Chapter 1

You should install NLTK, import NlTK and download the NLTK Book Collection at the beginning.

In [63]:
```
import nltk
```

In [62]:
```
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Question 1.1 Find the collocations in text2 from NLTK Book Collection

In [64]:
```
text2.collocations()
```

```
Colonel Brandon; Sir John; Lady Middleton; Miss Dashwood; every thing;
thousand pounds; dare say; Miss Steeles; said Elinor; Miss Steele;
every body; John Dashwood; great deal; Harley Street; Berkeley Street;
Miss Dashwoods; young man; Combe Magna; every day; next morning
```

Question 1.2 Review the discussion of conditionals in 4. Find all unique words in the Chat Corpus (text5) ending with the letter l. Show the first 20 words in alphabetical order.

In [65]:
```
l_end = sorted(item.lower() for item in set(text5) if item.endswith('l') and item.isalpha())
l_end[:20]
```

Out[65]:
```
['al',
 'alcohol',
 'all',
 'anal',
 'angel',
 'animal',
 'anygirl',
 'april',
 'asl',
 'astral',
 'atl',
 'bacl',
 'bagel',
 'ball',
 'barrel',
 'bbl',
 'beautiful',
 'bell',
 'betrayal',
 'bisexual']
```

Question 1.3 What is the difference between the following two lines of codes? Which one will give a larger value and why? Will this be the case for other texts?

sorted(set(w.lower() for w in text2))

sorted(w.lower() for w in set(text2))

ANSWER:

The difference in these two lines of code is that the first uses the set function on the words once they become lower whereas the second uses the set() function on the whole text. The second line will yield more results because it is taking the set of all words in the text whether they are lower or not. This is the case for all texts because of where the set function is being placed, you are counting the unique words at different times before and after the .lower() function is being put to use.

In [30]:
```
len(sorted(set(w.lower() for w in text6)))
```

Out[30]: 1855

In [29]:
```
len(sorted(w.lower() for w in set(text6)))
```

Out[29]: 2166

Question 1.4 Find all the Twelve-letter words in the Chat Corpus (text5). With the help of a frequency distribution (FreqDist), show these words in decreasing order of frequency.

In [99]:
```
text_5_len_12 = FreqDist([w for w in text5 if w.isalpha() and len(w) == 12])
fdist = FreqDist(text_5_len_12)
fdist.most_common()
```

Out[99]:
```
[('conversation', 5),
 ('construction', 3),
 ('entertaining', 3),
 ('multitasking', 2),
 ('Considerably', 1),
 ('ihavehotnips', 1),
 ('christianity', 1),
 ('ooooooooooo', 1),
 ('disappointed', 1),
 ('outrageously', 1),
 ('neighborhood', 1),
 ('thanksgiving', 1),
 ('interruption', 1),
 ('alternatives', 1),
 ('hugsssssssss', 1),
 ('butterscotch', 1),
 ('constituents', 1),
 ('subscription', 1),
 ('catastrophic', 1),
 ('Constitution', 1),
 ('Christianity', 1),
 ('freeeezinggg', 1),
 ('bachelorette', 1),
 ('hahahahahaha', 1),
 ('sweeeeeeeeet', 1),
 ('heyyyyyyyyyy', 1),
 ('masturbating', 1),
 ('yummylicious', 1),
 ('denomination', 1),
 ('registration', 1),
 ('necromancers', 1),
 ('psychologist', 1),
 ('descriminate', 1),
 ('slkfjsldkfjs', 1),
 ('passionately', 1),
 ('heartbreaker', 1),
 ('Connecticutt', 1),
```

```
 ('tooooooooooo', 1),
 ('periodically', 1),
 ('Fergalicious', 1),
 ('hhaaaaatttee', 1),
 ('Blooooooooood', 1)]
```

I am using fdist.most_common() here so that the results are printed each on a new line and you can see all of them. I could also just print out the frequency distribution but that would not list all results.

Question 1.5 Review the discussion of looping with conditions in Section 4 in Chapter 1. Use a combination of for and if statements to loop over the tokens of text2 and print all the distinct numbers, one per line.

In [100…
```
sorted([t for t in set(text2) if t.isdigit()])
```

Out[100…
```
['1',
 '10',
 '11',
 '12',
 '13',
 '14',
 '15',
 '16',
 '17',
 '18',
 '1811',
 '19',
 '2',
 '20',
 '200',
 '21',
 '22',
 '23',
 '24',
 '25',
 '26',
 '27',
 '28',
 '29',
 '3',
 '30',
 '31',
 '32',
 '33',
 '34',
 '35',
 '36',
 '37',
 '38',
 '39',
 '4',
 '40',
 '41',
 '42',
 '43',
 '44',
 '45',
 '46',
 '47',
 '48',
 '49',
 '5',
 '50',
 '6',
```

```
'7',
'8',
'9']
```

Chapter 2

Question 2.1 Use Gutenberg Corpus Module to explore 'shakespeare-caesar.txt'.a. How many word tokens does this book have? b.How many word types? c. What is the lexical richness?

In [101…
```
caesar = nltk.corpus.gutenberg.words("shakespeare-caesar.txt")
print (f"a. Word Tokens:{len(caesar)}")
print (f"b. Word Types:{len(set(caesar))}")
print (f"c. Lexical Richness:{len(set(caesar))/len(caesar):.3f}")
```

```
a. Word Tokens:25833
b. Word Types:3560
c. Lexical Richness:0.138
```

Question 2.2 Explore the Section "Hobbies" in Brown Corpus. Count the number of all "wh-" words (words starting with "wh"). Please make sure we are not double-counting words like “What” and “what”, which differ only in capitalization.

In [102…
```
from nltk.corpus import brown
```

In [103…
```
hobbies_text = brown.words(categories = 'hobbies')
```

In [104…
```
wh_fdist = nltk.FreqDist(w.lower() for w in hobbies_text if w.lower().startswith('wh'))
```

I am using wh_fdist.most_common() here so that the results are printed each on a new line and you can see all of them. I could also just print out the frequency distribution but that would not list all results.

In [105…
```
wh_fdist.most_common()
```

Out[105…
```
[('which', 253),
 ('when', 164),
 ('what', 108),
 ('who', 104),
 ('where', 77),
 ('while', 38),
 ('whole', 20),
 ('why', 17),
 ('white', 15),
 ('whose', 14),
 ('whether', 14),
 ('whatever', 6),
 ('wheel', 4),
 ('wheels', 3),
 ('wherever', 3),
 ('whenever', 2),
 ('whom', 2),
 ('wheeled', 2),
 ('whitetail', 2),
 ('whipped', 2),
 ('wholly', 2),
 ('whereas', 2),
 ('whaddya', 1),
 ('wholesale', 1),
 ('wherein', 1),
 ('whippet', 1),
 ('whiskey', 1),
 ('whips', 1),
```

```
 ('whip', 1),
 ('whaling', 1),
 ('whirlwind', 1),
 ('whole-house', 1),
 ("what's", 1),
 ('whereby', 1)]
```

Question 2.3 Explore movie reviews corpus which contains 2k movie reviews with sentiment polarity classification (positive or negative reviews). Please find out the 20 most common words in the negative reviews and positive reviews separately,please get rid of stop words, number and punctuations from reviews. Please make sure we are not double-counting words like “This” and “this”, which differ only in capitalization.

Hint:

from nltk.corpus import movie_reviews

from nltk.corpus import stopwords

In [106…
```
from nltk.corpus import movie_reviews
from nltk.corpus import stopwords
from nltk import FreqDist
```

In [107…
```
stopwords = nltk.corpus.stopwords.words('english')
negative_reviews = movie_reviews.words(categories = 'neg')
positive_reviews = movie_reviews.words(categories = 'pos')
```

In [108…
```
neg_words = FreqDist([w for w in negative_reviews if (w.lower() not in stopwords and w.isalpha())])
neg_words.most_common(20)
```

Out[108…
```
[('film', 4287),
 ('movie', 3246),
 ('one', 2800),
 ('like', 1888),
 ('even', 1386),
 ('time', 1168),
 ('good', 1163),
 ('would', 1090),
 ('get', 1052),
 ('bad', 1034),
 ('much', 1011),
 ('character', 942),
 ('story', 923),
 ('plot', 917),
 ('two', 912),
 ('characters', 873),
 ('make', 851),
 ('first', 832),
 ('could', 791),
 ('see', 784)]
```

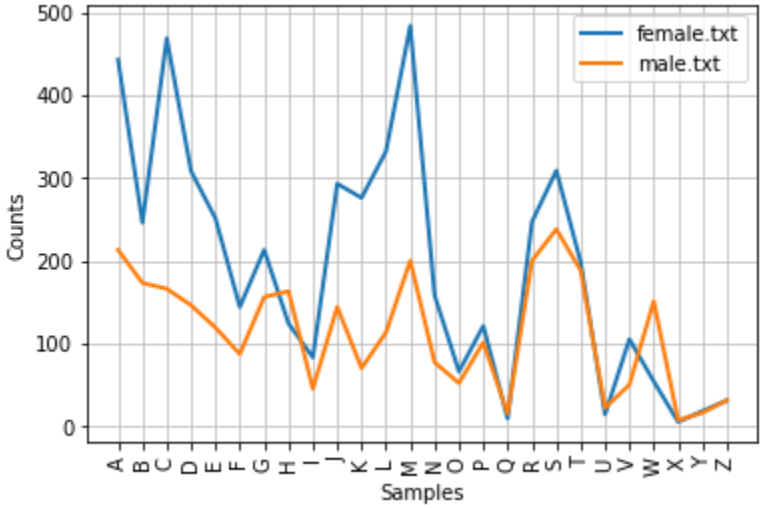
In [109…
```
pos_words = FreqDist([w for w in positive_reviews if (w.lower() not in stopwords and w.isalpha())])
pos_words.most_common(20)
```

Out[109…
```
[('film', 5230),
 ('one', 3052),
 ('movie', 2525),
 ('like', 1802),
 ('good', 1248),
 ('story', 1246),
 ('time', 1243),
 ('also', 1200),
 ('even', 1179),
 ('well', 1123),
```

```
('character', 1078),
('life', 1057),
('much', 1038),
('would', 1019),
('first', 1004),
('two', 999),
('characters', 986),
('see', 965),
('way', 929),
('get', 897)]
```

Question 2.4 Explore Names Corpus, which initial letters are more frequent for males vs. females? ((Hint: Plot of a Conditional Frequency Distribution will be useful for answering this question)

In [91]:
```
names = nltk.corpus.names
cfd = nltk.ConditionalFreqDist(
 (fileid, name[0])
 for fileid in names.fileids()
 for name in names.words(fileid))
cfd.plot()
```



Out[91]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>

The above plot shows which genders have the most common initial letters. Most female names start with C as well as the middle letters like J, K, L, and many for M. For males it is more spread it with many starting with W, S, and M as well. It looks like from the graph, M and S are popular across genders.

Question 2.5 Use the Gutenberg Corpus module to explore austen-persuasion.txt. Write a function to find out the 20 most frequent occurring words of this text file that are not stopwords.Please get rid of numbers and punctuations. Please make sure we are not double-counting words like "This" and "this", which differ only in capitalization.

In [92]:
```
def most_frequent(text):
    stopwords = nltk.corpus.stopwords.words('english')
    fdist = FreqDist([w for w in text if w.lower() not in stopwords and w.isalpha()])
    most_freq = fdist.most_common(20)
    return most_freq

most_frequent(nltk.corpus.gutenberg.words('austen-persuasion.txt'))
```

Out[92]:
```
[('Anne', 497),
 ('could', 444),
 ('would', 351),
 ('Captain', 297),
 ('Mrs', 291),
```

```
('Elliot', 288),
('Mr', 256),
('must', 228),
('one', 221),
('Wentworth', 218),
('much', 205),
('Lady', 191),
('good', 181),
('little', 175),
('said', 173),
('Charles', 166),
('might', 166),
('never', 153),
('time', 151),
('think', 149)]
```

Question 2.6 Use one of the path similarity measures to score the similarity of each of the following pairs of words. Rank the pairs in order of decreasing similarity: car-automobile, journey-voyage, boy-lad, coast-shore, midday-noon, furnace-stove, food-fruit, bird-cock, bird-crane, tool-implement, journey-car, cemetery-woodland, food-rooster, coast-hill, forest-graveyard, shore-woodland, coast-forest, lad-wizard, chord-smile, glass-magician, noon-string.

In [93]:
```
from nltk.corpus import wordnet as wn
pairs = [('car', 'automobile'),('journey', 'voyage'),
         ('boy', 'lad'),('coast', 'shore'),
         ('midday', 'noon'),('furnace', 'stove'),
         ('food', 'fruit'),('bird', 'cock'),
         ('bird', 'crane'),('tool', 'implement'),
         ('journey', 'car'),('cemetery', 'woodland'),
         ('food', 'rooster'),('coast', 'hill'),
         ('forest', 'graveyard'),('shore', 'woodland'),
         ('coast', 'forest'),('lad', 'wizard'),
         ('chord', 'smile'),('glass', 'magician'),
         ('noon', 'string')]

def similarity(pairs):
    word_dict = {}
    for word1, word2 in pairs:
        synset1 = wn.synset(word1 + '.n.01')
        synset2 = wn.synset(word2 + '.n.01')
        similarity = synset1.path_similarity(synset2)
        similarity_rounded = round(similarity,3)
        word_dict[word1 + '-' + word2] = similarity_rounded
        word_list = sorted(word_dict.items(), key=lambda item:item[1], reverse = True)
        sorted_word_dict = dict(word_list)
    for k in sorted_word_dict:
        print(k, sorted_word_dict[k])



similarity(pairs)
```

```
car-automobile 1.0
midday-noon 1.0
coast-shore 0.5
tool-implement 0.5
boy-lad 0.333
journey-voyage 0.25
coast-hill 0.2
shore-woodland 0.2
lad-wizard 0.2
bird-crane 0.111
cemetery-woodland 0.111
glass-magician 0.111
```

food-fruit 0.091
coast-forest 0.091
chord-smile 0.091
furnace-stove 0.077
forest-graveyard 0.071
bird-cock 0.062
food-rooster 0.062
noon-string 0.059
journey-car 0.05