Assignment 2

Chapter 3

In [30]:
```
import nltk
import re
from nltk import word_tokenize, sent_tokenize
import pprint
```

In [4]:
```
#### 1.Describe the class of strings matched by the following regular expressions.No code is neede

#### a.[a-zA-Z]+
#### b.[A-Z][a-z]*
#### c.p[aeiou]{,2}t
#### d.\d+(\.\d+)?
#### e.([^aeiou][aeiou][^aeiou])*
#### f.\w+(?:[-']\w+)*|[-.(]+|\S\w*
```

In [ ]:
```
#### a. String has one or more instances of uppercase or lowercase letters
#### b. String has one instance of A-Z(uppercase) and then zero or more instances of a-z(lowercase
#### c. String starts with the letter p, followed by two vowels, and ends with t
#### d. String contains one or more digits followed by a . and then followed by one or more digits
####    the . can happen once or not at all
#### e. String contains zero or more instances of not starting with a vowel, then containing a vow
####    not ending with a vowel
#### f. String contains one or more word character and then groups but does not capture any charac
####    followed by one or more word characters - this can be done zero or more times or expressio
####    the -,., or ( OR contains zero or more instances of a character that is not a white space
```

2.Rewrite the following loop as a list comprehension:

In [5]:
```
# sent = ['This', 'is', 'an', 'introduction', 'class']
#result = []
# for word in sent:
#    word_len = (word, len(word))
#    result.append(word_len)
# result
```

Out[5]:
```
[('This', 4), ('is', 2), ('an', 2), ('introduction', 12), ('class', 5)]
```

In [133…
```
sent = ['This', 'is', 'an', 'introduction', 'class']
result = [(word, len(word)) for word in sent]
result
```

Out[133…
```
[('This', 4), ('is', 2), ('an', 2), ('introduction', 12), ('class', 5)]
```

3.Read in some text from your own document in your local disk, tokenize it, and use the regular expressions to print the list of all wh-word types that occur. (wh-words in English are used in questions, relative clauses and exclamations: who, which, what, and so on.) (hint: import nltk; import re; from nltk import word_tokenize) (use lower() and set())

In [1]:
```
doc = open('sample_text_2_3.txt')
```

In [9]:
```
doc.read()
```

Out[9]:
```
'What questions often run through my head? What will I eat tonight? When is the sky green? Why doe
s the sun not have a smile like it does on the Raisin Bran box. These are the thoughts in which I
am at odds with, and need to understand.'
```

```
In [10]: raw_doc = open('sample_text_2_3.txt').read()
         raw_doc
```

```
Out[10]: 'What questions often run through my head? What will I eat tonight? When is the sky green? Why doe
         s the sun not have a smile like it does on the Raisin Bran box. These are the thoughts in which I
         am at odds with, and need to understand.'
```

```
In [26]: doc_tokens = word_tokenize(raw_doc)
         set(t for t in doc_tokens if re.search('^wh', t.lower()))
```

```
Out[26]: {'What', 'When', 'Why', 'which'}
```

4. Create your own file consisting of words and (made up) frequencies, where each line consists of a word, the space character, and a positive integer, e.g. fuzzy 53. Read the file into a Python list using open(filename).readlines(). Next, break each line into its two fields using split(), and convert the number into an integer using int(). The result should be a list of the form: [['fuzzy', 53], ...].

```
In [30]: new_list = []
         text_4 = open('text_2_4.txt').readlines()
```

```
In [31]: for line in text_4:
             line = line.split()
             new_list.append([line[0],int(line[1])])
```

```
In [32]: new_list
```

```
Out[32]: [['kermit', 12], ['taboo', 94], ['yahoo', 23], ['wahoo', 76], ['monkeypox', 9]]
```

5 .Readability measures are used to score the reading difficulty of a text, for the purposes of selecting texts of appropriate difficulty for language learners. Let us define μw to be the average number of letters per word, and μs to be the average number of words per sentence, in a given text. The Automated Readability Index (ARI) of the text is defined to be: 4.71 μw + 0.5 μs - 21.43. Compute the ARI score for each section of the Brown Corpus (i.e. News, Editorial,..., Humor). Please use two ways introduced in the class to calculate the average number of letters per word μw and the average number of words per sentences μs. You are supposed to get two different results.

```
In [151…: from nltk.corpus import brown
          for category in nltk.corpus.brown.categories():
              num_chars = len(nltk.corpus.brown.raw(categories = category))
              num_words = len(nltk.corpus.brown.words(categories = category))
              num_sents = len(nltk.corpus.brown.sents(categories = category))
              mu_w = (num_chars) / (num_words)
              mu_s = (num_words) / (num_sents)
              print(category, ((4.71 * mu_w) + (.5 * mu_s) - 21.43))
```

```
adventure 24.40926324686098
belles_lettres 31.095382353662615
editorial 30.03411587902211
fiction 25.150998441603406
government 32.8804612028414
hobbies 29.139922001742477
humor 28.06731665647837
learned 31.912251710365197
lore 30.387050143555797
mystery 24.159818927038003
news 30.83395561360988
religion 30.382863961972696
reviews 31.13075310191683
romance 24.713170386382934
science_fiction 25.292578032827997
```

In [150…
```
for category in nltk.corpus.brown.categories():
    words = nltk.corpus.brown.words(categories = category)
    sents = nltk.corpus.brown.sents(categories = category)
    mu_w = sum(len(w) for w in words) / len(words)
    mu_s = sum(len(s) for s in sents) / len(sents)
    print(category, ((4.71 * mu_w) + (.5 * mu_s) - 21.43))
```

```
adventure 4.0841684990890705
belles_lettres 10.987652885621749
editorial 9.471025332953673
fiction 4.9104735321302115
government 12.08430349501021
hobbies 8.922356393630267
humor 7.887805248319808
learned 11.926007043317348
lore 10.254756197101155
mystery 3.8335518942055167
news 10.176684595052684
religion 10.203109907301261
reviews 10.769699888473433
romance 4.34922419804213
science_fiction 4.978058336905399
```

I tried to use the slides but not sure if these produced the calculations you were hoping. Hope to go over this question in class as it took me a long time to finally get.

6.Use the Porter Stemmer to normalize some tokenized text (see below), calling the stemmer on each word. Do the same thing with the Lancaster Stemmer and describe any difference you observe by using these two stemmers. Finally, please try Lemmatization by using WordNet Lemmatizer and describe any difference you observe compared to Porter Stemmer and Lancaster Stemmer.

text='Technologies based on NLP are becoming increasingly widespread. For example, phones and handheld computers support predictive text and handwriting recognition; web search engines give access to information locked up in unstructured text; machine translation allows us to retrieve texts written in Chinese and read them in Spanish; text analysis enables us to detect sentiment in tweets and blogs. By providing more natural human-machine interfaces, and more sophisticated access to stored information, language processing has come to play a central role in the multilingual information society'

In [9]:
```
porter = nltk.PorterStemmer()
lancaster = nltk.LancasterStemmer()
```

In [7]:
```
text_6 ='Technologies based on NLP are becoming increasingly widespread. For example, phones and h
```

In [11]:
```
tokens_6 = word_tokenize(text_6)
```

In [22]:
```
[porter.stem(t) for t in tokens_6]
```

Out[22]:
```
['technolog',
 'base',
 'on',
 'nlp',
 'are',
 'becom',
 'increasingli',
 'widespread',
 '.',
 'for',
 'exampl',
 ',',
 'phone',
 'and',
 'handheld',
 'comput',
 'support',
 'predict',
 'text',
 'and',
 'handwrit',
 'recognit',
 ';',
 'web',
 'search',
 'engin',
 'give',
 'access',
 'to',
 'inform',
 'lock',
 'up',
 'in',
 'unstructur',
 'text',
 ';',
 'machin',
 'translat',
 'allow',
 'us',
 'to',
 'retriev',
 'text',
 'written',
 'in',
 'chines',
 'and',
 'read',
 'them',
 'in',
 'spanish',
 ';',
 'text',
 'analysi',
 'enabl',
 'us',
 'to',
 'detect',
 'sentiment',
 'in',
 'tweet',
 'and',
 'blog',
 '.',
 'By',
 'provid',
 'more',
 'natur',
 'human-machin',
 'interfac',
 ',',
 'and',
 'more',
 'sophist',
 'access',
 'to',
 'store',
 'inform',
 ',',
 'languag',
 'process',
 'ha',
 'come',
 'to',
 'play',
 'a',
 'central',
 'role',
 'in',
 'the',
 'multilingu',
 'inform',
 'societi']
```

```
In [21]: [lancaster.stem(t) for t in tokens_6]
```

```
Out[21]: ['technolog',
 'bas',
 'on',
 'nlp',
 'ar',
 'becom',
 'increas',
 'widespread',
 '.',
 'for',
 'exampl',
 ',',
 'phon',
 'and',
 'handheld',
 'comput',
 'support',
 'predict',
 'text',
 'and',
 'handwrit',
 'recognit',
 ';',
 'web',
 'search',
 'engin',
 'giv',
 'access',
 'to',
 'inform',
 'lock',
 'up',
 'in',
 'unstruct',
 'text',
 ';',
 'machin',
 'transl',
 'allow',
 'us',
 'to',
 'retriev',
 'text',
 'writ',
 'in',
 'chines',
 'and',
 'read',
 'them',
 'in',
 'span',
 ';',
 'text',
 'analys',
 'en',
 'us',
 'to',
 'detect',
 'senty',
 'in',
 'tweet',
 'and',
 'blog',
 '.',
 'by',
 'provid',
 'mor',
 'nat',
 'human-machine',
 'interfac',
 ',',
 'and',
 'mor',
 'soph',
 'access',
 'to',
 'stor',
 'inform',
 ',',
 'langu',
 'process',
 'has',
 'com',
 'to',
 'play',
 'a',
 'cent',
 'rol',
 'in',
 'the',
 'multil',
 'inform',
 'socy']
```

ANSWER:

Porter seems like a much less aggressive stemmer because it cuts off words much later on than with lancaster that cuts off words much earlier.

In [24]:
```
wnl = nltk.WordNetLemmatizer()
```

In [25]:
```
[wnl.lemmatize(t) for t in tokens_6]
```

Out[25]:
```
['Technologies',
 'based',
 'on',
 'NLP',
 'are',
 'becoming',
 'increasingly',
 'widespread',
 '.',
 'For',
 'example',
 ',',
 'phone',
 'and',
 'handheld',
 'computer',
 'support',
 'predictive',
 'text',
 'and',
 'handwriting',
 'recognition',
 ';',
 'web',
 'search',
 'engine',
 'give',
 'access',
 'to',
 'information',
 'locked',
 'up',
 'in',
 'unstructured',
 'text',
 ';',
 'machine',
 'translation',
 'allows',
 'u',
 'to',
 'retrieve',
 'text',
 'written',
 'in',
 'Chinese',
 'and',
 'read',
 'them',
 'in',
 'Spanish',
 ';',
 'text',
 'analysis',
 'enables',
 'u',
 'to',
 'detect',
 'sentiment',
 'in',
 'tweet',
 'and',
 'blog',
 '.',
 'By',
 'providing',
 'more',
 'natural',
 'human-machine',
 'interface',
 ',',
 'and',
 'more',
 'sophisticated',
 'access',
 'to',
 'stored',
 'information',
 ',',
 'language',
 'processing',
 'ha',
 'come',
 'to',
 'play',
 'a',
 'central',
 'role',
 'in',
 'the',
 'multilingual',
 'information',
 'society']
```

ANSWER:

This lemmatizer seems much less aggressive than the other 2 because it keeps words only if they are still a word in the dictionary after they have been reduced.

7. Please try to retreive some text from any web page in the form of HTML documents, tokenize the text, create an NLTK text and then use similar( ) function on any word of your interest.

In [148…

```
from urllib import request
from bs4 import BeautifulSoup
url='https://www.rogerebert.com/reviews/great-movie-on-the-waterfront-1954'
html = request.urlopen(url).read().decode('utf8')
raw_7 = BeautifulSoup(html, 'html.parser').get_text()
tokens_7 = word_tokenize(raw_7)
text_7 = nltk.Text(tokens_7)
text_7.similar('film')
```

```
waterfront tone acting godfather gun roof front way
```

8.Rewrite the following nested loop by using a list comprehension and regular expressions:

In [1]:

```
  words = ['attribution', 'confabulation', 'elocution',
         'sequoia', 'tenacious', 'unidirectional']
  vsequences = set()
  for word in words:
      vowels = []
      for char in word:
          if char in 'aeiou':
              vowels.append(char)
      vsequences.add(''.join(vowels))
  sorted(vsequences)
```

Out[1]:

```
['aiuio', 'eaiou', 'eouio', 'euoia', 'oauaio', 'uiieioa']
```

In [147…

```
words = ['attribution', 'confabulation', 'elocution', 'sequoia', 'tenacious', 'unidirectional']
sorted([''.join([char for char in word if re.search('[aeiou]',char)]) for word in words])
```

Out[147…

```
['aiuio', 'eaiou', 'eouio', 'euoia', 'oauaio', 'uiieioa']
```

9.Try to refer the following sample code to print the following sentences in a formatted way.(Hint: you should use str.format() method in print() and for loop; For more information, please read the textbook section 3.9 in chapter 3)

Output should exactly look:

In [64]:

```
The Tragedie of Hamlet was written by William Shakespeare in 1599
```

```
Leaves of Grass        was written by Walt Whiteman       in 1855
Emma                   was written by Jane Austen         in 1816
# sample code:
template = 'Lee wants a {} right now'
menu = ['sandwich', 'spam fritter', 'pancake']
for snack in menu:
    print(template.format(snack))
```

```
Lee wants a sandwich right now
Lee wants a spam fritter right now
Lee wants a pancake right now
```

In [146…

```
title = ['The Tragedie of Hamlet', 'Leaves of Grass','Emma']
author = ['William Shakespeare','Walt Whitman','Jane Austen']
year = ['1599', '1855', '1816']
template_9 = '{:22s} was written by {:19s} in {}'
for i in range (len(title)):
    print(template_9.format(title[i],author[i],year[i]))
```

```
The Tragedie of Hamlet was written by William Shakespeare in 1599
Leaves of Grass        was written by Walt Whitman        in 1855
Emma                   was written by Jane Austen         in 1816
```

10. Please first use sentence tokenization to print out the first 10 sentences in the "carroll-alice.txt" in the Gutenberg corpus. And then use basic corpus functionality sents() to return the first 10 sentences in this book. Please describe the difference between the two results. (hint: use sent_tokenzie (), pprint(), sents())

In [144…

```
alice_text = nltk.corpus.gutenberg.raw('carroll-alice.txt')
alice_sents = sent_tokenize(alice_text)
pprint.pprint(alice_sents[:10])
```

```
["[Alice's Adventures in Wonderland by Lewis Carroll 1865]\n\nCHAPTER I.",
 'Down the Rabbit-Hole\n'
 '\n'
 'Alice was beginning to get very tired of sitting by her sister on the\n'
 'bank, and of having nothing to do: once or twice she had peeped into the\n'
 'book her sister was reading, but it had no pictures or conversations in\n'
 "it, 'and what is the use of a book,' thought Alice 'without pictures or\n"
 "conversation?'",
 'So she was considering in her own mind (as well as she could, for the\n'
 'hot day made her feel very sleepy and stupid), whether the pleasure\n'
 'of making a daisy-chain would be worth the trouble of getting up and\n'
 'picking the daisies, when suddenly a White Rabbit with pink eyes ran\n'
 'close by her.',
 'There was nothing so VERY remarkable in that; nor did Alice think it so\n'
 "VERY much out of the way to hear the Rabbit say to itself, 'Oh dear!",
 'Oh dear!',
 "I shall be late!'",
 '(when she thought it over afterwards, it\n'
 'occurred to her that she ought to have wondered at this, but at the time\n'
 'it all seemed quite natural); but when the Rabbit actually TOOK A WATCH\n'
 'OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on,\n'
 'Alice started to her feet, for it flashed across her mind that she had\n'
 'never before seen a rabbit with either a waistcoat-pocket, or a watch\n'
 'to take out of it, and burning with curiosity, she ran across the field\n'
 'after it, and fortunately was just in time to see it pop down a large\n'
 'rabbit-hole under the hedge.',
 'In another moment down went Alice after it, never once considering how\n'
 'in the world she was to get out again.',
 'The rabbit-hole went straight on like a tunnel for some way, and then\n'
 'dipped suddenly down, so suddenly that Alice had not a moment to think\n'
 'about stopping herself before she found herself falling down a very deep\n'
 'well.',
 'Either the well was very deep, or she fell very slowly, for she had\n'
 'plenty of time as she went down to look about her and to wonder what was\n'
 'going to happen next.']
```

```
In [145…  alice = nltk.corpus.gutenberg.sents('carroll-alice.txt')
          alice[:12]
```

```
Out[145…  [['[',
            'Alice',
            "'",
            's',
            'Adventures',
            'in',
            'Wonderland',
            'by',
            'Lewis',
            'Carroll',
            '1865',
            ']'],
           ['CHAPTER', 'I', '.'],
           ['Down', 'the', 'Rabbit', '-', 'Hole'],
           ['Alice',
            'was',
            'beginning',
            'to',
            'get',
            'very',
            'tired',
            'of',
            'sitting',
            'by',
            'her',
            'sister',
            'on',
            'the',
            'bank',
            ',',
            'and',
            'of',
            'having',
            'nothing',
            'to',
            'do',
            ':',
            'once',
            'or',
            'twice',
            'she',
            'had',
            'peeped',
            'into',
            'the',
            'book',
            'her',
            'sister',
            'was',
            'reading',
            ',',
            'but',
            'it',
            'had',
            'no',
            'pictures',
            'or',
            'conversations',
            'in',
            'it',
            ',',
            "'",
            'and',
            'what',
            'is',
            'the',
            'use',
            'of',
            'a',
            'book',
            ",'",
            'thought',
            'Alice',
            "'",
            'without',
            'pictures',
            'or',
            'conversation',
            "?'"],
           ['So',
            'she',
            'was',
            'considering',
            'in',
            'her',
            'own',
            'mind',
            '(',
            'as',
            'well',
            'as',
            'she',
            'could',
            ',',
            'for',
            'the',
            'hot',
            'day',
            'made',
            'her',
            'feel',
            'very',
            'sleepy',
            'and',
            'stupid',
            '),',
            'whether',
            'the',
            'pleasure',
            'of',
            'making',
            'a',
            'daisy',
            '-',
            'chain',
            'would',
            'be',
            'worth',
            'the',
            'trouble',
            'of',
            'getting',
            'up',
            'and',
            'picking',
            'the',
            'daisies',
            ',',
            'when',
            'suddenly',
            'a',
            'White',
            'Rabbit',
            'with',
            'pink',
            'eyes',
            'ran',
            'close',
            'by',
            'her',
            '.'],
           ['There',
            'was',
            'nothing',
            'so',
            'VERY',
            'remarkable',
            'in',
            'that',
            ';',
            'nor',
            'did',
            'Alice',
            'think',
            'it',
            'so',
            'VERY',
            'much',
            'out',
            'of',
            'the',
            'way',
            'to',
            'hear',
            'the',
            'Rabbit',
            'say',
            'to',
            'itself',
            ',',
            "'",
            'Oh',
            'dear',
            '!'],
           ['Oh', 'dear', '!'],
           ['I', 'shall', 'be', 'late', "!'"],
           ['(',
            'when',
            'she',
            'thought',
            'it',
            'over',
            'afterwards',
            ',',
            'it',
            'occurred',
            'to',
            'her',
            'that',
            'she',
            'ought',
            'to',
            'have',
            'wondered',
            'at',
            'this',
            ',',
            'but',
            'at',
            'the',
            'time',
            'it',
            'all',
            'seemed',
            'quite',
            'natural',
            ');',
            'but',
            'when',
            'the',
            'Rabbit',
            'actually',
            'TOOK',
            'A',
            'WATCH',
            'OUT',
            'OF',
            'ITS',
            'WAISTCOAT',
            '-',
            'POCKET',
            ',',
            'and',
            'looked',
            'at',
            'it',
            ',',
            'and',
            'then',
            'hurried',
            'on',
            ',',
            'Alice',
            'started',
            'to',
            'her',
            'feet',
            ',',
            'for',
            'it',
            'flashed',
            'across',
            'her',
            'mind',
            'that',
            'she',
            'had',
            'never',
            'before',
            'seen',
            'a',
            'rabbit',
            'with',
            'either',
            'a',
            'waistcoat',
            '-',
            'pocket',
            ',',
            'or',
            'a',
            'watch',
            'to',
            'take',
            'out',
            'of',
            'it',
            ',',
            'and',
            'burning',
            'with',
            'curiosity',
            ',',
            'she',
            'ran',
            'across',
            'the',
            'field',
            'after',
            'it',
            ',',
            'and',
            'fortunately',
            'was',
            'just',
            'in',
            'time',
            'to',
            'see',
            'it',
            'pop',
            'down',
            'a',
            'large',
            'rabbit',
            '-',
            'hole',
            'under',
            'the',
            'hedge',
            '.'],
           ['In',
            'another',
            'moment',
            'down',
            'went',
            'Alice',
            'after',
            'it',
            ',',
            'never',
            'once',
            'considering',
            'how',
            'in',
            'the',
            'world',
            'she',
            'was',
            'to',
            'get',
            'out',
            'again',
            '.'],
           ['The',
            'rabbit',
            '-',
            'hole',
            'went',
            'straight',
            'on',
            'like',
            'a',
            'tunnel',
            'for',
            'some',
            'way',
            ',',
            'and',
            'then',
            'dipped',
            'suddenly',
            'down',
            ',',
            'so',
            'suddenly',
            'that',
            'Alice',
            'had',
            'not',
            'a',
            'moment',
            'to',
            'think',
            'about',
            'stopping',
            'herself',
            'before',
            'she',
            'found',
            'herself',
            'falling',
            'down',
            'a',
            'very',
            'deep',
            'well',
            '.'],
           ['Either',
            'the',
            'well',
            'was',
            'very',
            'deep',
            ',',
            'or',
            'she',
            'fell',
            'very',
            'slowly',
            ',',
            'for',
            'she',
            'had',
            'plenty',
            'of',
            'time',
            'as',
            'she',
            'went',
            'down',
            'to',
            'look',
            'about',
            'her',
            'and',
            'to',
            'wonder',
            'what',
            'was',
            'going',
            'to',
            'happen',
            'next',
            '.']]
```

ANSWER:

The difference in these two ways of retrieving the first 10 sentences is that the second way creates a list of sentences that each get printed on a separate line. This also does not show the new line character. In the first way we are using sentence segmentation to tokenize each sentence of the text. The sent_tokenize function also splits sentences in quotes into individual strings. With the sent() function we can get more granular with the data, and delve into each sentence of the first 10 in the book. We do not have that same liberty when using the sent_tokenize() function as it only creates one list