A SIMPLE INTRO TO *NLTK*

For a full intro, look at the book: http://www.nltk.org/book/

Download data

```
>>> import nltk
>>> nltk.download()
```

Arrays in Python

```
>>> saying = ['After', 'all', 'is', 'said', 'and', 'done', 'more', 'is', 'said', 'than',
'done']
>>> tokens = list(set(saying))
>>> tokens2 = sorted(tokens)
>>> tokens
>>> tokens2
>>> tokens[0]
>>> tokens[0:2]
>>> tokens[-2:]
>>> tokens2[-2:]
```

Num of tokens; num of unique words; frequencies; cumulative graph

```
>>> from nltk.book import *
>>> len(text1)   # num. tokens
260819
>>> fdist1 = FreqDist(text1)     # FreqDist() viene importata da nltk.book...
>>> len(fdist1) # num. unique words; i.e., num. word types
19317
>>> fdist1.most_common(50)
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024),
('a', 4569), ('to', 4542), (';', 4072), ('in', 3916), ('that', 2982),
("'", 2684), ('-', 2552), ('his', 2459), ('it', 2209), ('I', 2124),
('s', 1739), ('is', 1695), ('he', 1661), ('with', 1659), ('was', 1632),
('as', 1620), ('"', 1478), ('all', 1462), ('for', 1414), ('this', 1280),
('!', 1269), ('at', 1231), ('by', 1137), ('but', 1113), ('not', 1103),
('--', 1070), ('him', 1058), ('from', 1052), ('be', 1030), ('on', 1005),
('so', 918), ('whale', 906), ('one', 889), ('you', 841), ('had', 767),
('have', 760), ('there', 715), ('But', 705), ('or', 697), ('were', 680),
('now', 646), ('which', 640), ('?', 637), ('me', 627), ('like', 624)]
>>> fdist1['whale']
906
>>> fdist1.freq('whale')
0.003473673313677301
>>> fdist1.plot(50)  # can you see the Zipf's law?
>>> fdist1.plot(50, cumulative=True)
```

Cumulative Frequency Plot for 50 Most Frequently Words in Moby Dick: these account for nearly half of the tokens

Find long tokens

```
>>> V = set(text1)
>>> long_words = [w for w in V if len(w) > 15]  # the 'list comprehension' syntax
>>> sorted(long_words)
['CIRCUMNAVIGATION', 'Physiognomically', 'apprehensiveness', 'cannibalistically',
'characteristically', 'circumnavigating', 'circumnavigation', 'circumnavigations',
'comprehensiveness', 'hermaphroditical', 'indiscriminately', 'indispensableness',
'irresistibleness', 'physiognomically', 'preternaturalness', 'responsibilities',
'simultaneousness', 'subterraneousness', 'supernaturalness', 'superstitiousness',
'uncomfortableness', 'uncompromisedness', 'undiscriminating', 'uninterpenetratingly']
```

Find most frequent word types wigh length>7 and count>7

```
>>> fdist5 = FreqDist(text5)
>>> sorted(w for w in set(text5) if len(w) > 7 and fdist5[w] > 7)
['#14-19teens', '#talkcity_adults', '((((((((((', '........', 'Question',
'actually', 'anything', 'computer', 'cute.-ass', 'everyone', 'football',
'innocent', 'listening', 'remember', 'seriously', 'something', 'together',
'tomorrow', 'watching']
```

Bigrams

```
>>> from nltk.util import bigrams
>>> list(bigrams(['more', 'is', 'said', 'than', 'done']))
[('more', 'is'), ('is', 'said'), ('said', 'than'), ('than', 'done')]
```

Counting token lengths and length distribution

```
>>> [len(w) for w in text1]
[1, 4, 4, 2, 6, 8, 4, 1, 9, 1, 1, 8, 2, 1, 4, 11, 5, 2, 1, 7, 6, 1, 3, 4, 5, 2, ...]
>>> fdist = FreqDist(len(w) for w in text1)
>>> fdist.most_common()
[(3, 50223), (1, 47933), (4, 42345), (2, 38513), (5, 26597), (6, 17111), (7, 14399),
(8, 9966), (9, 6428), (10, 3528), (11, 1873), (12, 1053), (13, 567), (14, 177),
(15, 70), (16, 22), (17, 12), (18, 1), (20, 1)]
>>> fdist.max()   #most frequent word length is 3
3
>>> fdist[3]     # count corresponding to length=3
50223
>>> fdist.freq(3)      # frequency corresponding to length=3
0.19255882431878046
```

CORPORA: Gutenberg - number of tokens into the austen-emma.txt file

```
>>> import nltk
>>> nltk.corpus.gutenberg.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt',
'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt',
'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt',
'chesterton-thursday.txt', 'edgeworth-parents.txt', 'melville-moby_dick.txt',
'milton-paradise.txt', 'shakespeare-caesar.txt', 'shakespeare-hamlet.txt',
'shakespeare-macbeth.txt', 'whitman-leaves.txt']
>>> emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
192427
```

This program displays three statistics for each text: average word length, average sentence length, and the number of times each vocabulary item appears in the text on average (our lexical diversity score)

```
>>> for fileid in gutenberg.fileids():
...     num_chars = len(gutenberg.raw(fileid))
...     num_words = len(gutenberg.words(fileid))
...     num_sents = len(gutenberg.sents(fileid))
...     num_vocab = len(set(w.lower() for w in gutenberg.words(fileid)))
...     print(round(num_chars/num_words), round(num_words/num_sents), round(num_words/
num_vocab), fileid)
...
5 25 26 austen-emma.txt
5 26 17 austen-persuasion.txt
5 28 22 austen-sense.txt
4 34 79 bible-kjv.txt
5 19 5 blake-poems.txt
4 19 14 bryant-stories.txt
4 18 12 burgess-busterbrown.txt
4 20 13 carroll-alice.txt
5 20 12 chesterton-ball.txt
5 23 11 chesterton-brown.txt
5 18 11 chesterton-thursday.txt
4 21 25 edgeworth-parents.txt
5 26 15 melville-moby_dick.txt
5 52 11 milton-paradise.txt
4 12 9 shakespeare-caesar.txt
4 12 8 shakespeare-hamlet.txt
4 12 7 shakespeare-macbeth.txt
5 36 12 whitman-leaves.txt
```

CORPORA: Brown - count modal verbs; tabulate the distribution (actually, counts) of words given category

```
>>> from nltk.corpus import brown
>>> news_text = brown.words(categories='news')
>>> fdist = FreqDist(w.lower() for w in news_text)
>>> modals = ['can', 'could', 'may', 'might', 'must', 'will']
>>> for m in modals:
...     print(m + ':', fdist[m], end=' ')
...
can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
>>> cfd = nltk.ConditionalFreqDist(
...     (genre, word)
...     for genre in brown.categories()
...     for word in brown.words(categories=genre))
>>> genres = ['news', 'religion', 'hobbies', 'science_fiction', 'romance', 'humor']
>>> cfd.tabulate(conditions=genres, samples=modals)
                  can could  may might must will
            news   93   86   66   38   50  389
        religion   82   59   78   12   54   71
         hobbies  268   58  131   22   83  264
science_fiction   16   49    4   12    8   16
         romance   74  193   11   51   45   43
           humor   16   30    8    8    9   13
>>> cfd['news'].freq('can')   # P(sample='can' | condition='news')
0.0009248761859299481
>>> dist = FreqDist(brown.words(categories='news'))
>>> dist.freq('can')
0.0009248761859299481
```