Regular Expressions & Web Scraping

<http://cs.brown.edu/~jadrian/thatcamp/>

* [Regular Expression tutorial](http://gnosis.cx/publish/programming/regular_expressions.html)
* [Another Regular Expressions tutorial](http://www.regular-expressions.info/)
* [Python tutorial](http://docs.python.org/tutorial/introduction.html)
* [Python documentation](http://docs.python.org/library/index.html)
* [Python web-scraping library, urllib](http://docs.python.org/library/urllib.html)
* [Python regular expressions library, re](http://docs.python.org/library/re.html)

Command to open: python

>>> import zombify

>>> zombify.zombify\_text('pnp.txt', 'pnp.csv', ‘pnpnz.txt')

**Regular Expressions**

GUTENBERG\_DIVIDER = '(\\\*)+\\s\*(START|END) OF THIS PROJECT GUTENBERG.\*$'

* Consistent Pattern: \*\*\*Start of this Project Guetenberg…. \*\*\*
* (\\\*)+//s\*(START|END)
* +\\s\* = any number of spaces

Tool Kit

* a 🡪 a – match letter
* a+ 🡪 a; aa; aaa – match letter or letter repeats
* a\* 🡪 “; a; aa; aaa
* . 🡪 any character
* [axbrj] 🡪 any one character from this list
* + 🡪 one or more
* \* 🡪 zero or more
* \ 🡪 escape character
* \s 🡪 space
* ‘’’ 🡪 exact match
* \n 🡪 line break
* \\ 🡪 indicates that \ is a character not a command (escape from the escape charcter) e.g. \\s to indicate character of \s (space)
* ? 🡪 zero or one
* +?, \*? 🡪 the non-greedy versions/shortest versions
* ^ 🡪 inverted (e.g. [^”] = anything that is not quotation marks)

Terminal Input:

>>> import re

>>> re.findall('e', 'the quick brown fox')

['e']

>>> re.findall('e', 'the quick brown fox walked slowly to school')

['e', 'e']

>>> re.findall('.e', 'the quick brown fox walked slowly to school')

['he', 'ke']

>>> re.findall('[aeiou].', 'the quick brown fox walked slowly to school')

['e ', 'ui', 'ow', 'ox', 'al', 'ed', 'ow', 'o ', 'oo']

>>> re.findall('.[aeiou]+.', 'the quick brown fox walked slowly to school')

['he ', 'quic', 'row', 'fox', 'wal', 'ked', 'low', 'to ', 'hool']

>>> re.findall('.[aeiou]\*.', 'the quick brown fox walked slowly to school')

['th', 'e ', 'quic', 'k ', 'br', 'ow', 'n ', 'fox', ' w', 'al', 'ked', ' s', 'low', 'ly', ' t', 'o ', 'sc', 'hool']

>>> re.findall('''.[aeiou]\*.''', 'the quick brown fox walked slowly to school')

['th', 'e ', 'quic', 'k ', 'br', 'ow', 'n ', 'fox', ' w', 'al', 'ked', ' s', 'low', 'ly', ' t', 'o ', 'sc', 'hool']

>>> re.findall('o[aeiou]?.', 'the quick brown fox walked slowly to school')

['ow', 'ox', 'ow', 'o ', 'ool']

>>> re.findall('o.\*l', 'the quick brown fox walked slowly to school')

['own fox walked slowly to school']

>>> re.findall('o.\*?l', 'the quick brown fox walked slowly to school')

['own fox wal', 'owl', 'o school']

>>> print('''Four score and\nseven years ago''')

Four score and

seven years ago

>>> print('Four score and\nseven years ago')

Four score and

seven years ago

GUTENBERG\_DIVIDER = '(\\\*)+\\s\*(START|END) OF THIS PROJECT GUTENBERG.\*$'

(\\\*) -- “I really mean star”

(\\\*)+ -- one ore more stars

\\s\* -- one ore more spaces

(START|END) -- either one

OF THIS PROJECT GUTENBERG -- exact text

.\* -- any characters after that

$ -- to the end of the line

>>> filename = 'pnp.txt'

>>> file = open(filename, 'r')

>>> file

<open file 'pnp.txt', mode 'r' at 0x1004bdc68>

>>> text = file.read()

>>> text[0:90] #read first 90 characters, just text would read entirety

'The Project Gutenberg EBook of Pride and Prejudice, by Jane Austen\r\n\r\nThis eBook is for th'

text = re.sub('\r?\n', '\n', text)

replaces extra characters added by Windows

>>> GUTENBERG\_DIVIDER = '(\\\*)+\\s\*(START|END) OF THIS PROJECT GUTENBERG.\*$'

>>> header\_match = re.search(GUTENBERG\_DIVIDER, text, re.MULTILINE)

>>> header\_match.groups()

('\*', 'START')

>>> header\_match.group(0)

'\*\*\* START OF THIS PROJECT GUTENBERG EBOOK PRIDE AND PREJUDICE \*\*\*'

PROPER\_NAME\_PATT = '(?<=[a-z])[-\\s,\_]\*[\\s][-\\s,\_]\*([A-Z][a-z]+((-[A-Za-z]|\\s[A-Z])[a-z]+)\*)'

PROPER\_NAME\_PATT = '(?<=[a-z])[-\\s,\_]\*[\\s][-\\s,\_]\*

([A-Z] 🡪 capital letter

[a-z]+ 🡪 one or more lowercase letters

((-[A-Za-z] 🡪 might not have last name

|\\s[A-Z])[a-z]+)\*)' 🡪 or might have last name following same pattern

text = text[header\_match.end()+1 : len(text)]

text is everything from the end of the header to the length of text

**Web Scraping**

<http://pitchfork.com/artists/1742-godspeed-you-black-emperor/>

urllib gives one page at a time

regular expressions allow us to parse that page

>>> from pitchfork\_scraper import \*

pretend that I just typed the contents of this file into Terminal

>>> art\_id = '1742-godspeed-you-black-emperor'

>>> gybe = Artist(art\_id)

>>> print(gybe)

Artist('1742-godspeed-you-black-emperor')

>>> gybe.albums()

set(['17283-allelujah-dont-bend-ascend', '3487-yanqui-uxo', '3489-lift-your-skinny-fists-like-antennas-to-heaven'])

>>> Artist.url\_patt % art\_id

'http://pitchfork.com/artists/1742-godspeed-you-black-emperor/'

>>> art\_id

'1742-godspeed-you-black-emperor'

>>> Artist.url\_patt

'http://pitchfork.com/artists/%s/'

fid.close() 🡪 closes connection to website

To import webpage content:

fid = urllib.urlopen(‘actualurl’)

text = fid.read()

fid.close()

text

re.search(‘<[^>]+>’, text).groups(0)

Finds metadata

re.findall(‘<[^>]+>’, text).groups(0)

Finds all metadata

text = re.sub(‘<[^>]+>’, ‘’, text)

Removes metadata

re.findall(‘\\w+’, text)

Grabs words, not punctuation, not spaces

words = re.findall(‘\\w+’, text)

can then create dictionary & generate word frequencies

dictionary: maps a value to a key

example = dict({‘alice’ : 26, ‘bob’ : 33})

bob = key, 33 = value

For more complex parsing, there are html parsing libraries, re is not best