Book 1 (Language Processing and Python)

SIDE-39-GAB

Bastomy - 1301178418 - Text Mining

1. Language Processing and Python

1.Computing with Language: Texts and Words

1.1 Getting Started with Python

Perhitungan sederhana

In [1]:
```
(1 + 5 * 2) - 3
```

Out[1]: 8

In [2]:
```
(2**3) + 6
```

Out[2]: 14

perhitungan pada pemograman python sama dengan pemograman lainnya, untuk pangkat kita bisa menggunakan tanda ** sebagai perhitungan pangkat

In [3]:
```
import math

print('akar dari 36 = ', math.sqrt(36))
print('cos dari 45 = ', math.cos(45))
print('faktorial dari 4 = ',math.factorial(4))
```

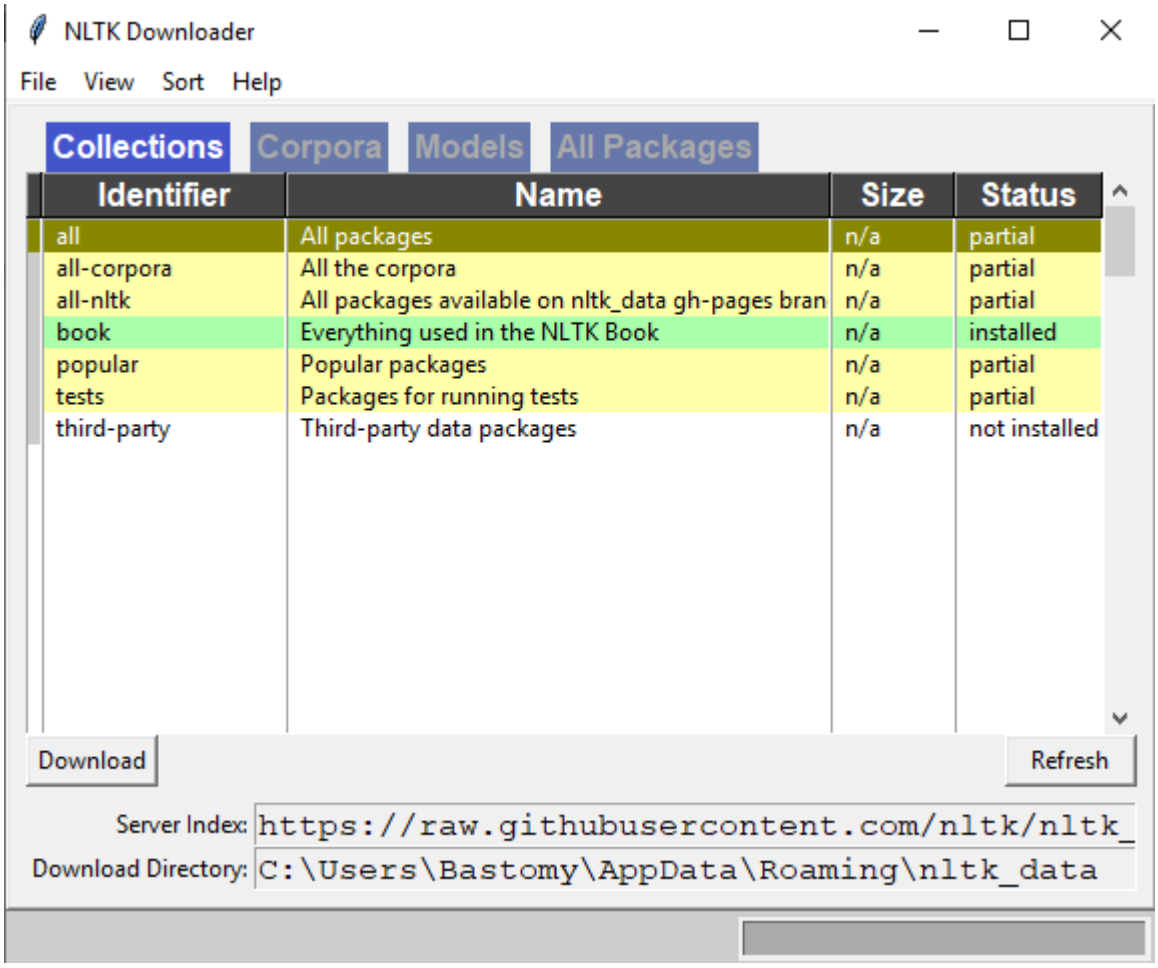
```
akar dari 36 =  6.0
cos dari 45 =  0.5253219888177297
faktorial dari 4 =  24
```

untuk mempermudah proses perhitungan kita bisa menggunakan library bernama math seperti beberapa contoh di atas

1.2 Getting Started with NLTK

NLTK adalah sebuah library yang biasanya banyak digunakan untuk pengerjaan tugas NLP dimana pada NLTK ini terdapat banyak fitur yang sangat membantu seperti untuk preprocessing dan lainnya. sebelum menggunakan NLTK kita diharuskan mendownload terlebih dahulu seperti code dibawah ini

In [4]:
```
import nltk
# nltk.download()
```



nltk.download() hanya di jalannkan sekali untuk mendownload seperti gambar diatas, jika sudah menginstallnya kita bisa skip dengan cara mengkomen code **nltk.download()**

In [5]:
```
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
```

sintak di atas berfungsi untuk mengimport NLTK dan beberapa buku yg tersedia, dapat dilihat terdapat 9 buku yang di sediakan nltk, untuk memanggil text tersebut dapat digunakan sintax

text+no buku seperti contoh dibawah

In [6]:
```
text1, text2, text3, text4, text5, text6, text7, text8, text9
```

Out[6]:
```
(<Text: Moby Dick by Herman Melville 1851>,
 <Text: Sense and Sensibility by Jane Austen 1811>,
 <Text: The Book of Genesis>,
 <Text: Inaugural Address Corpus>,
 <Text: Chat Corpus>,
 <Text: Monty Python and the Holy Grail>,
 <Text: Wall Street Journal>,
 <Text: Personals Corpus>,
 <Text: The Man Who Was Thursday by G . K . Chesterton 1908>)
```

1.3 Searching Text

Untuk melihat isi text kita dapat menggunakan tokens seperti contoh dibawah ini, dan menggunakan join untuk merubah bentuk array menjadi string. dikarenakan isi buku terlalu besar maka kita akan mengpreview 500 tokens pertama saja

In [7]:
```
a = text1.tokens
data1 = " ".join(a[:500])
print(data1)
```

```
[ Moby Dick by Herman Melville 1851 ] ETYMOLOGY . ( Supplied by a Late Consumpt
ive Usher to a Grammar School ) The pale Usher -- threadbare in coat , heart ,
body , and brain ; I see him now . He was ever dusting his old lexicons and gra
mmars , with a queer handkerchief , mockingly embellished with all the gay flag
s of all the known nations of the world . He loved to dust his old grammars ; i
t somehow mildly reminded him of his mortality . " While you take in hand to sc
hool others , and to teach them by what name a whale - fish is to be called in
our tongue leaving out , through ignorance , the letter H , which almost alone
maketh the signification of the word , you deliver that which is not true ." --
HACKLUYT " WHALE . ... Sw . and Dan . HVAL . This animal is named from roundnes
s or rolling ; for in Dan . HVALT is arched or vaulted ." -- WEBSTER ' S DICTIO
NARY " WHALE . ... It is more immediately from the Dut . and Ger . WALLEN ; A .
S . WALW - IAN , to roll , to wallow ." -- RICHARDSON ' S DICTIONARY KETOS , GR
EEK . CETUS , LATIN . WHOEL , ANGLO - SAXON . HVALT , DANISH . WAL , DUTCH . HW
AL , SWEDISH . WHALE , ICELANDIC . WHALE , ENGLISH . BALEINE , FRENCH . BALLENA
, SPANISH . PEKEE - NUEE - NUEE , FEGEE . PEKEE - NUEE - NUEE , ERROMANGOAN . E
XTRACTS ( Supplied by a Sub - Sub - Librarian ). It will be seen that this mere
painstaking burrower and grub - worm of a poor devil of a Sub - Sub appears to
have gone through the long Vaticans and street - stalls of the earth , picking
up whatever random allusions to whales he could anyways find in any book whatso
ever , sacred or profane . Therefore you must not , in every case at least , ta
ke the higgledy - piggledy whale statements , however authentic , in these extr
acts , for veritable gospel cetology . Far from it . As touching the ancient au
thors generally , as well as the poets here appearing , these extracts are sole
ly valuable or entertaining , as affording a glancing bird ' s eye view of what
has been promiscuously said , thought , fancied , and sung of Leviathan , by ma
ny nations and generations , including our own . So fare thee well , poor devil
of a Sub - Sub , whose commentator I am . Thou belongest to that hopeless , sal
low tribe which no wine of this world will ever warm ; and for whom even Pale S
herry would be too rosy - strong ; but with whom one sometimes loves to sit , a
nd feel poor
```

contoh pencarian kata **Lucky**

In [8]:
```
text1.concordance("lucky")
```

```
Displaying 8 of 8 matches:
etter than nothing ; and if we had a lucky voyage , might pretty nearly pay for
 a Cape - Cod - man . A happy - go - lucky ; neither craven nor valiant ; takin
fore the wind . They are accounted a lucky omen . If you yourself can withstand
l heights ; here and there from some lucky point of view you will catch passing
olently making for one centre . This lucky salvation was cheaply purchased by t
h Sea . The voyage was a skilful and lucky one ; and returning to her berth wit
eat skull echoed -- and seizing that lucky chance , I quickly concluded my own
I ' ll be ready for them presently . Lucky now ( SNEEZES ) there ' s no knee -
```

contoh pencarian similarity **nation** pada text1

In [9]:
```
text1.similar("nation")
```

```
ship whaleman lake school world whale roll devil view dragon sea thing
land king vessel patient men grove man chance
```

contoh pencarian konteks yang paling sering muncul **sadness** pada text1

In [10]:
```
text1.common_contexts(["sadness"])
```

```
unpleasant_give helpless_than
```

untuk melihat kepadatan suatu kata tertentu kita dapat menggunakan fungsi **dispersion_plot** dengan fungsi ini akan ditampilkan suatu plot kemunculan kata tersebut, dan kita dapat menggunakan plot tersebut lebih dari satu kata yaitu dengan sebuah array kata-kata yang akan di plot, seperti contoh dibawah ini

In [11]:
```
text1.dispersion_plot(["nation","GREEK","devil","swim","nature"])
```

```
<Figure size 640x480 with 1 Axes>
```

1.4 Counting Vocabulary

contoh menghitung panjang text menggunakan fungsi **len**

In [12]:
```
print('panjang text 1 ',len(text1))
print('panjang text 2 ',len(text2))
print('panjang text 3 ',len(text3))
print('panjang text 4 ',len(text4))
print('panjang text 5 ',len(text5))
print('panjang text 6 ',len(text6))
print('panjang text 7 ',len(text7))
print('panjang text 8 ',len(text8))
print('panjang text 9 ',len(text9))
```

```
panjang text 1  260819
panjang text 2  141576
panjang text 3  44764
panjang text 4  145735
panjang text 5  45010
panjang text 6  16967
panjang text 7  100676
panjang text 8  4867
panjang text 9  69213
```

Mengurutkan 20 kata pertama text3

In [13]:
```
sorted(set(text3[:20]))
```

Out[13]:
```
[',',
 '.',
 'And',
 'God',
 'In',
 'and',
 'beginning',
 'created',
 'earth',
 'form',
 'heaven',
 'the',
 'void',
 'was',
 'without']
```

melakukan pembagian antara panjang **set (kata unik)** dalam text4 dengan **panjang** text4

In [14]:
```
len(set(text4)) / len(text4)
```

Out[14]:
```
0.06692970116993173
```

In [15]:
```
def lexical_diversity(text):
    return len(set(text)) / len(text)

def percentage(count, total):
    return 100 * count / total
```

kita bisa membuat fungsi agar dapat di panggil berulang-ulang seperti pada contoh fungsi **lexical diversity**

In [16]:
```
print("lexical diversity text1 = ",lexical_diversity(text1))
print("lexical diversity text2 = ",lexical_diversity(text2))
print("lexical diversity text3 = ",lexical_diversity(text3))
print("lexical diversity text4 = ",lexical_diversity(text4))
print("lexical diversity text5 = ",lexical_diversity(text5))
print("lexical diversity text6 = ",lexical_diversity(text6))
print("lexical diversity text7 = ",lexical_diversity(text7))
print("lexical diversity text8 = ",lexical_diversity(text8))
print("lexical diversity text9 = ",lexical_diversity(text9))
```

```
lexical diversity text1 =  0.07406285585022564
lexical diversity text2 =  0.04826383002768831
lexical diversity text3 =  0.06230453042623537
lexical diversity text4 =  0.06692970116993173
lexical diversity text5 =  0.13477005109975562
lexical diversity text6 =  0.1276595744680851
lexical diversity text7 =  0.12324685128531129
lexical diversity text8 =  0.22765564002465585
lexical diversity text9 =  0.0983485761345412
```

dengan menggunakan fungsi percentage di atas kita dapat menghitung persentasi jumlah kemunculan kata dalam sebuah text seperti menghitung kemunculan kata **nations** dalam text1 yaitu sebesar **0.0046** persen dan **devil** sebesar **0.019** persen

In [17]:
```
print("nations = ",percentage(text1.count('nations'), len(text1)))
print("devil = ",percentage(text1.count('devil'), len(text1)))
```

```
nations =  0.0046008918061951004
devil =  0.01955379017632918
```

2. A Closer Look at Python: Texts as Lists of Words

2.1 List

List merupakan sebuah array pada python dengan list kita dapat membuat sebuah array berisi banyak data, dengan pemisah data menggunakan koma (,) berikut contoh deklarasi sebuah list

In [18]:
```
sent1 = ['Call', 'me', 'Ishmael', '.']
sent2 = ['The', 'family', 'of', 'Dashwood', 'had', 'long','been', 'settled', 'in
sent3 = ['In', 'the', 'beginning', 'God', 'created', 'the','heaven', 'and', 'the
```

berikut merupakan contoh fungsi yang dapat digunakan pada sebuah list

In [19]:
```
print('panjang sent1 : ', len(sent1))
print('panjang sent2 : ', len(sent2))
print('panjang sent3 : ', len(sent3))
```

```
panjang sent1 :  4
panjang sent2 :  11
panjang sent3 :  11
```

In [20]:
```
sent1.append("ahmed")
print('menambah ahmed pada sent1 : ', sent1)
```

```
menambah ahmed pada sent1 :  ['Call', 'me', 'Ishmael', '.', 'ahmed']
```

In [21]:
```
print("ahmed berada pada index ke ",sent1.index("ahmed")," pada sent1")
```

```
ahmed berada pada index ke  4  pada sent1
```

In [22]:
```
sent1.sort()
print("mengurutkan list sent1 menjadi ",sent1)
```

```
mengurutkan list sent1 menjadi  ['.', 'Call', 'Ishmael', 'ahmed', 'me']
```

In [23]:
```
sent1.append("me")
print('sent1 ',sent1)
print("jumlah me pada sent1 ",sent1.count('me'))
```

```
sent1  ['.', 'Call', 'Ishmael', 'ahmed', 'me', 'me']
jumlah me pada sent1  2
```

In [24]:
```
sent1.clear()
print("mengosongkan sent1")
```

```
mengosongkan sent1
```

In [25]:
```
sent1
```

Out[25]:
```
[]
```

Daftar list sent2 dan sent 3

In [26]:
```
print('list sent2 = ', sent2)
print('list sent3 = ', sent3)
```

```
list sent2 =  ['The', 'family', 'of', 'Dashwood', 'had', 'long', 'been', 'settl
ed', 'in', 'Sussex', '.']
list sent3 =  ['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven', 'an
d', 'the', 'earth', '.']
```

penggabungan list sent2 dan sent3 ke variabel list4

In [27]:
```
list4 = sent2 + sent3
print('list 4 =', list4)
```

```
list 4 = ['The', 'family', 'of', 'Dashwood', 'had', 'long', 'been', 'settled',
'in', 'Sussex', '.', 'In', 'the', 'beginning', 'God', 'created', 'the', 'heave
n', 'and', 'the', 'earth', '.']
```

2.2 Indexing List

In [28]:
```
print('index 100 pada text1 adalah ',text1[100])
```

```
index 100 pada text1 adalah  and
```

In [29]:
```
print('nation berada pada index ', text1.index('nation'))
```

```
nation berada pada index  37631
```

In [30]:
```
print('menampilkan data text1 dari index 100 sampai dengan 110')
print(text1[100:110])
```

```
menampilkan data text1 dari index 100 sampai dengan 110
['and', 'to', 'teach', 'them', 'by', 'what', 'name', 'a', 'whale', '-']
```

menampilkan data dari index 141525 sampai akhir pada text2

In [31]:
```
print(text2[141525:])
```

```
['among', 'the', 'merits', 'and', 'the', 'happiness', 'of', 'Elinor', 'and', 'M
arianne', ',', 'let', 'it', 'not', 'be', 'ranked', 'as', 'the', 'least', 'consi
derable', ',', 'that', 'though', 'sisters', ',', 'and', 'living', 'almost', 'wi
thin', 'sight', 'of', 'each', 'other', ',', 'they', 'could', 'live', 'without',
'disagreement', 'between', 'themselves', ',', 'or', 'producing', 'coolness', 'b
etween', 'their', 'husbands', '.', 'THE', 'END']
```

In [32]:
```
sent = ['word1', 'word2', 'word3', 'word4', 'word5','word6', 'word7', 'word8', 'w
print('index ke 0 pada sent adalah ', sent[0])
```

```
index ke 0 pada sent adalah  word1
```

In [33]:
```
sent[5:8]
```

Out[33]:
```
['word6', 'word7', 'word8']
```

In [34]:
```
sent[5]
```

Out[34]:
```
'word6'
```

In [35]:
```
print('menampilkan 3 index pertama dari sent ', sent[:3])
```

```
menampilkan 3 index pertama dari sent  ['word1', 'word2', 'word3']
```

mengubah data sent pada index k 0 dengan "first" & index k 9 dengan "last"

In [36]:
```
sent[0]
```

Out[36]:
```
'word1'
```

In [37]:
```
sent[0]='first'
sent[9]="last"
print('index 0 ',sent[0])
print('index 9 ',sent[9])
```

```
index 0  first
index 9  last
```

In [38]:
```
print('panjang data sent = ',len(sent))
```

```
panjang data sent =  10
```

merubah data pada index 1 sampai 9 dengan 'Second' 'Third'

In [39]:
```
sent[1:9] = ['Second', 'Third']
```

In [40]:
```
sent
```

Out[40]:
```
['first', 'Second', 'Third', 'last']
```

2.3 Variables

In [41]:
```
sent1 = ['Call', 'me', 'Ishmael', '.']
my_sent = ['Bravely', 'bold', 'Sir', 'Robin', ',', 'rode','forth', 'from', 'Camel
```

membuat variable noun_phrasa dari variable my_sent index ke 1 sampai 4

In [42]:
```
noun_phrase = my_sent[1:4]
```

In [43]:
```
noun_phrase
```

Out[43]:
```
['bold', 'Sir', 'Robin']
```

mengurutkan data noun_phrasa

In [44]:
```
wOrDs = sorted(noun_phrase)
print(wOrDs)
```

```
['Robin', 'Sir', 'bold']
```

kita tidak bisa menggunakan sintax dari python untuk nama vaiable seperti **not** , **if** dan **import**

In [45]:
```
vocab = set(text1)
vocab_size = len(vocab)
```

In [46]:
```
print('panjang dari vocab adalah ', vocab_size)
```

```
panjang dari vocab adalah  19317
```

2.4 Strings

In [47]:
```
name = 'Monty'
print('index k 0 pada name = ', name[0])
```

```
index k 0 pada name =  M
```

In [48]:
```
print('menampilkan 4 index pertama = ',name[:4])
```

```
menampilkan 4 index pertama =  Mont
```

In [49]:
```
print(name * 2)
print(name + '!')
```

```
MontyMonty
Monty!
```

menggabungkan array menjadi string

In [50]:
```
' '.join(['Monty', 'Python'])
```

Out[50]:
```
'Monty Python'
```

memecah string menjadi array

In [51]:
```
'Monty Python'.split()
```

Out[51]:
```
['Monty', 'Python']
```

3. Computing with Language: Simple Statistics

In [52]:
```
saying = ['After', 'all', 'is', 'said', 'and', 'done','more', 'is', 'said', 'thar
tokens = set(saying)
print(tokens)
tokens = sorted(tokens)
print(tokens)
tokens[-2:]
```

```
{'After', 'all', 'is', 'done', 'said', 'than', 'more', 'and'}
['After', 'all', 'and', 'done', 'is', 'more', 'said', 'than']
```

Out[52]:
```
['said', 'than']
```

3.1 Frequency Distributions

In [53]:
```
fdist1 = FreqDist(text1)
```

menghitung jumlah kemunculan setiap kata

In [54]:
```
print(fdist1)
```

```
<FreqDist with 19317 samples and 260819 outcomes>
```

In [55]:
```
fdist1
```

Out[55]:
```
FreqDist({',': 18713, 'the': 13721, '.': 6862, 'of': 6536, 'and': 6024, 'a': 45
69, 'to': 4542, ';': 4072, 'in': 3916, 'that': 2982, ...})
```

menampilkan 50 kata yang paling sering muncul

In [56]:
```
print(fdist1.most_common(50))
```

```
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024), ('a',
4569), ('to', 4542), (';', 4072), ('in', 3916), ('that', 2982), ("'", 2684),
('-', 2552), ('his', 2459), ('it', 2209), ('I', 2124), ('s', 1739), ('is', 169
5), ('he', 1661), ('with', 1659), ('was', 1632), ('as', 1620), ('"', 1478), ('a
ll', 1462), ('for', 1414), ('this', 1280), ('!', 1269), ('at', 1231), ('by', 11
37), ('but', 1113), ('not', 1103), ('--', 1070), ('him', 1058), ('from', 1052),
('be', 1030), ('on', 1005), ('so', 918), ('whale', 906), ('one', 889), ('you',
841), ('had', 767), ('have', 760), ('there', 715), ('But', 705), ('or', 697),
('were', 680), ('now', 646), ('which', 640), ('?', 637), ('me', 627), ('like',
624)]
```

mengecek kemunculan kata good

In [57]:
```
fdist1['good']
```

Out[57]:
```
192
```

3.2 Fine-grained Selection of Words

In [58]:
```
V = set(text1)
long_words = [w for w in V if len(w) > 15]
print(sorted(long_words))
```

```
['CIRCUMNAVIGATION', 'Physiognomically', 'apprehensiveness', 'cannibalisticall
y', 'characteristically', 'circumnavigating', 'circumnavigation', 'circumnaviga
tions', 'comprehensiveness', 'hermaphroditical', 'indiscriminately', 'indispens
ableness', 'irresistibleness', 'physiognomically', 'preternaturalness', 'respon
sibilities', 'simultaneousness', 'subterraneousness', 'supernaturalness', 'supe
rstitiousness', 'uncomfortableness', 'uncompromisedness', 'undiscriminating',
'uninterpenetratingly']
```

In [59]:
```
fdist5 = FreqDist(text5)
print(sorted(w for w in set(text5) if len(w) > 7 and fdist5[w] > 7))
```

```
['#14-19teens', '#talkcity_adults', '((((((((((', '........', 'Question', 'actu
ally', 'anything', 'computer', 'cute.-ass', 'everyone', 'football', 'innocent',
'listening', 'remember', 'seriously', 'something', 'together', 'tomorrow', 'wat
ching']
```

3.3 Collocations and Bigrams

In [60]:
```
list(bigrams(['more', 'is', 'said', 'than', 'done']))
```

Out[60]:
```
[('more', 'is'), ('is', 'said'), ('said', 'than'), ('than', 'done')]
```

In [61]:
```
text4.collocations()
```

```
United States; fellow citizens; four years; years ago; Federal
Government; General Government; American people; Vice President; Old
World; Almighty God; Fellow citizens; Chief Magistrate; Chief Justice;
God bless; every citizen; Indian tribes; public debt; one another;
foreign nations; political parties
```

In [62]:
```
text8.collocations()
```

```
would like; medium build; social drinker; quiet nights; non smoker;
long term; age open; Would like; easy going; financially secure; fun
times; similar interests; Age open; weekends away; poss rship; well
presented; never married; single mum; permanent relationship; slim
build
```

3.4 Counting Other Things

menghitung panjang string di setiap index (10 index pertama)

In [63]:
```
print([len(w) for w in text1[:10]])
```

```
[1, 4, 4, 2, 6, 8, 4, 1, 9, 1]
```

In [64]:
```
fdist = FreqDist(len(w) for w in text1)
print(fdist)
fdist
```

```
<FreqDist with 19 samples and 260819 outcomes>
```

Out[64]:
```
FreqDist({3: 50223, 1: 47933, 4: 42345, 2: 38513, 5: 26597, 6: 17111, 7: 14399,
8: 9966, 9: 6428, 10: 3528, ...})
```

In [65]:
```
print(fdist.most_common())
```

```
[(3, 50223), (1, 47933), (4, 42345), (2, 38513), (5, 26597), (6, 17111), (7, 14
399), (8, 9966), (9, 6428), (10, 3528), (11, 1873), (12, 1053), (13, 567), (14,
177), (15, 70), (16, 22), (17, 12), (18, 1), (20, 1)]
```

menampilkan index data yg sering muncul pada fdist

In [66]:
```
fdist.max()
```

Out[66]:
```
3
```

frekuensi kemungkinan kemunculan data

In [67]:
```
fdist.freq(3)
```

Out[67]:
```
0.19255882431878046
```

4. Back to Python: Making Decisions and Taking Control

4.1 Conditionals

Operator Relationship
< less than
<= less than or equal to
== equal to (note this is two "=" signs, not one)
!= not equal to
> greater than
>= greater than or equal to

In [68]:
```
print(sent7)
```

```
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'bo
ard', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']
```

menampilkan semua data sent7 dengan panjang kurang dari 4

In [69]:
```
[w for w in sent7 if len(w) < 4]
```

Out[69]:
```
[',', '61', 'old', ',', 'the', 'as', 'a', '29', '.']
```

menampilkan semua data sent7 dengan panjang kurang dari sama dengan 4

In [70]:
```
[w for w in sent7 if len(w) <= 4]
```

Out[70]:
```
[',', '61', 'old', ',', 'will', 'join', 'the', 'as', 'a', 'Nov.', '29', '.']
```

menampilkan semua data sent7 dengan panjang sama dengan 4

In [71]:
```
[w for w in sent7 if len(w) == 4]
```

Out[71]:
```
['will', 'join', 'Nov.']
```

menampilkan semua data sent7 dengan panjang tidak sama dengan 4

In [72]:
```
[w for w in sent7 if len(w) != 4]
```

Out[72]:
```
['Pierre',
 'Vinken',
 ',',
 '61',
 'years',
 'old',
 ',',
 'the',
 'board',
 'as',
 'a',
 'nonexecutive',
 'director',
 '29',
 '.']
```

menampilkan data sent7 yang diawali huruf P

In [73]:
```
[w for w in sent7 if w.startswith("P")]
```

Out[73]:
```
['Pierre']
```

In [74]:
```
sorted(w for w in set(text1) if w.endswith('ableness'))
```

Out[74]:
```
['comfortableness',
 'honourableness',
 'immutableness',
 'indispensableness',
 'indomitableness',
 'intolerableness',
 'palpableness',
 'reasonableness',
 'uncomfortableness']
```

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)

In [75]:
```
sorted(term for term in set(text4) if 'gnt' in term)
```

Out[75]:
```
['Sovereignty', 'sovereignties', 'sovereignty']
```

In [76]:
```
sorted(item for item in set(text6[:20]) if item.istitle())
```

Out[76]:
```
['Whoa']
```

In [77]:
```
sorted(item for item in set(sent7) if item.isdigit())
```

Out[77]:
```
['29', '61']
```

4.2 Operating on Every Element

menghitung panjang setiap element 10 pertama text1

In [78]:
```
[len(w) for w in text1[:10]]
```

Out[78]:
```
[1, 4, 4, 2, 6, 8, 4, 1, 9, 1]
```

merubah 10 element text1 menjadi huruf besar

In [79]:
```
[w.upper() for w in text1[:10]]
```

Out[79]:
```
['[',
 'MOBY',
 'DICK',
 'BY',
 'HERMAN',
 'MELVILLE',
 '1851',
 ']',
 'ETYMOLOGY',
 '.']
```

In [80]:
```
len(text1)
```

Out[80]:
```
260819
```

In [81]:
```
len(set(text1))
```

Out[81]:
```
19317
```

In [82]:
```
len(set(word.lower() for word in text1))
```

Out[82]:
```
17231
```

In [83]:
```
len(set(word.lower() for word in text1 if word.isalpha()))
```

Out[83]:
```
16948
```

4.3 Nested Code Blocks

mengecek apakah word kurang dari 5

In [84]:
```
word = 'cat'
if len(word) < 5:
    print('word length is less than 5')
```

```
word length is less than 5
```

In [85]:
```
if len(word) >= 5:
    print('word length is greater than or equal to 5')
```

In [86]:
```
for word in ['Call', 'me', 'Ishmael', '.']:
    print(word)
```

```
Call
me
Ishmael
.
```

4.4 Looping with Conditions

mengecek setiap element ygberakhiran l

In [87]:
```
sent1 = ['Call', 'me', 'Ishmael', '.']
for xyzzy in sent1:
    if xyzzy.endswith('l'):
        print(xyzzy)
```

```
Call
Ishmael
```

In [88]:
```
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
```

In [89]:
```
tricky = sorted(w for w in set(text2) if 'cie' in w or 'cei' in w)
for word in tricky:
    print(word, end=' ')
```

```
ancient ceiling conceit conceited conceive conscience conscientious conscientio
usly deceitful deceive deceived deceiving deficiencies deficiency deficient del
icacies excellencies fancied insufficiency insufficient legacies perceive perce
ived perceiving prescience prophecies receipt receive received receiving societ
y species sufficient sufficiently undeceive undeceiving
```