

```
import nltk
```

```
nltk.download('wordnet')
```

```
nltk.download('punkt')
```

```
text = "This is Andrew's text, isn't it?"
```

1) How many tokens:-

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
```

```
tokens = tokenizer.tokenize(text)
```

```
print(len(tokens))
```

```
print(tokens)
```

2) How many tokens: treebankword tokenizer?

```
tokenizer = nltk.tokenize.treebankwordWhitespaceTokenizer()
```

```
tokens = tokenizer.tokenize(text)
```

```
print(len(tokens))
```

```
print(tokens)
```

3) How many tokens there are if you use WordPunct.
Tokenizer?

```
tokenizer = nltk.tokenize.WordPunctTokenizer()
```

```
tokens = tokenizer.tokenize(text)
```

```
print(len(tokens))
```

```
print(tokens)
```

Natural Language Processing Lab

Lab1. Understanding Large Text Files

EXERCISE-1

Consider the following text.

```
import nltk
nltk.download('wordnet')
text = "This is Andrew's text, isn't it?"
```

1. How many tokens are there if you use WhitespaceTokenizer?. Print tokens.

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
tokens = tokenizer.tokenize(text)
print(len(tokens))
print(tokens)
```

2. How many tokens are there if you use TreebankWordTokenizer?. Print tokens.

```
tokenizer = nltk.tokenize.TreebankWordTokenizer()
```

3. How many tokens there are if you use WordPunctTokenizer?. Print tokens.

```
tokenizer = nltk.tokenize.WordPunctTokenizer()
```

EXERCISE-2

1. Open the file: O. Henry's The Gift of the Magi (**gift-of-magi.txt**).

2. Write a Python script to print out the following:

1. How many word tokens there are
2. How many word types there are, (word types are a unique set of words)
3. Top 20 most frequent words and their counts
4. Words that are at least 10 characters long and their counts
5. 10+ characters-long words that occur at least twice, sorted from most frequent to least

1. Open the file:

```
import re
```

```
f = open("gift-of-magi.txt", encoding="utf-8")
```

```
con = f.read()
```

```
print(len(con))
```

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
```

```
tokens = tokenizer.tokenize(con)
```

```
print(len(tokens))
```

```
from nltk import  
data = FreqDist(tokens)  
data.
```

```
3)  
data.most_common(20)
```

```
A) from nltk import.
```

```
text = [w for w in tokens if len(w) > 10]
```

```
print(text)
```

```
freq = FreqDist(text)
```

```
freq
```

```
text = [w for w in tokens if len(w) > 10]
```

```
s = FreqDist(text)
```

```
s
```

```
for i, j in freq.items():
```

```
    if len(i) > 10 and j > 2:
```

```
        print(i, j)
```

Exercise - 3

List Comprehension

Step-1

```
fname = ". /data/ austen-emma.txt"
```

```
d = open("austen-emma.txt", encoding="utf-8")
```

```
etxt = f.read()
```

```
f.close()
```

```
etxt[-200:]
```

EXERCISE -3: List Comprehension

STEP-1

Download the document Austen's *Emma* ("**austen-emma.txt**"). Read it in and apply the usual text processing steps, building three objects: *etoks* (a list of word tokens, all in lowercase), *etypes* (an alphabetically sorted word type list), and *efreq* (word frequency distribution).

```
>>> fname = "./data/austen-emma.txt"
>>> f = open(fname, 'r')
>>> etxt = f.read()
>>> f.close()
>>> etxt[-200:]
'e deficiencies, the wishes,\nthe hopes, the confidence, the predictions of the
small band\nof true friends who witnessed the ceremony, were fully answered\nin
the perfect happiness of the union.\n\nFINIS\n'
>>> etoks = nltk.word_tokenize(etxt.lower())
>>> etoks[-20:]
['of', 'true', 'friends', 'who', 'witnessed', 'the', 'ceremony', ',', 'were',
'fully', 'answered', 'in', 'the', 'perfect', 'happiness', 'of', 'the', 'union',
',', 'finis']
>>> len(etoks)
191781
>>> etypes = sorted(set(etoks))
>>> etypes[-10:]
['younger', 'youngest', 'your', 'yours', 'yourself', 'yourself.', 'youth', 'youthful',
'zeal', 'zigzags']
>>> len(etypes)
7944
>>> efreq = nltk.FreqDist(etoks)
>>> efreq['beautiful']
24
```

STEP 2: list-comprehend *Emma*

Now, explore the three objects *wlist*, *efreq*, and *etypes* to answer the following questions. Do NOT use the for loop! Every solution must involve use of LIST COMPREHENSION.

Question 1: Words with prefix and suffix

What are the words that start with 'un' and end in 'able'?

Question 2: Length

How many Emma word types are 15 characters or longer? Exclude hyphenated words.

tokenizer = nltk.tokenize.WhitespaceTokenizer()

tokens = tokenizer.tokenize(text)

tokens[-20:]

etoks = nltk.WordTokenizer(text.lower())

etoks[-20:]

len(etoks)

etypes = sorted(set(etoks))

etypes[-10:]

len(etypes)

efreq = nltk.FreqDist(etoks)

efreq['beautiful']

Step 2:
etoks

Question 1: Words with prefix and suffix:

[word for word in tokens if word.startswith("un")
or word.endswith("able")]

Question 2: length

How many Emma word types are 15 characters or longer?

tokenizer = nltk.tokenize.WordPunctTokenizer()

toks = tokenizer.tokenize(text)

[word for word in toks if len(word) > 15]

Question 3: Average word length

What's the average length of all Emma word types?

Question 4: Word frequency

How many Emma word types have a frequency count of 200 or more? How many word types appear only once?

Question 5: Emma words not in wlist

Of the Emma word types, how many of them are not found in our list of ENABLE English words, i.e., wlist?

STEP 3: bigrams in Emma

Let's now try out bigrams. Build two objects: e2grams (a list of word bigrams; make sure to cast it as a list) and e2gramfd (a frequency distribution of bigrams) as shown below, and then answer the following questions.

```
>>> e2grams = list(nltk.bigrams(etoks))
>>> e2gramfd = nltk.FreqDist(e2grams)
>>>
```

Question 6: Bigrams

What are the last 10 bigrams?

Question 7: Bigram top frequency

What are the top 20 most frequent bigrams?

Question 8: Bigram frequency count

How many times does the bigram 'so happy' appear?

Question 9: Word following 'so'

What are the words that follow 'so'? What are their frequency counts? (For loop will be easier; see if you can utilize list comprehension for this.)

Question 10: Trigrams

What are the last 10 trigrams? (You can use `nltk.util.ngrams()` method)

Question 11: Trigram top frequency

What are the top 10 most frequent trigrams?

Question 12: Trigram frequency count

How many times does the trigram 'so happy to' appear?

Average word length

What's the average length of all Emma word types?

average = $\frac{\sum (\text{len}(\text{word}) \text{ for word in tokens})}{\text{len}(\text{tokens})}$
average.

```
lg = []
```

```
for i in take:
```

```
    if len(i) > 15:
```

```
        lg.append(i)
```

```
print(lg)
```

Question 4:- Word frequency

How many .emma. Word types have a frequency count.

```
from nltk import
```

```
fdiemm = FreqDist(take)
```

```
for i, j in fdiemm.items():
```

```
    if j > 200:
```

```
        print(i, j)
```

How many word types appears only once?

```
for i, j in fdiemm.items():
```

```
    if j == 1:
```

```
        print(i, j).
```

What are the last 10 bigrams,

```
e2grams = list(nltk.bigrams(take))
```

```
e2gramsfreq = nltk.FreqDist(e2grams)
```

```
e2gramsfreq
```

```
last_ten = FreqDist(dict(e2gramsfreq.most_common(10)))
```

```
last_ten
```

NOTES

Question 7: Bigram for frequency.

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
```

```
tokens = tokenizer.tokenize(text)
```

```
ezgrams = list(nltk.bigrams(tokens))
```

```
ezgramfd = nltk.FreqDist(ezgrams)
```

```
ezgramfd.most_common(20)
```

Question 8: Bigram frequency count.

How many times does the bigram 'so happy' appear?

```
for i, j in ezgramfd.items():
```

```
    for i == ('so', 'happy'):
```

```
        print(i, j)
```

Question 9: Word following 'so'

```
import re
```

```
from collections import Counter
```

```
words = re.findall(r'so (\w+)', open('austen-emma.txt').read())
```

```
ab = Counter(zip(words))
```

```
print(ab)
```

Question 10: ezgrams = list(nltk.bigrams(tokens))

```
ezgramfd = nltk.FreqDist(ezgrams)
```