BUSINESS REPORT

NTLK

統計110葉兆穎
統計110盧映竹

NLTK

01

什麼是NLTK？

什麼是NLTK？

NLTK：Natural Language Toolkit
NLTK是一個用於自然語言處理(NLP)的Python庫

那NLP到底是什麼？簡單的說，NLP其實就是開發能夠理解人類語言的應用程式和服務。語音辨識、語音翻譯、語法辨識等等都是NLP的應用。
又例如；Facebook 的資訊流。新聞饋送演算法通過自然語言處理了解到你的興趣，並向你展示相關的廣告以及訊息，而不是一些無關的資訊。

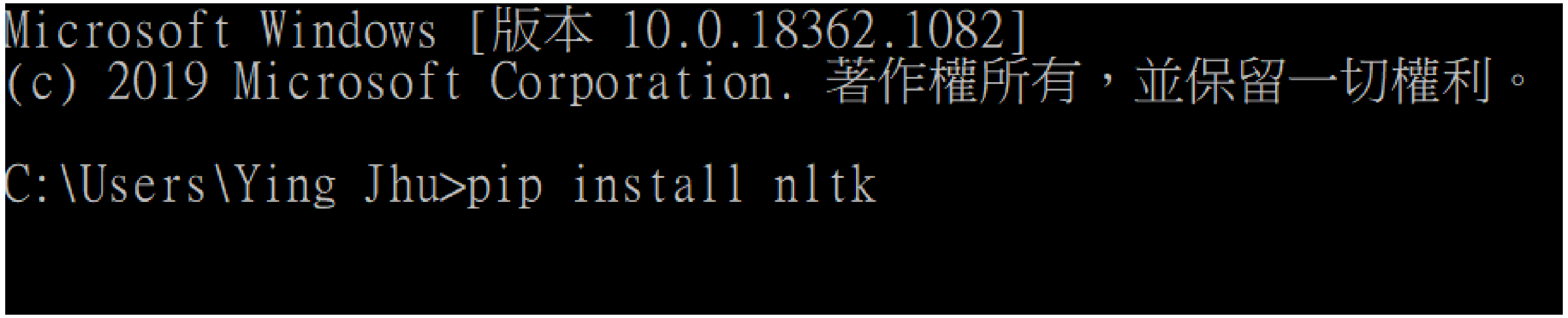
NLTK是最受歡迎的自然語言處理（NLP）庫。它是用 Python 語言編寫的，背後有強大的社群支援。

02

安裝及基本功能介紹

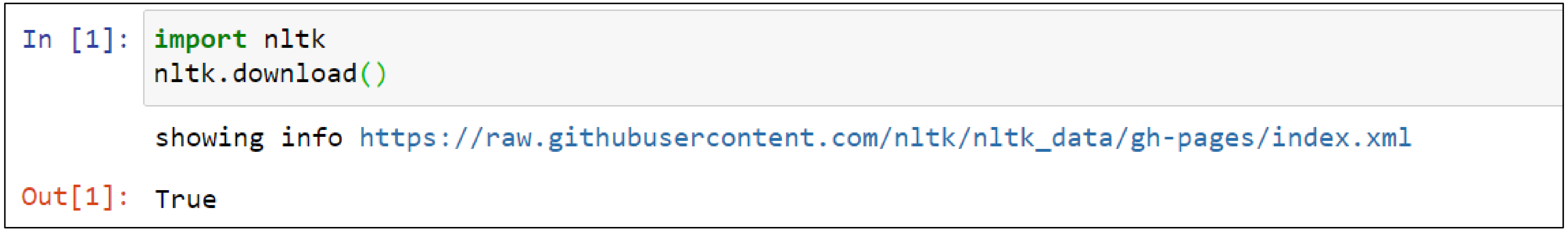
安裝NLTK

在終端機輸入pip install nltk，並在程式裡輸入import nltk即可使用



```
Microsoft Windows [版本 10.0.18362.1082]
(c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\Ying Jhu>pip install nltk
```

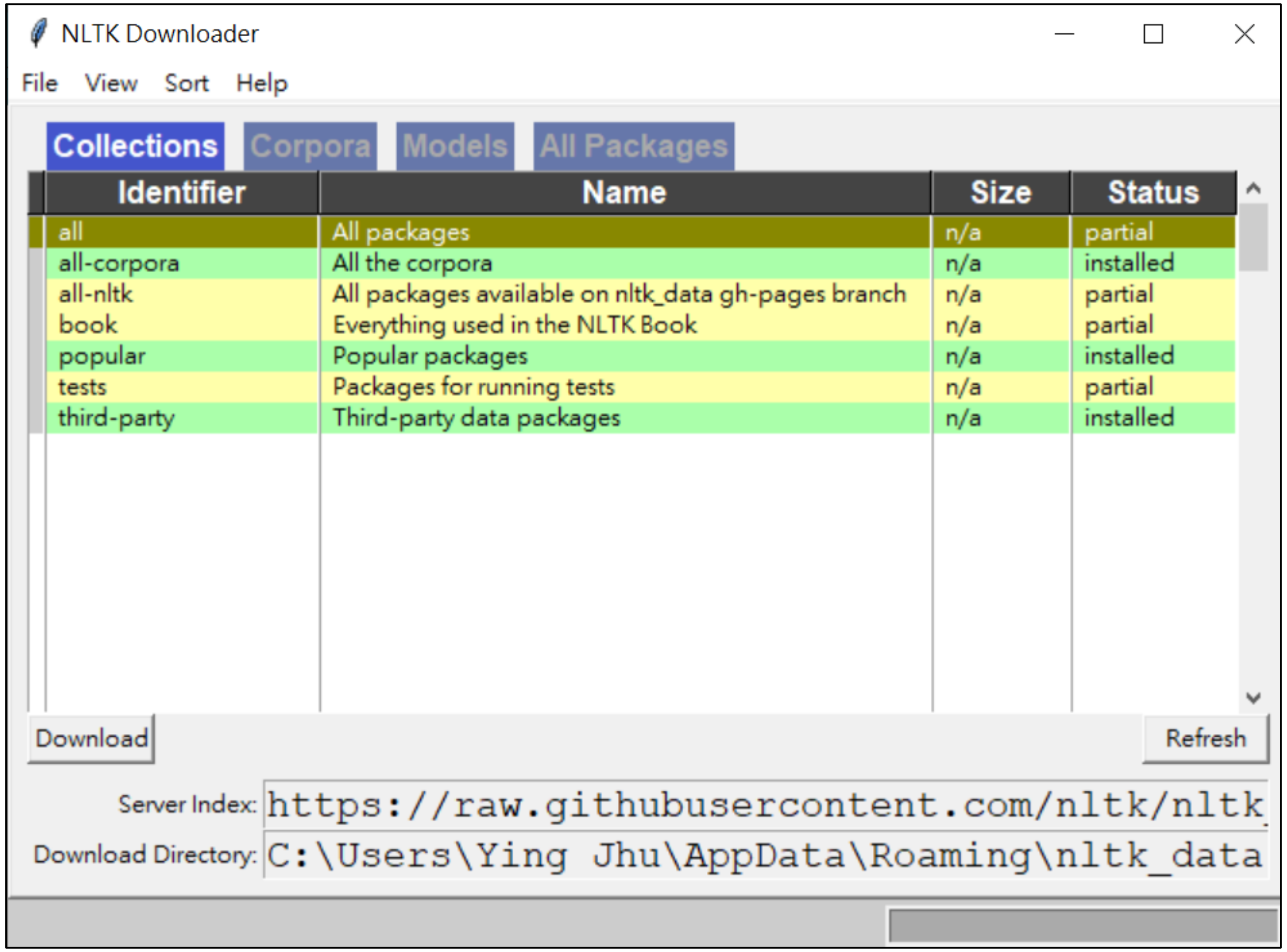
在程式裡輸入nltk.download()即可下載NLTK相關套件



```
In [1]: import nltk
        nltk.download()

        showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml

Out[1]: True
```

安裝NLTK

NLTK Downloader



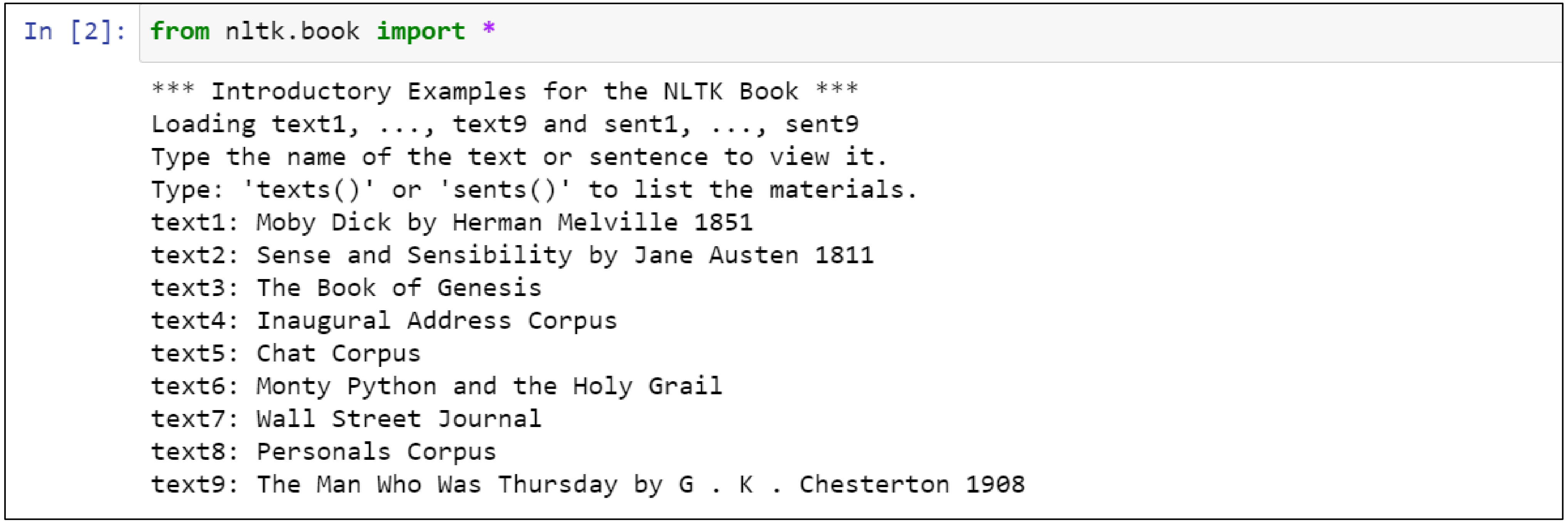
Identifier	Name	Size	Status
all	All packages	n/a	partial
all-corpora	All the corpora	n/a	installed
all-nltk	All packages available on nltk_data gh-pages branch	n/a	partial
book	Everything used in the NLTK Book	n/a	partial
popular	Popular packages	n/a	installed
tests	Packages for running tests	n/a	partial
third-party	Third-party data packages	n/a	installed

Server Index: https://raw.githubusercontent.com/nltk/nltk

Download Directory: C:\Users\Ying Jhu\AppData\Roaming\nltk_data

基本功能介紹

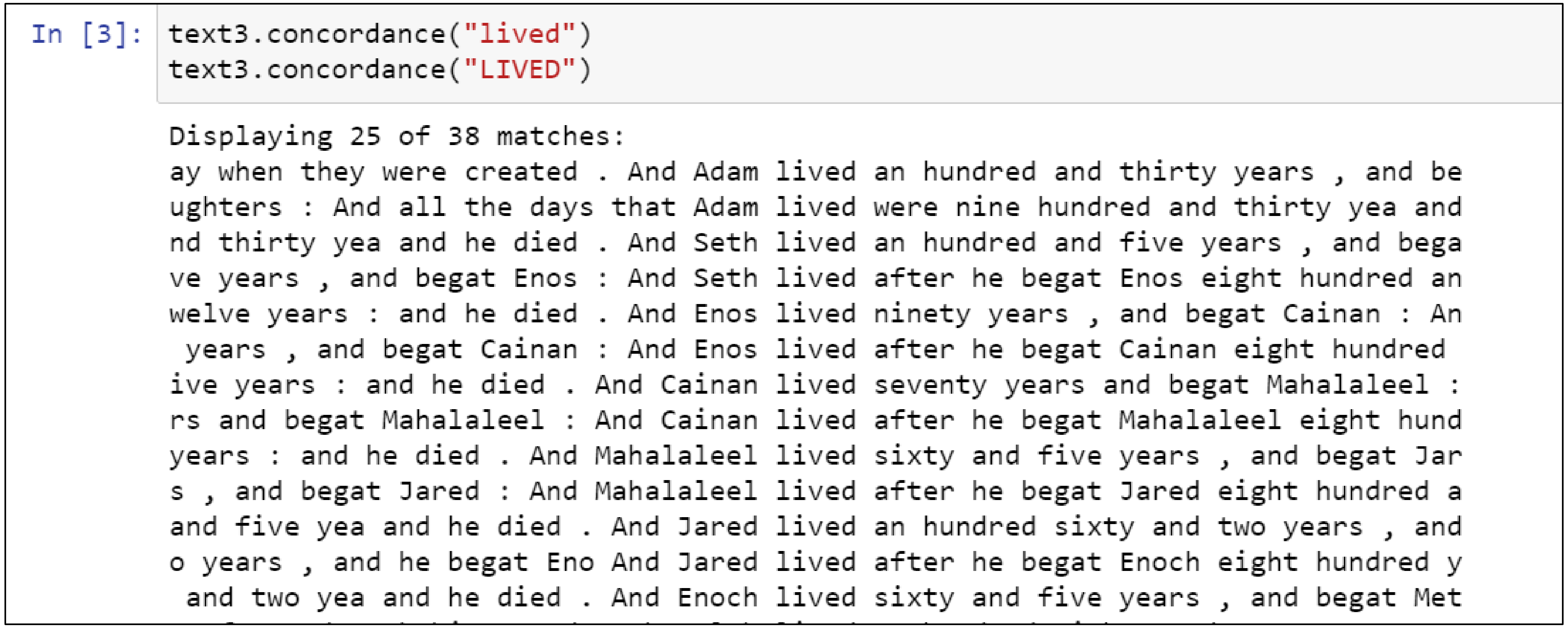
由於 NLTK 本身就是一個以自然語言處理為名的工具箱，因此可以很方便地透過工具箱去使用前面預先下載好的文本，讓我們做接下來的練習。



```
In [2]: 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
```

基本功能介紹 concordance

當我們想知道某個單字的上下文時，可以這樣：



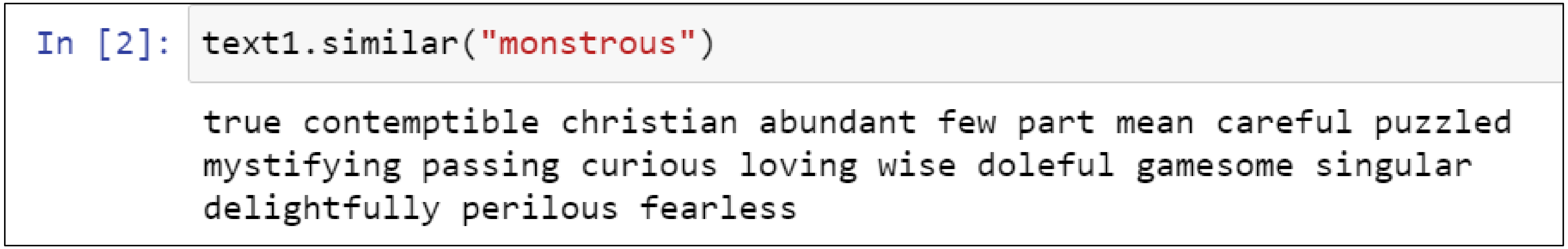
```
In [3]: text3.concordance("lived")
        text3.concordance("LIVED")
```

```
Displaying 25 of 38 matches:
ay when they were created . And Adam lived an hundred and thirty years , and be
ughters : And all the days that Adam lived were nine hundred and thirty yea and
nd thirty yea and he died . And Seth lived an hundred and five years , and bega
ve years , and begat Enos : And Seth lived after he begat Enos eight hundred an
welve years : and he died . And Enos lived ninety years , and begat Cainan : An
 years , and begat Cainan : And Enos lived after he begat Cainan eight hundred
ive years : and he died . And Cainan lived seventy years and begat Mahalaleel :
rs and begat Mahalaleel : And Cainan lived after he begat Mahalaleel eight hund
years : and he died . And Mahalaleel lived sixty and five years , and begat Jar
s , and begat Jared : And Mahalaleel lived after he begat Jared eight hundred a
and five yea and he died . And Jared lived an hundred sixty and two years , and
o years , and he begat Eno And Jared lived after he begat Enoch eight hundred y
 and two yea and he died . And Enoch lived sixty and five years , and begat Met
```

此外，輸入的關鍵字詞與搜尋結果都有「不分英文大小寫」的特性。

基本功能介紹 similar

當我們想知道某個單字的近似字，可以這樣：



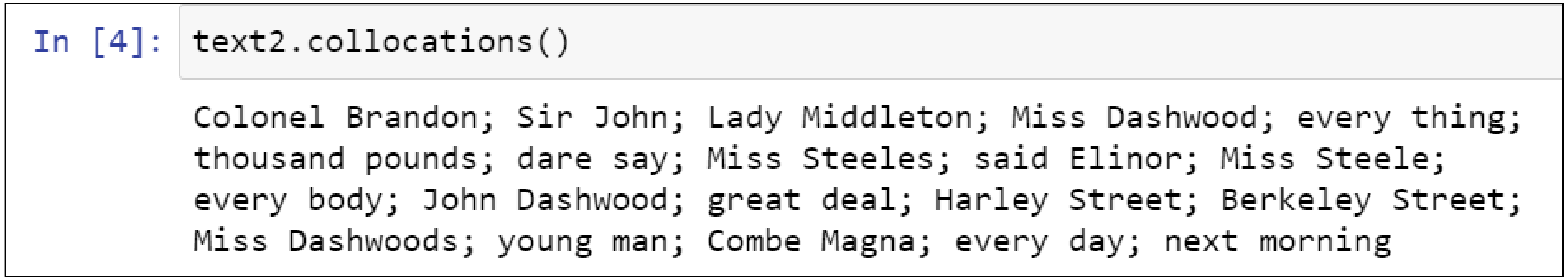
```
In [2]: text1.similar("monstrous")

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

根據該詞的上下文，找到類似結構，就認定他們為近似字。

假設我們在text1裡找 monstrous，而 monstrous 會出現在 the ___ pictures 以及 a ___ size 這樣的結構當中，透過這個方法去比對。

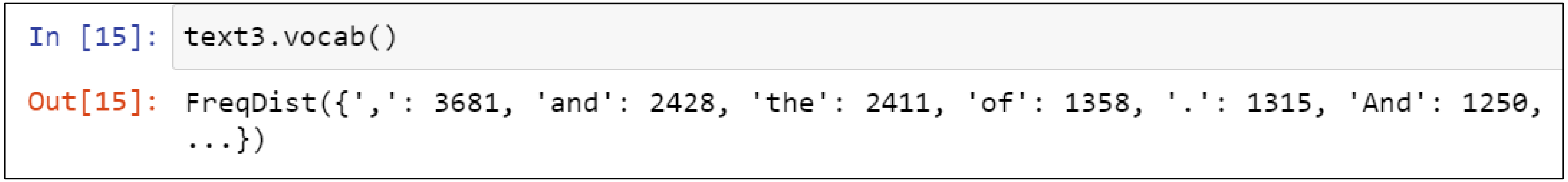
基本功能介紹 collocations、vocab

當我們想知道最常見的兩字搭配，可以這樣：



```
In [4]: text2.collocations()

        Colonel Brandon; Sir John; Lady Middleton; Miss Dashwood; every thing;
        thousand pounds; dare say; Miss Steeles; said Elinor; Miss Steele;
        every body; John Dashwood; great deal; Harley Street; Berkeley Street;
        Miss Dashwoods; young man; Combe Magna; every day; next morning
```

當我們想知道每一個單字出現在文章裡的次數，可以這樣：

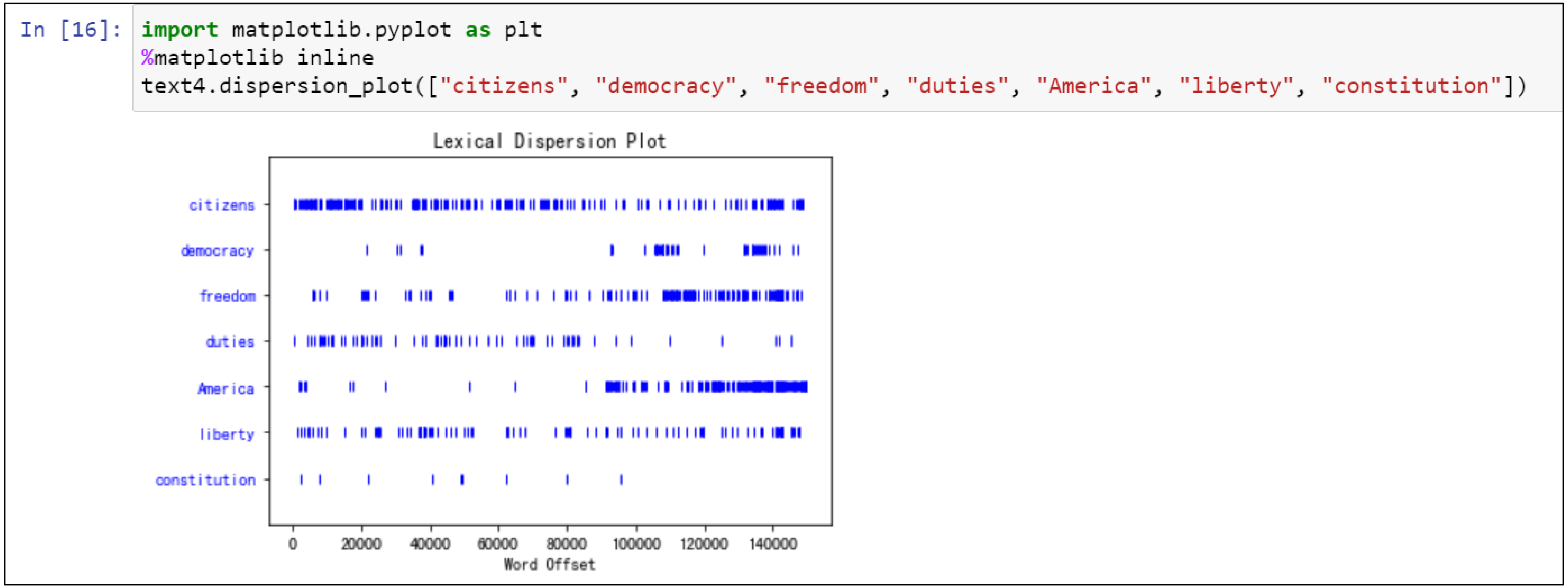


```
In [15]: text3.vocab()

Out[15]: FreqDist({',': 3681, 'and': 2428, 'the': 2411, 'of': 1358, '.': 1315, 'And': 1250,
         ...})
```

會以字典的形式返回。

基本功能介紹 dispersion_plot

當我們想看一些單字出現在文章裡的頻率圖，可以這樣：

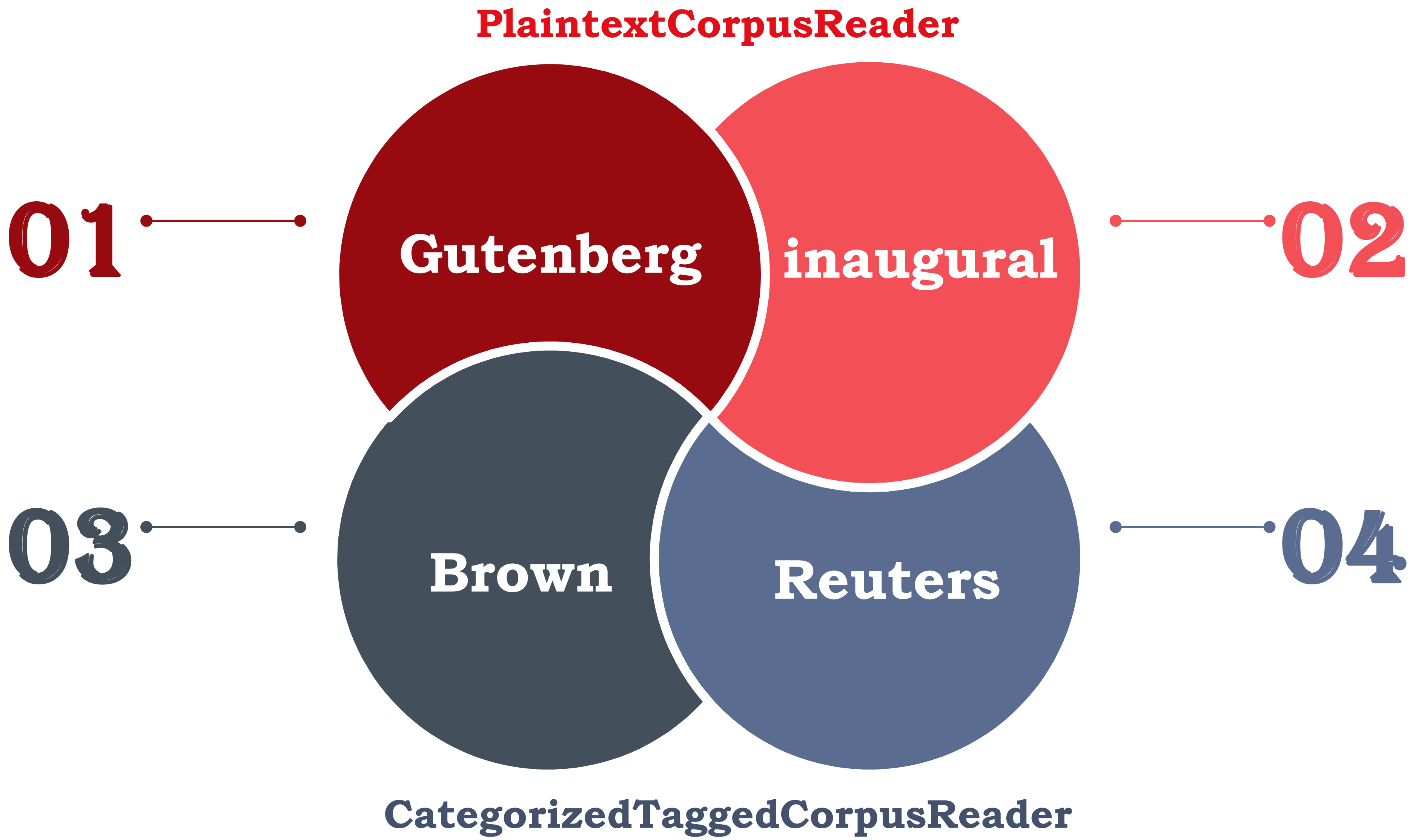


```
In [16]: import matplotlib.pyplot as plt
         %matplotlib inline
         text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America", "liberty", "constitution"])
```

03

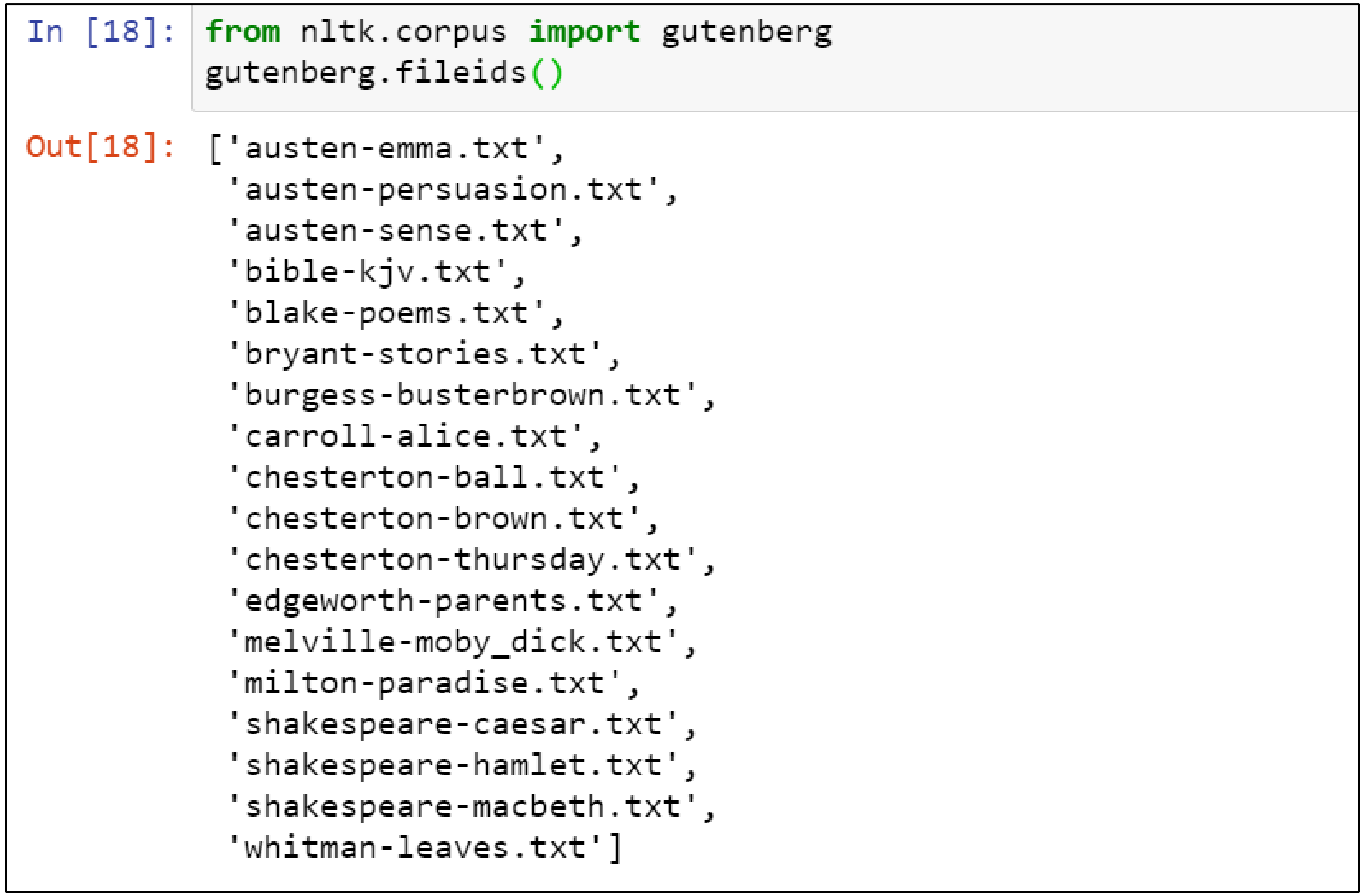
語料庫及字詞處理

Corpus語料庫


PlaintextCorpusReader
01
Gutenberg
inaugural
02
03
Brown
Reuters
04
CategorizedTaggedCorpusReader

Corpus語料庫 PlaintextCorpusReader

我們先用gutenberg.fileids()查有多少文本：



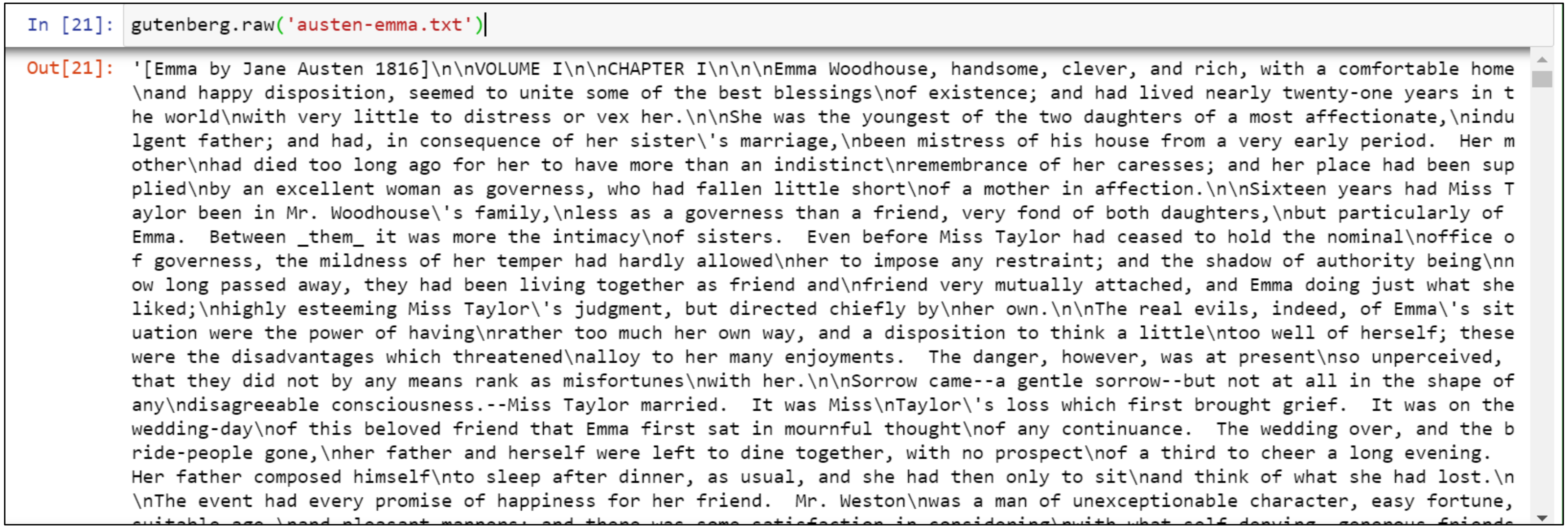
```
In [18]: from nltk.corpus import gutenberg
         gutenberg.fileids()

Out[18]: ['austen-emma.txt',
          'austen-persuasion.txt',
          'austen-sense.txt',
          'bible-kjv.txt',
          'blake-poems.txt',
          'bryant-stories.txt',
          'burgess-busterbrown.txt',
          'carroll-alice.txt',
          'chesterton-ball.txt',
          'chesterton-brown.txt',
          'chesterton-thursday.txt',
          'edgeworth-parents.txt',
          'melville-moby_dick.txt',
          'milton-paradise.txt',
          'shakespeare-caesar.txt',
          'shakespeare-hamlet.txt',
          'shakespeare-macbeth.txt',
          'whitman-leaves.txt']
```

gutenberg是第一個提供免費的網路電子書平台。

根據官方網站，project gutenberg 已經有超過57000本免費的電子書，NLTK的package僅納入部分語料。

Corpus語料庫 corpus.raw(fileids)

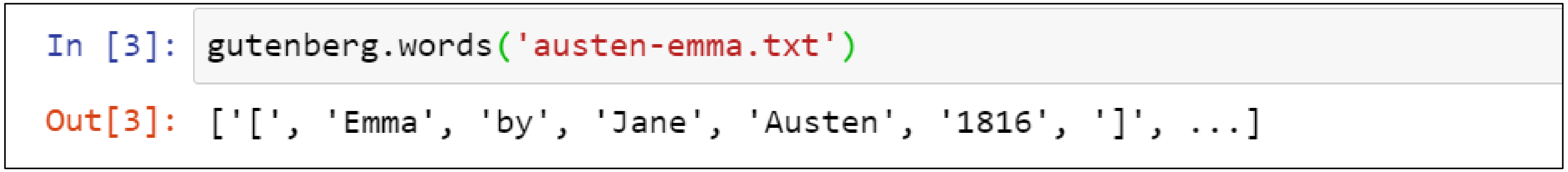
當我們想查看第一篇語料的原始內容，可以這樣：



```
In [21]:  gutenberg.raw('austen-emma.txt')

Out[21]:  '[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, and rich, with a comfortable home
          \nand happy disposition, seemed to unite some of the best blessings\nof existence; and had lived nearly twenty-one years in t
          he world\nwith very little to distress or vex her.\n\nShe was the youngest of the two daughters of a most affectionate,\nindu
          lgent father; and had, in consequence of her sister\'s marriage,\nbeen mistress of his house from a very early period.  Her m
          other\nhad died too long ago for her to have more than an indistinct\nremembrance of her caresses; and her place had been sup
          plied\nby an excellent woman as governess, who had fallen little short\nof a mother in affection.\n\nSixteen years had Miss T
          aylor been in Mr. Woodhouse\'s family,\nless as a governess than a friend, very fond of both daughters,\nbut particularly of
          Emma.  Between _them_ it was more the intimacy\nof sisters.  Even before Miss Taylor had ceased to hold the nominal\noffice o
          f governess, the mildness of her temper had hardly allowed\nher to impose any restraint; and the shadow of authority being\nn
          ow long passed away, they had been living together as friend and\nfriend very mutually attached, and Emma doing just what she
          liked;\nhighly esteeming Miss Taylor\'s judgment, but directed chiefly by\nher own.\n\nThe real evils, indeed, of Emma\'s sit
          uation were the power of having\nrather too much her own way, and a disposition to think a little\ntoo well of herself; these
          were the disadvantages which threatened\nalloy to her many enjoyments.  The danger, however, was at present\nso unperceived,
          that they did not by any means rank as misfortunes\nwith her.\n\nSorrow came--a gentle sorrow--but not at all in the shape of
          any\ndisagreeable consciousness.--Miss Taylor married.  It was Miss\nTaylor\'s loss which first brought grief.  It was on the
          wedding-day\nof this beloved friend that Emma first sat in mournful thought\nof any continuance.  The wedding over, and the b
          ride-people gone,\nher father and herself were left to dine together, with no prospect\nof a third to cheer a long evening.
          Her father composed himself\nto sleep after dinner, as usual, and she had then only to sit\nand think of what she had lost.\n
          \nThe event had every promise of happiness for her friend.  Mr. Weston\nwas a man of unexceptionable character, easy fortune,
```

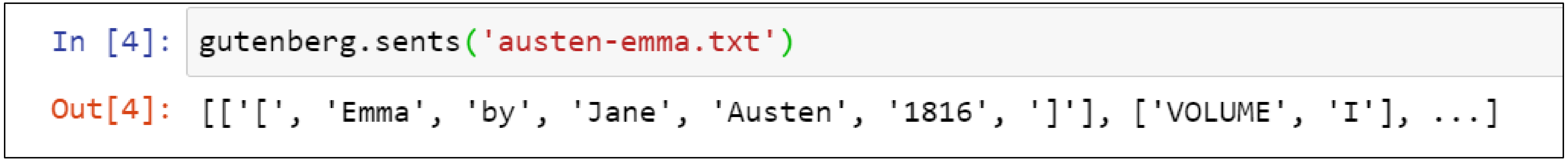
Corpus語料庫 corpus.words(fileids)、corpus.sents(fileids)

當我們想查看第一篇語料的單詞列表，可以這樣：



```
In [3]: gutenberg.words('austen-emma.txt')

Out[3]: ['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', ...]
```

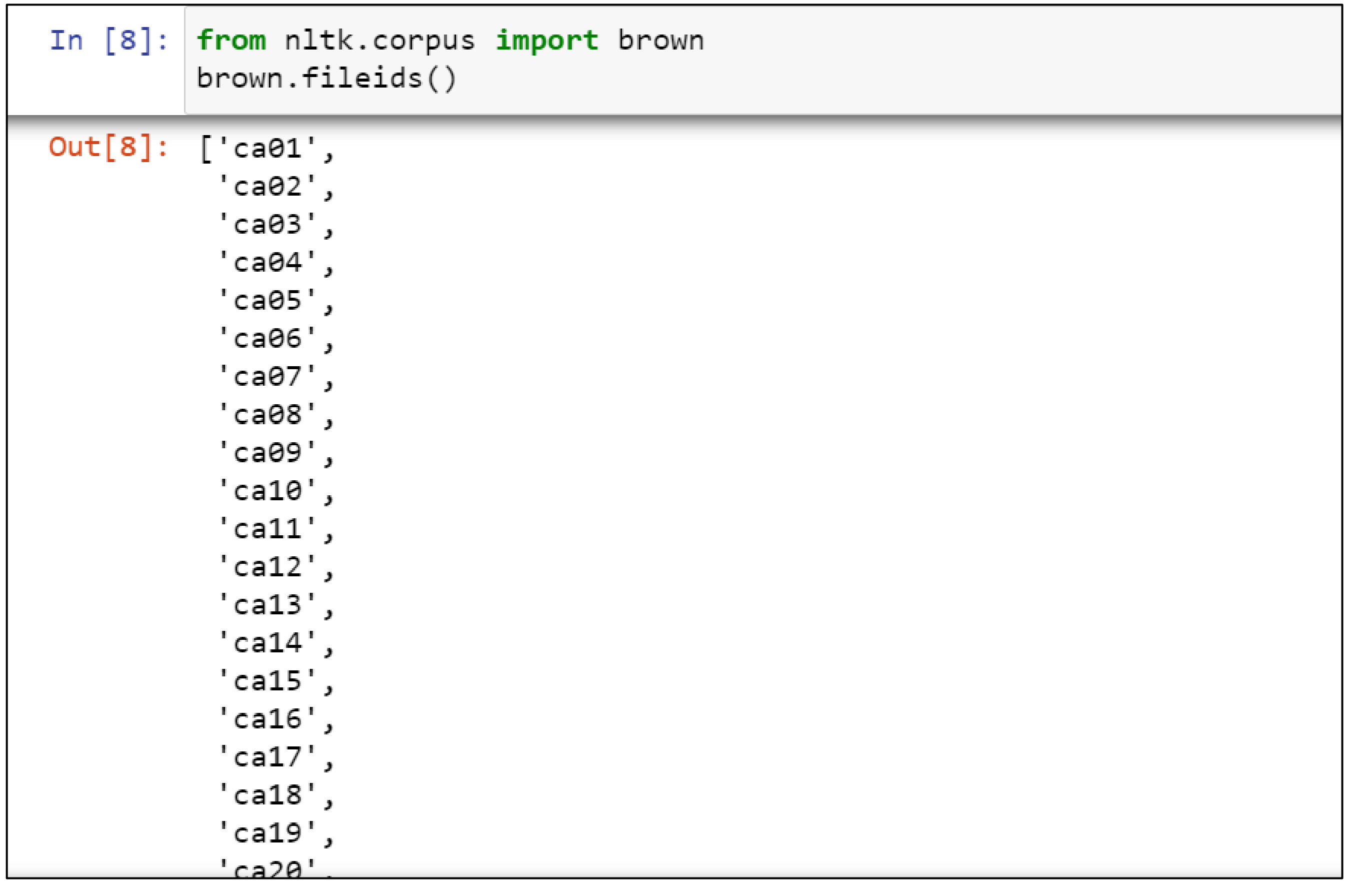
當我們想查看第一篇語料的句子列表，可以這樣：



```
In [4]: gutenberg.sents('austen-emma.txt')

Out[4]: [['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']'], ['VOLUME', 'I'], ...]
```

Corpus語料庫 CategorizedTaggedCorpusReader

我們先用brown.fileids()查有多少文本：

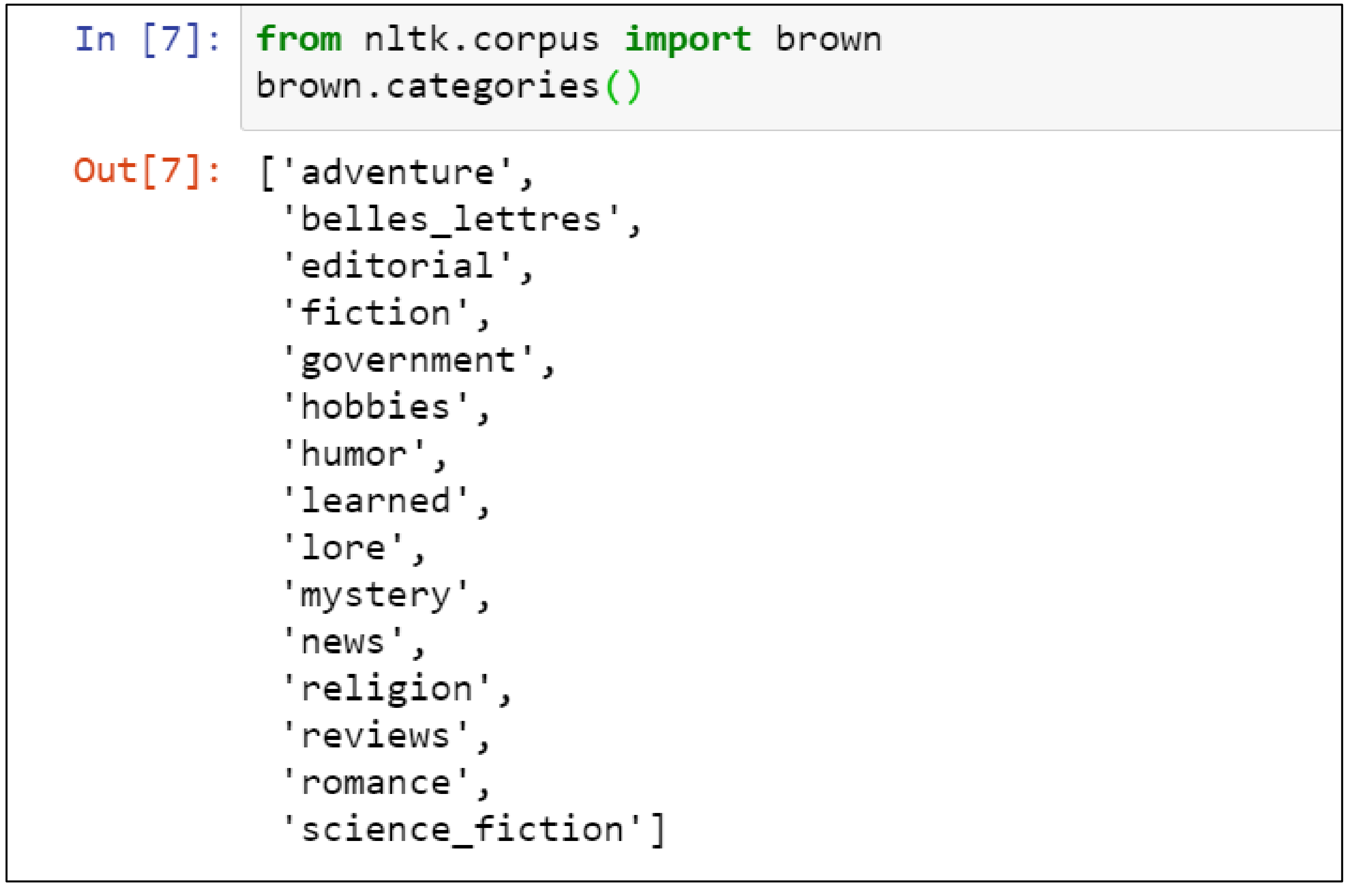


```
In [8]: from nltk.corpus import brown
        brown.fileids()

Out[8]: ['ca01',
         'ca02',
         'ca03',
         'ca04',
         'ca05',
         'ca06',
         'ca07',
         'ca08',
         'ca09',
         'ca10',
         'ca11',
         'ca12',
         'ca13',
         'ca14',
         'ca15',
         'ca16',
         'ca17',
         'ca18',
         'ca19',
         'ca20',
```

brown 語料庫是第一個百萬等級的電子語料庫(英文)，1961年由Brown University所整理，這個語料庫包含的字詞來自 500 個資料源，並參考資料源的種類做分類，例如：adventure、news、reviews…。

Corpus語料庫 corpus.categories(fileids)

當我們想查看裡面文本的分類屬性，可以這樣：

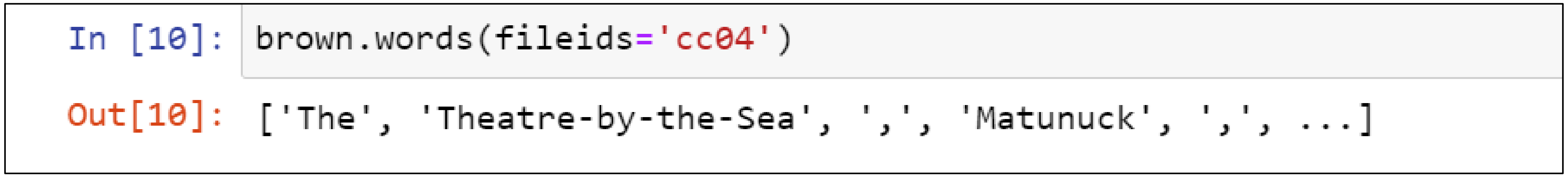


```
In [7]: from nltk.corpus import brown
        brown.categories()

Out[7]: ['adventure',
         'belles_lettres',
         'editorial',
         'fiction',
         'government',
         'hobbies',
         'humor',
         'learned',
         'lore',
         'mystery',
         'news',
         'religion',
         'reviews',
         'romance',
         'science_fiction']
```

也可以輸入特定文本來看其屬性。

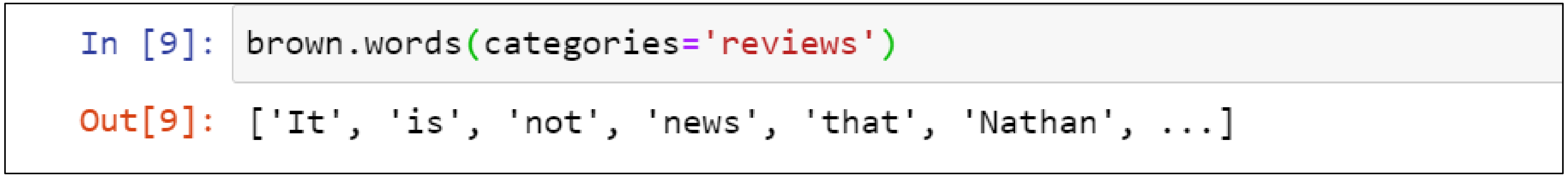
Corpus語料庫 corpus.words(fields, categories)

當我們想查看其中一本語料的單詞列表，可以這樣：



```
In [10]: brown.words(fileids='cc04')

Out[10]: ['The', 'Theatre-by-the-Sea', ',', 'Matunuck', ',', ...]
```

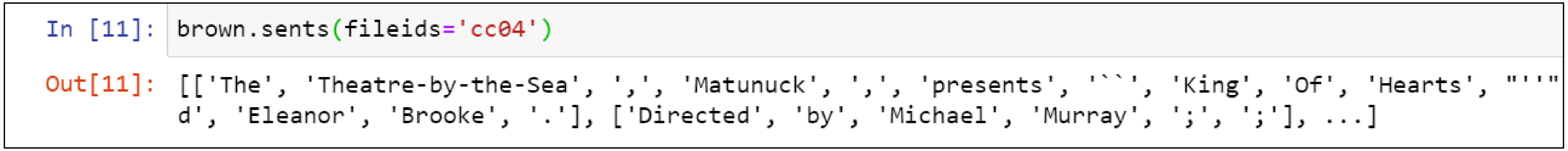
當我們想在語料庫尋找特定字詞，可以這樣：



```
In [9]: brown.words(categories='reviews')

Out[9]: ['It', 'is', 'not', 'news', 'that', 'Nathan', ...]
```

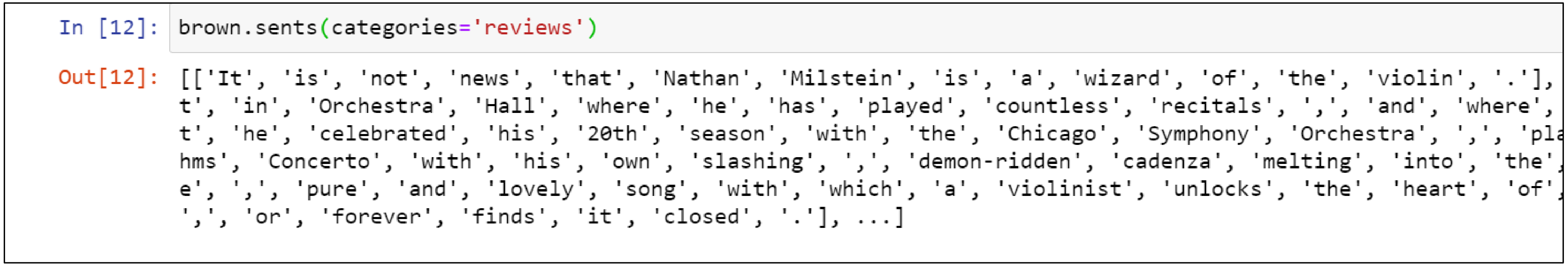
Corpus語料庫 corpus.sents(fields, categories)

當我們想查看其中一本語料的句子列表，可以這樣：



```
In [11]: brown.sents(fileids='cc04')

Out[11]: [['The', 'Theatre-by-the-Sea', ',', 'Matunuck', ',', 'presents', '``', 'King', 'Of', 'Hearts', "''"
         d', 'Eleanor', 'Brooke', '.'], ['Directed', 'by', 'Michael', 'Murray', ';', ';'], ...]
```

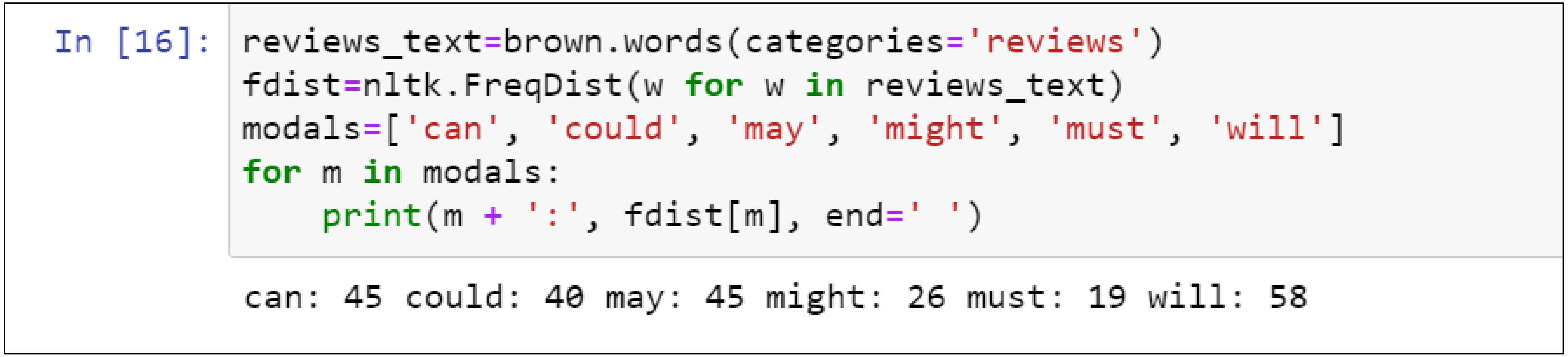
當我們想在語料庫尋找特定句子，可以這樣：



```
In [12]: brown.sents(categories='reviews')

Out[12]: [['It', 'is', 'not', 'news', 'that', 'Nathan', 'Milstein', 'is', 'a', 'wizard', 'of', 'the', 'violin', '.'],
         t', 'in', 'Orchestra', 'Hall', 'where', 'he', 'has', 'played', 'countless', 'recitals', ',', 'and', 'where',
         t', 'he', 'celebrated', 'his', '20th', 'season', 'with', 'the', 'Chicago', 'Symphony', 'Orchestra', ',', 'pla
         hms', 'Concerto', 'with', 'his', 'own', 'slashing', ',', 'demon-ridden', 'cadenza', 'melting', 'into', 'the',
         e', ',', 'pure', 'and', 'lovely', 'song', 'with', 'which', 'a', 'violinist', 'unlocks', 'the', 'heart', 'of',
         ',', 'or', 'forever', 'finds', 'it', 'closed', '.'], ...]
```

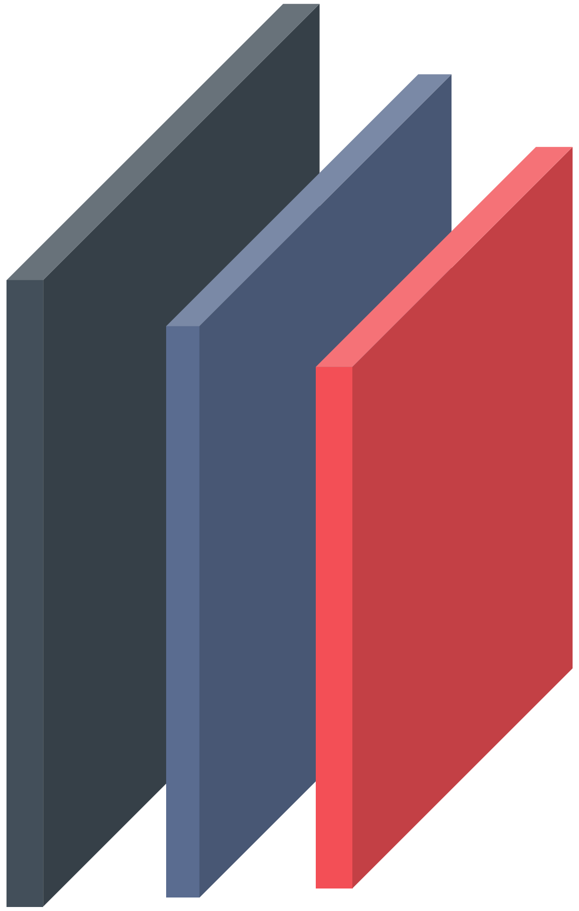
Corpus語料庫 nltk.FreqDist()

當我們想在特定類文本裡，計算特定字詞個數，可以這樣：



```
In [16]: reviews_text=brown.words(categories='reviews')
         fdist=nltk.FreqDist(w for w in reviews_text)
         modals=['can', 'could', 'may', 'might', 'must', 'will']
         for m in modals:
             print(m + ':', fdist[m], end=' ')

         can: 45 could: 40 may: 45 might: 26 must: 19 will: 58
```

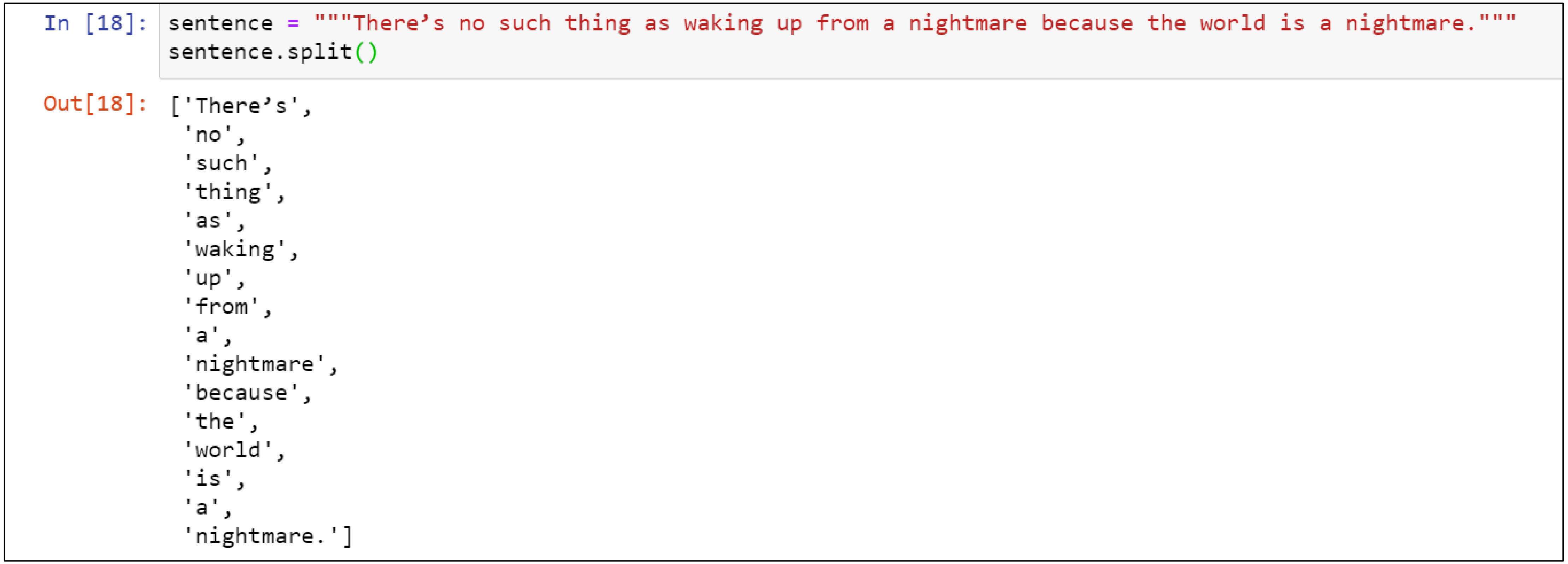
字詞處理



- Tokenize
- stopwords
- pos_tag

字詞處理 tokenize

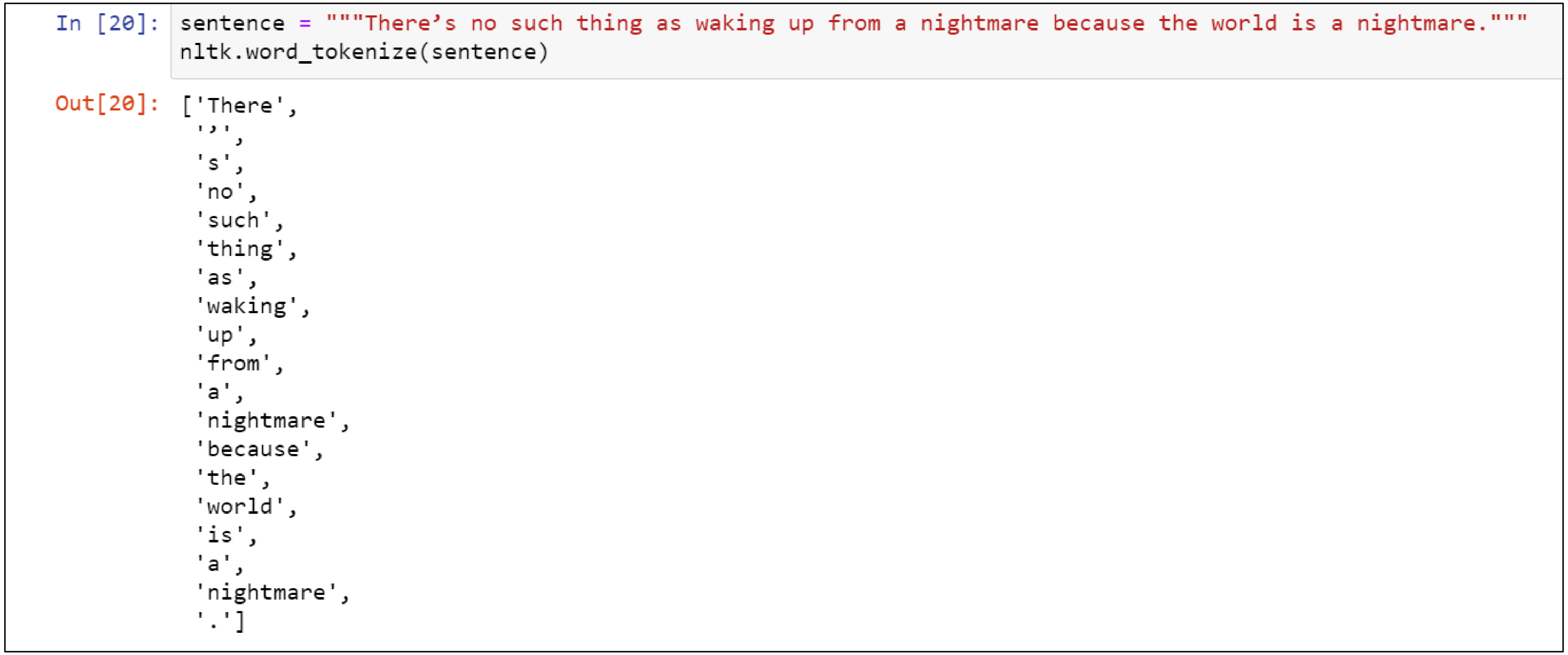
如果用split進行字詞拆解：



```
In [18]: sentence = """There’s no such thing as waking up from a nightmare because the world is a nightmare."""
         sentence.split()

Out[18]: ['There’s',
          'no',
          'such',
          'thing',
          'as',
          'waking',
          'up',
          'from',
          'a',
          'nightmare',
          'because',
          'the',
          'world',
          'is',
          'a',
          'nightmare.']
```

字詞處理 tokenize

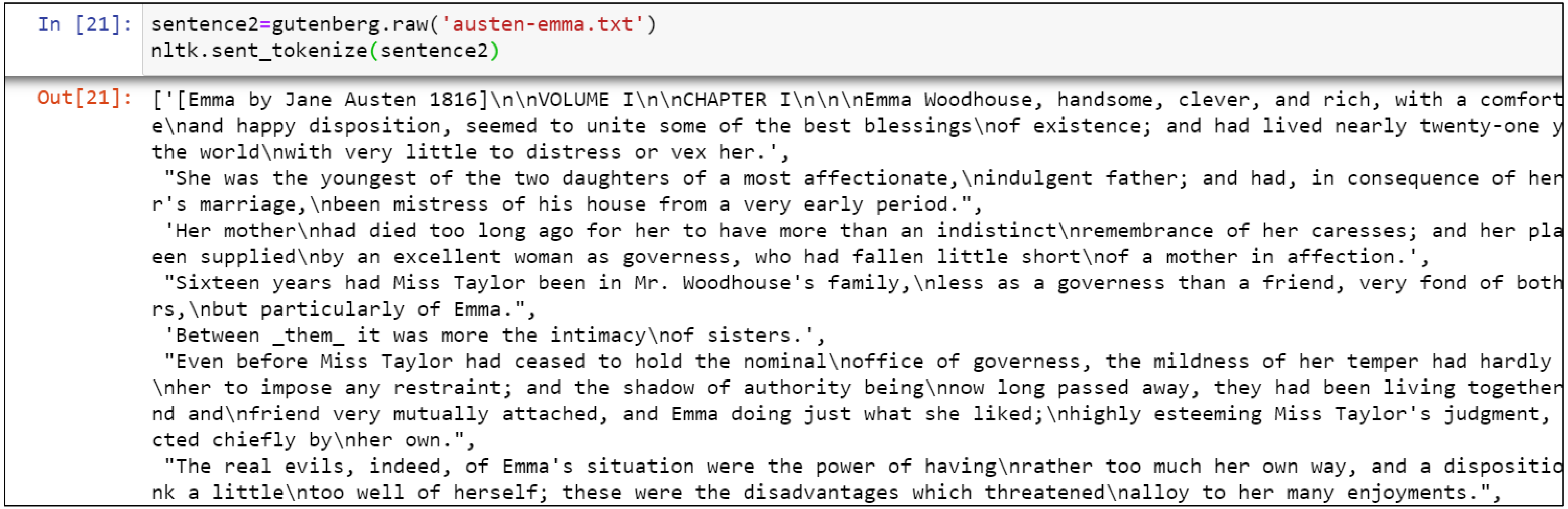
但如果用tokenize進行拆解：



```
In [20]: sentence = """There’s no such thing as waking up from a nightmare because the world is a nightmare."""
         nltk.word_tokenize(sentence)

Out[20]: ['There',
          '’',
          's',
          'no',
          'such',
          'thing',
          'as',
          'waking',
          'up',
          'from',
          'a',
          'nightmare',
          'because',
          'the',
          'world',
          'is',
          'a',
          'nightmare',
          '.']
```

字詞處理 tokenize

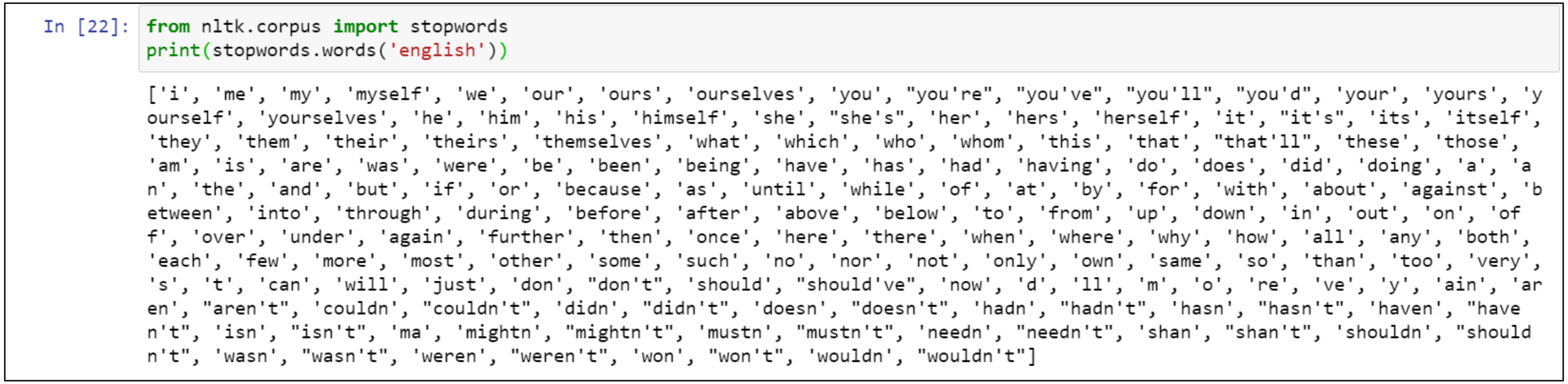
也可以運用在斷句：



```
In [21]: sentence2=gutenberg.raw('austen-emma.txt')
         nltk.sent_tokenize(sentence2)

Out[21]: ['[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, and rich, with a comfort
         e\nand happy disposition, seemed to unite some of the best blessings\nof existence; and had lived nearly twenty-one y
         the world\nwith very little to distress or vex her.',
          "She was the youngest of the two daughters of a most affectionate,\nindulgent father; and had, in consequence of her
         r's marriage,\nbeen mistress of his house from a very early period.",
          'Her mother\nhad died too long ago for her to have more than an indistinct\nremembrance of her caresses; and her pla
         een supplied\nby an excellent woman as governess, who had fallen little short\nof a mother in affection.',
          "Sixteen years had Miss Taylor been in Mr. Woodhouse's family,\nless as a governess than a friend, very fond of both
         rs,\nbut particularly of Emma.",
          'Between _them_ it was more the intimacy\nof sisters.',
          "Even before Miss Taylor had ceased to hold the nominal\noffice of governess, the mildness of her temper had hardly
         \nher to impose any restraint; and the shadow of authority being\nnow long passed away, they had been living together
         nd and\nfriend very mutually attached, and Emma doing just what she liked;\nhighly esteeming Miss Taylor's judgment,
         cted chiefly by\nher own.",
          "The real evils, indeed, of Emma's situation were the power of having\nrather too much her own way, and a dispositio
         nk a little\ntoo well of herself; these were the disadvantages which threatened\nalloy to her many enjoyments.",
```

字詞處理 stopwords

停用字像是that, then, such等等，通常沒有什麼詞彙價值，因此會讓我們在區別文本時出現混淆，因此在處理文本前可以先過濾掉：



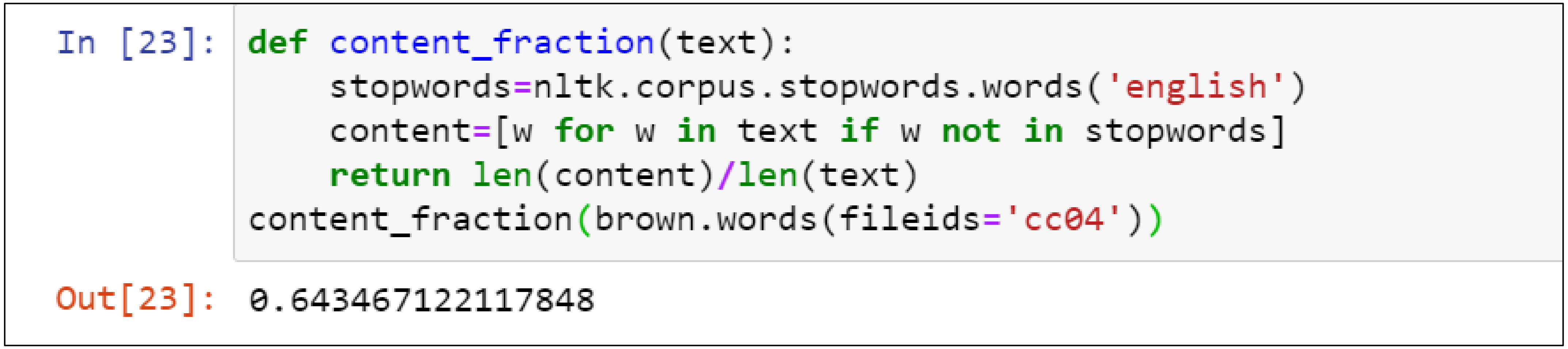
```
In [22]: from nltk.corpus import stopwords
         print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

以上是在英文裡規範的停用詞。

字詞處理 stopwords

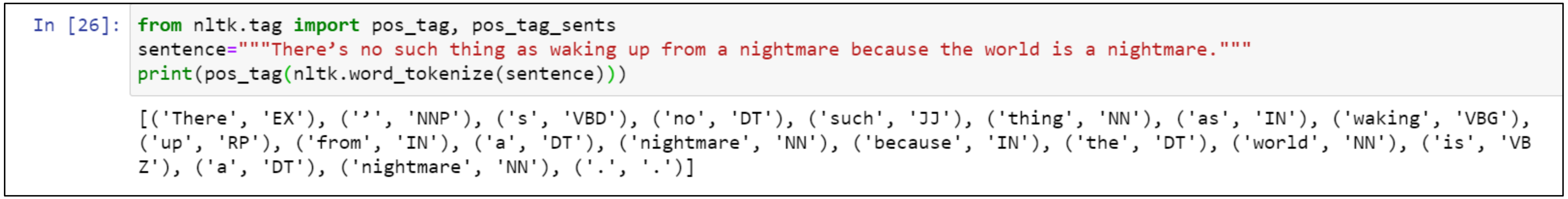
我們可以寫一個function來看非停用詞佔整篇文本的比例：



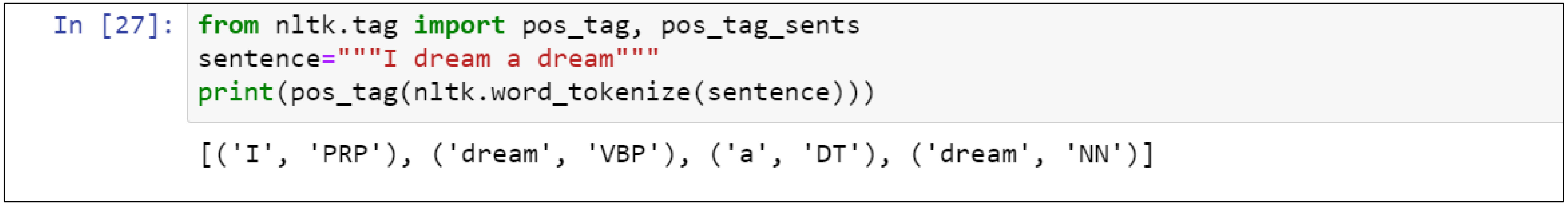
```
In [23]: def content_fraction(text):
             stopwords=nltk.corpus.stopwords.words('english')
             content=[w for w in text if w not in stopwords]
             return len(content)/len(text)
         content_fraction(brown.words(fileids='cc04'))

Out[23]: 0.643467122117848
```

字詞處理 pos_tag

此外，使用pos_tag()還可以看句子裡的詞性：



```
In [26]: from nltk.tag import pos_tag, pos_tag_sents
         sentence="""There’s no such thing as waking up from a nightmare because the world is a nightmare."""
         print(pos_tag(nltk.word_tokenize(sentence)))

         [('There', 'EX'), ('’', 'NNP'), ('s', 'VBD'), ('no', 'DT'), ('such', 'JJ'), ('thing', 'NN'), ('as', 'IN'), ('waking', 'VBG'),
         ('up', 'RP'), ('from', 'IN'), ('a', 'DT'), ('nightmare', 'NN'), ('because', 'IN'), ('the', 'DT'), ('world', 'NN'), ('is', 'VB
         Z'), ('a', 'DT'), ('nightmare', 'NN'), ('.', '.')]
```



```
In [27]: from nltk.tag import pos_tag, pos_tag_sents
         sentence="""I dream a dream"""
         print(pos_tag(nltk.word_tokenize(sentence)))

         [('I', 'PRP'), ('dream', 'VBP'), ('a', 'DT'), ('dream', 'NN')]
```

04

WordNet介紹

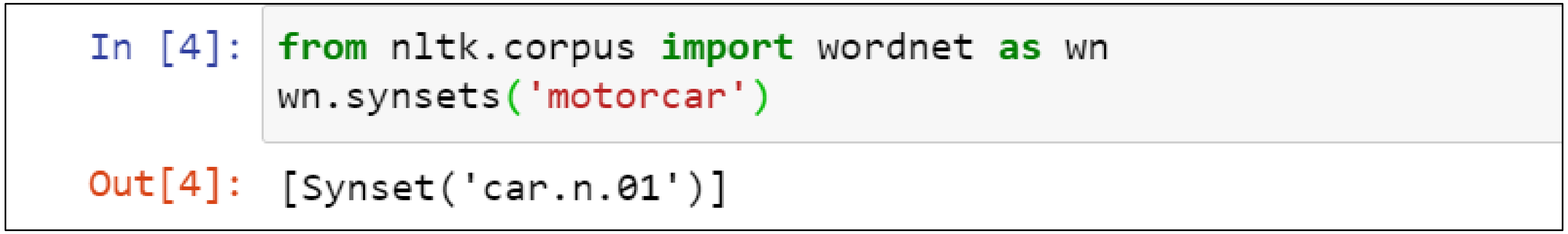
WordNet介紹

WordNet是語意導向的英文字典，類似含有同義詞的字典。

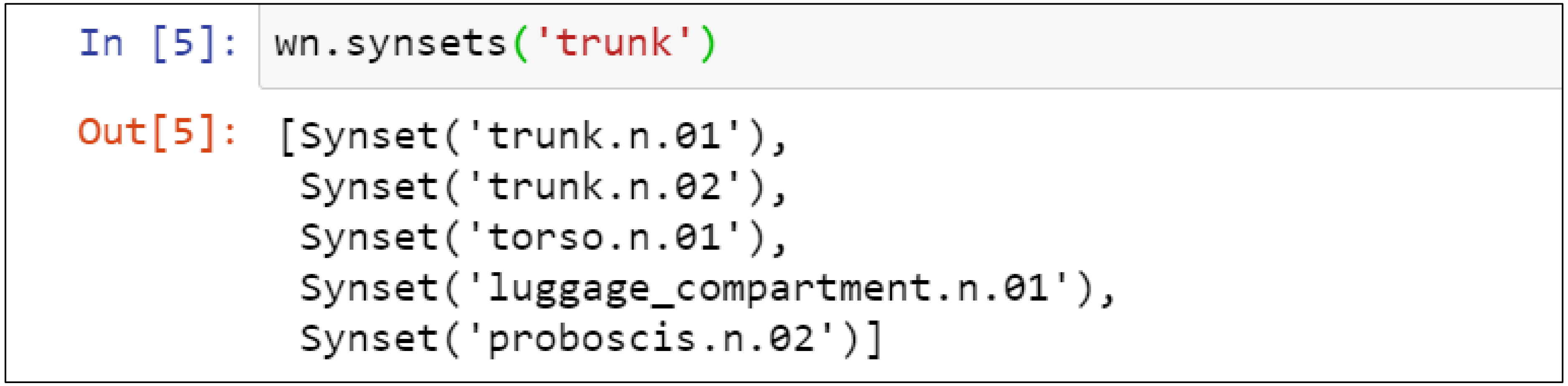
WordNet將每一個字詞進行分組，而分組後得到的每一個具有相同意義的字詞組，就稱為synset（同義詞集合）。
每一個synset都有一個定義，記錄著不同synset的語義關係。

WordNet介紹 wn. synsets()

我們想找出單字的同義詞集合，可以這樣：



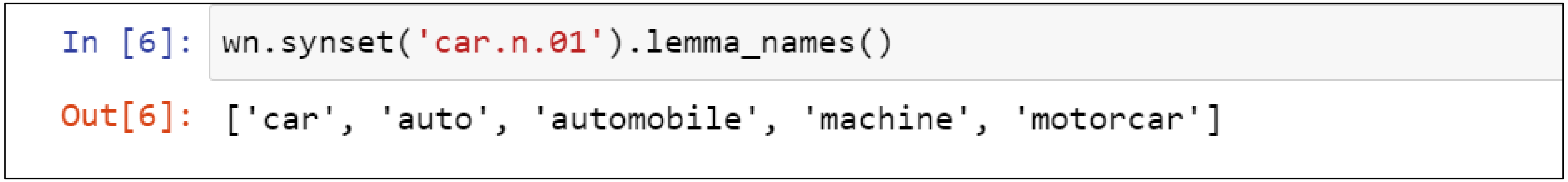
```
In [4]:  from nltk.corpus import wordnet as wn
         wn.synsets('motorcar')

Out[4]:  [Synset('car.n.01')]
```



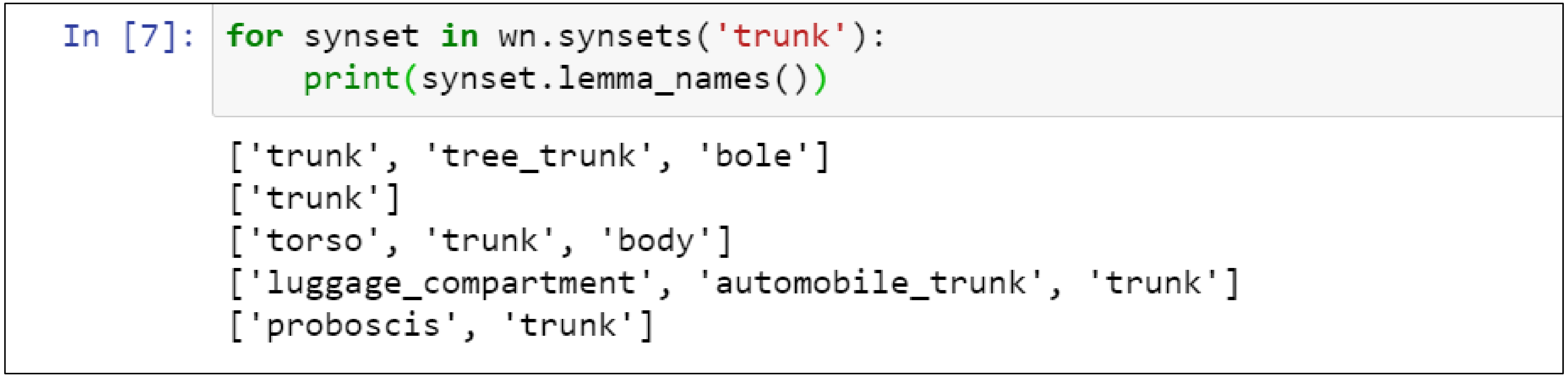
```
In [5]:  wn.synsets('trunk')

Out[5]:  [Synset('trunk.n.01'),
          Synset('trunk.n.02'),
          Synset('torso.n.01'),
          Synset('luggage_compartment.n.01'),
          Synset('proboscis.n.02')]
```

WordNet介紹 wn.synset().lemma_names()

我們想找出同義詞組裡有哪些字詞，可以這樣：



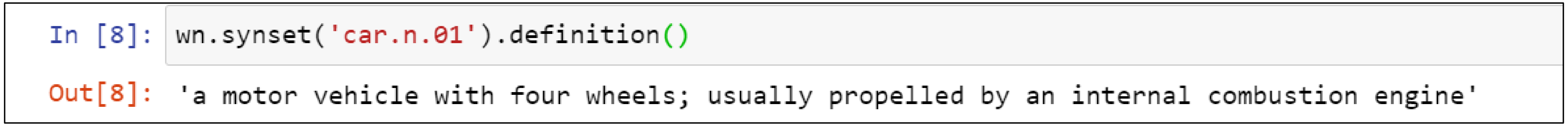
```
In [6]: wn.synset('car.n.01').lemma_names()

Out[6]: ['car', 'auto', 'automobile', 'machine', 'motorcar']
```



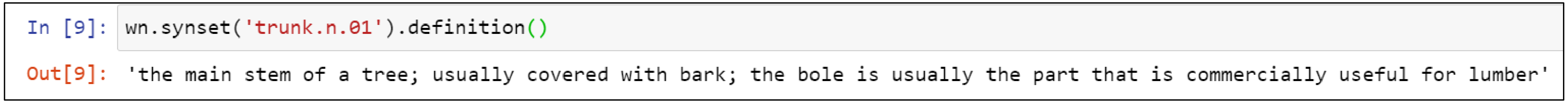
```
In [7]: for synset in wn.synsets('trunk'):
            print(synset.lemma_names())

        ['trunk', 'tree_trunk', 'bole']
        ['trunk']
        ['torso', 'trunk', 'body']
        ['luggage_compartment', 'automobile_trunk', 'trunk']
        ['proboscis', 'trunk']
```

WordNet介紹 wn.synset().definiton()

我們想知道該synset的定義，可以這樣：



```
In [8]: wn.synset('car.n.01').definition()

Out[8]: 'a motor vehicle with four wheels; usually propelled by an internal combustion engine'
```

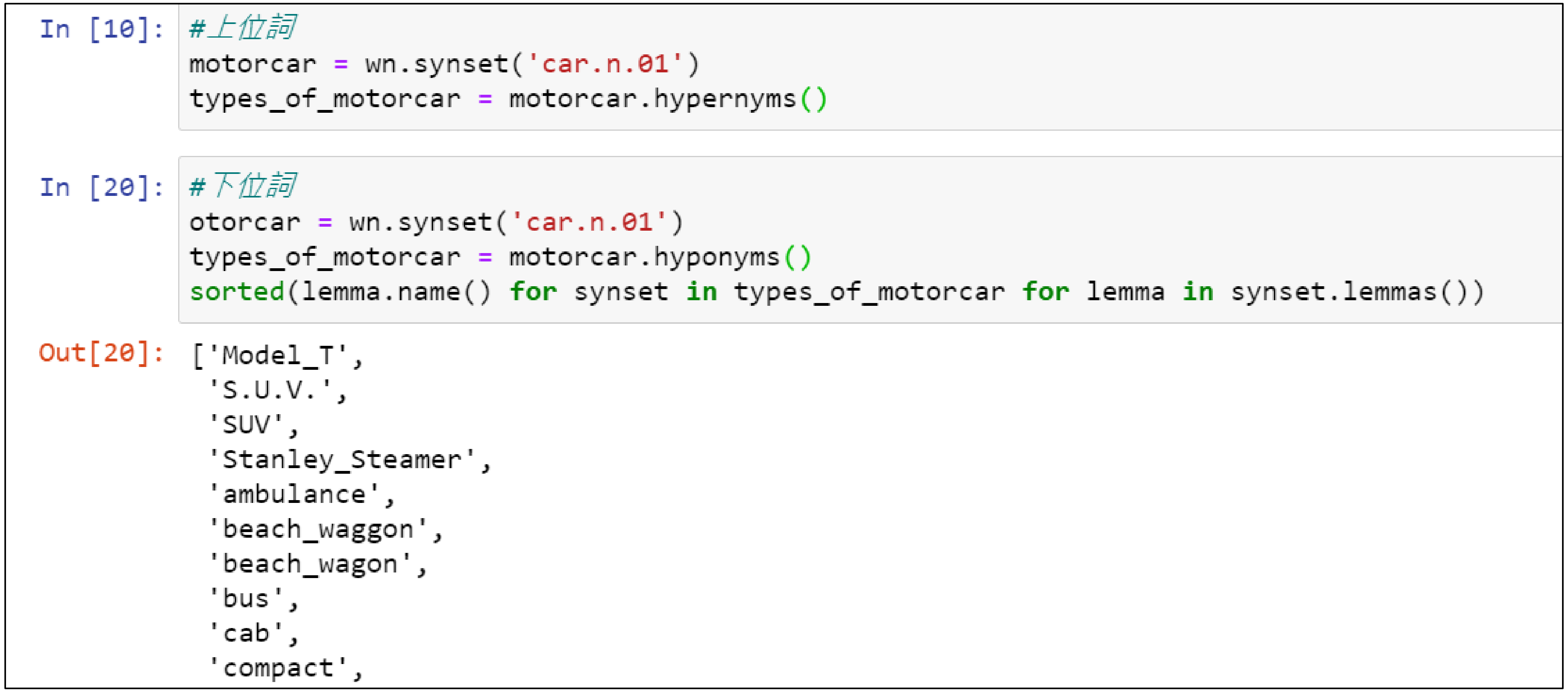


```
In [9]: wn.synset('trunk.n.01').definition()

Out[9]: 'the main stem of a tree; usually covered with bark; the bole is usually the part that is commercially useful for lumber'
```

WordNet介紹

wn. synset(). hypernyms()
wn. synset(). hyponyms()

hypernym/hyponym （上位詞與下位詞）
對字詞做分類，也是WordNet的一大特色與亮點，就像界門綱目科屬種。
如果我們想找出上位詞或下位詞，可以這樣：

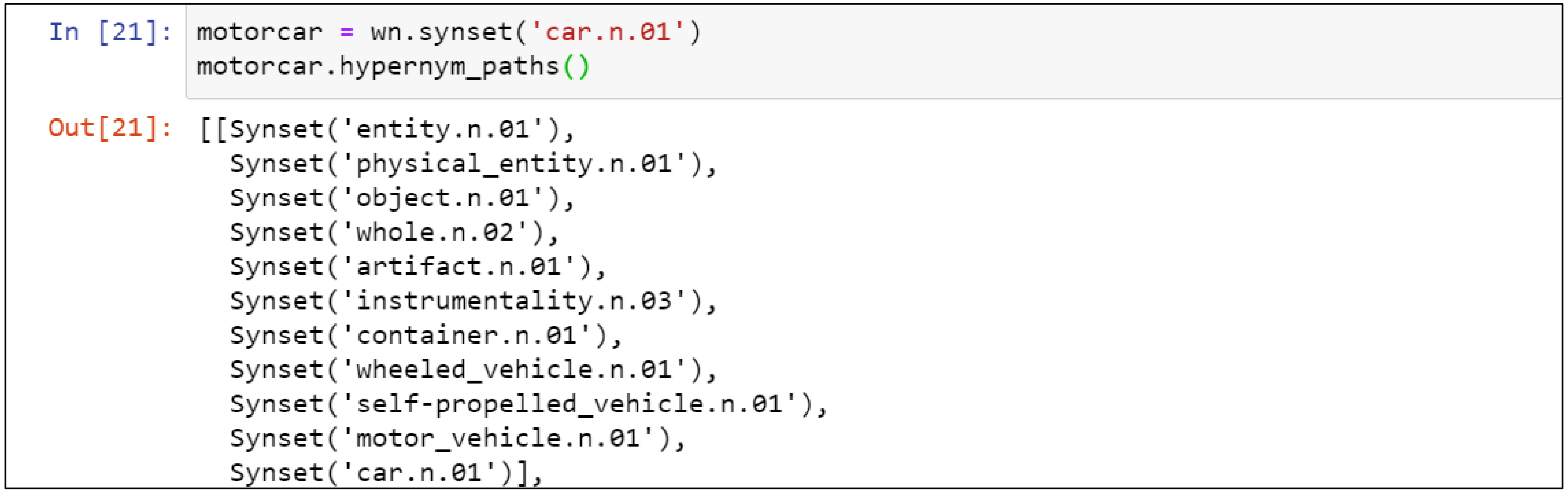


```
In [10]: #上位詞
         motorcar = wn.synset('car.n.01')
         types_of_motorcar = motorcar.hypernyms()

In [20]: #下位詞
         otorcar = wn.synset('car.n.01')
         types_of_motorcar = motorcar.hyponyms()
         sorted(lemma.name() for synset in types_of_motorcar for lemma in synset.lemmas())

Out[20]: ['Model_T',
          'S.U.V.',
          'SUV',
          'Stanley_Steamer',
          'ambulance',
          'beach_waggon',
          'beach_wagon',
          'bus',
          'cab',
          'compact',
```

WordNet介紹

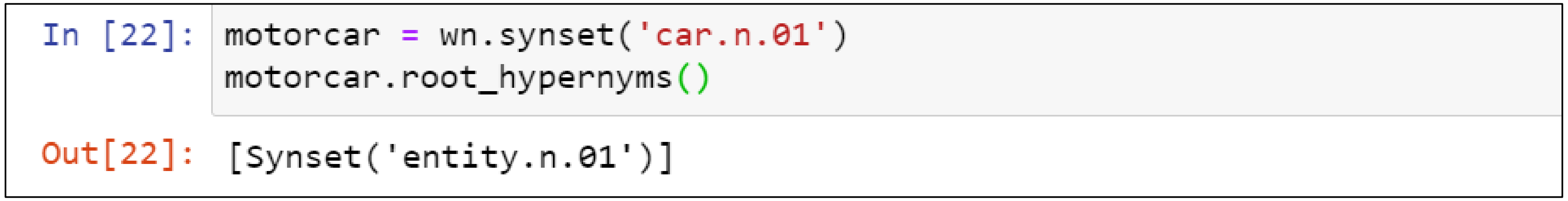
wn. synset(). hypernym_paths()
wn. synset(). root_hypernym

如果我們想找出該詞組的路徑(也就是上位詞組在往上走)，可以這樣：



```
In [21]: motorcar = wn.synset('car.n.01')
         motorcar.hypernym_paths()

Out[21]: [[Synset('entity.n.01'),
           Synset('physical_entity.n.01'),
           Synset('object.n.01'),
           Synset('whole.n.02'),
           Synset('artifact.n.01'),
           Synset('instrumentality.n.03'),
           Synset('container.n.01'),
           Synset('wheeled_vehicle.n.01'),
           Synset('self-propelled_vehicle.n.01'),
           Synset('motor_vehicle.n.01'),
           Synset('car.n.01')],
```

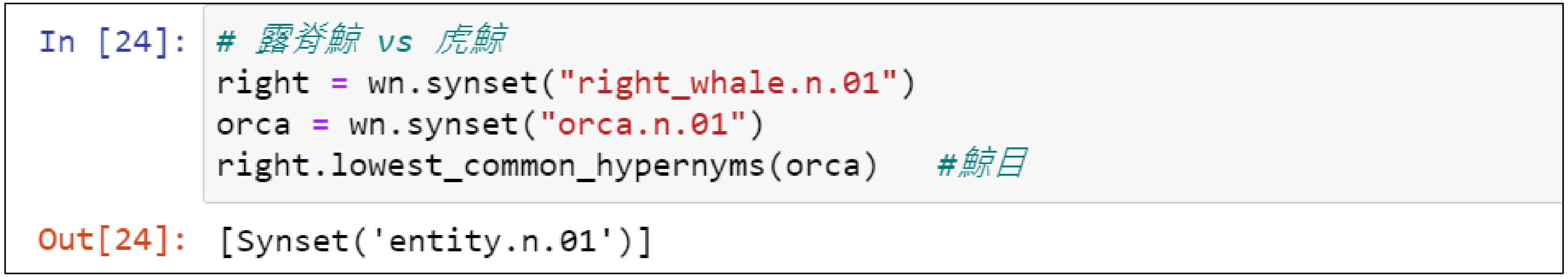
如果我們想找出最頂端的上位詞組，可以這樣：



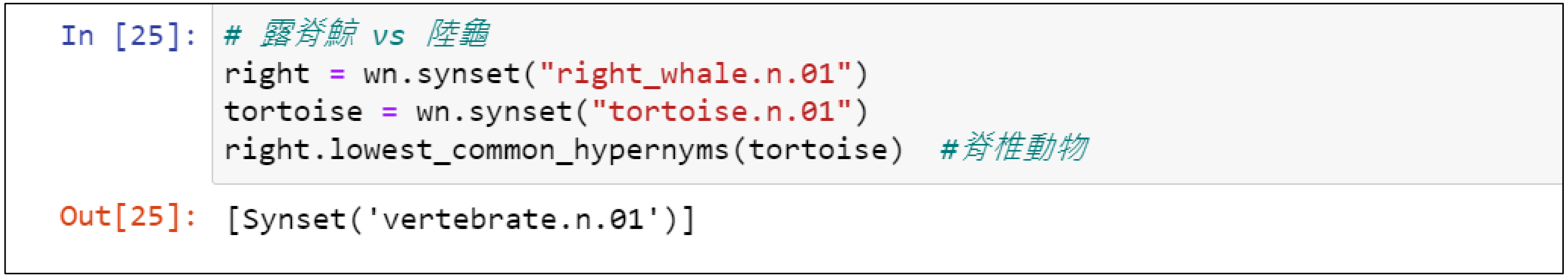
```
In [22]: motorcar = wn.synset('car.n.01')
         motorcar.root_hypernyms()

Out[22]: [Synset('entity.n.01')]
```

WordNet介紹 wn.synset().lowest_common_hypernyms()

如果我們想找出兩詞組間的最低位共同詞組，可以這樣：



```
In [24]: # 露脊鯨 vs 虎鯨
         right = wn.synset("right_whale.n.01")
         orca = wn.synset("orca.n.01")
         right.lowest_common_hypernyms(orca)    #鯨目

Out[24]: [Synset('entity.n.01')]
```

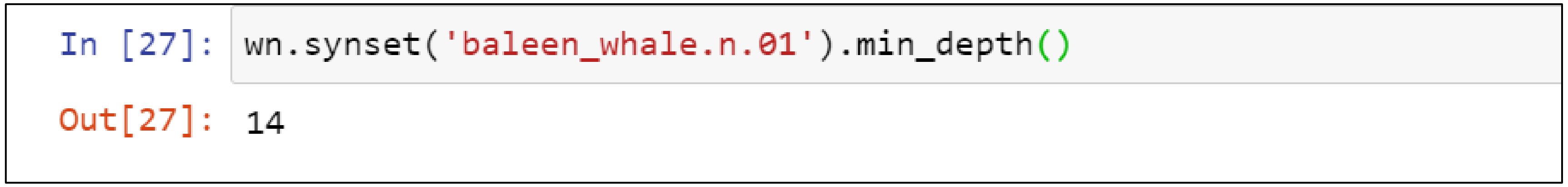


```
In [25]: # 露脊鯨 vs 陸龜
         right = wn.synset("right_whale.n.01")
         tortoise = wn.synset("tortoise.n.01")
         right.lowest_common_hypernyms(tortoise)  #脊椎動物

Out[25]: [Synset('vertebrate.n.01')]
```

WordNet介紹

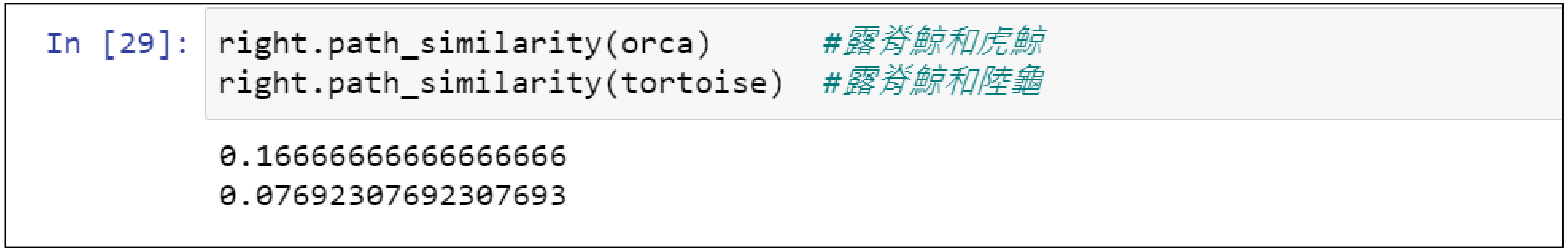
wn.synset().min_depth()
wn.synset().path_similarity

如果我們想知道該詞組的階層等級，可以這樣：



```
In [27]: wn.synset('baleen_whale.n.01').min_depth()

Out[27]: 14
```

如果我們想知道上下位詞組結構的相似度，可以這樣：



```
In [29]: right.path_similarity(orca)       #露脊鯨和虎鯨
         right.path_similarity(tortoise)   #露脊鯨和陸龜

         0.16666666666666666
         0.07692307692307693
```

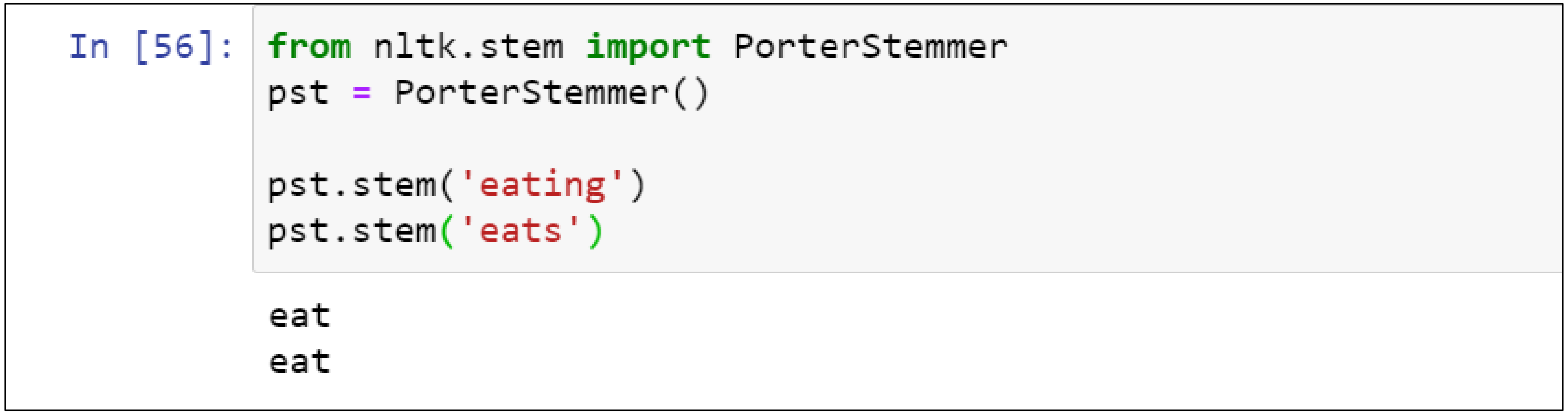
05

補充

補充 stemming

詞幹提取(stemming)：

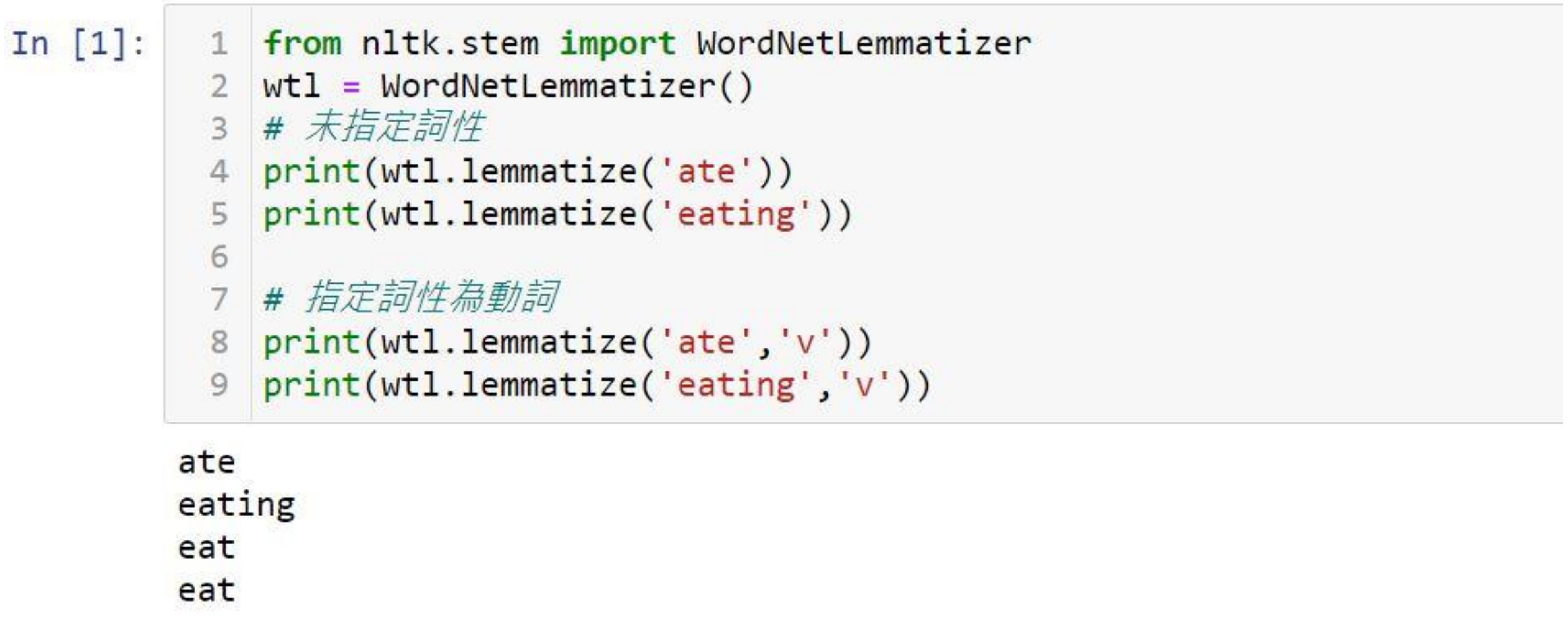
一個動詞會有很多變化，像是過去是、完成式。詞幹提取器可以移除 -s/es、-ing、-ed。



```
In [56]: from nltk.stem import PorterStemmer
         pst = PorterStemmer()

         pst.stem('eating')
         pst.stem('eats')

         eat
         eat
```

補充 Lemmatization

有別於詞幹提取，詞性還原(Lemmatization)涵蓋了詞根的文法和變化形式，它運用了不同的標準化規則（像是上下文情境和詞性），來獲取相關的詞根(lemma)，將各種類型的詞的變形，歸成一致的形式。

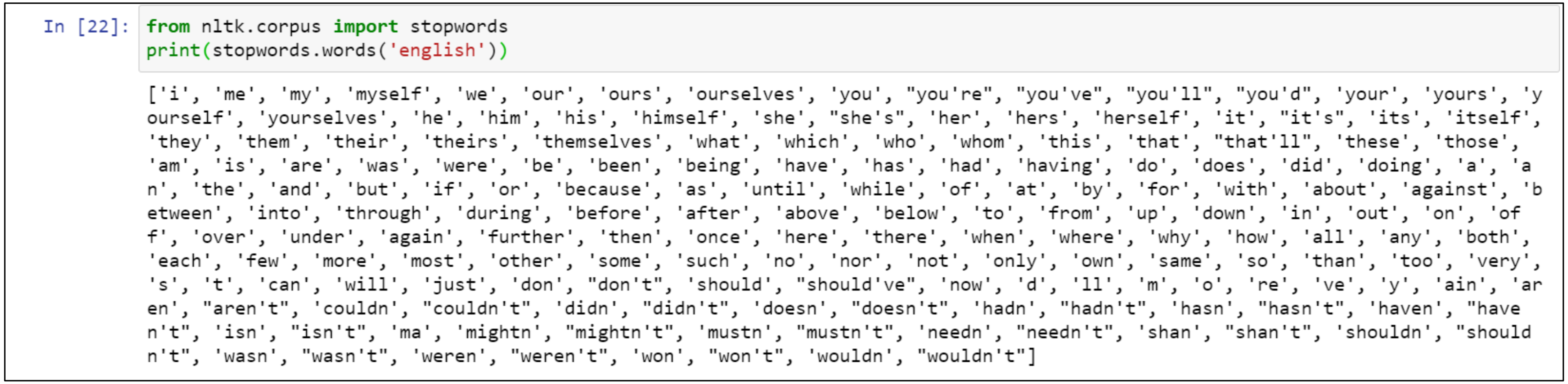


```
In [1]:
from nltk.stem import WordNetLemmatizer
wtl = WordNetLemmatizer()
# 未指定詞性
print(wtl.lemmatize('ate'))
print(wtl.lemmatize('eating'))

# 指定詞性為動詞
print(wtl.lemmatize('ate','v'))
print(wtl.lemmatize('eating','v'))
```

```
ate
eating
eat
eat
```

補充 RMSW(Remove Stop words)

在每篇文章中這些單字（定冠詞、介系詞、代名詞）都會是高頻單字，因此似乎沒什麼必要保留這些不太帶有資訊的單詞。



```
In [22]: from nltk.corpus import stopwords
         print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

相反的，動詞、名詞、形容詞、副詞這四類單字包含者份文件的絕大多數訊息，應予以保留。

補充 RMSW(Remove Stop words)

至於有一些觀點說明 RMSW 會影響效能，例如：

This is not a great movie (negative)
This is a great movie (positive)

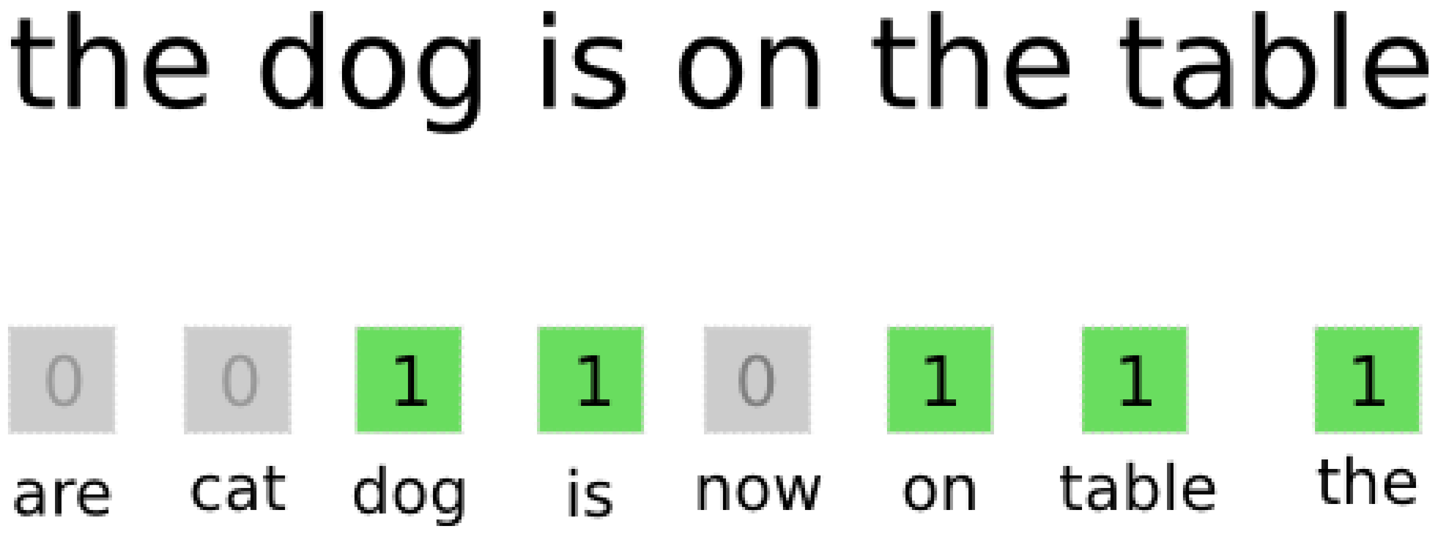
由於 This, is, not, a 都是 stop words ，所以這兩句話會變成
great movie (positive)
great movie (positive)

雖然大部分的情況 RMSW會讓模型更專注在訊息量較大的單詞，但上述的狀況仍很常發生，那究竟要不要 remove stop words 呢？這沒有唯一答案，但或許可以只使用部分的停用詞就好(he, she, him, her 等人稱代名詞)

補充 Bag of words

如果我們要讓機器學習如何對影評，或者餐廳評論做分類（正面、負面），
第一個出現在我們眼前的問題是：機器要怎麼看懂文字？

一種方法就是詞袋(Bag of words)

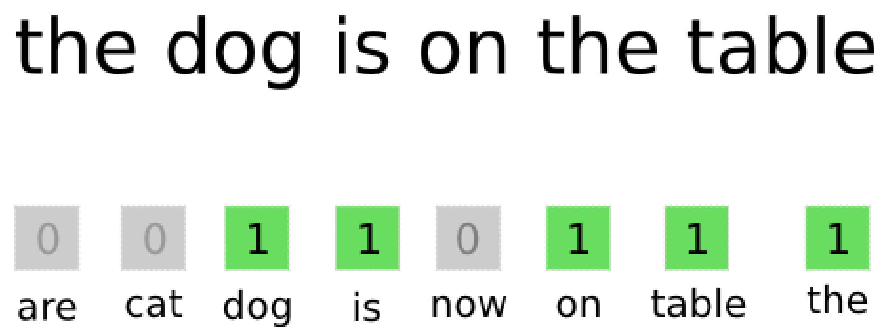


補充 One-hot encoding

詞袋的優點：以簡單的向量形式來紀錄單詞出現的頻率

詞袋的缺點：

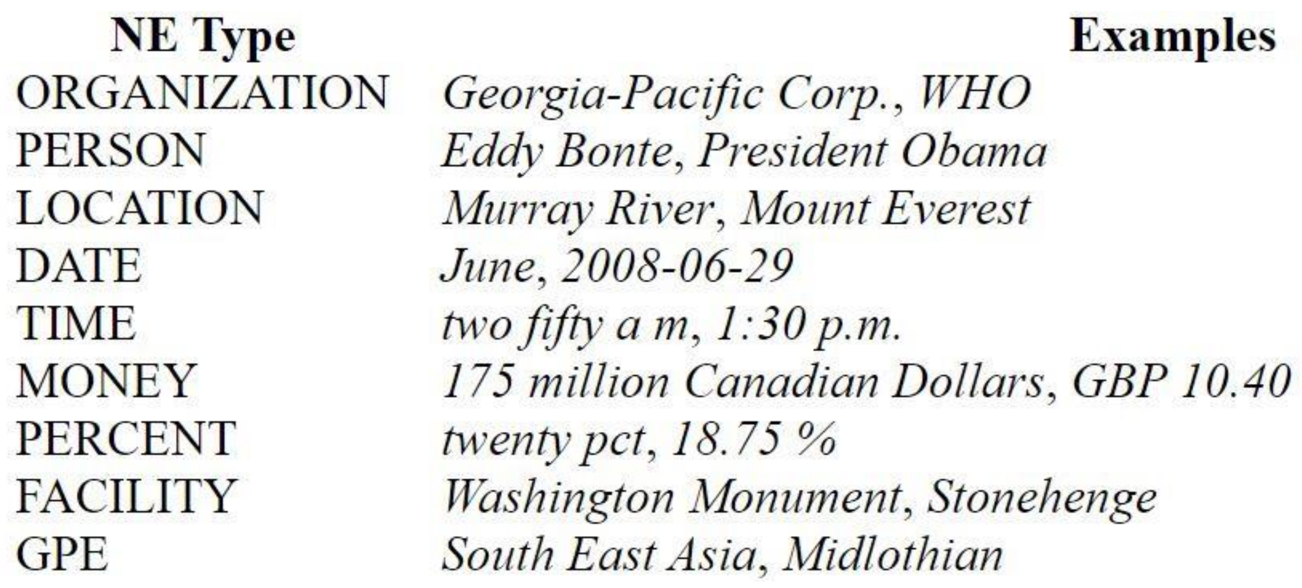
1. 文字之間的順序在詞袋之中無法被保存；
2. 文字之間的「意義」沒有被保存，「貓」和「狗」以及「貓」和「飛機」，貓和狗應該是較為相似的，但在詞袋之中也沒辦法體現這種「距離」的概念；
3. 動詞及名詞的變化(複數、過去分詞)等等的情況要再提前轉換成單數動詞等等。



補充 Named Entity Recognition(NER)

命名實體識別指識別文本中具有特定意義的實體，主要包括人名、地名、機構名、專有名詞等，以及時間、數量、貨幣、比例數值等文字。

例如，「歐巴馬是美國總統」的「歐巴馬」和「美國」都代表一個具體事物，因此都是命名實體。而「總統」不代表一個具體事物，因此不是命名實體。



NE Type	Examples
ORGANIZATION	*Georgia-Pacific Corp., WHO*
PERSON	*Eddy Bonte, President Obama*
LOCATION	*Murray River, Mount Everest*
DATE	*June, 2008-06-29*
TIME	*two fifty a m, 1:30 p.m.*
MONEY	*175 million Canadian Dollars, GBP 10.40*
PERCENT	*twenty pct, 18.75 %*
FACILITY	*Washington Monument, Stonehenge*
GPE	*South East Asia, Midlothian*

BUSINESS REPORT

THANK YOU