1homework.py

First Homework : NLP with Python

Due 22 FEB 2012

Chapter1

Exercises 4, 19, 20, 22, 24, 26

Your turn p6 p8 p24

```
from __future__ import division
from nltk.book import *
```

```
def write_doc(fn):

    def wrapper():
        print "%s\n" % (fn.__doc__)
        fn()
        print "\n"*2
    return wrapper
```

ex4

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

```
@write_doc
def ex4():
```

number of words in text2

```
    print "number of words in text 2 : %i" % len(text2)
```

number of distinct words

```
    print "number of distinct words in text2 : %i" % len(set(text2))
```

ex19

What is the difference between the following two lines? Which one will give a larger value? Will this be the case for other texts?

```
sorted(set([w.lower() for w in text1]))
sorted([w.lower() for w in set(text1)])
```

In the first line we first transform all the words to lowercase then remove the duplicates.

In the second line we remove the duplicates and then transform to lowercase the first line will give less results

examples :

```
input = ["a","A","b","C","a","b","B","A"]
> sorted(set([w.lower() for w in a]))
["a","b"]
> sorted([w.lower() for w in set(a)])
["a", "b", "A", "B"]
```

```
@write_doc
def ex19():
```

```
    pass
```

ex20

What is the difference between the following two tests: w.isupper() and not w.islower()?

- `isupper` will check if the word is uppercase it will return true if there are letters in the string and they are all uppercase

  ```
  > "2".isupper()
  False
  ```

- `not islower` will return True even if the string doesn't contain letters

  ```
  > not "2".islower()
  True
  ```

```
@write_doc
def ex20():
```

```
    pass
```

ex22

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

```
@write_doc
def ex22():
```

```
    four_letter_words = [w for w in text5 if len(w) == 4]
    freq_dist = FreqDist(four_letter_words)
```

we don't need to sort because the FreqDist() already is sorted by frequency

```
    print "frequency of four letter words in text5"
    print freq_dist.keys()[0:50]
```

ex24

Write expressions for finding all words in text6 that meet the following conditions The result should be in the form of a list of words: ['word1', 'word2', ...].

a. Ending in ize

b. Containing the letter z

c. Containing the sequence of letters pt

d. All lowercase letters except for an initial capital (i.e., titlecase)

```
@write_doc
def ex24():




    print "words finishing in ''ize''"
    print [w for w in text5 if w.endswith("ize")]
    print

    print "words with a 'z'"
    print [w for w in text5 if "z" in w]
    print

    print "words with 'pt'"
    print [w for w in text5 if "pt" in w]
    print

    print "words that look like titles"
    print [w for w in text5 if w.istitle()][0:10]
    print
```

ex26

What does the following Python code do? sum([len(w) for w in text1]) Can you use it to work out the average word length of a text ?

This code computes the sum of the length of all the words in text1

```
@write_doc
def ex26():




    def avg_word_length(text):
        return sum([len(w) for w in text])/len(text)

    print "average word length of text1 : %i\n" % avg_word_length(text1)
```

Your turn p6

text1: Moby Dick by Herman Melville 1851

text2: Sense and Sensibility by Jane Austen 1811

text3: The Book of Genesis

text4: Inaugural Address Corpus

```
@write_doc
def your_turn_p6():


    texts = (text3, text1)
    second_text = text1
    word = "god"
    print "words that are similar to %s in %s" % (word, texts[0])
    texts[0].similar(word)
    print "words that are similar to %s in %s" % (word, texts[1])
    texts[1].similar(word)

    context = ["great", "god"]
    print "common context of %s in %s" % (context, texts[0])
    texts[0].common_contexts(context)
    print "common context of %s in %s" % (context, texts[1])
    texts[1].common_contexts(context)
```

Your turn p8

```
@write_doc
def your_turn_p8():


    word = "lol"
    text = text5
    freq_dist = FreqDist(text)
    len_text = len(text)
    lexical_diversity = freq_dist[word]/len_text*100

    print "%s appears %i times in %s" % (word, freq_dist[word], text)
    print "it represents %.2f percent of the total number of words" % lexical_diversi
```

Your turn p24

`sorted([w for w in set(text7) if '-' in w and 'index' in w])` returns the alphabetically sorted-without duplicates list of words in text7 which got a dash in them and index

`sorted([wd for wd in set(text3) if wd.istitle() and len(wd) > 10])` returns the alphabetically sorted-duplicates free list of words in text7 that look like titles and whose length is superior than 10

`sorted([w for w in set(sent7) if not w.islower()])` return the alphabetically sorted-duplicate free list of words in sentence #7 who are not lowercase

`sorted([t for t in set(text2) if 'cie' in t or 'cei' in t])` returns the alphabetically sorted-duplicate free list of words in text2 which contains "cie" or "cei"

```
@write_doc
def your_turn_p24():
```

```
ex4()
ex19()
ex20()
ex22()
ex24()
```

```
    ex26()
    your_turn_p6()
    your_turn_p8()
    your_turn_p24()
```