```
# https://www.nltk.org/book/ch08.html

import nltk


groucho_grammar = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'an' | 'my'
N -> 'elephant' | 'pajamas'
V -> 'shot'
P -> 'in'
""")


sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
parser = nltk.ChartParser(groucho_grammar)
for tree in parser.parse(sent):
  print(tree)
```

```
    (S
      (NP I)
      (VP
        (VP (V shot) (NP (Det an) (N elephant)))
        (PP (P in) (NP (Det my) (N pajamas)))))
    (S
      (NP I)
      (VP
        (V shot)
        (NP (Det an) (N elephant) (PP (P in) (NP (Det my) (N pajamas))))))
```

```
# https://github.com/jalajthanaki/NLPython


nltk.download('punkt')
```

```
    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
    True
```

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

```
    [nltk_data] Downloading package wordnet to /root/nltk_data...
    [nltk_data]   Package wordnet is already up-to-date!
    True
```

```
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

```
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer

def wordtokenization():
    content = """Stemming is funnier than a bummer says the sushi loving computer
    She really wants to buy cars. She told me angrily. It is better for you.
    Man is walking. We are meeting tomorrow. You really don't know..!"""
    print(word_tokenize(content))

def wordlemmatization():
    wordlemma = WordNetLemmatizer()
    print(wordlemma.lemmatize('cars'))
    print(wordlemma.lemmatize('walking',pos='v'))
    print(wordlemma.lemmatize('meeting',pos='n'))
    print(wordlemma.lemmatize('meeting',pos='v'))
    print(wordlemma.lemmatize('better',pos='a'))
    print(wordlemma.lemmatize('is',pos='v'))
    print(wordlemma.lemmatize('funnier',pos='a'))
    print(wordlemma.lemmatize('expected',pos='v'))
    print(wordlemma.lemmatize('fantasized',pos='v'))

if __name__ =="__main__":
    wordtokenization()
    print("\n")
    print("----------Word Lemmatization----------")
    wordlemmatization()
```

```
['Stemming', 'is', 'funnier', 'than', 'a', 'bummer', 'says', 'the', 'sushi', 'loving',


----------Word Lemmatization----------
car
walk
meeting
meet
good
be
funny
expect
fantasize
```

```
text = """Wikis are enabled by wiki
software, otherwise known as wiki engines. A wiki engine, being a
form of a content management system, differs from other web-based systems such as
content is created without any defined owner or leader, and wikis have little inhe
 to emerge according to the needs of the users.[1] Wiki engines usually allow cont
  markup language and sometimes edited with the help of a rich-text editor.[2] The
   in use, both standalone and part of other software, such as bug tracking syster
   whereas others are proprietary. Some permit control over different functions (
may permit access without enforcing access control. Other rules may be imposed to

data = text.split('.')
for i in data:
  print(i)
```

```
Wikis are enabled by wiki
software, otherwise known as wiki engines
 A wiki engine, being a
form of a content management system, differs from other web-based systems such as blog s
content is created without any defined owner or leader, and wikis have little inherent s
 to emerge according to the needs of the users
[1] Wiki engines usually allow content to be written using a simplified
  markup language and sometimes edited with the help of a rich-text editor
[2] There are dozens of different wiki engines
   in use, both standalone and part of other software, such as bug tracking systems
 Some wiki engines are open-source,
   whereas others are proprietary
 Some permit control over different functions (levels of access); for example, editing r
 Others
may permit access without enforcing access control
 Other rules may be imposed to organize content
```

```
import nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize

str = "I love to study Natuaral Languague Processing in Python"
print("--------Sent tokenize--------------")
print(sent_tokenize(str))
print("\n")
print("--------Word tokenize--------------")
print(word_tokenize(str))
```

```
--------Sent tokenize--------------
['I love to study Natuaral Languague Processing in Python']


--------Word tokenize--------------
['I', 'love', 'to', 'study', 'Natuaral', 'Languague', 'Processing', 'in', 'Python']
```

```
import nltk
from nltk.tokenize import RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps = True)
str = "I love to study Natuaral Languague Processing in Python. Wikis are enabled
tokens = tk.tokenize(str)
print(tokens)
```

```
['I', 'love', 'to', 'study', 'Natuaral', 'Languague', 'Processing', 'in', 'Python.', 'Wi
```

```
import nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize


nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
word_tokenize("can't")
```

```
['ca', "n't"]
```

```
from nltk.tokenize import TreebankWordTokenizer
tokenizer = TreebankWordTokenizer()
tokenizer.tokenize('Hello World.')
```

```
['Hello', 'World', '.']
```

```
tokenizer.tokenize("can't")
```

```
['ca', "n't"]
```

```
word_tokenize("Hello World.")
```

```
['Hello', 'World', '.']
```

```
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
tokenizer.tokenize("Can't is a contradiction")
```

```
['Can', "'", 't', 'is', 'a', 'contradiction']
```

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer("[\w']+")
tokenizer.tokenize("Can't is a contradiction")
```

```
["Can't", 'is', 'a', 'contradiction']
```

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer("[\w ']+")
print(tokenizer.tokenize("abc@gmail.com"))
print(tokenizer.tokenize("xyz@rediffmail.com"))
print(tokenizer.tokenize("rjc@rjcollege.edu.in"))
```

```
['abc', 'gmail', 'com']
['xyz', 'rediffmail', 'com']
['rjc', 'rjcollege', 'edu', 'in']
```

```
address = ['abc@gmail.com', 'xyz@rediffmail.com', 'rjc@rjcollege.edu.in']
ls1 = []
for i in address:
  wr = tokenizer.tokenize(i)
  ls1.append(wr)
```

```
ls1
```

```
[['abc', 'gmail', 'com'],
 ['xyz', 'rediffmail', 'com'],
 ['rjc', 'rjcollege', 'edu', 'in']]
```

```
# H.W.
# store username, domain and webaddress seperately
```

```
text4 = ('The students of MSc IT are using historical data for data for data wareh
from nltk.stem import PorterStemmer as ps
print(text4)
```

```
['The', 'students', 'of', 'MSc', 'IT', 'are', 'using', 'historical', 'data', 'for', 'dat
```

```
lst = []
for w in text4:
  rootWord=ps().stem(w)
  print(rootWord)
  lst.append(rootWord)
```

```
the
student
```

```
of
msc
it
are
use
histor
data
for
data
for
data
wareh
project
```

```
type(lst)
```

```
list
```

```
lst
```

```
['the',
 'student',
 'of',
 'msc',
 'it',
 'are',
 'use',
 'histor',
 'data',
 'for',
 'data',
 'for',
 'data',
 'wareh',
 'project']
```

```
words = ['Unexpected','disagreement','disagee','agreement','quirkiness','historica
for w in words:
  stemPrint = ps().stem(w)
  print(w," -Stem- ", stemPrint)
```

```
Unexpected  -Stem-  unexpect
disagreement  -Stem-  disagr
disagee  -Stem-  disage
agreement  -Stem-  agreement
quirkiness  -Stem-  quirki
historical  -Stem-  histor
canonical  -Stem-  canon
```

```
import nltk
nltk.__file__
```

```
nltk.download('popular')
nltk.download('gutenberg')
from nltk.corpus import gutenberg
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to /root/nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package omw to /root/nltk_data...
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]    |   Package omw-1.4 is already up-to-date!
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
```

```
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection popular
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
```

```
import nltk
from nltk import sent_tokenize
from nltk import word_tokenize


#read single file
text = gutenberg.raw('austen-emma.txt')
text
```

```
'[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, cle
ver, 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 the world\nw
ith very little to distress or vex her.\n\nShe was the youngest of the two daughters of
a most affectionate,\nindulgent father; and had, in consequence of her sister\'s marria
ge,\nbeen mistress of his house from a very early period.  Her mother\nhad died too lon
g ago for her to have more than an indistinct\nremembrance of her caresses; and her pla
ce had been supplied\nby an excellent woman as governess, who had fallen little short\n
```

```
textBlob = '''Emma Woodhouse, handsome, clever, and rich, with a comfortable home
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.'''.split()


from nltk.stem import PorterStemmer as ps
for k in textBlob:
  wordStem = ps().stem(k)
  print(k," -Stem- ", wordStem)
```

```
Emma  -Stem-  emma
Woodhouse,  -Stem-  woodhouse,
handsome,  -Stem-  handsome,
clever,  -Stem-  clever,
and  -Stem-  and
rich,  -Stem-  rich,
with  -Stem-  with
a  -Stem-  a
comfortable  -Stem-  comfort
home  -Stem-  home
and  -Stem-  and
happy  -Stem-  happi
disposition,  -Stem-  disposition,
seemed  -Stem-  seem
to  -Stem-  to
unite  -Stem-  unit
some  -Stem-  some
```

```
of  -Stem-  of
the  -Stem-  the
best  -Stem-  best
blessings  -Stem-  bless
of  -Stem-  of
existence;  -Stem-  existence;
and  -Stem-  and
had  -Stem-  had
lived  -Stem-  live
nearly  -Stem-  nearli
twenty-one  -Stem-  twenty-on
years  -Stem-  year
in  -Stem-  in
the  -Stem-  the
world  -Stem-  world
with  -Stem-  with
very  -Stem-  veri
little  -Stem-  littl
to  -Stem-  to
distress  -Stem-  distress
or  -Stem-  or
vex  -Stem-  vex
her.  -Stem-  her.
```

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()


tex = '''Emma Woodhouse, handsome, clever, and rich, with a comfortable home
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.'''
word_list = nltk.word_tokenize(tex)
print(word_list)
```

```
['Emma', 'Woodhouse', ',', 'handsome', ',', 'clever', ',', 'and', 'rich', ',', 'with',
```

```
for w in word_list:
  lem = lemmatizer.lemmatize(w)
  print(lem)
```

```
Emma
Woodhouse
,
handsome
,
clever
,
and
```

```
rich
,
with
a
comfortable
home
and
happy
disposition
,
seemed
to
unite
some
of
the
best
blessing
of
existence
;
and
had
lived
nearly
twenty-one
year
in
the
world
with
very
little
to
distress
or
vex
her
.
```

```
# https://medium.com/mlearning-ai/nlp-tokenization-stemming-lemmatization-and-part

#
```

Colab paid products - Cancel contracts here