion to be in this part of the town, on the 6th of June in the evening? - I dined with my brother who lives opposite Mr. Akerman's house. They attacked Mr. Akerman's house precisely at seven o'clock; they were preceded by a man better dressed than the rest, who went up to Mr. Akerman's door; he rapped three times, and I believe pulled the bell as often. Mr. Akerman had barrocadoed his house. When the man found that no one came, he went down the steps, made his obeisance to the mob, and pointed to the door, and then retired.

Have you any recollection how that man who you say had a better appearance than the rest was dressed? - I think he had on a dark brown coat and a round ha, but I cannot be particular as to that; the mob immediately following in that formidable manner made such an impression upon me, that I did not take notice. The mob approached about thirty in number, three

# The re Module

## Regular Expressions (Regex)

- match patterns and can process text, removing stray symbols

## Regex filters can remove those symbols

Regex combines `r` (raw code) and `"\"` with a special code

```
import re
```

```
sampletext = "(that's okay). I want to .know #what''s going on?"
```

```
re.compile(r'[A-Z]', re.UNICODE).split(sampletext) # remove caps
```

```
["(that's okay). ", " want to .know #what''s going on?"]
```

```
re.compile(r'\W+', re.UNICODE).split(sampletext) # remove non-characters
```

```
['', 'that', 's', 'okay', 'I', 'want', 'to', 'know', 'what', 's', 'going', 'on', '']
```

# We need the request module

Find location of python.exe on Lab computers, should be c:\..\36-32\

```
import sys
```

```
sys.path
```

```
# at command prompt on your own PC:(must point to python.exe), so cd or include file location:
```

```
"C:\Users\Steve9\AppData\Local\Programs\Python\Python36-32\python.exe" -m pip install -U pip requests
```

```
# change to python.exe directory, then execute
```

```
cd C:\Users\sousley\AppData\Local\Programs\Python\Python36-32\
```

```
python -m pip install -U pip setuptools
```

Install requests module on your computer:

```
python.exe -m pip install -U pip requests
```

Install requests module on Lab computers:

```
"C:\Program Files (x86)\Python36-32\python.exe" -m pip install -U pip requests -t c:/users/sousley/Documents
```

```
- will be same directory where you can save python scripts
```

# Web Scrapping

Open and rrint a web page - open-webpage.py

```
# open-webpage.py
import requests
url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'
pagetext = requests.get(url) # entire page to string pagetext

# pagetext itself returns 200 if successful
print(pagetext.text)
```

and run it

# Web Scrapping

## Alternate method in case requests library will not load:

```
# load built-in Python module
```

```
import urllib.request
```

```
url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'
```

```
with urllib.request.urlopen(url) as webpage:
```

```
    for line in webpage:
```

```
        line = line.strip()
```

```
        line = line.decode('utf-8')
```

```
        print(line) # one line at a time
```

OR:

```
import urllib.request
```

```
url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'
```

```
with urllib.request.urlopen(url) as webpage:
```

```
    print(webpage.read().decode('utf-8'))
```

# Web Scrapping

**We can save the html file too: save-webpage.py**

```
# save-webpage.py
import requests

url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'
pagetext = requests.get(url)

# pagetext itself returns 200 if successful
#print(pagetext.text)

with open('obo-t17800628-33.html', 'w') as writefile:
    writefile.write(pagetext.text)
```

**and run it, look at contents**

# Web Scraping

Header, lots of html junk at beginning and end

create file obo.py

```
# obo.py
```

```
def stripTags(pageContents):
```

```
    startLoc = pageContents.find("<p>") # where paragraph text begins
```

```
    endLoc = pageContents.rfind("<br/>") # where paragraph text ends
```

```
    pageContents = pageContents[startLoc:endLoc]
```

```
    return pageContents
```

then...

# Web Scraping

Header, lots of html junk at beginning and end  
create file trial-content.py

```
# trial-content.py
import requests, obo
url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'
pagetext = requests.get(url)

HTML = pagetext.text

print(obo.stripTags(HTML))
```

and run it



# Web Scraping

Still markup junk. Modify obo.py: add text below

```
def stripTags(pageContents):
    startLoc = pageContents.find("<p>")
    endLoc = pageContents.rfind("<br/>")
    pageContents = pageContents[startLoc:endLoc]

# commented out: return pageContents
# strip html markup between beg and end
inside = 0
text = ''
for char in pageContents:
    if char == '<':
        inside = 1
    elif (inside == 1 and char == '>'):
        inside = 0
    elif inside == 1:
        continue
    else:
        text += char

return text
```

then run trial-content.py

# Web Scrapping

Now we will get a word list

create `html-to-list1.py`

```
#html-to-list1.py
import requests, obo

url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'

pagetext = requests.get(url)
HTML = pagetext.text
wordlist = HTML.split()
print(wordlist)  # print all words
print(wordlist[0:100])  # print first 100 words ( 0 to 99)
```

and run it...

# Web Scrapping

The word list is messy - mixed cases

edit `html-to-list1.py`

```
#html-to-list1.py
import requests, obo

url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'

pagetext = requests.get(url)
HTML = pagetext.text
text = obo.stripTags(HTML).lower() # convert to lower case
wordlist = text.split()
print(wordlist)
```

and run...

# Web Scraping

The word list is messy, due to \$#@%!&#\* punctuation!

edit obo.py, add at bottom:

```
# added to obo.py
# Given a text string, remove all non-alphanumeric
# characters (using Unicode definition of alphanumeric) and RegEx.
# return split text ( a Python list (array) of words)
def stripNonAlphaNum(text):
    import re
    return re.compile(r'\W+', re.UNICODE).split(text)
```

then edit html-list1.py ...

# Web Scrapping

## Edit html-to-list1.py:

```
#html-to-list1.py
import requests, obo

url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'
pagetext = requests.get(url)
HTML = pagetext.text
text = obo.stripTags(HTML).lower()
#wordlist = text.split() # now commented out
wordlist = obo.stripNonAlphaNum(text) # RegEx and split done together

print(wordlist[0:150])
```

# Web Scraping

Now we will get word counts .. but how?

First part: use a list and count words

create count-list.items-1.py

```
# count-list-items-1.py
wordstring = 'it was the best of times it was the worst of times '
wordstring += 'it was the age of wisdom it was the age of foolishness'
wordlist = wordstring.split()
wordfreq = [] # create empty Python list
for w in wordlist:
    wordfreq.append(wordlist.count(w)) # count of the times word appears in the list

print("String\n" + wordstring + "\n")
print("List\n" + str(wordlist) + "\n")
print("Frequencies\n" + str(wordfreq) + "\n")
print("Pairs\n" + str(list(zip(wordlist, wordfreq)))) # zips together paired lists 22
```

# Web Scraping

Now we will get word counts .. but how?

Second part: use Python **dictionaries**: {key:value} that are paired

```
# in the shell:
# set up a small dictionary - note curly brackets, colon {key : value}
>>> dct = {'hello': 0, 'Erie': 1} #
>>> print(dct['hello']) # look up value using key; note SQUARE brackets used
0
>>> print(dct['Erie']) # look up value using key; note SQUARE brackets used
1
>>> print(dct['typo']) # look up value using key; note SQUARE brackets used
KeyError: 'typo'
>>> print(dct.keys()) # give list of keys; note PARENS used
['world', 'Erie']
```

# Web Scraping

Now we will get word counts

edit obo.py, add at bottom:

```
# Given a list of words, return a dictionary of word-frequency pairs.  
def wordListToFreqDict(wordlist):  
    wordfreq = [wordlist.count(p) for p in wordlist]  
    return dict(list(zip(wordlist,wordfreq))) # zips together paired lists
```

also add a sorting routine to obo.py:

```
# Sort a dictionary of word-frequency pairs in order of descending frequency.  
def sortFreqDict(freqdict):  
    aux = [(freqdict[key], key) for key in freqdict]  
    aux.sort()  
    aux.reverse()  
    return aux
```



# Web Scraping

Now we will get word counts

create `html-to-freq.py`

```
#html-to-freq.py

import requests, obo

url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'

pagetext = requests.get(url)

HTML = pagetext.text

text = obo.stripTags(HTML).lower() # make lower case


wordlist = obo.stripNonAlphaNum(text) # convert to list of words, no punctuation


dictionary = obo.wordListToFreqDict(wordlist) # add words, counts to dictionary
sorteddict = obo.sortFreqDict(dictionary)      # sort word list by frequency
for s in sorteddict: print(str(s))
```

and run...

# Web Scrapping

## The word counts start with "the"!

(192, 'the')

(105, 'i')

(74, 'to')

(71, 'was')

(67, 'of')

(62, 'in')

(53, 'a')

(52, 'and')

(50, 'you')

(50, 'he')

(40, 'that')

(39, 'his')

(36, 'it')

(34, 'did')

(27, 'him')

# Web Scraping

**We can remove the less informative words using a list of them (part 1)**  
**(add to the beginning of obo.py)**

```
stopwords = ['a', 'about', 'above', 'across', 'after', 'afterwards']
stopwords += ['again', 'against', 'all', 'almost', 'alone', 'along']
stopwords += ['already', 'also', 'although', 'always', 'am', 'among']
stopwords += ['amongst', 'amoungst', 'amount', 'an', 'and', 'another']
stopwords += ['any', 'anyhow', 'anyone', 'anything', 'anyway', 'anywhere']
stopwords += ['are', 'around', 'as', 'at', 'back', 'be', 'became']
stopwords += ['because', 'become', 'becomes', 'becoming', 'been']
stopwords += ['before', 'beforehand', 'behind', 'being', 'below']
stopwords += ['beside', 'besides', 'between', 'beyond', 'bill', 'both']
stopwords += ['bottom', 'but', 'by', 'call', 'can', 'cannot', 'cant']
stopwords += ['co', 'computer', 'con', 'could', 'couldnt', 'cry', 'de']
stopwords += ['describe', 'detail', 'did', 'do', 'done', 'down', 'due']
stopwords += ['during', 'each', 'eg', 'eight', 'either', 'eleven', 'else']
stopwords += ['elsewhere', 'empty', 'enough', 'etc', 'even', 'ever']
stopwords += ['every', 'everyone', 'everything', 'everywhere', 'except']
stopwords += ['few', 'fifteen', 'fifty', 'fill', 'find', 'fire', 'first']
stopwords += ['five', 'for', 'former', 'formerly', 'forty', 'found']
stopwords += ['four', 'from', 'front', 'full', 'further', 'get', 'give']
stopwords += ['go', 'had', 'has', 'hasnt', 'have', 'he', 'hence', 'her']
stopwords += ['here', 'hereafter', 'hereby', 'herein', 'hereupon', 'hers']
stopwords += ['herself', 'him', 'himself', 'his', 'how', 'however']
stopwords += ['hundred', 'i', 'ie', 'if', 'in', 'inc', 'indeed']
stopwords += ['interest', 'into', 'is', 'it', 'its', 'itself', 'keep']
stopwords += ['last', 'latter', 'latterly', 'least', 'less', 'ltd', 'made']
stopwords += ['many', 'may', 'me', 'meanwhile', 'might', 'mill', 'mine']
stopwords += ['more', 'moreover', 'most', 'mostly', 'move', 'much']
stopwords += ['must', 'my', 'myself', 'name', 'namely', 'neither', 'never']
stopwords += ['nevertheless', 'next', 'nine', 'no', 'nobody', 'none']
```

# Web Scraping

**We can remove the less informative words using a list of them (part 2)**  
**(add to the beginning of obo.py)**

```
stopwords += ['noone', 'nor', 'not', 'nothing', 'now', 'nowhere', 'of']
stopwords += ['off', 'often', 'on', 'once', 'one', 'only', 'onto', 'or']
stopwords += ['other', 'others', 'otherwise', 'our', 'ours', 'ourselves']
stopwords += ['out', 'over', 'own', 'part', 'per', 'perhaps', 'please']
stopwords += ['put', 'rather', 're', 's', 'same', 'see', 'seem', 'seemed']
stopwords += ['seeming', 'seems', 'serious', 'several', 'she', 'should']
stopwords += ['show', 'side', 'since', 'sincere', 'six', 'sixty', 'so']
stopwords += ['some', 'somehow', 'someone', 'something', 'sometime']
stopwords += ['sometimes', 'somewhere', 'still', 'such', 'system', 'take']
stopwords += ['ten', 'than', 'that', 'the', 'their', 'them', 'themselves']
stopwords += ['then', 'thence', 'there', 'thereafter', 'thereby']
stopwords += ['therefore', 'therein', 'thereupon', 'these', 'they']
stopwords += ['thick', 'thin', 'third', 'this', 'those', 'though', 'three']
stopwords += ['three', 'through', 'throughout', 'thru', 'thus', 'to']
stopwords += ['together', 'too', 'top', 'toward', 'towards', 'twelve']
stopwords += ['twenty', 'two', 'un', 'under', 'until', 'up', 'upon']
stopwords += ['us', 'very', 'via', 'was', 'we', 'well', 'were', 'what']
stopwords += ['whatever', 'when', 'whence', 'whenever', 'where']
stopwords += ['whereafter', 'whereas', 'whereby', 'wherein', 'whereupon']
stopwords += ['wherever', 'whether', 'which', 'while', 'whither', 'who']
stopwords += ['whoever', 'whole', 'whom', 'whose', 'why', 'will', 'with']
stopwords += ['within', 'without', 'would', 'yet', 'you', 'your']
stopwords += ['yours', 'yourself', 'yourselves']
```

# Web Scraping

Add a function to the end of obo.py:

```
# Given a list of words, remove any that are in a list of stop words.  
def removeStopwords(wordlist, stopwords):  
    return [w for w in wordlist if w not in stopwords]
```

# Web Scrapping

Create `html-to-freq2.py` (one line added to `html-to-freq1.py`) so it reads:

```
# html-to-freq-2.py
import requests, obo
url = 'http://www.oldbaileyonline.org/browse.jsp?id=t17800628-33&div=t17800628-33'
pagetext = requests.get(url)
HTML = pagetext.text

text = obo.stripTags(HTML).lower() # convert to lower case

fullwordlist = obo.stripNonAlphaNum(text) # only words, into list

wordlist = obo.removeStopwords(fullwordlist, obo.stopwords) # remove common useless words

dictionary = obo.wordListToFreqDict(wordlist) # add words and counts to dictionary
sorteddict = obo.sortFreqDict(dictionary) # sort word list by frequency

for s in sorteddict: print(str(s))
```

and run...

# No homework

**But see if you can extract text and get word counts from another site**