Python Import

import nltk	
nltk.download()	This step will bring up a window in which you can download ‘All Corpora’
from nltk.book import *	

BASICS

Tokens

text1[0:100]	-first 101 tokens
text2[5]	- fifth token

Concordance

> text3.concordance(‘begat’)	- basic keyword-in-context
text1.concordance(‘sea’, lines=100)	-- show other than default 25 lines
> text1.concordance(‘sea’, lines=100)	- show other than default 25 lines
text1.concordance(‘sea’, lines=all)	- show all results
text1.concordance(‘sea’, 10, lines=all) -	- change left and right context width to 10 characters and show all results

common_contexts

text1.common_contexts([‘sea’,’ocean’])

COUNTING

COUNTING

Count a String	len(‘this is a string of text’) – number of characterslen(‘this is a string of text’) – number of characters
Count a list of tokens	len(text1) –number of tokens
Make and Count a list of unique tokens	len(set(text1)) – notice that set return a list of unique tokens
Count Occurences	text1.count(‘heaven’) – how many times does a word occur?
Frequency	
	fd = nltk.FreqDist(text1) – creates a new data object that contains information about word frequency
	fd[‘the’] – how many occurences of the word ‘the’
	fd.keys() – show the keys in the data object
	fd.values() – show the values in the data object

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com

COUNTING (cont)

	fd.items() – show everything
	fd.keys()[0:50] – just show a portion of the info
Frequency Plots	fd.plot(50,cumulative=False) – generate a chart of the 50 most frequent words
Other FreqDist functions	fd.hapaxes()
	fd.freq('the')
Get word lengths	lengths = [len(w) for w in text1]
And do FreqDist	fd = nltk.FreqDist(lengths)
FreqDist as Table	fd.tabulate()

PARTS OF SPEACH CODES

CC Coordinating conjunction
CD Cardinal number
DT Determiner
EX Existential there
FW Foreign word
IN Preposition or subordinating conjunction
JJ Adjective
JJR Adjective, comparative
JJS Adjective, superlative
LS List item marker
MD Modal
NN Noun, singular or mass

PARTS OF SPEACH CODES

NNS Noun, plural
NNP Proper noun, singular
NNPS Proper noun, plural
PDT Predeterminer
POS Possessive ending
PRP Personal pronoun
PRP$ Possessive pronoun
RB Adverb
RBR Adverb, comparative
RBS Adverb, superlative
RP Particle
SYM Symbol
TO to

PARTS OF SPEACH CODES

UH Interjection
VB Verb, base form
VBD Verb, past tense
VBG Verb, gerund or present participle
VBN Verb, past participle
VBP Verb, non-3rd person singular present
VBZ Verb, 3rd person singular present
WDT Wh-determiner
WP Wh-pronoun
WP$ Possessive wh-pronoun
WRB Wh-adverb

NORMALIZING

De-punctuate	[w for w in text1 if w.isalpha()] – not so much getting rid of punctuation, but keeping alphabetic characters
De-uppercaseify (?)	>[w.lower() for w in text] – make each word in the tokenized list lowercase
	[w.lower() for w in text if w.isalpha()] – all in one go
Sort	sorted(text1) – careful with this!
Unique Words	set(text1) – set is oddly named, but very powerful. Leaves you with a list of only one of each word.
Exclude Stopwords	Make your own list of word to be excluded:
	stopwords = ['the','it','she','he']
	mynewtext = [w for w in text1 if w not in stopwords]
	Or you can also use predefined stopword lists from NLTK:

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com

NORMALIZING (cont)

from nltk.corpus import stopwords
stopwords = stopwords.words('english')
mynewtext = [w for w in text1 if w not in stopwords]

SEARCHING

Dispersion Plot	text4.dispersion_plot(['American','Liberty','Government'])
Find Word that ends with...	[w for w in text4 if w.endswith('ness')]
Find Word that start with...	[w for w in text4 if w.startsswith('ness')]
Find Word that contain...	[w for w in text4 if 'ee' in w]
Combine them together	[w for w in text4 if 'ee' in w and w.endswith('ing')] Regular expressions 'Regular expressions' is a syntax for describing sequences
Regular Expressions	'Regular expressions' is a syntax for describing sequences of characters usually used to construct search queries. The Python 're' module must first be imported:
Import	

SEARCHING (cont)

>>>import re >>>[w for w in text1 if re.search('^ab',w)] – 'Regular expressions' is too big of a topic to cover here. Google it!

CHUNKING

	>>>import re >>>[w for w in text1 if re.search('^ab',w)] – 'Regular expressions' is too big of a topic to cover here. Google it!
Collocations	> text4.collocations() - multi-word expressions that commonly co-occur. Notice that is not necessarily related to the frequency of the words
	>text4.collocations(num=100) – alter the number of phrases returned Bigrams, Trigrams, and n-grams are useful for comparing texts, particularly for plagiarism detection and collation
Bi-grams	>nltk.bigrams(text4) – returns every string of two words
Tri-grams	nltk.trigrams(text4) – return every string of three word
n-grams	nltk.ngrams(text4, 5)

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com

TAGGING

part-of-speach tagging	mytext = nltk.word_tokenize(“This is my sentence”)
	nltk.pos_tag(mytext)

Working with your own texts:

Open a file for reading	>file = open(‘myfile.txt’) – make sure you are in the correct directory before starting Python
Read the file	t = file.read();
Tokenize the file	tokens = nltk.word_tokenize(t)
Convert to NLTK text object	text = nltk.Text(tokens)

QUITTING PYTHON

Quit	quit()

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com