1. How many classes are in the `nltk.text` module? How many are there in the `nltk.probability` module?

   **Answer:** Five and twenty respectively.

2. How many arguments does the `nltk.text.Text.collocations()` method have? How many are arguments are obligatory? Why does the call `text6.collocations()` work even though no arguments are specified?

   **Answer:** According to the documentation it has three arguments, but one of them (`self`) is a “fake” argument that every Python method has. (Technically, a method’s first argument is the object that the method is called from.) As a result, it has only two “real” arguments, and none are obligatory. That’s also the reason that the call `text6.collocations()` works.

3. Choose a plain-text corpus of your own, load it into Python with `PlaintextCorpusReader`, and construct and plot a (non-cumulative) word frequency distribution for the corpus. Compare this word frequency distribution with those from `text1` through `text8`. Which words appear as the most frequent words in all the corpora? Which words appear only in some corpora but not in others? Why?

   The corpus you choose can be anything—a class paper or papers that you wrote, a Wikipedia or newspaper article(s), a set of emails—you name it. In your answer to this problem, provide a brief description of the corpus that you used, including a URL or URLs if it can be obtained from the Web.

4. Find the twenty most frequent words in each of `text2` and `text4`. How many words appear in both top-twenty lists?

   Now find the twenty most frequent bigrams in each of `text2` and `text4`. How many bigrams appear in both top-twenty lists?
How many of the bigrams in either list contain a word that is contained in either of
the single-word top-twenty lists? Why do you think this is the case? What do your
results tell you in general about the nature of frequent bigrams?

**Answer:** Code for obtaining the unigrams occurring in both the top-20 most frequent
lists for `text2` and `text4` (note that [...] means that I'm eliding output that isn't
pertinent to the solution):

```python
>>> from nltk.book import *
[...]  
>>> fd2 = FreqDist(text2)
>>> top20_text2 = fd2.keys()[0:20]
>>> fd4 = FreqDist(text4)
>>> top20_text4 = fd4.keys()[0:20]
>>> [w for w in top20_text2 if w in top20_text4]
['.', 'to', '.', 'the', 'of', 'and', 'a', 'in', 'it', 'be', 'that', 'for', 'not', 'as']

14 out of 20 overlap between the two.

Now code for obtaining the overlap in top-20 bigram lists:

```python
>>> top20_bigrams_text4 = fd4_bigrams.keys()[0:20]
>>> fd2_bigrams = FreqDist(bigrams(text2))
>>> top20_bigrams_text2 = fd2_bigrams.keys()[0:20]
>>> fd4_bigrams = FreqDist(bigrams(text4))
>>> [b for b in top20_bigrams_text2 if b in top20_bigrams_text4]
[('.', 'and'), ('of', 'the'), ('to', 'be'), ('in', 'the'), ('.', 'I'), ('to', 'the')]
```

Only 6 out of 20 overlap between the two.

Code for determining how many bigrams occurring in either list contain a word con-
tained in either of the single-word top-20 lists:

```python
>>> total = 0
>>> top20_unigrams_merged = top20_text2 + top20_text4
>>> for bigram in (top20_bigrams_text2 + top20_bigrams_text4):
...     if (bigram[0] in top20_unigrams_merged) or (bigram[1] in top20_unigrams_merged):
...         total = total + 1
...     
>>> print str(total)
39
```
All but one of the bigrams contains a word contained in one of the single-word top-20 lists. If you take a look at the most frequent bigrams, you’ll see that something even stronger holds: almost all the words in the top 20 bigram lists are top 20 words as well.

Regarding interpretation of these results: almost all the most frequent bigrams are combinations of closed-class function words that are themselves highly frequent. There’s less overlap in the top-20 lists simply because there are so many more possible bigrams than there are unigrams.

5. Consider the following word pairs:

<table>
<thead>
<tr>
<th>every</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>it</td>
</tr>
<tr>
<td>one</td>
<td>hundred</td>
</tr>
<tr>
<td>king</td>
<td>people</td>
</tr>
<tr>
<td>thought</td>
<td>made</td>
</tr>
<tr>
<td>already</td>
<td>quickly</td>
</tr>
<tr>
<td>in</td>
<td>near</td>
</tr>
<tr>
<td>brave</td>
<td>good</td>
</tr>
</tbody>
</table>

Write a function to compute and print out the frequencies of each of the words in each of the nine texts, as well as in the corpus that you chose yourself. What generalizations do you see regarding the variability in word frequencies for each of the words in each of the pairs? Pay particular attention to the ratio of word frequencies across texts. For example, both every and a appear more in text 4 than in text 5:

<table>
<thead>
<tr>
<th>Text 4</th>
<th>Text 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>0.187%</td>
</tr>
<tr>
<td>a</td>
<td>1.464%</td>
</tr>
</tbody>
</table>

The absolute difference in word frequency between the two texts is bigger for a than for every, but the relative difference is much greater for every, which appears 17 times as often in Text 4 than in Text 5, than for a, which appears only 1.16 times as often in Text 4 than in Text 5

What does the two words in each pair have in common with one another? What patterns do you see in terms of the properties (grammatical and/or meaning) that each word has and how variable its frequency is across texts? Are there any texts that are particular outliers for some (or for many) words? Why? Please include the Python code you write, as well as the observations you make together with any explanation for them you can find, in your answer to this problem.

Answer: If you execute this Python code:

```python
from nltk.book import *
```
def percentage(count, total):
    return 100 * float(count) / float(total)

## this is a little function that just makes printing prettier -- you
## don't need to understand it to get the solution to this problem
def pad_text(x):
    result = x
    while len(result) < 10:
        result = result + " 
    return(result)

print " ", " 
      .join(["text" + str(i) for i in range(1,10)])
for (a,b) in [("every","a"),("I","it"),("one","hundred"),("people","king"),"made","thought"),("already","quickly"),("in","near"),("good","brave")]:
    pa = []
    pb = []
    for text in [text1,text2,text3,text4,text5,text6,text7,text8,text9]:
        n = len(text)
        pa = pa + [percentage(text.count(a),n)]
        pb = pb + [percentage(text.count(b),n)]
    ## "%.3f" % x is Python syntax for rendering x to three decimal
    ## places. You don't need this to give a good answer, but it's
    ## worth learning to do this type of formatting
    print pad_text(a), " 
      .join(["%.3f" % x for x in pa])
print pad_text(b), " 
      .join(["%.3f" % x for x in pb])

You will get this result:

    text1 text2 text3 text4 text5 text6 text7 text8 text9
    every  0.085  0.255  0.197  0.187  0.011  0.018  0.019  0.000  0.029
    a        1.752  1.443  0.764  1.464  1.262  1.108  1.865  1.068  2.475
    I        0.814  1.415  1.081  0.570  1.280  1.503  0.114  0.452  1.279
    it       0.847  1.108  0.648  0.694  0.738  0.631  0.473  0.000  0.816
    one      0.341  0.215  0.156  0.155  0.193  0.189  0.131  0.021  0.270
    hundred  0.019  0.015  0.136  0.007  0.000  0.006  0.003  0.000  0.022
    people   0.016  0.038  0.078  0.384  0.091  0.035  0.056  0.000  0.058
    king     0.010  0.000  0.071  0.001  0.002  0.059  0.000  0.000  0.007
made 0.068 0.097 0.161 0.104 0.018 0.024 0.056 0.000 0.088
thought 0.057 0.082 0.009 0.027 0.053 0.024 0.008 0.000 0.075
already 0.011 0.035 0.000 0.022 0.036 0.024 0.035 0.000 0.029
quickly 0.008 0.002 0.004 0.002 0.000 0.000 0.013 0.000 0.013
in 1.501 1.345 1.314 1.734 0.793 0.507 1.561 0.555 1.283
near 0.026 0.023 0.054 0.006 0.029 0.006 0.009 0.021 0.009
good 0.074 0.117 0.098 0.095 0.289 0.065 0.042 0.431 0.088
brave 0.007 0.000 0.000 0.005 0.000 0.100 0.000 0.000 0.006

The ratios in word frequencies are much more variable for some words (e.g., every, it, hundred, brave) than for others (e.g., a, made, in). There is one complete outlier text: text8, which doesn’t even have some of the most common words in the other texts (notably it). When you check the identity of this text, you realize why: it’s personals ads, which have totally different language use than normal English (e.g., they often omit function words altogether, and good is particularly common). In general, the less frequent a word the more its frequency varies across texts. Also, words with more “specific” and “concrete” semantic content (e.g., king, brave) seem to have more variability in their frequency than more semantically “bleached” words (e.g., people, good).

The words are paired by like grammatical category: the first pair is determiners, the second pronouns, the third numbers, the fourth common nouns, the fifth verbs, the sixth adverbs, the seventh prepositions, the eighth adjectives.

One specific word/text outlier pair of note is hundred’s high frequency in text3: this is Biblical language and presumably the number 100’s symbolic significance plays a role in its high frequency in this text.