NLTK : Natural Language Toolkit

Prepared by Prof. Youn-Sik Hong

본 강의 자료는 http://www.nltk.org/book/ 내용을 참고하여 작성하였음

목차

nltk 소개 및 참고 사이트

NLTK 란?

- 자연어 처리를 위한 Python 프로그램 개발을 위한 플랫폼
  - **A wonderful tool for teaching, and working in, computational linguistics *using Python***
  - **An *amazing library* to play with natural language**
- A free, open source, community-driven project
- Windows, Mac OS X 및 Linux에서 사용 가능
- 테스트 용으로 활용 가능한 text 리소스(= *text corpus*(단수), *text corpora*(복수))
  - http://www.nltk.org/nltk_data/
- 참고 사이트 : http://www.nltk.org/book/

Natural Language Processing with Python

– Analyzing Text with the Natural Language Toolkit

Steven Bird, Ewan Klein, and Edward Loper

Free ebooks - Project Gutenberg

Book search · Book categories · Browse catalog · Mobile site · Report errors · Terms of use

Some of the Latest Books

http://www.gutenberg.org

nltk 설치

- **커맨드 창에서 아래 명령어 실행**

  pip install nltk

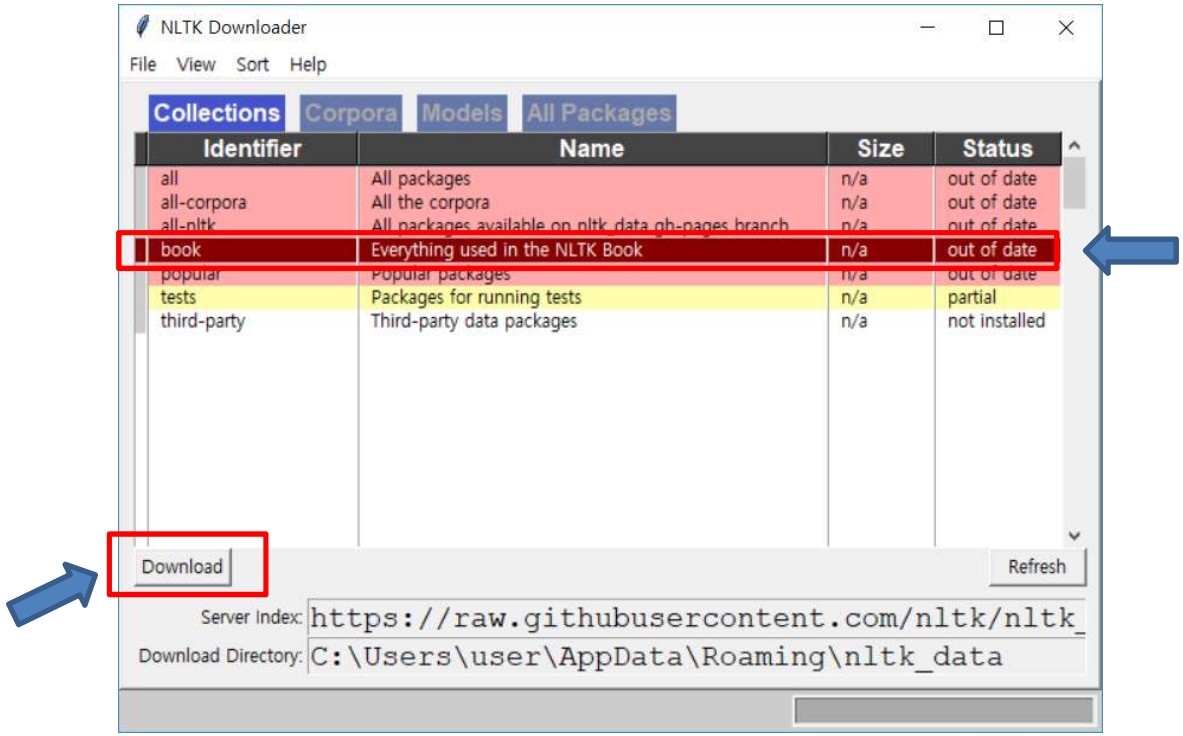
- **잠깐 : pip가 뭔가요?**
  - Python에서 package(라이브러리)를 관리하는 SW
    - 3.4(또는 2.7.9) 이상 버전부터 python 설치할 때 함께 설치됨.
  - pip 설치 여부 및 버전 확인 : **pip show pip**

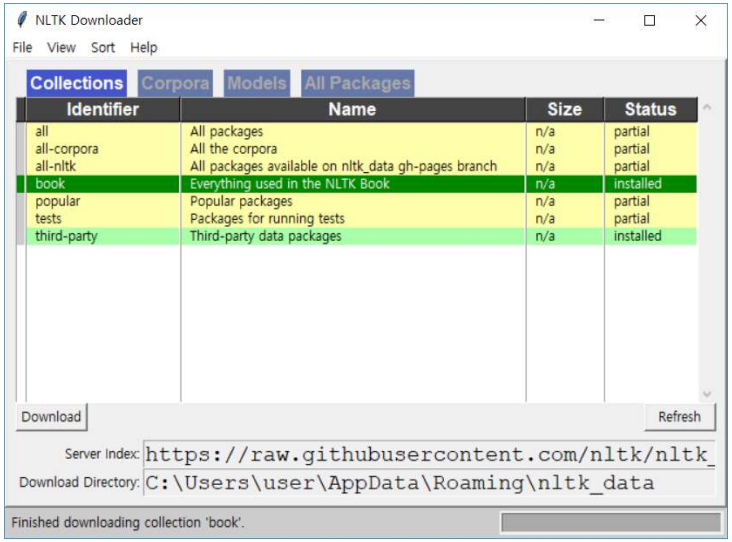
```
C:\Users\user>pip show pip
Name: pip
Version: 19.1.1
Summary: The PyPA recommended tool for installing Python packages.
Home-page: https://pip.pypa.io/
Author: The pip developers
Author-email: pypa-dev@groups.google.com
License: MIT
Location: c:\users\user\appdata\local\programs\python\python37\lib\site-packages
Requires:
Required-by:
```

샘플 book 다운로드 (1/2)

- **Python shell에서**

```
>>> import nltk
>>> nltk.download()
```



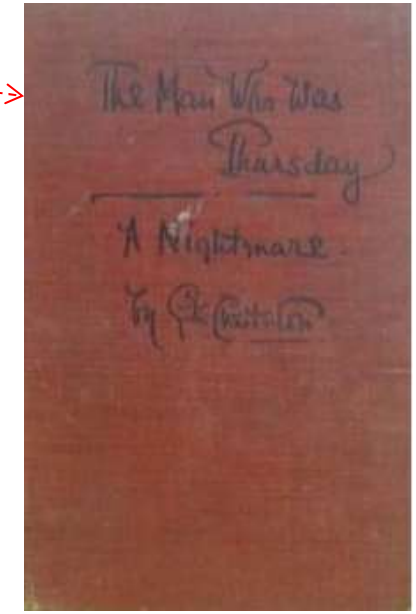


샘플 book 다운로드 (2/2)

- Python shell에서

```
>>> 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
```

창세기
취임 연설문
채팅



참고 : https://www.nltk.book/ch01.html

Practice #1 – nltk(intro). ipynb

- 아래 명령을 실행시키고, 결과를 확인 하시오.

```
text1.concordance("monstrous")
text2.concordance("affection")
text3.concordance("lived")
text4.concordance("nation") # terror, god
text5.concordance("lol") # im, ur
```

Melville의 Moby Dick

Austen의 Sense and Sensibility

성경: 창세기

1789년 대통령 연설문

채팅 대화 내용

concordance(일치)
monstrous(거대한, 괴이한)

similar 함수 – 유사한 형태로 문장에 등장하는 단어 찾기

```
text1.concordance("monstrous")
```

```
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

```
text1.similar("monstrous")
```

```
true contemptible christian abundant few part mean careful puzzled
mystifying passing curious loving wise doleful gamesome singular
delightfully perilous fearless
```

```
text1.concordance("abundant")
```

```
Displaying 3 of 3 matches:
ars , must have possessed the most abundant and the most convenient opportunit
w alarmed . I had allowed him such abundant time ; I thought he might have had
ngs ; though the affair still left abundant room for all manner of wild conjec
```

Practice #2 - nltk(intro). ipynb

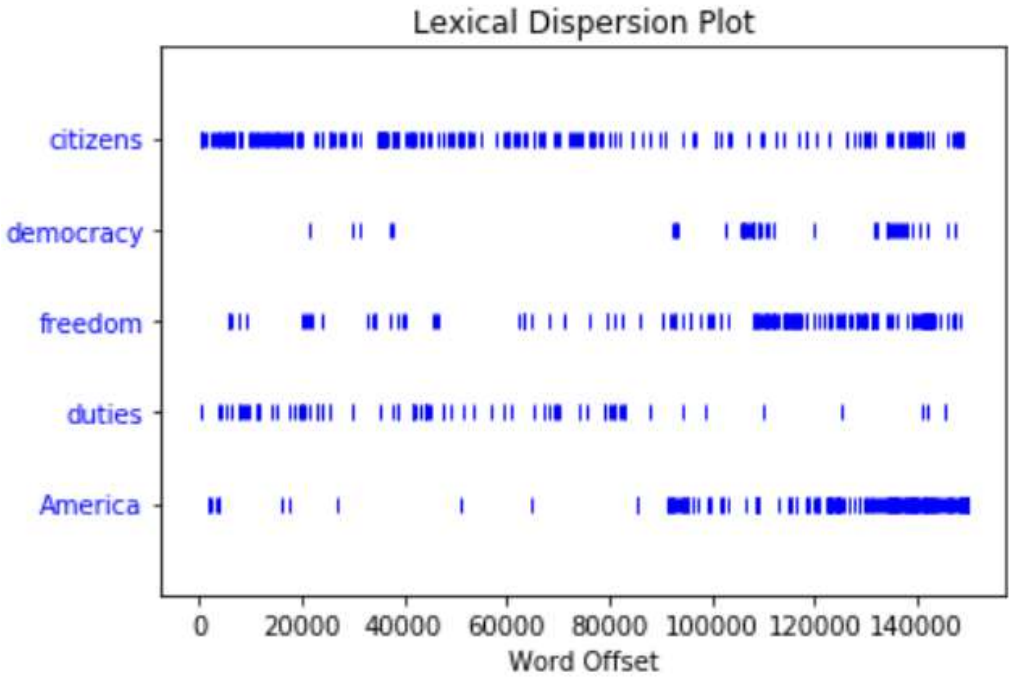
- 아래 명령을 실행시키고, 결과를 확인 하시오.

```
text1.concordance('monstrous')
text1.similar('monstrous')
text2.similar('monstrous')
text2.common_contexts(['monstrous', 'very'])
```

dispersion_plot 함수 – 단어 발생 분포

- **말뭉치에서 단어가 어느 위치에 나타났는지 표시**
  - '**citizen**'은 처음부터 끝까지 일정하게 나타나지만,
  - '**America**'는 연설 끝부분에 집중적으로 등장

```
text4.dispersion_plot(['citizens', 'democracy', 'freedom', 'duties', 'America'])
```



dispersion(분산)

Practice #3 – 어휘 다양성(lexical richness)

■ 아래 명령을 실행시키고, 결과를 확인 하시오.

```
len(text3)

tmp3 = sorted(set(text3))
print('구둣점 기호 제거 전:', len(text3), len(tmp3))

tmp3[:13] #13개 문자는 구둣점 기호(punctuation symbols)

punct_list = ["!", "'", "(", ")", ",", ",)", ".", ".)", ":", ";", ";)", "?", "?)"]
tmp4 = [x for x in tmp3 if x not in punct_list]
tmp42 = [x for x in text3 if x not in punct_list]

print('구둣점 기호 제거 후:', len(tmp42), len(tmp4))
```

```
import numpy as np

print('어휘 다양성: 구두점 기호 제거 전')
np.round(len(tmp3)/len(text3), 5)

print('어휘 다양성: 구두점 기호 제거 후')
np.round(len(tmp4)/len(tmp42), 5)
```

Practice #4 – 단어 발생 빈도와 어휘 다양성

■ 아래 명령을 실행시키고, 결과를 확인 하시오.

```
text3.count('Abel')
```

```
def lexical_diversity(txt):
    print(txt, np.round(len(set(txt))/len(txt), 5))

def percentage(w, txt):
    count = txt.count(w)
    total = len(txt)
    print('문자=', w, '빈도=', np.round(100*count/total, 5))
```

```
lexical_diversity(text1)
lexical_diversity(text2)
lexical_diversity(text3)
lexical_diversity(text4)
lexical_diversity(text6)
lexical_diversity(text9)
```

```
percentage('a', text1)
percentage('A', text1)
percentage('.', text1)
percentage(',', text1)
```

목차

sample text 다운로드

■ 샘플 텍스트 다운로드

```
>>> 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
```

■ sents(), sent1, sent2, ..., sent9 – 테스트용 문장

```
>>> sents()
sent1: Call me Ishmael .
sent2: The family of Dashwood had long been settled in Sussex .
sent3: In the beginning God created the heaven and the earth .
sent4: Fellow - Citizens of the Senate and of the House of Representatives :
sent5: I have a problem with people PMing me to lol JOIN
sent6: SCENE 1 : [ wind ] [ clop clop clop ] KING ARTHUR : Whoa there !
sent7: Pierre Vinken , 61 years old , will join the board as a nonexecutive dire
ctor Nov. 29 .
sent8: 25 SEXY MALE , seeks attrac older single lady , for discreet encounters .
sent9: THE suburb of Saffron Park lay on the sunset side of London , as red and
ragged as a cloud of sunset .
```

문자열 연산 : list, join, split – nltk(string-1). ipynb

list와 문자열

```
>>> ex1 = ['A','closer','look','at','Python']
>>> ex1
['A', 'closer', 'look', 'at', 'Python']
>>> sentence = 'Texts as List of Words'
>>> sentence
'Texts as List of Words'
```

join

```
>>> ex1join = ';'.join(ex1)
>>> print(type(ex1join), ':', ex1join)
<class 'str'> : A;closer;look;at;Python
>>> ex1join
'A;closer;look;at;Python'
```

split

```
>>> wordlist = sentence.split(' ')
>>> print(type(wordlist), ':', wordlist)
<class 'list'> : ['Texts', 'as', 'List', 'of', 'Words']
```

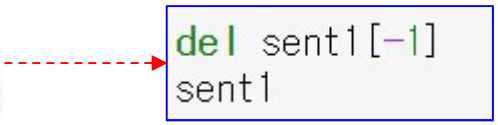
문자열 연산 : concatenation, append, multiplication

■ concatenation (+)

```
>>> sent1
['Call', 'me', 'Ishmael', '.']
>>> sent2
['The', 'family', 'of', 'Dashwood', 'had', 'long', 'been', 'settled', 'in', 'Sus
sex', '.']
>>> sent2+sent1
['The', 'family', 'of', 'Dashwood', 'had', 'long', 'been', 'settled', 'in', 'Sus
sex', '.', 'Call', 'me', 'Ishmael', '.']
```

■ append

```
>>> sent1.append("Some")
>>> sent1
['Call', 'me', 'Ishmael', '.', 'Some']
```



■ multiplication

```
>>> ex2 = 'python' + 'python' + 'python'
>>> ex2
'pythonpythonpython'
>>> ex3 = 'python' * 2
>>> ex3
'pythonpython'
```

문자열 연산 : index와 slicing(부분 문자열 추출)

■ 문자열 index

```
>>> str = 'Python NLTK'
>>> str[1]
'y'
>>> str[-3]
'L'
```

■ 부분 문자열

```
>>> str2 = 'NLTK Dolly Python'
>>> str2[:4]
'NLTK'
>>> str2[11:]
'Python'
>>> str2[5:10]
'Dolly'
>>> str2[-12:-7]
'Dolly'
```

```
ref = len('NLTK')+1
offset = ref+len('Python')-1
str2[ref:offset]
```

```
last = len(str2)
ref = (len('Python')-1)-last
offset = ref+len('Dolly')
str2[ref:offset]
```

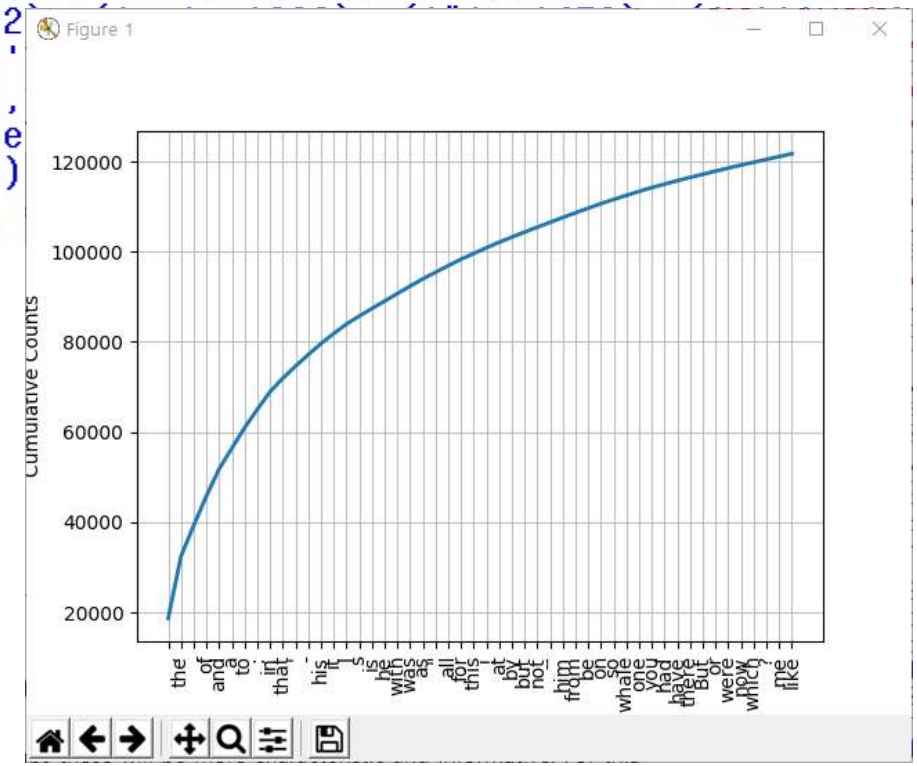
빈도 분포(frequency distribution) – nltk(string-2). ipynb

■ FreqDist, most_common

```
>>> fdist1 = FreqDist(text1)
>>> print(fdist1)
<FreqDist with 19317 samples and 260819 outcomes>
>>> fdist1.most_common(50)
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024), ('a', 4
569), ('to', 4542), (';', 4072), ('in', 3916), ('that', 2982), ("'", 2684), ('-'
, 2552), ('his', 2459), ('it', 2209), ('I', 2124), ('s', 1739), ('is', 1695), ('
he', 1661), ('with', 1659), ('was', 1632
62), ('for', 1414), ('this', 1280), ('!'
ut', 1113), ('not', 1103), ('--', 1070),
030), ('on', 1005), ('so', 918), ('whale
ad', 767), ('have', 760), ('there', 715)
0), ('now', 646), ('which', 640), ('?',
>>> fdist1['whale']
906
```

■ 누적 빈도 분포 출력(50개 단어)

```
>>> fdist1.plot(50, cumulative=True)
```



조건을 만족하는 단어 찾기

- **소설 Moby Dick(=text1)에서 길이가 15이상인 단어를 찾아보자.**
  - Python 코딩 스타일

$\{w \mid w \in V \& P(w)\}$ ⇒ `[w for w in V if p(w)]`

```
>>> V = set(text1)
>>> long_words = [w for w in V if len(w)>15]
>>> sorted(long_words)
['CIRCUMNAVIGATION', 'Physiognomically', 'apprehensiveness', 'cannibalistically'
, 'characteristically', 'circumnavigating', 'circumnavigation', 'circumnavigatio
ns', 'comprehensiveness', 'hermaphroditical', 'indiscriminately', 'indispensable
ness', 'irresistibleness', 'physiognomically', 'preternaturalness', 'responsibil
ities', 'simultaneousness', 'subterraneousness', 'supernaturalness', 'superstiti
ousness', 'uncomfortableness', 'uncompromisedness', 'undiscriminating', 'uninter
penetratingly']
```

단어 길이에 따른 빈도 분포

■ 단어 길이를 기준으로 빈도 분포를 조사해 보자!

```
>>> fdist = FreqDist(len(w) for w in text1)
>>> fdist
FreqDist({3: 50223, 1: 47933, 4: 42345, 2: 38513, 5: 26597, 6: 17111, 7: 14399,
8: 9966, 9: 6428, 10: 3528, ...})
>>> print(fdist)
<FreqDist with 19 samples and 260819 outcomes>
>>> fdist.most_common()
[(3, 50223), (1, 47933), (4, 42345), (2, 38513), (5, 26597), (6, 17111), (7, 143
99), (8, 9966), (9, 6428), (10, 3528), (11, 1873), (12, 1053), (13, 567), (14, 1
77), (15, 70), (16, 22), (17, 12), (18, 1), (20, 1)]
>>> fdist.max()
3
>>> fdist[3]
50223
>>> fdist.freq(3)
0.19255882431878046
```

문자열 비교 연산

Function	Meaning
s.startswith(t)	test if s starts with t
s.endswith(t)	test if s ends with t
t in s	test if t is a substring of s
s.islower()	test if s contains cased characters and all are lowercase
s.isupper()	test if s contains cased characters and all are uppercase
s.isalpha()	test if s is non-empty and all characters in s are alphabetic
s.isalnum()	test if s is non-empty and all characters in s are alphanumeric
s.isdigit()	test if s is non-empty and all characters in s are digits
s.istitle()	test if s contains cased characters and is titlecased (i.e. all words in s have initial capitals)

단어 비교 함수

endswith, istitle, isdigit, w in list

```
>>> sorted(w for w in set(text1) if w.endswith('ableness'))
['comfortableness', 'honourableness', 'immutableness', 'indispensableness', 'ind
omitableness', 'intolerableness', 'palpableness', 'reasonableness', 'uncomfortab
leness']
>>> sorted (term for term in set(text4) if 'gnt' in term)
['Sovereignty', 'sovereignties', 'sovereignty']
>>> sorted(item for item in set(text6) if item.istitle())
```

Squeezed text (53 lines).

- istitle 함수는 첫 글자가 대문자인 경우를 찾음.

```
>>> sorted(item for item in set(sent7) if item.isdigit())
['29', '61']
```

출력

■ 조건에 따른 출력

```
>>> for token in sent1:
        if token.islower():
                print(token, 'is a lowercase word')
        elif token.istitle():
                print(token, 'is a titlecase word')
        else:
                print(token, 'is punctuation')
```

```
Call is a titlecase word
me is a lowercase word
Ishmael is a titlecase word
. is punctuation
```

■ 출력 제어: 단어 뒤에 빈칸(공백)을 덧붙임

```
>>> for word in tricky:
        print(word, end=' ')
```

```
ancient ceiling conceit conceited conceive conscience conscientious conscientiou
sly deceitful deceive deceived deceiving deficiencies deficiency deficient delic
acies excellencies fancied insufficiency insufficient legacies perceive perceive
d perceiving prescience prophecies receipt receive received receiving society sp
ecies sufficient sufficiently undeceive undeceiving
```