文字資料分析

NLTK

報告大綱

1. 什麼是 NLTK ?
2. 安裝 NLTK
3. Nltk.text.Text() 類別
4. 標註詞性與結構
5. 邏輯判斷

What is NLTK ?

NLTK：Natural Language Toolkit

所謂的「自然語言」，就是我們平時在與人溝通時用的語言，舉凡中文、英文、法文、德文、日文...都是自然語言。

在分析資料的時候，如何讓電腦可以分析非結構化、無欄位值的資料，就是自然語言處理的領域。

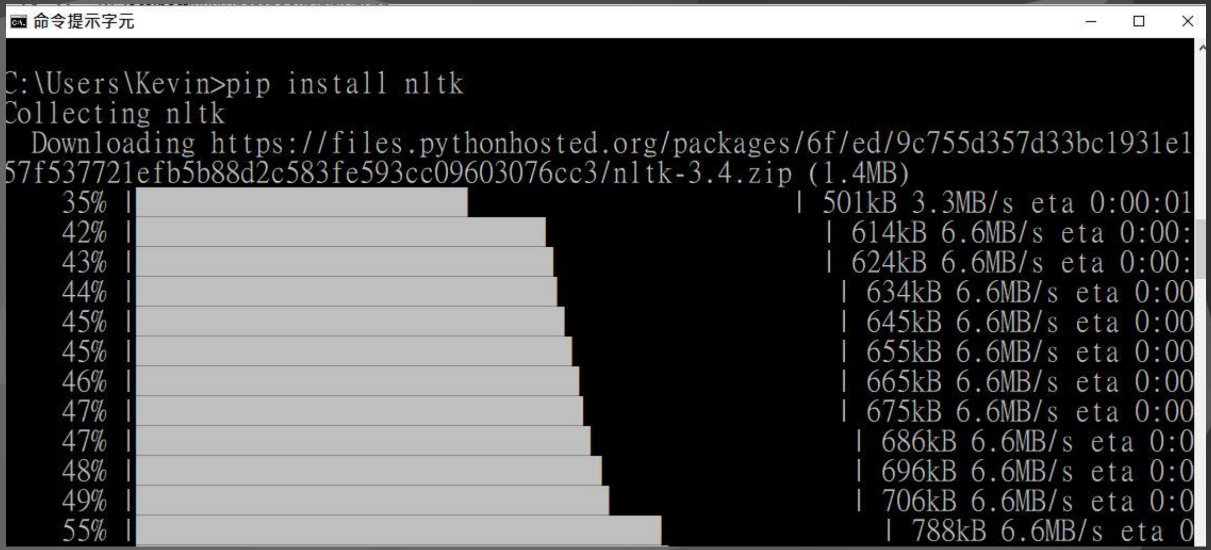
而 NLTK 就是為了處理自然語言而生的套件！

安裝 NLTK

與一般套件相同，使用 pip install 即可安裝

在終端機輸入 pip install nltk

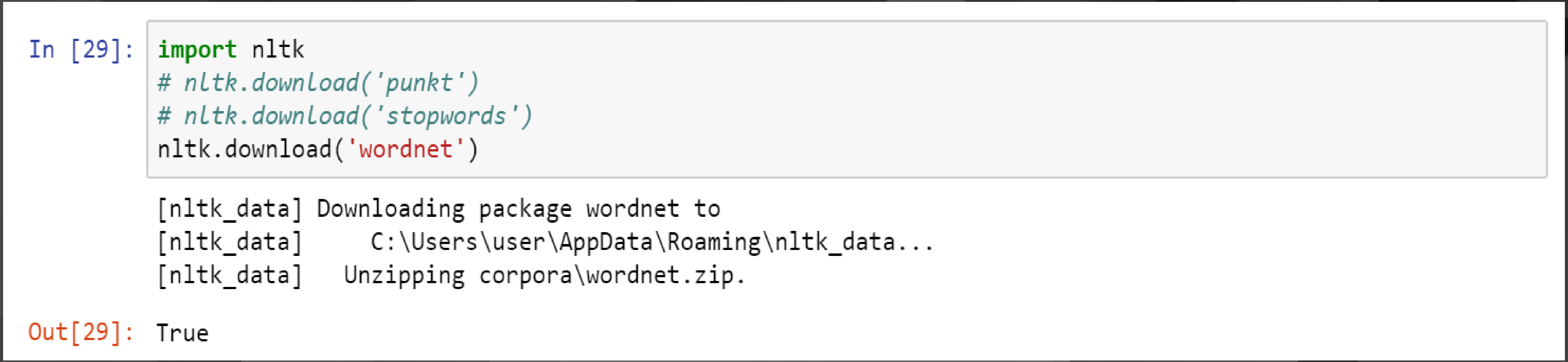
並在程式中 import nltk 即可


命令提示字元

```
C:\Users\Kevin>pip install nltk
Collecting nltk
  Downloading https://files.pythonhosted.org/packages/6f/ed/9c755d357d33bc1931e1
57f537721efb5b88d2c583fe593cc09603076cc3/nltk-3.4.zip (1.4MB)
    35% |███████████▌                    | 501kB 3.3MB/s eta 0:00:01
    42% |█████████████▋                    | 614kB 6.6MB/s eta 0:00:
    43% |█████████████▉                    | 624kB 6.6MB/s eta 0:00:
    44% |██████████████                    | 634kB 6.6MB/s eta 0:00
    45% |██████████████▍                    | 645kB 6.6MB/s eta 0:00
    45% |██████████████▋                    | 655kB 6.6MB/s eta 0:00
    46% |██████████████▉                    | 665kB 6.6MB/s eta 0:00
    47% |███████████████                    | 675kB 6.6MB/s eta 0:00
    47% |███████████████▏                    | 686kB 6.6MB/s eta 0:0
    48% |███████████████▌                    | 696kB 6.6MB/s eta 0:0
    49% |███████████████▊                    | 706kB 6.6MB/s eta 0:0
    55% |█████████████████▌                    | 788kB 6.6MB/s eta 0
```

下載 額外的辭庫

NLTK 是一個很龐大的套件，單單使用 pip 安裝 NLTK，有時不足以應付我們的需求

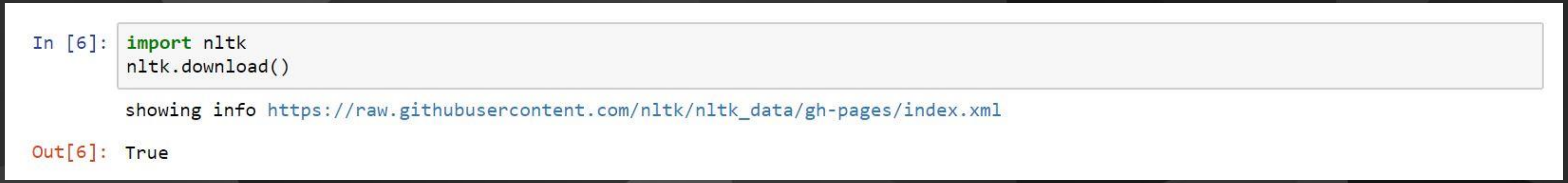


```
In [29]: import nltk
         # nltk.download('punkt')
         # nltk.download('stopwords')
         nltk.download('wordnet')

         [nltk_data] Downloading package wordnet to
         [nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
         [nltk_data]   Unzipping corpora\wordnet.zip.

Out[29]: True
```

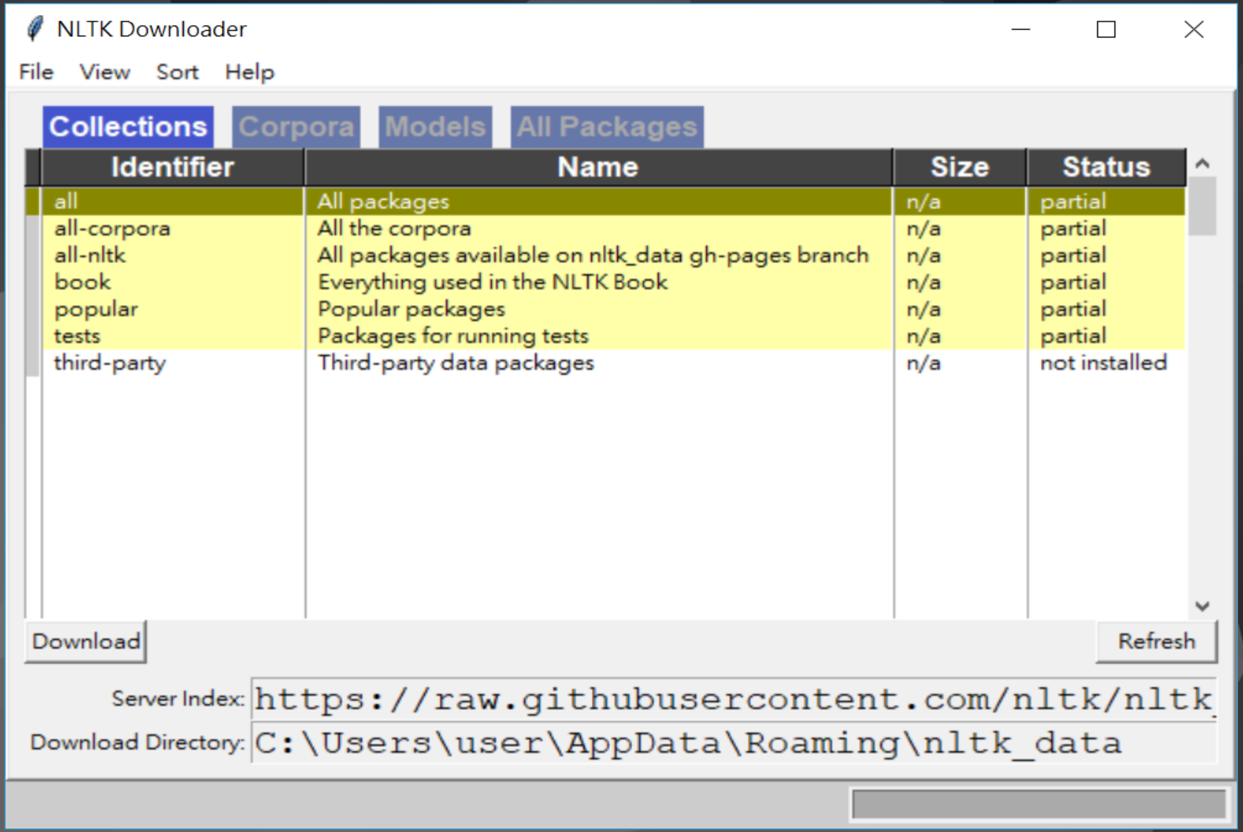
需要額外下載什麼，Compiler 都會告訴你

下載 額外套件的管理工具

NLTK Downloader



```
In [6]: import nltk
        nltk.download()

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

Out[6]: True
```



Identifier	Name	Size	Status
all	All packages	n/a	partial
all-corpora	All the corpora	n/a	partial
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	partial
tests	Packages for running tests	n/a	partial
third-party	Third-party data packages	n/a	not installed

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

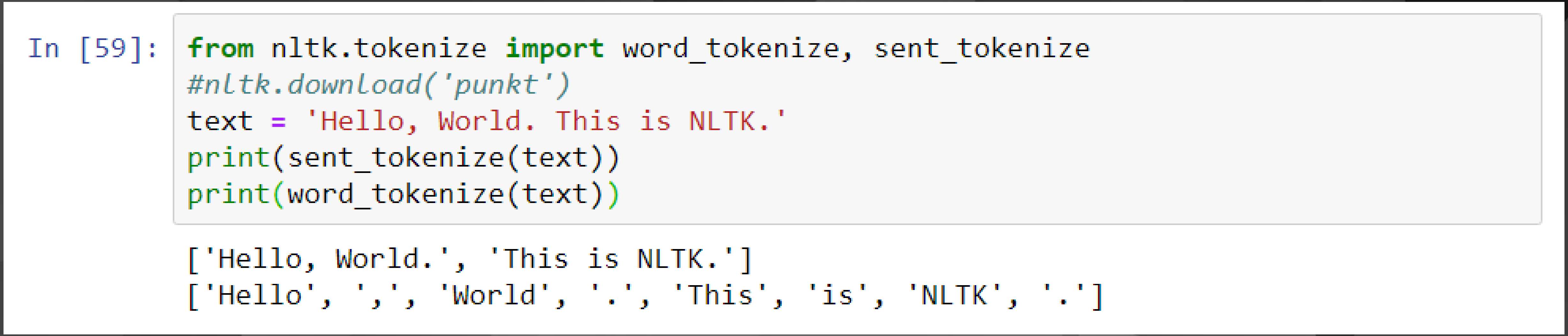
Download Directory: C:\Users\user\AppData\Roaming\nltk_data

nltk.text.Text() 類別

方法	作用
text(words)	對象構造
concordance(word, width = 79, lines = 25)	顯示word出現的上下文
common_contexts(word)	顯示word出現的相同模式
similar(word)	顯示word的相似詞
collocations(num = 20, window_size = 2)	顯示最常見的二詞搭配
count(word)	Word出現的詞數
dispersion_plot(words)	繪製words中文檔中出現的位置圖
vocab() or FreqDist()	返回文章單字出現頻率的字典

Tokenize

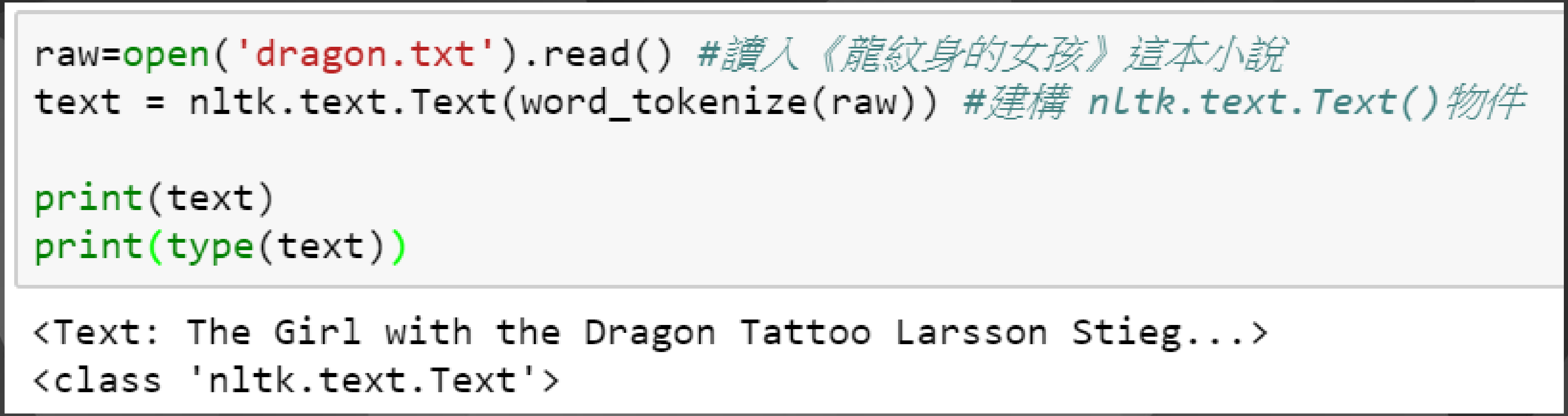
sent_tokenize() ：拆成個別句子
word_tokenize()：拆成個別單字



```
In [59]: from nltk.tokenize import word_tokenize, sent_tokenize
         #nltk.download('punkt')
         text = 'Hello, World. This is NLTK.'
         print(sent_tokenize(text))
         print(word_tokenize(text))

         ['Hello, World.', 'This is NLTK.']
         ['Hello', ',', 'World', '.', 'This', 'is', 'NLTK', '.']
```

Text(word)

先建構出一個 Text 類別，之後有許多的函數可以使用！

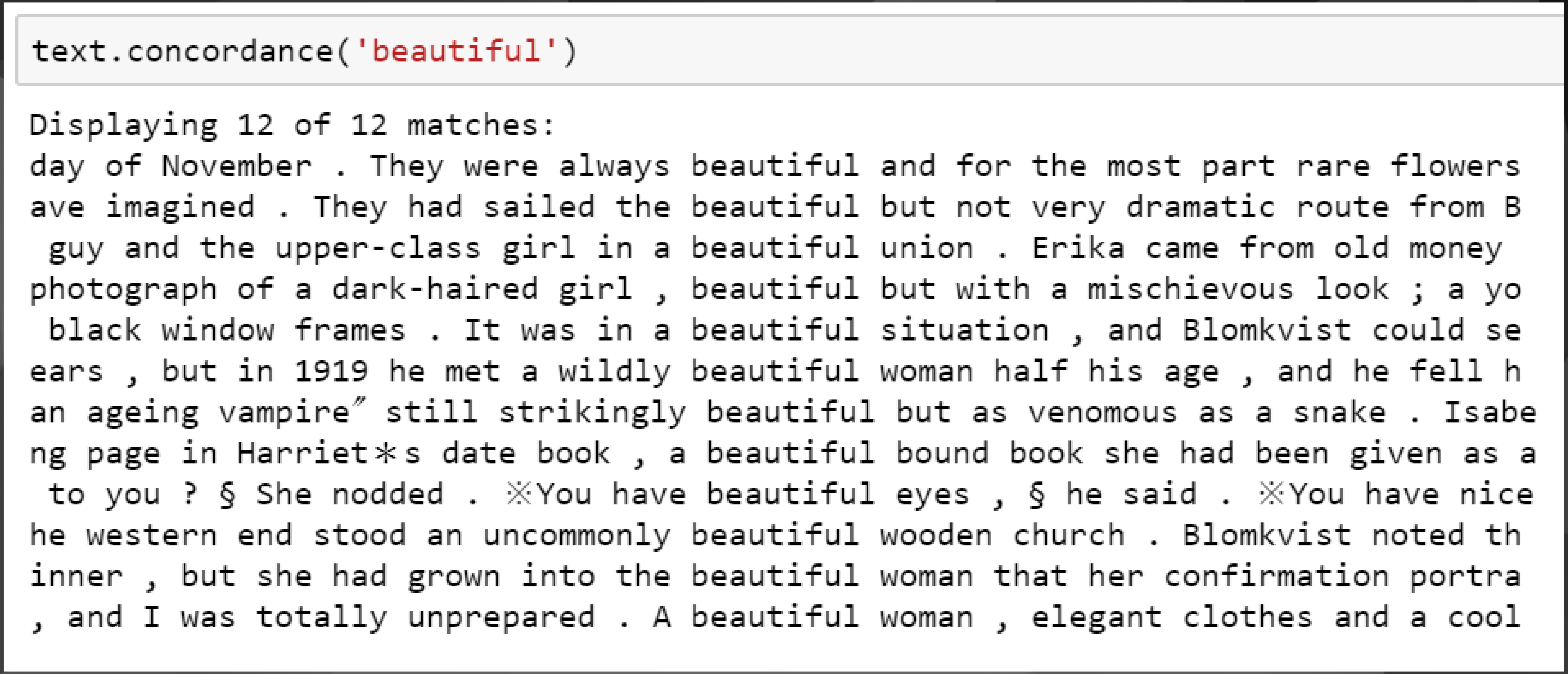


```
raw=open('dragon.txt').read() #讀入《龍紋身的女孩》這本小說
text = nltk.text.Text(word_tokenize(raw)) #建構 nltk.text.Text()物件

print(text)
print(type(text))
```

```
<Text: The Girl with the Dragon Tattoo Larsson Stieg...>
<class 'nltk.text.Text'>
```

concordance

我想知道某個單字所對應的上下文，就這麼做：

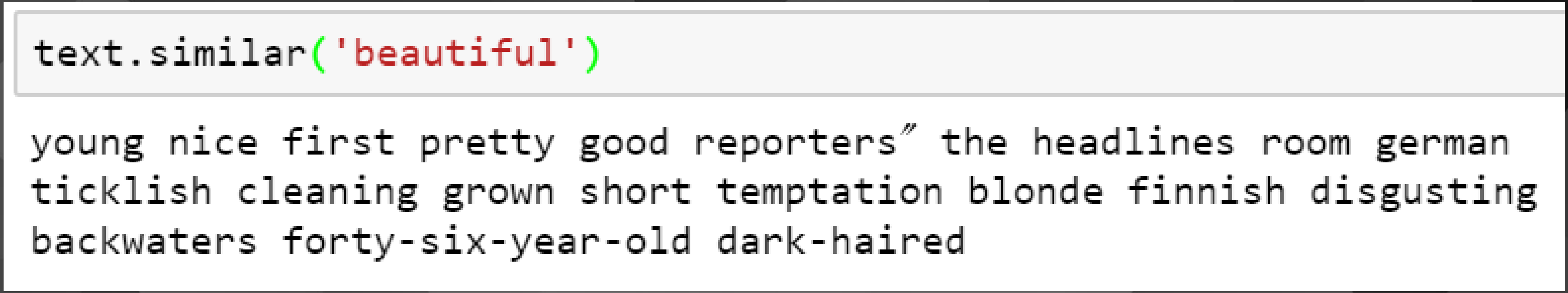


```
text.concordance('beautiful')
```

```
Displaying 12 of 12 matches:
day of November . They were always beautiful and for the most part rare flowers
ave imagined . They had sailed the beautiful but not very dramatic route from B
 guy and the upper-class girl in a beautiful union . Erika came from old money
photograph of a dark-haired girl , beautiful but with a mischievous look ; a yo
 black window frames . It was in a beautiful situation , and Blomkvist could se
ears , but in 1919 he met a wildly beautiful woman half his age , and he fell h
an ageing vampire〃 still strikingly beautiful but as venomous as a snake . Isabe
ng page in Harriet＊s date book , a beautiful bound book she had been given as a
 to you ? § She nodded . ※You have beautiful eyes , § he said . ※You have nice
he western end stood an uncommonly beautiful wooden church . Blomkvist noted th
inner , but she had grown into the beautiful woman that her confirmation portra
, and I was totally unprepared . A beautiful woman , elegant clothes and a cool
```

similar

我想知道某個單字的同義詞，怎麼辦？

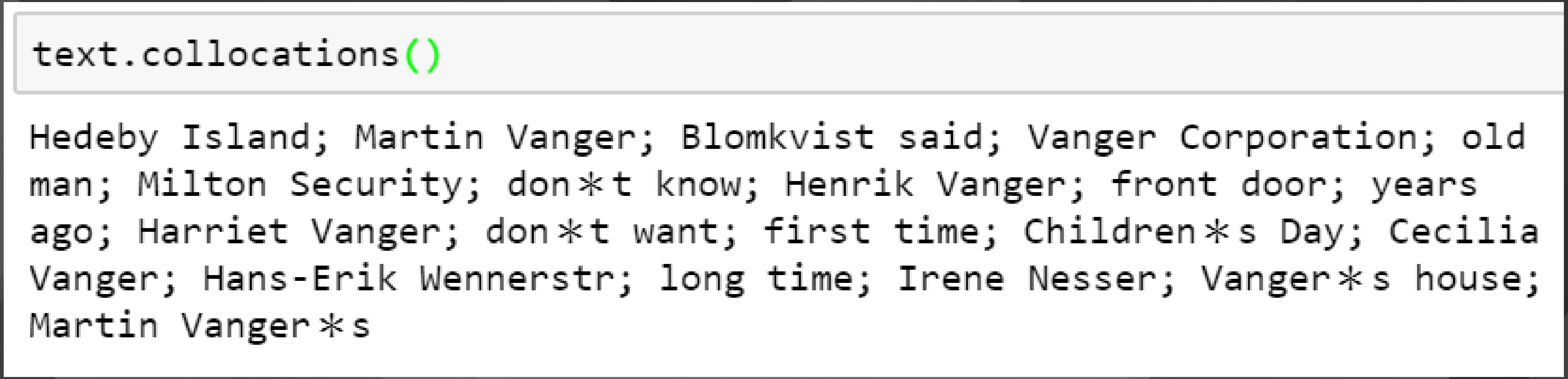


```
text.similar('beautiful')
```

```
young nice first pretty good reporters" the headlines room german
ticklish cleaning grown short temptation blonde finnish disgusting
backwaters forty-six-year-old dark-haired
```

collocations

最常見的兩字搭配

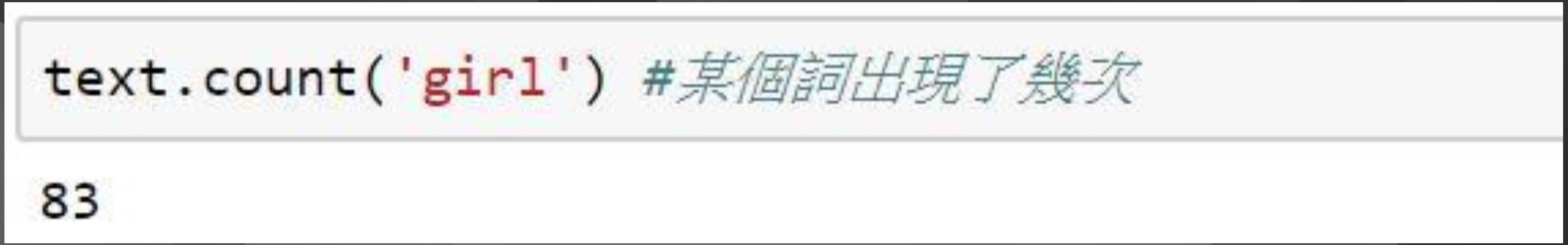


```
text.collocations()
```

```
Hedeby Island; Martin Vanger; Blomkvist said; Vanger Corporation; old
man; Milton Security; don＊t know; Henrik Vanger; front door; years
ago; Harriet Vanger; don＊t want; first time; Children＊s Day; Cecilia
Vanger; Hans-Erik Wennerstr; long time; Irene Nesser; Vanger＊s house;
Martin Vanger＊s
```

Count

單字出現的次數

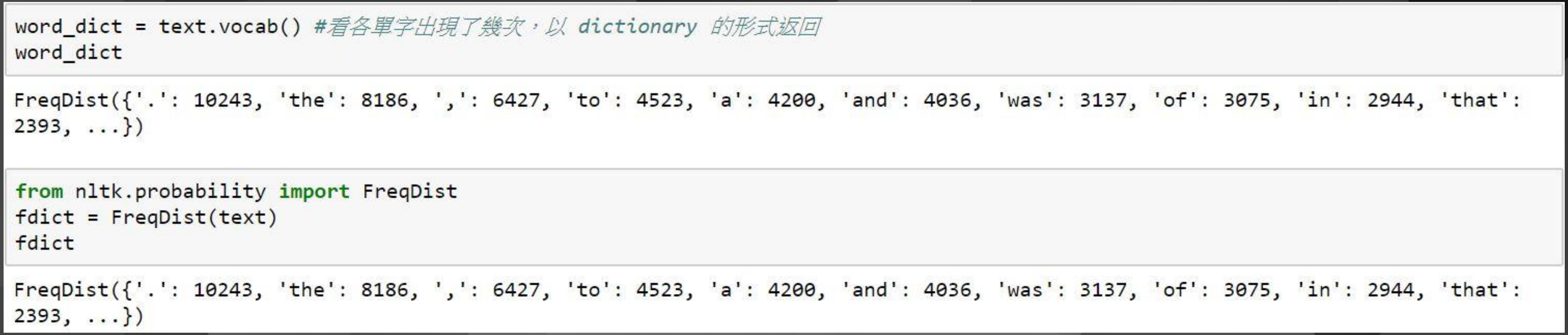


```
text.count('girl') #某個詞出現了幾次
```

```
83
```

Vocab() or FreqDist()

看每一個單字在文章中出現的次數



```
word_dict = text.vocab() #看各單字出現了幾次，以 dictionary 的形式返回
word_dict
```

```
FreqDist({'.': 10243, 'the': 8186, ',': 6427, 'to': 4523, 'a': 4200, 'and': 4036, 'was': 3137, 'of': 3075, 'in': 2944, 'that':
2393, ...})
```

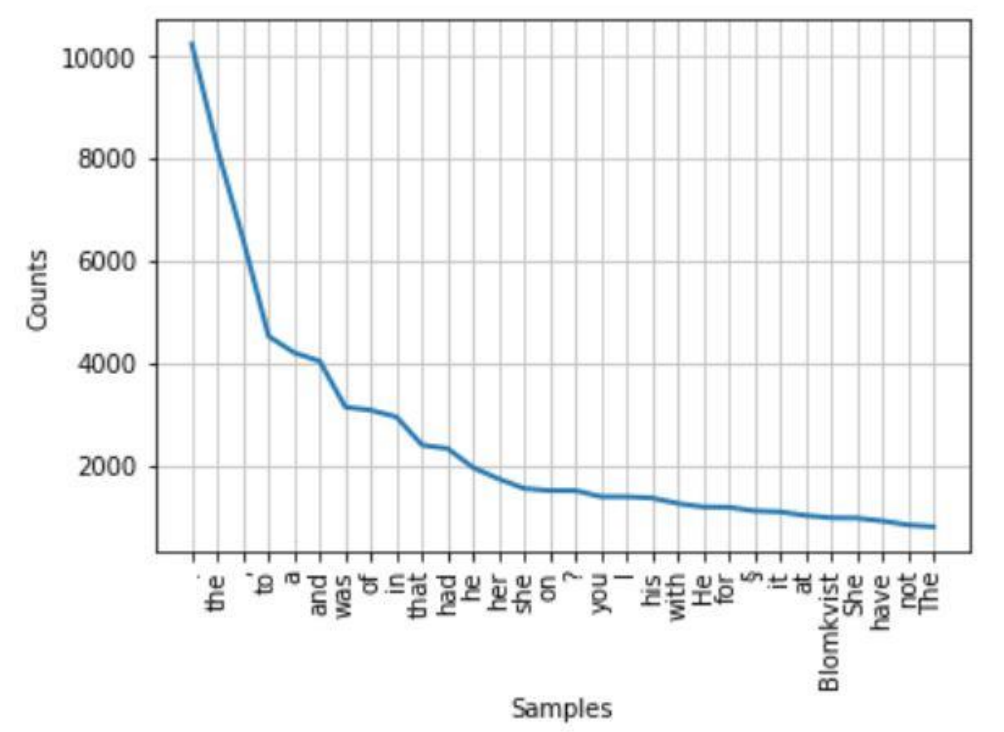
```
from nltk.probability import FreqDist
fdict = FreqDist(text)
fdict
```

```
FreqDist({'.': 10243, 'the': 8186, ',': 6427, 'to': 4523, 'a': 4200, 'and': 4036, 'was': 3137, 'of': 3075, 'in': 2944, 'that':
2393, ...})
```

Count

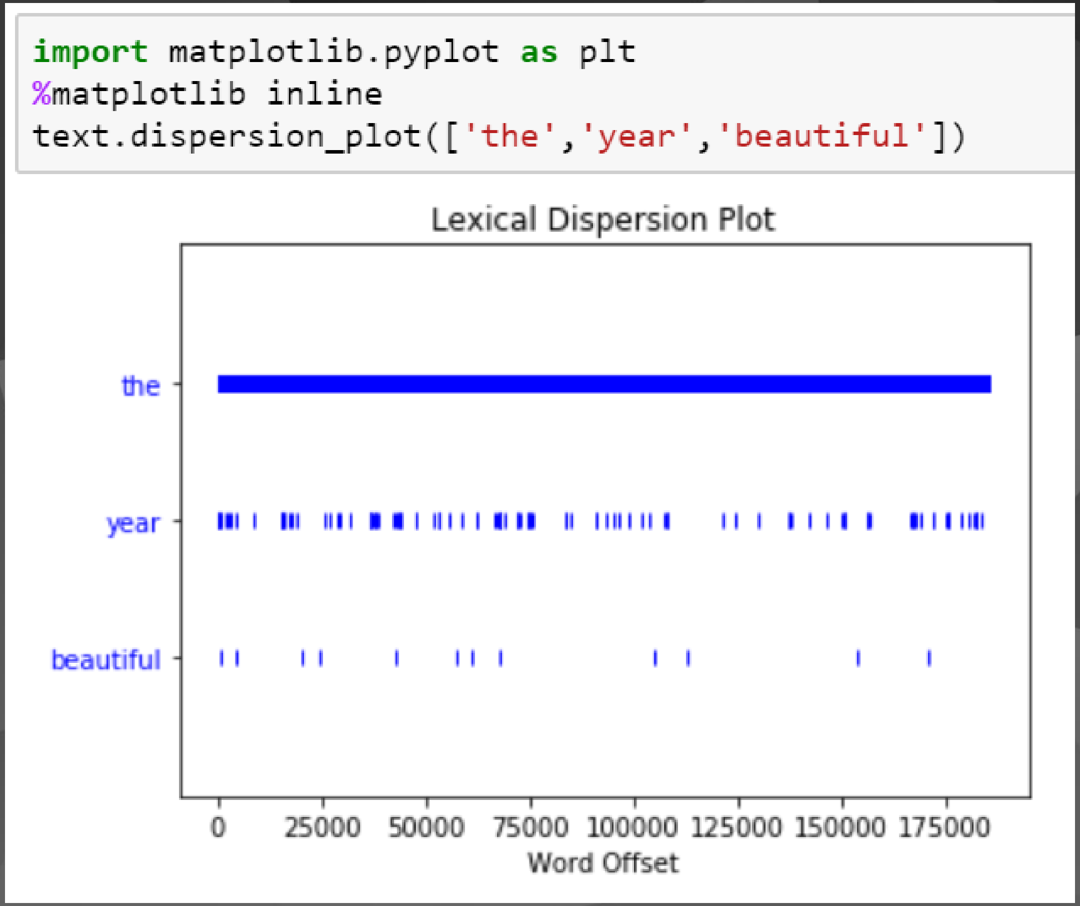
畫出出現次數前30多的單字的次數分布圖

```
fdict.plot(30)
```



dispersion_plot

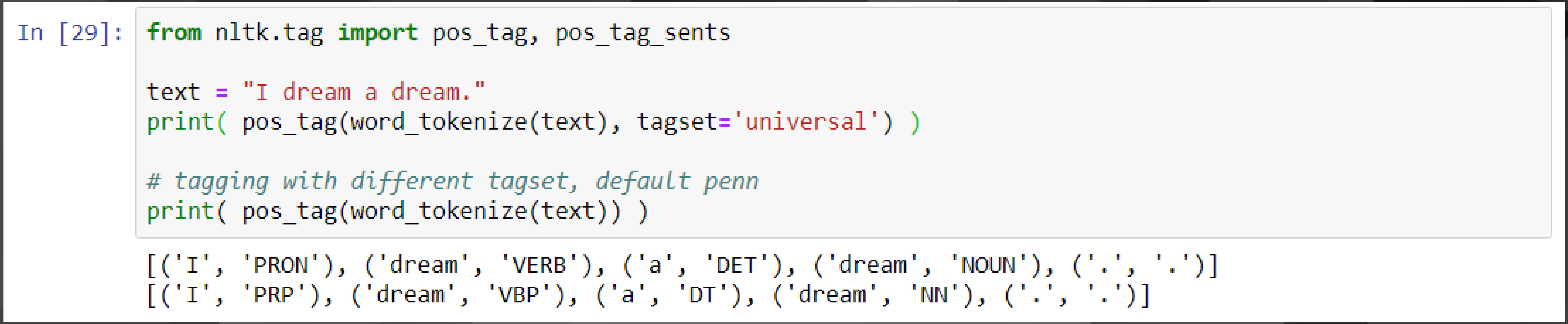
查看單字在文章中出現的地方



```
import matplotlib.pyplot as plt
%matplotlib inline
text.dispersion_plot(['the','year','beautiful'])
```

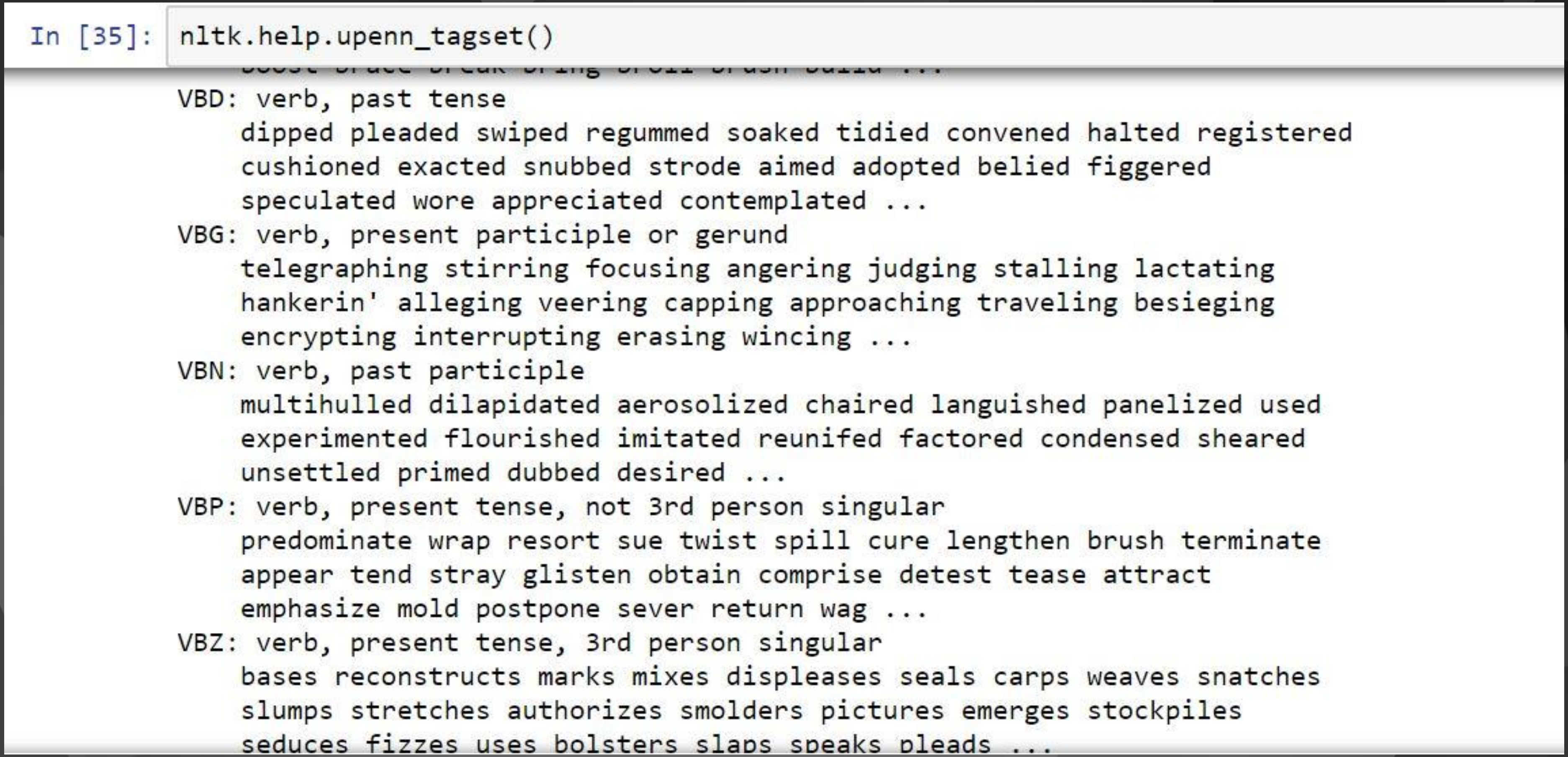
Tagging

pos_tag()：標註詞性



```
In [29]: from nltk.tag import pos_tag, pos_tag_sents

         text = "I dream a dream."
         print( pos_tag(word_tokenize(text), tagset='universal') )

         # tagging with different tagset, default penn
         print( pos_tag(word_tokenize(text)) )

         [('I', 'PRON'), ('dream', 'VERB'), ('a', 'DET'), ('dream', 'NOUN'), ('.', '.')]
         [('I', 'PRP'), ('dream', 'VBP'), ('a', 'DT'), ('dream', 'NN'), ('.', '.')]
```

詞性解釋

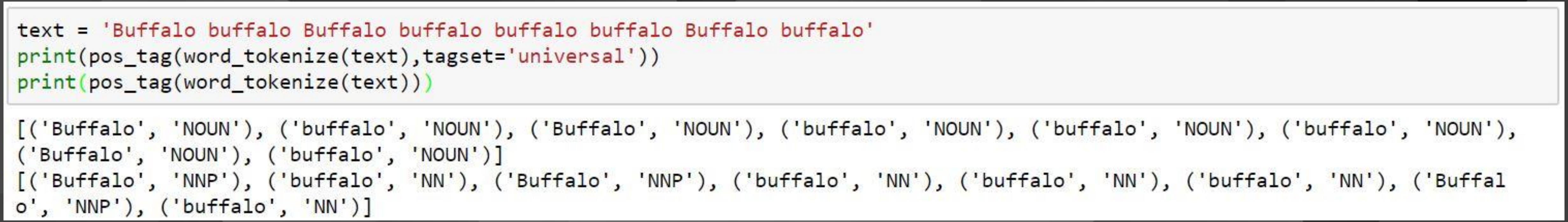


```
In [35]: nltk.help.upenn_tagset()
```

```
VBD: verb, past tense
    dipped pleaded swiped regummed soaked tidied convened halted registered
    cushioned exacted snubbed strode aimed adopted belied figgered
    speculated wore appreciated contemplated ...
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lactating
    hankerin' alleging veering capping approaching traveling besieging
    encrypting interrupting erasing wincing ...
VBN: verb, past participle
    multihulled dilapidated aerosolized chaired languished panelized used
    experimented flourished imitated reunifed factored condensed sheared
    unsettled primed dubbed desired ...
VBP: verb, present tense, not 3rd person singular
    predominate wrap resort sue twist spill cure lengthen brush terminate
    appear tend stray glisten obtain comprise detest tease attract
    emphasize mold postpone sever return wag ...
VBZ: verb, present tense, 3rd person singular
    bases reconstructs marks mixes displeases seals carps weaves snatches
    slumps stretches authorizes smolders pictures emerges stockpiles
    seduces fizzes uses bolsters slaps speaks pleads ...
```

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo

水牛城中某些被其他美洲野牛所恐嚇的美洲野牛，
又去恐嚇了另一些水牛城的美洲野牛。



```
text = 'Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo'
print(pos_tag(word_tokenize(text),tagset='universal'))
print(pos_tag(word_tokenize(text)))
```

```
[('Buffalo', 'NOUN'), ('buffalo', 'NOUN'), ('Buffalo', 'NOUN'), ('buffalo', 'NOUN'), ('buffalo', 'NOUN'), ('buffalo', 'NOUN'),
('Buffalo', 'NOUN'), ('buffalo', 'NOUN')]
[('Buffalo', 'NNP'), ('buffalo', 'NN'), ('Buffalo', 'NNP'), ('buffalo', 'NN'), ('buffalo', 'NN'), ('buffalo', 'NN'), ('Buffal
o', 'NNP'), ('buffalo', 'NN')]
```

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.

is a grammatically correct sentence used as an example of how homonyms and homophones can be used to create complicated constructs. The sentence is unpunctuated and uses three different readings of the word "buffalo." In order of their first use, these are:

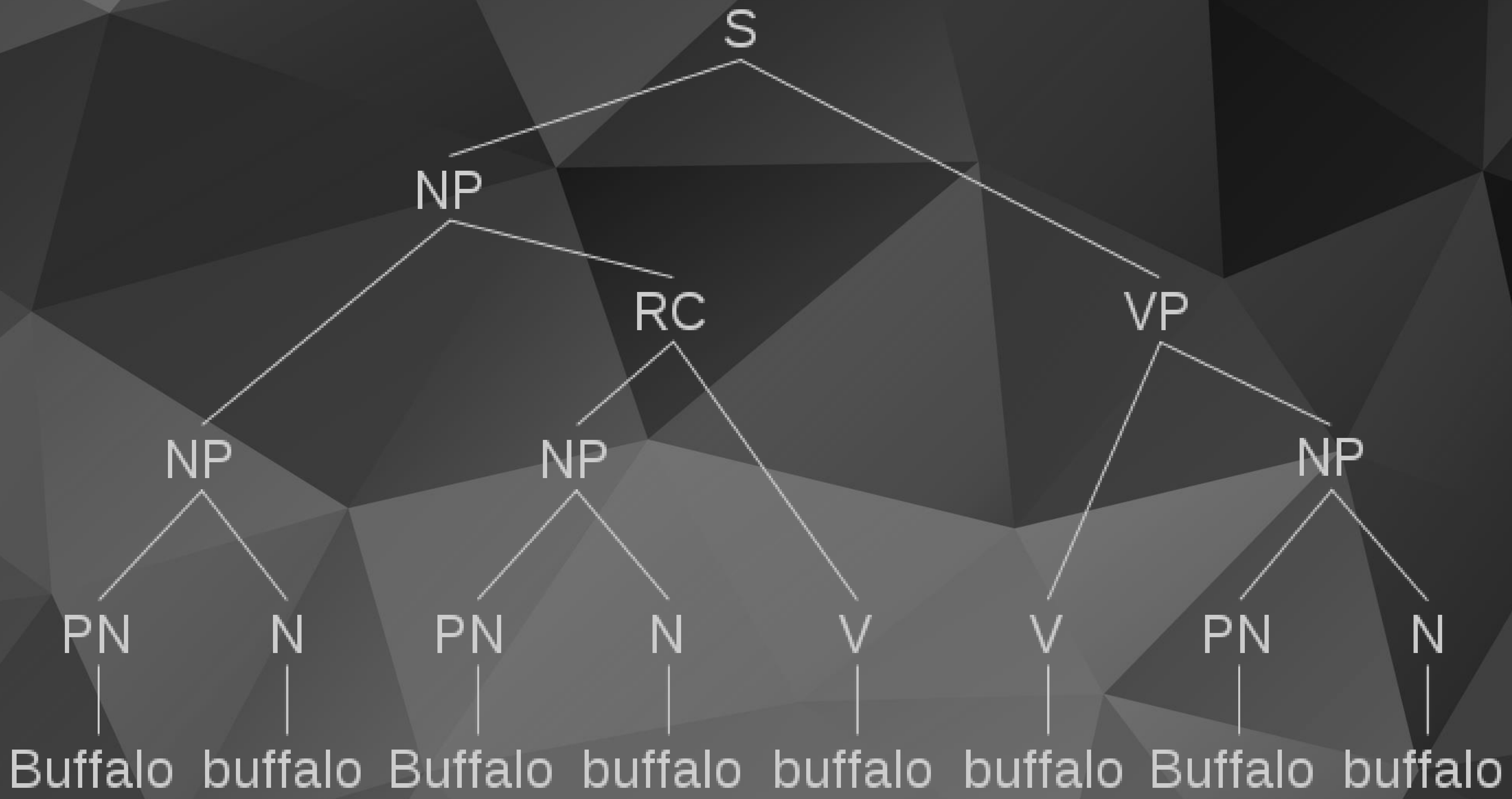
- The city of Buffalo, New York.
- The animal "buffalo," in the plural (equivalent to "buffaloes"), in order to avoid articles.
- The verb "buffalo," meaning to confuse, deceive or intimidate.

Homonym = a word form that has two or more distinct meanings

Homophone = a word which is pronounced the same as another word but differs in meaning

Substituting the synonym "bison" for "buffalo" (animal), "bully" for "buffalo" (verb) and leaving "Buffalo" to mean the city, yields:

Buffalo bison, whom other Buffalo bison bully, themselves bully Buffalo bison.


S
NP
RC
VP
NP
NP
NP
PN
N
PN
N
V
V
PN
N
Buffalo
buffalo
Buffalo
buffalo
buffalo
buffalo
Buffalo
buffalo

A ship-shipping ship ships shipping-ships.

一艘載了船的船載了載了船的船。

Regular Expression

抓取字元

. 任何字元

| 或

次數

? 出現0或1次

* 出現0或更多次

+ 出現1或更多次

Example

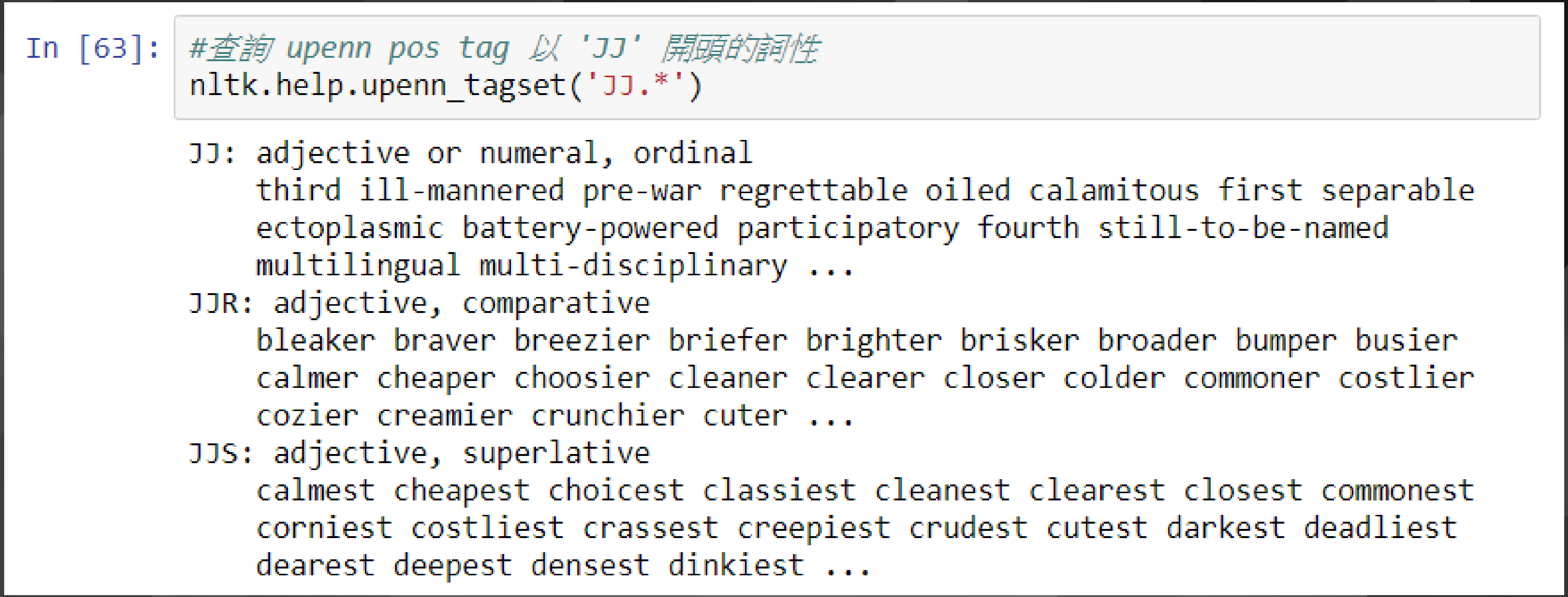
可以用a*去抓取 a或是aaaaaa

可以用ab?去抓取 a或是ab

可以用JJ.去抓取 JJR或是JJS

Regular Expression

Example : 查詢 upenn pos tag 以 'JJ' 開頭的詞性



```
In [63]: #查詢 upenn pos tag 以 'JJ' 開頭的詞性
         nltk.help.upenn_tagset('JJ.*')
```

```
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first separable
    ectoplasmic battery-powered participatory fourth still-to-be-named
    multilingual multi-disciplinary ...
JJR: adjective, comparative
    bleaker braver breezier briefer brighter brisker broader bumper busier
    calmer cheaper choosier cleaner clearer closer colder commoner costlier
    cozier creamier crunchier cuter ...
JJS: adjective, superlative
    calmest cheapest choicest classiest cleanest clearest closest commonest
    corniest costliest crassest creepiest crudest cutest darkest deadliest
    dearest deepest densest dinkiest ...
```

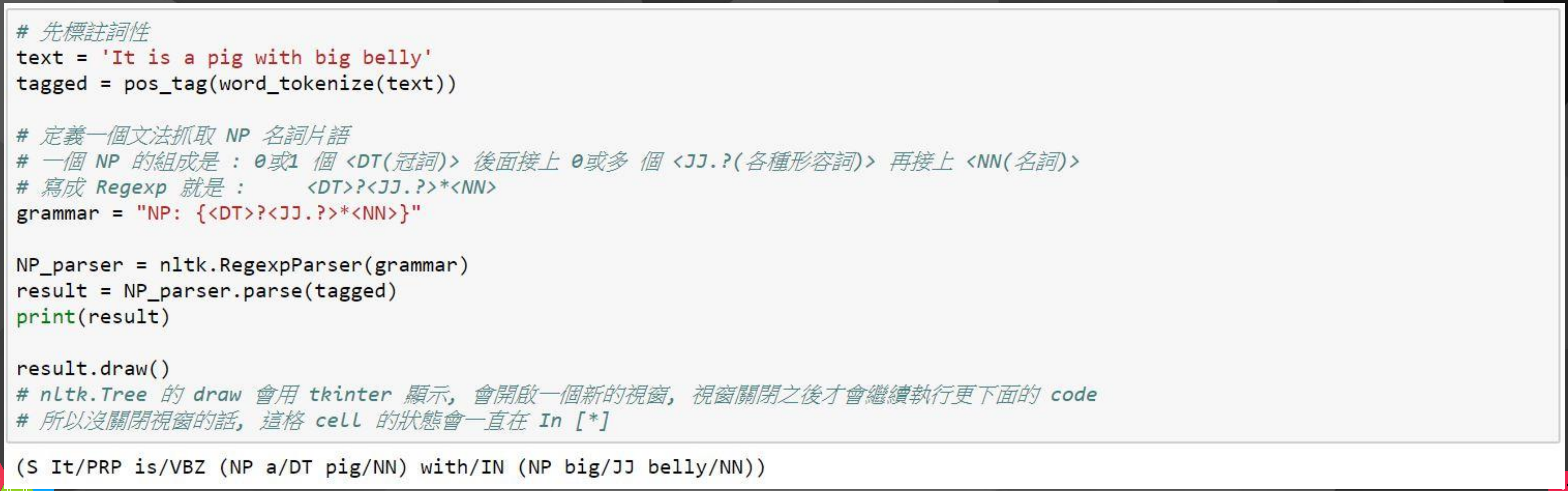
Chunk 方塊

Example

定義 一個 NP 的組成是：

0或1 個 <DT(冠詞)> 後面接上 0或多 個 <JJ.?(各種形容詞)> 再接上 <NN(普通名詞)>

寫成 Regexp 就是： <DT>?<JJ.?>*<NN>

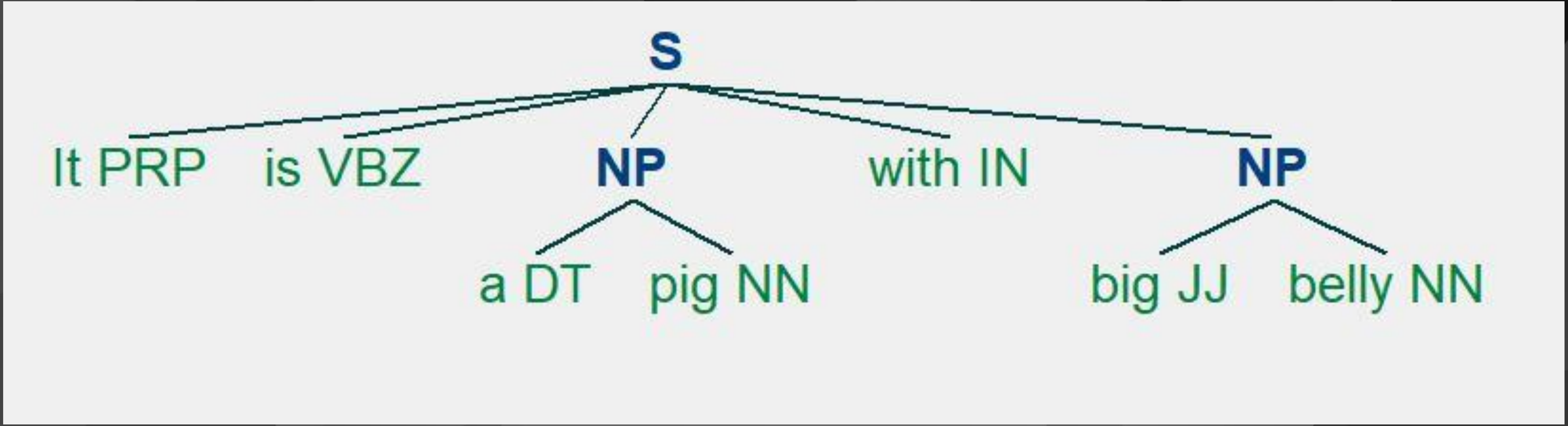


```
# 先標註詞性
text = 'It is a pig with big belly'
tagged = pos_tag(word_tokenize(text))

# 定義一個文法抓取 NP 名詞片語
# 一個 NP 的組成是 : 0或1 個 <DT(冠詞)> 後面接上 0或多 個 <JJ.?(各種形容詞)> 再接上 <NN(名詞)>
# 寫成 Regexp 就是 :     <DT>?<JJ.?>*<NN>
grammar = "NP: {<DT>?<JJ.?>*<NN>}"

NP_parser = nltk.RegexpParser(grammar)
result = NP_parser.parse(tagged)
print(result)

result.draw()
# nltk.Tree 的 draw 會用 tkinter 顯示, 會開啟一個新的視窗, 視窗關閉之後才會繼續執行更下面的 code
# 所以沒關閉視窗的話, 這格 cell 的狀態會一直在 In [*]
```

```
(S It/PRP is/VBZ (NP a/DT pig/NN) with/IN (NP big/JJ belly/NN))
```

Chunk 方塊

句構圖



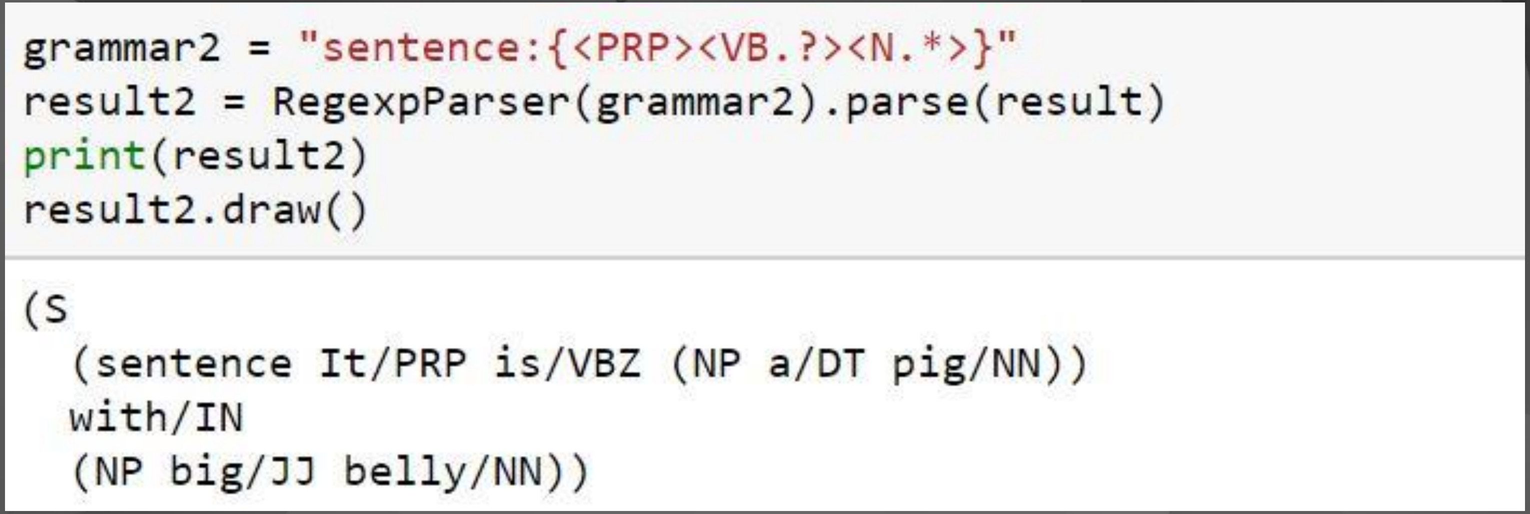
Chunk 方塊

Example

定義 一個 sentence 的組成是：

1 個 <DT(冠詞)> 後面接上 1 個 <VB.?(各種動詞)> 再接上 <N.*(各種名詞)>

寫成 Regexp 就是： <PRP><VB.?><N.*>

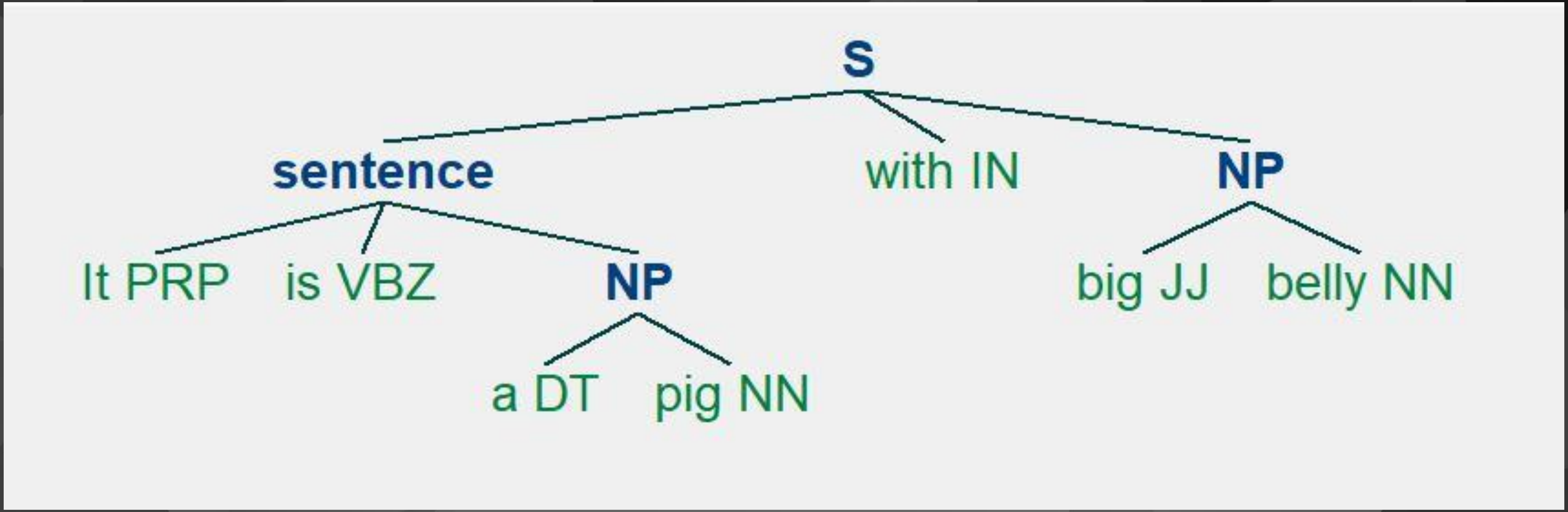


```
grammar2 = "sentence:{<PRP><VB.?><N.*>}"
result2 = RegexpParser(grammar2).parse(result)
print(result2)
result2.draw()
```

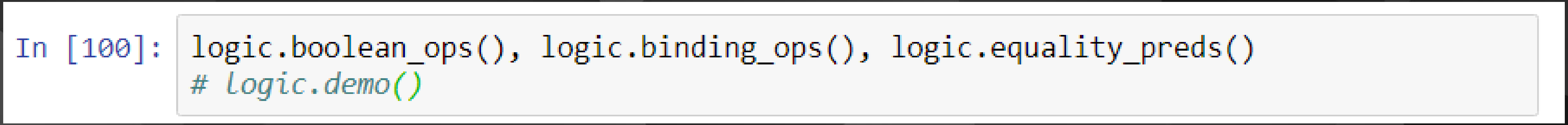
```
(S
  (sentence It/PRP is/VBZ (NP a/DT pig/NN))
  with/IN
  (NP big/JJ belly/NN))
```

Chunk 方塊

句構圖



Logic & Prove



```
In [100]: logic.boolean_ops(), logic.binding_ops(), logic.equality_preds()
          # logic.demo()
```

意思	標示
negative	-
conjuction	&
disjunction	|
implication	->
equivalence	<->

意思	標示
existential	exists
universal	all
lambda	\
equality	=
inequality	!=

Logic & Prove

Propositions

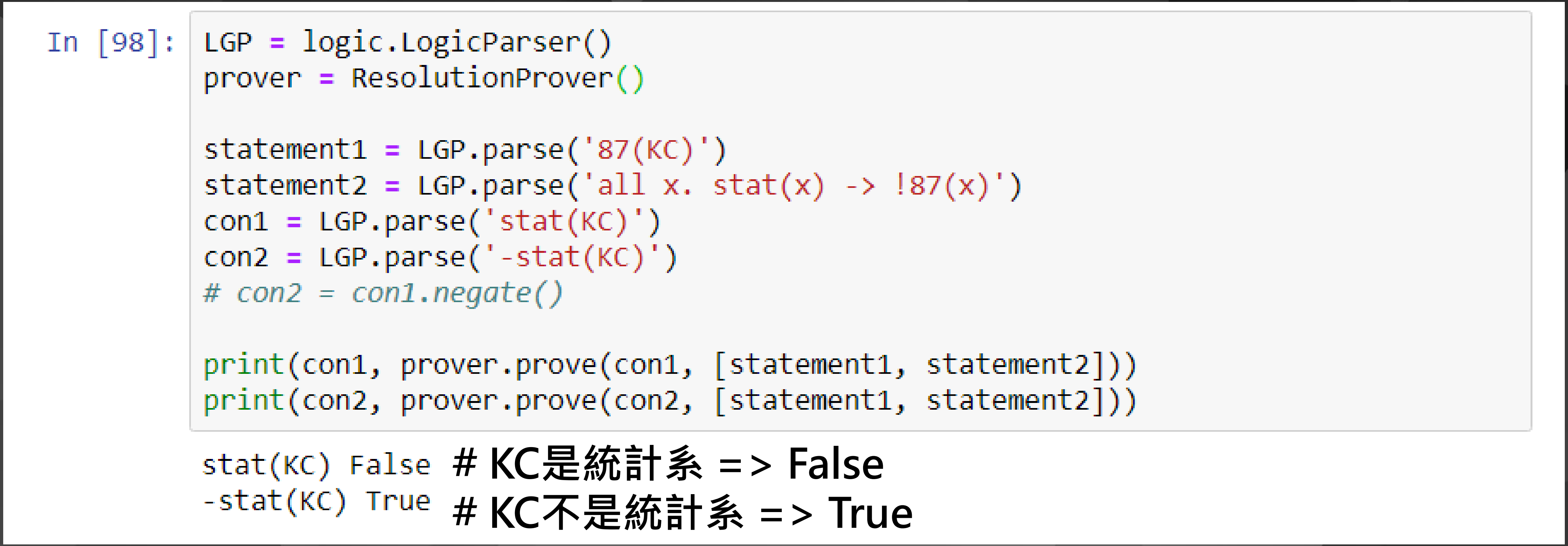
Example :

statement 1 : KC is 87
statement 2 : ∀x, x is stat → ¬(x is 87) # "¬"表 否定

Find whether following conclusion is True

conclusion 1 : KC is stat
conclusion 2 : ¬(KC is stat)

Logic & Prove



```
In [98]: LGP = logic.LogicParser()
         prover = ResolutionProver()

         statement1 = LGP.parse('87(KC)')
         statement2 = LGP.parse('all x. stat(x) -> !87(x)')
         con1 = LGP.parse('stat(KC)')
         con2 = LGP.parse('-stat(KC)')
         # con2 = con1.negate()

         print(con1, prover.prove(con1, [statement1, statement2]))
         print(con2, prover.prove(con2, [statement1, statement2]))

         stat(KC) False
         -stat(KC) True
```

# KC是統計系 => False
# KC不是統計系 => True

THANK YOU