

Chapter One Exercises

Set up the environment

In [6]:

```
import nltk
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
```

1. Try using the Python interpreter as a calculator, and typing expressions like:

$$12 / (4 + 1) .$$

In [1]:

$$12 \div (4 + 1)$$

Out[1]:

2.4

2. Given an alphabet of 26 letters, there are 26 to the power 10, or `26 ** 10`, 10-letter strings we can form. That works out to 14116709553376L (The L at the end just indicates that this is Python's lone-number format). How many hundred-letter strings are possible?

In [2]:

```
26 ** 100
```

Out[2]:

314293064158293883017435778850162642728266998876247525637417317539899590842010402346543259906970228
964075081611719197835869803511992549376

3. The Python multiplication operation can be applied to lists. What happens when you type `['Monty', 'Python'] * 20`, or `3 * sent1`?

In [4]:

```
['Monty', 'Python'] * 20
```

Out[4]:

```
[ 'Monty',
  'Python',
  'Monty',
  'Python',
  'Monty',
  'Python',
  'Monty',
```

[illegible]

In [7]:

```
3 * sent1
```

Out[7]:

```
['Call',
 'me',
 'Ishmael',
 '.',
 'Call',
 'me',
 'Ishmael',
 '.',
 'Call',
 'me',
 'Ishmael',
 '.']
```

4. Review Section 1.1 on computing with language. How many words are there in `text2`? How many distinct words are there?

In [12]:

```
len(text2) # Number of words (tokens)
```

Out[12]:

141576

In [14]:

```
len(set(text2)) # Number of word types
```

Out[14]:

6833

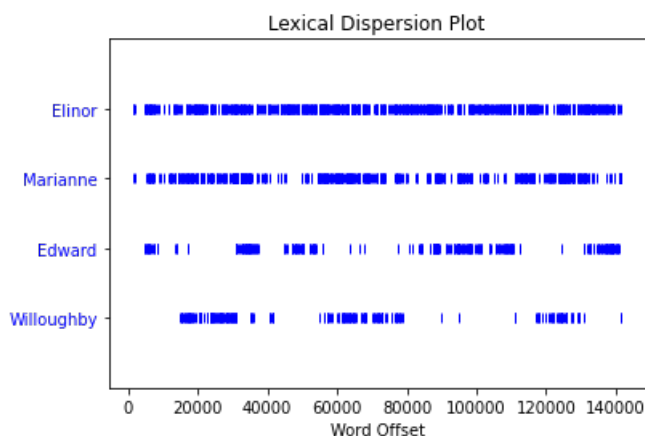
5. Compare the lexical diversity scores for humor and romance fiction in Table 1-1. Which genre is more lexically diverse?

Romance fiction is more lexically diverse than humor since it has a higher score.

6. Produce a dispersion plot of the four main protagonist in *Sense and Sensibility*: Elinor, Marianne, Edward, and Willoughby. What can you observe about the different roles played by the males and females in this novel? Can you identify the couples?

In [15]:

```
text2.dispersion_plot(['Elinor', 'Marianne', 'Edward', 'Willoughby'])
```



Clearly, the two main roles are those of Elinor and Marianne, with Elinor being the most prominent.

The word Edward appears often with both Elinor and Marianne, but on closer look the word Elinor appears more often with Edward than Marianne does. So, I could guess that Elinor and Edward are a couple. Similarly, Marianne and Willoughby appear together more often, so I could guess that Marianne and Willoughby are a couple.

7. Find the collocations in text5.

In [16]:

```
text5.collocations()
```

```
wanna chat; PART JOIN; MODE #14-19teens; JOIN PART; PART PART;
cute.-ass MP3; MP3 player; JOIN JOIN; times .. .; ACTION watches; guys
wanna; song lasts; last night; ACTION sits; -...)...- S.M.R.; Lime
Player; Player 12%; dont know; lez gurls; long time
```

8. Consider the following Python expression: `len(set(text4))`. State the purpose of this expression. Describe the two steps involved in performing this computation.

In [18]:

```
# The purpose of this expression is to count the number of tokens
# in text4, without counting repetitions.

# The first step, which is the set(text4) command, is to obtain a set of
# elements consisting of tokens from text4, but without any token appearing
# in the set more than once.
# The second step, len(), is to count the number of tokens in the set.

len(set(text4))
```

Out[18]:

9913

9. Review Section 1.2 on lists and strings.

a. Define a string and assign it to a variable, e.g. `my_string = 'My String'` (but put something more interesting in the string). Print the contents of this variable in two ways, first by simply typing the variable name and pressing **Enter**, then by using the `print` statement.

In [24]:

```
my_string = 'This is a string with something more interesting in it than what was originally suggested.'
```

In [25]:

```
my_string
```

Out[25]:

```
'This is a string with something more interesting in it than what was originally suggested.'
```

In [23]:

```
print(my_string)
```

```
This is a string with something more interesting in it than what was originally suggested.
```

b. Try adding the string to itself using `my_string + my_string`, or by multiplying it by a number, e.g., `my_string * 3`. Notice that the strings are joined together without any spaces. How could you fix this?

In [26]:

```
my_string + my_string          # Adding string to itself
```

Out[26]:

```
'This is a string with something more interesting in it than what was originally suggested.This is a string with something more interesting in it than what was originally suggested.'
```

In [27]:

```
my_string * 3                  # Multiplying string by 3
```

Out[27]:

```
'This is a string with something more interesting in it than what was originally suggested.This is a string with something more interesting in it than what was originally suggested.This is a string with something more interesting in it than what was originally suggested.'
```

In [28]:

```
# One solution could be to add a "buffer", like a space character ' '
# in between the two string, like so:
my_string + ' ' + my_string
```

Out[28]:

```
'This is a string with something more interesting in it than what was originally suggested. This is a string with something more interesting in it than what was originally suggested.'
```

In [32]:

```
# Same solution would work with multiplication, with the caveat that
# the space character ' ' would need to be added to the string first.
(my_string + ' ') * 3
```

Out[32]:

```
'This is a string with something more interesting in it than what was originally suggested. This is a string with something more interesting in it than what was originally suggested. This is a string with something more interesting in it than what was originally suggested.'
```

ing with something more interesting in it than what was originally suggested. '

10. Define a variable `my_sent` to be a list of words, using the syntax `my_sent = ["My", "sent"]` (but with your own words, or a favorite saying).

In [48]:

```
my_sent = ["The", "limits", "of", "my", "language", "mean", "the", "limits", "of", "my", "world"]
```

a. Use `' '.join(my_sent)` (note the space between the single quotes) to convert this into a string.

In [49]:

```
my_sent = ' '.join(my_sent)
my_sent
```

Out[49]:

```
'The limits of my language mean the limits of my world'
```

b. Use `split()` to split the string back into the list form you had to start with.

In [50]:

```
my_sent = my_sent.split()
my_sent
```

Out[50]:

```
['The',
 'limits',
 'of',
 'my',
 'language',
 'mean',
 'the',
 'limits',
 'of',
 'my',
 'world']
```