# Python-Machine Learning

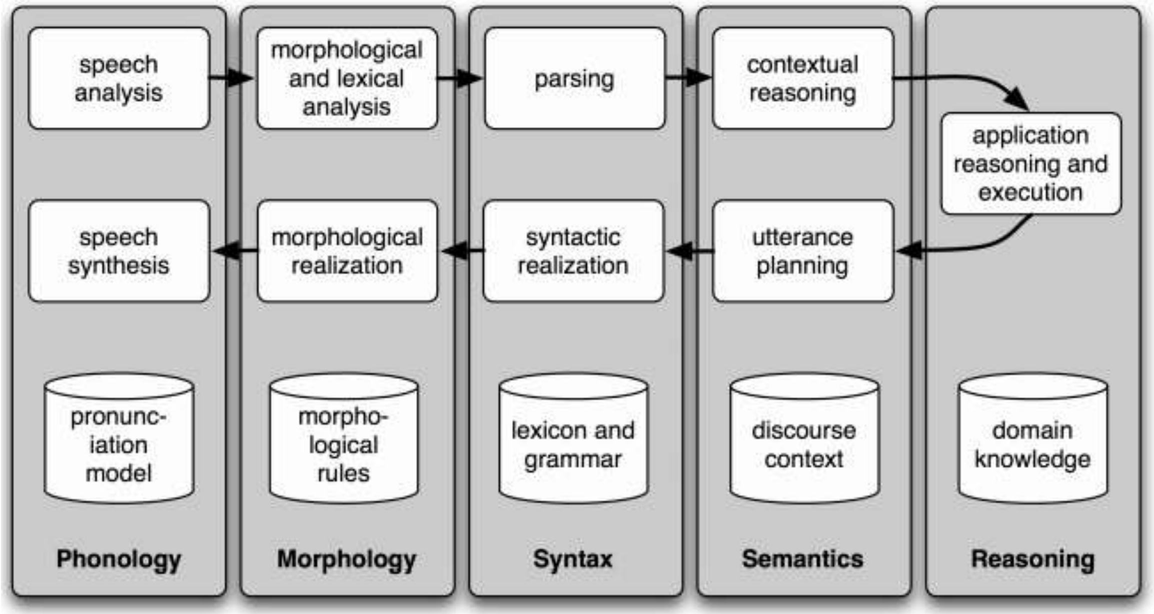
## Natural Language Processing (NLP)

## using NLTK package

Dr. Sarwan Singh

# long-standing challenge within computer science has been to build intelligent machines

*The chief measure of machine intelligence has been a linguistic one, namely the Turing Test: can a dialogue system, responding to a user's typed input with its own textual output, perform so naturally that users cannot distinguish it from a human interlocutor using the same interface? Today, there is substantial ongoing research and development in such areas as machine translation and spoken dialogue, and significant commercial systems are in widespread use*



Simple Pipeline Architecture for a Spoken Dialogue System

Natural Language Analyses with NLTK

Artificial Intelligence

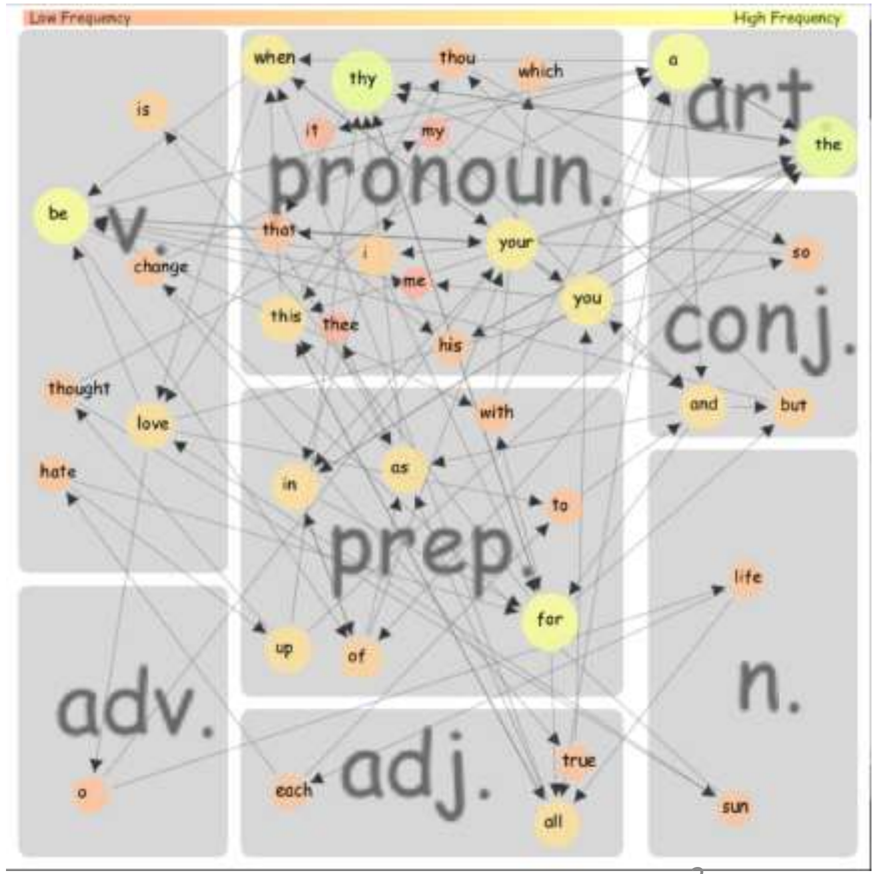
Machine Learning

Deep Learning

# Agenda

- Introduction (NLTK Toolkit)
- History,  benefits ,
- Libraries, Uses
- Tokenize Text Using NLTK
- Wordnet, Lemmatizing Words

*Machine learning is a branch in computer science that studies the design of algorithms that can learn.*



## History

- **Natural Language Toolkit**, or **NLTK**
- NLTK was developed by Steven Bird and Edward Loper in 2001 as part of computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania
- NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning
- There are 32 universities in the US and 25 countries using NLTK in their courses.
- NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities

Source : wikipedia

## Introduction

- **Natural Language Toolkit**, or **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.
- Natural language processing (NLP) is about developing applications and services that are able to understand human languages.
- Practical examples of natural language processing (NLP) are :
  - understanding complete sentences,
  - understanding synonyms of matching words,
  - writing complete grammatically correct sentences and paragraphs,
  - speech recognition, speech translation.

## Benefits of NLP

- As all of you know, millions of gigabytes every day are generated by blogs, social websites, and web pages.
- There are many companies gathering all of this data to better understand users and their passions and make appropriate changes.
- These data could show that the people of India are happy with product A, while the people of the Nepal are happier with product B.
- With NLP, this knowledge can be found instantly (i.e. a real-time result).
- For example, search engines are a type of NLP that give the appropriate results to the right people at the right time.

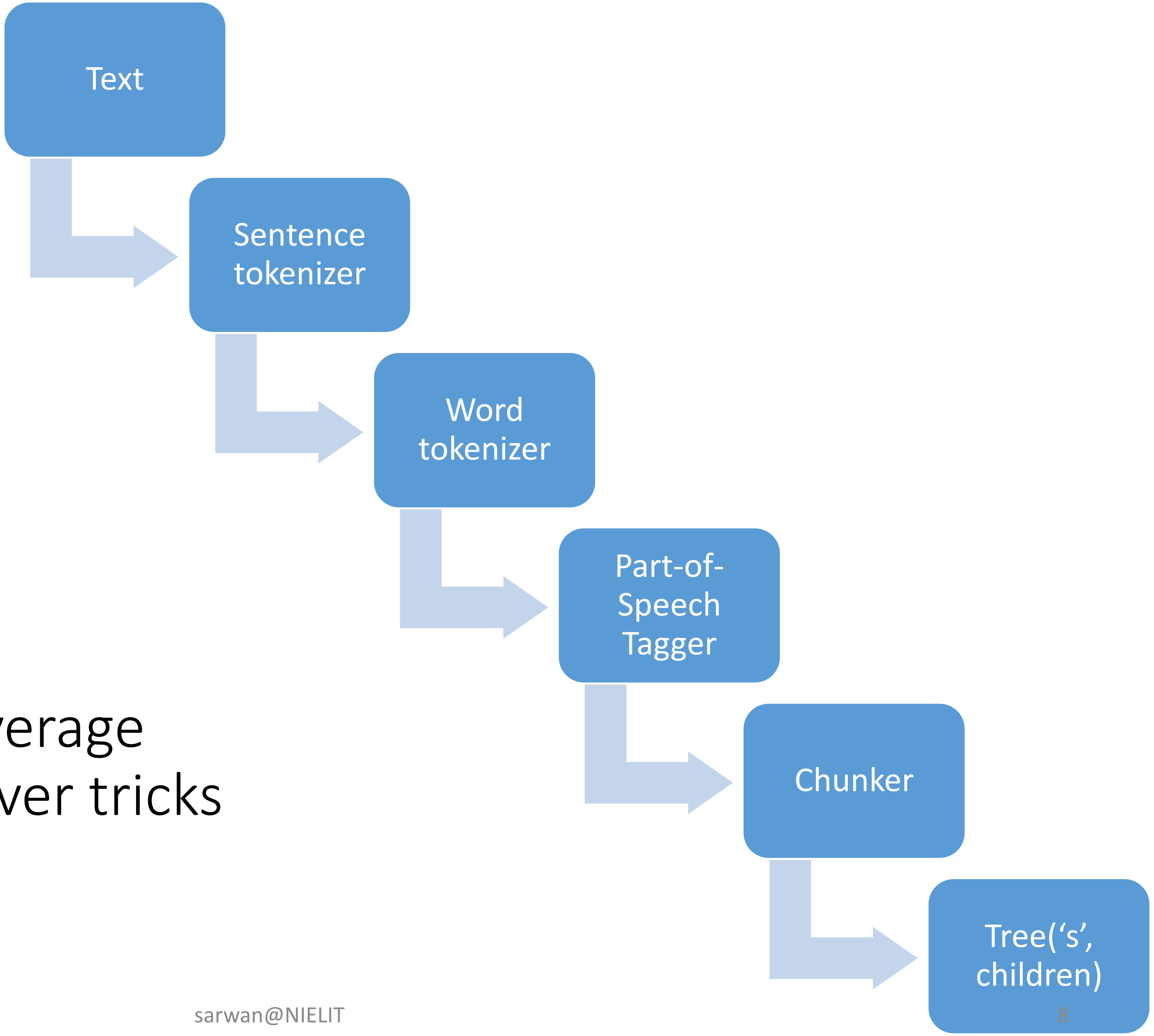
## NLTK Modules & Functionality

| NLTK Modules | Functionality |
|---|---|
| nltk.corpus | Corpus |
| nltk.tokenize, nltk.stem | Tokenizers, stemmers |
| nltk.collocations | t-test, chi-squared, mutual-info |
| nltk.tag | n-gram, backoff,Brill, HMM, TnT |
| nltk.classify, nltk.cluster | Decision tree, Naive bayes, K-means |
| nltk.chunk | Regex,n-gram, named entity |
| nltk.parsing | Parsing |
| nltk.sem, nltk.interence | Semantic interpretation |
| nltk.metrics | Evaluation metrics |
| nltk.probability | Probability & Estimation |
| nltk.app, nltk.chat | Applications |

- Goals of NLTK
  - Simplicity
  - Consistency
  - Extensibility
  - Modularity

<u>other desirables</u>

- Encyclopedic coverage
- Optimization/clever tricks



# NLP Libraries

- There are many open source Natural Language Processing (NLP) libraries. These are some of them:
  - Natural language toolkit (NLTK)
  - Apache OpenNLP
  - Stanford NLP suite
  - Gate NLP library
- Natural language toolkit (NLTK) is the most popular library for natural language processing (NLP).
- It was written in Python and has a big community behind it.
- NLTK also is very easy to learn, actually, it's the easiest natural language processing (NLP) library

## Installing NLTK

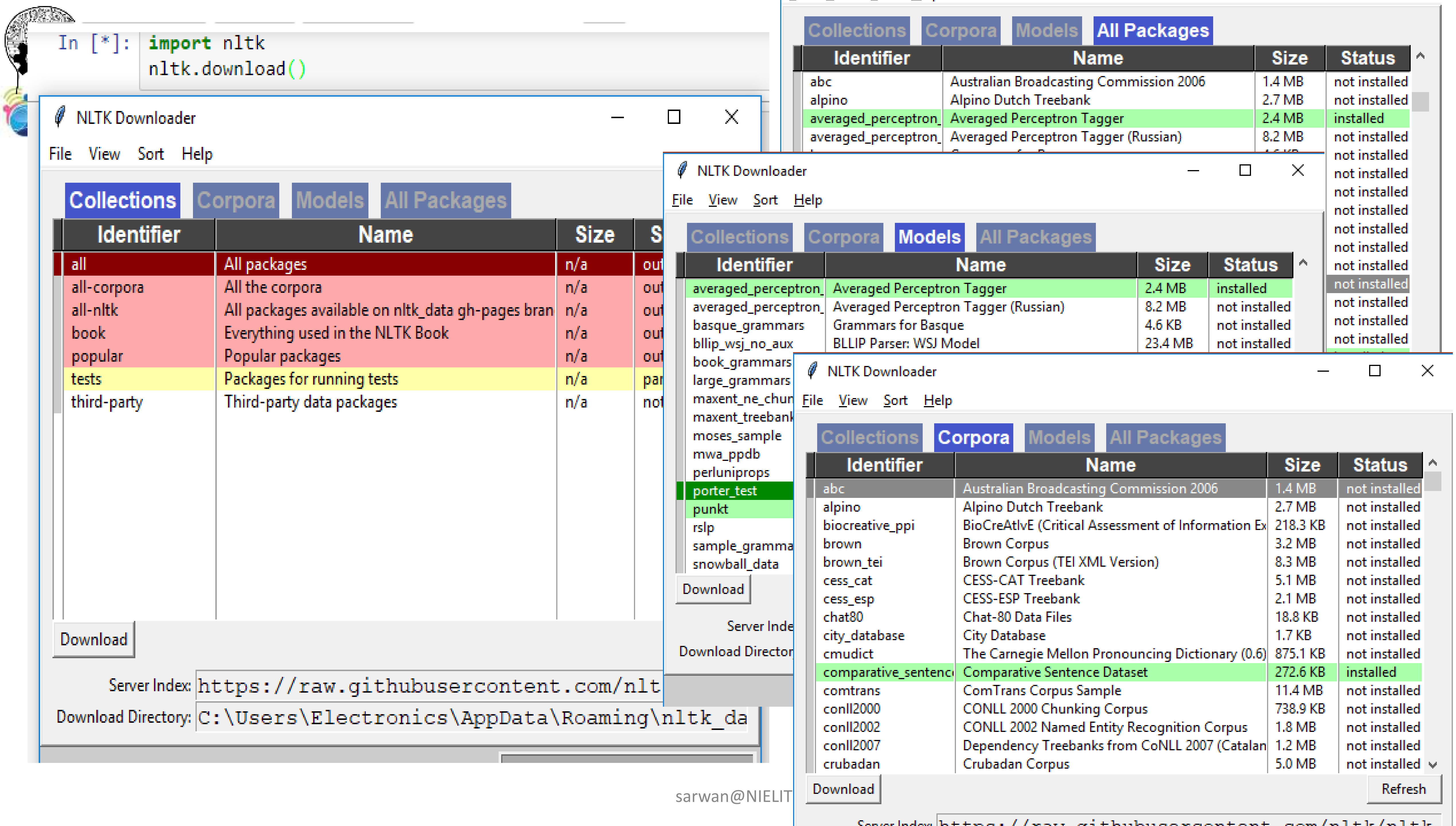
- Install NLTK using pip: # pip install nltk.
- To check if NLTK has installed correctly, you can open your Python terminal and type the following:

```
import nltk
```

- If everything goes fine, that means you've successfully installed NLTK library.
- Once you've installed NLTK, you should install the NLTK packages by running the following code:

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

- This will show the NLTK downloader to choose what packages need to be installed.
- You can install all packages since they all have small sizes with no problem.



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

NLTK Downloader

File View Sort Help

Collections | Corpora | Models | All Packages

| Identifier | Name | Size | S |
|---|---|---|---|
| all | All packages | n/a | out |
| all-corpora | All the corpora | n/a | out |
| all-nltk | All packages available on nltk_data gh-pages bran | n/a | out |
| book | Everything used in the NLTK Book | n/a | out |
| popular | Popular packages | n/a | out |
| tests | Packages for running tests | n/a | par |
| third-party | Third-party data packages | n/a | not |

Download

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

Download Directory: C:\Users\Electronics\AppData\Roaming\nltk_da

Collections | Corpora | Models | All Packages

| Identifier | Name | Size | Status |
|---|---|---|---|
| abc | Australian Broadcasting Commission 2006 | 1.4 MB | not installed |
| alpino | Alpino Dutch Treebank | 2.7 MB | not installed |
| averaged_perceptron_ | Averaged Perceptron Tagger | 2.4 MB | installed |
| averaged_perceptron_ | Averaged Perceptron Tagger (Russian) | 8.2 MB | not installed |

NLTK Downloader

File View Sort Help

Collections | Corpora | Models | All Packages

| Identifier | Name | Size | Status |
|---|---|---|---|
| averaged_perceptron_ | Averaged Perceptron Tagger | 2.4 MB | installed |
| averaged_perceptron_ | Averaged Perceptron Tagger (Russian) | 8.2 MB | not installed |
| basque_grammars | Grammars for Basque | 4.6 KB | not installed |
| bllip_wsj_no_aux | BLLIP Parser: WSJ Model | 23.4 MB | not installed |
| book_grammars | | | |
| large_grammars | | | |
| maxent_ne_chun | | | |
| maxent_treebank | | | |
| moses_sample | | | |
| mwa_ppdb | | | |
| perluniprops | | | |
| porter_test | | | |
| punkt | | | |
| rslp | | | |
| sample_gramma | | | |
| snowball_data | | | |

Download

Server Inde

Download Director

NLTK Downloader

File View Sort Help

Collections | Corpora | Models | All Packages

| Identifier | Name | Size | Status |
|---|---|---|---|
| abc | Australian Broadcasting Commission 2006 | 1.4 MB | not installed |
| alpino | Alpino Dutch Treebank | 2.7 MB | not installed |
| biocreative_ppi | BioCreAtIvE (Critical Assessment of Information Ex | 218.3 KB | not installed |
| brown | Brown Corpus | 3.2 MB | not installed |
| brown_tei | Brown Corpus (TEI XML Version) | 8.3 MB | not installed |
| cess_cat | CESS-CAT Treebank | 5.1 MB | not installed |
| cess_esp | CESS-ESP Treebank | 2.1 MB | not installed |
| chat80 | Chat-80 Data Files | 18.8 KB | not installed |
| city_database | City Database | 1.7 KB | not installed |
| cmudict | The Carnegie Mellon Pronouncing Dictionary (0.6) | 875.1 KB | not installed |
| comparative_sentenc | Comparative Sentence Dataset | 272.6 KB | installed |
| comtrans | ComTrans Corpus Sample | 11.4 MB | not installed |
| conll2000 | CONLL 2000 Chunking Corpus | 738.9 KB | not installed |
| conll2002 | CONLL 2002 Named Entity Recognition Corpus | 1.8 MB | not installed |
| conll2007 | Dependency Treebanks from CoNLL 2007 (Catalan | 1.2 MB | not installed |
| crubadan | Crubadan Corpus | 5.0 MB | not installed |

Download | Refresh

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

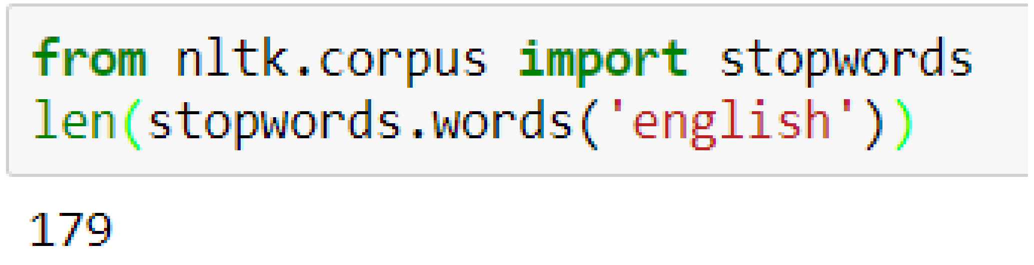
- **Corpus** - Body of text, singular. Corpora is the plural of this. Example: A collection of medical journals.
- **Lexicon** - Words and their meanings. Example: English dictionary. Consider, however, that various fields will have different lexicons
- **Token** - Each "entity" that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.

## Tokenize Text Using NLTK

- Tokenizing text is important since text can't be processed without tokenization.
- Tokenization process means splitting bigger parts to small parts.
- You can tokenize paragraphs to sentences and tokenize sentences to words according to your needs.
- NLTK is shipped with a sentence tokenizer and a word tokenizer.

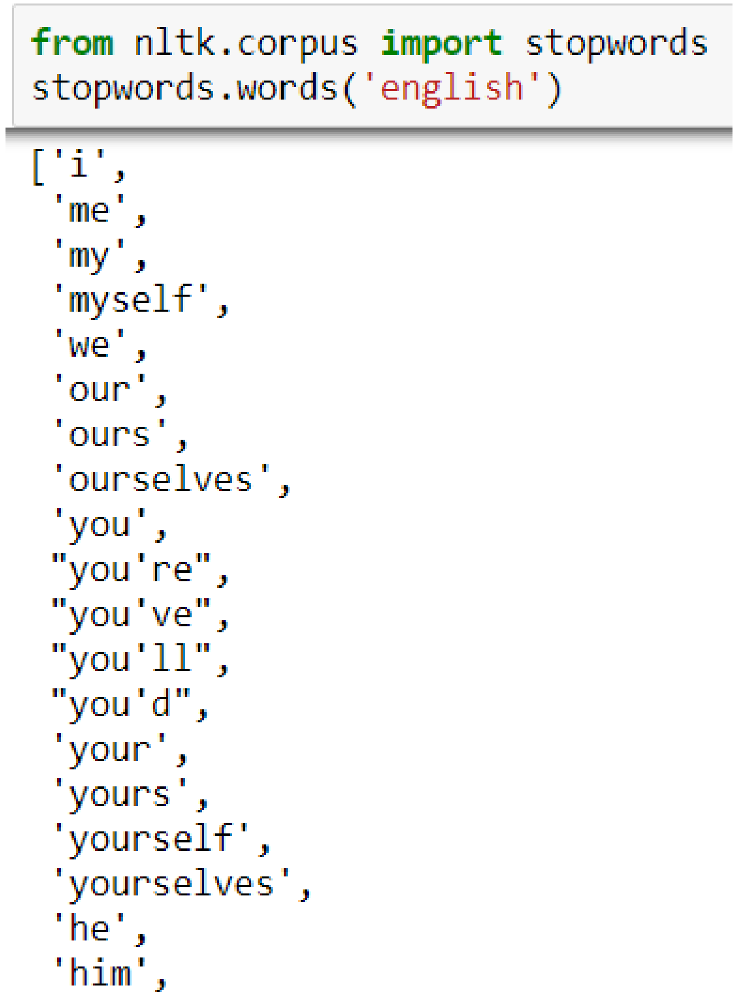
# NLTK stop words

- The foremost challenge with Natural language processing (nlp) is natural language understanding
- Text may contain stop words like ‘the’, ‘is’, ‘are’.
- Stop words should be filtered from the text to be processed. There is no universal list of stop words in nlp research, however the nltk module contains a list of stop words.



```
from nltk.corpus import stopwords
len(stopwords.words('english'))
```

```
179
```

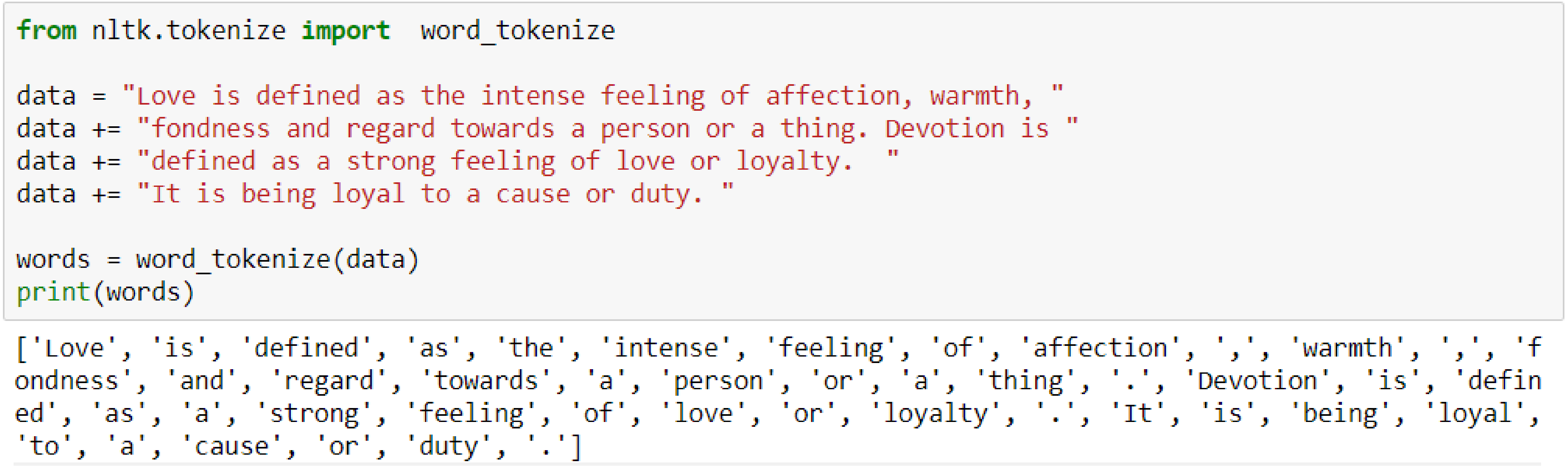


```
from nltk.corpus import stopwords
stopwords.words('english')
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
```

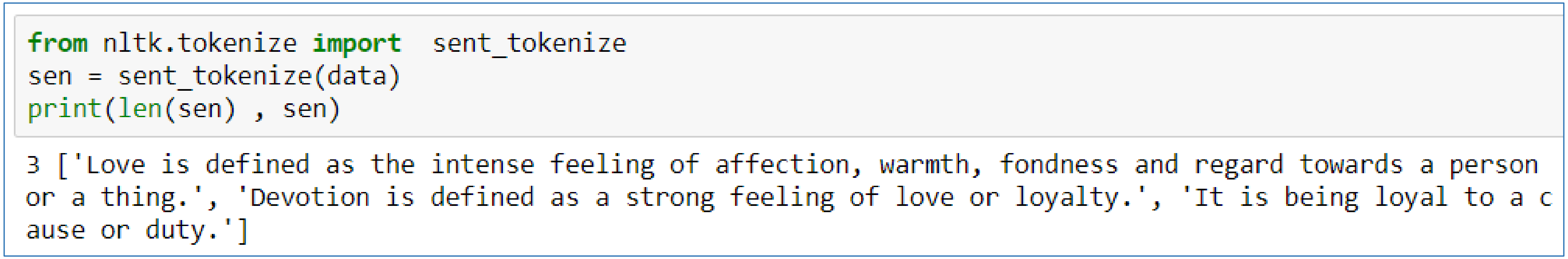
# Word tokenizer

- Tokenizing - Splitting sentences and words from the body of text.



```
from nltk.tokenize import  word_tokenize

data = "Love is defined as the intense feeling of affection, warmth, "
data += "fondness and regard towards a person or a thing. Devotion is "
data += "defined as a strong feeling of love or loyalty.  "
data += "It is being loyal to a cause or duty. "

words = word_tokenize(data)
print(words)
```

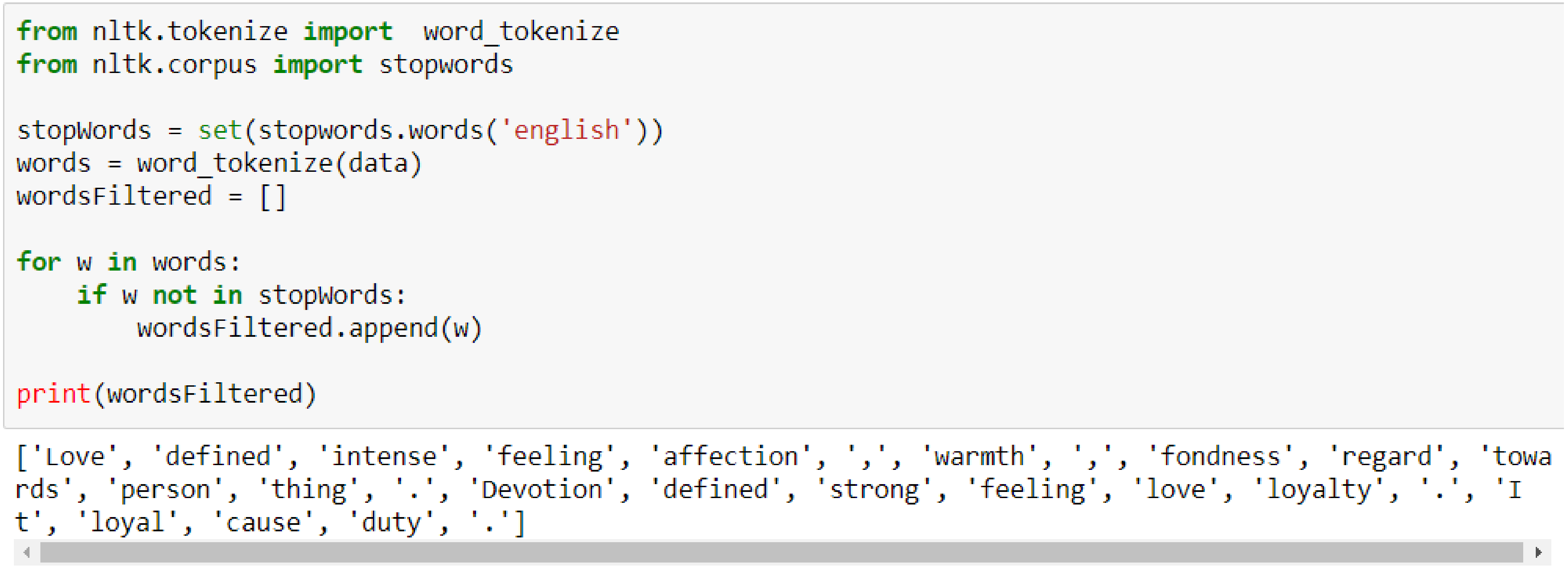
```
['Love', 'is', 'defined', 'as', 'the', 'intense', 'feeling', 'of', 'affection', ',', 'warmth', ',', 'f
ondness', 'and', 'regard', 'towards', 'a', 'person', 'or', 'a', 'thing', '.', 'Devotion', 'is', 'defin
ed', 'as', 'a', 'strong', 'feeling', 'of', 'love', 'or', 'loyalty', '.', 'It', 'is', 'being', 'loyal',
'to', 'a', 'cause', 'or', 'duty', '.']
```



```
from nltk.tokenize import  sent_tokenize
sen = sent_tokenize(data)
print(len(sen) , sen)
```

```
3 ['Love is defined as the intense feeling of affection, warmth, fondness and regard towards a person
or a thing.', 'Devotion is defined as a strong feeling of love or loyalty.', 'It is being loyal to a c
ause or duty.']
```

- create a new list called wordsFiltered which contains all words which are not stop words.
- To create it we iterate over the list of words and only add it if its not in the stopWords list.



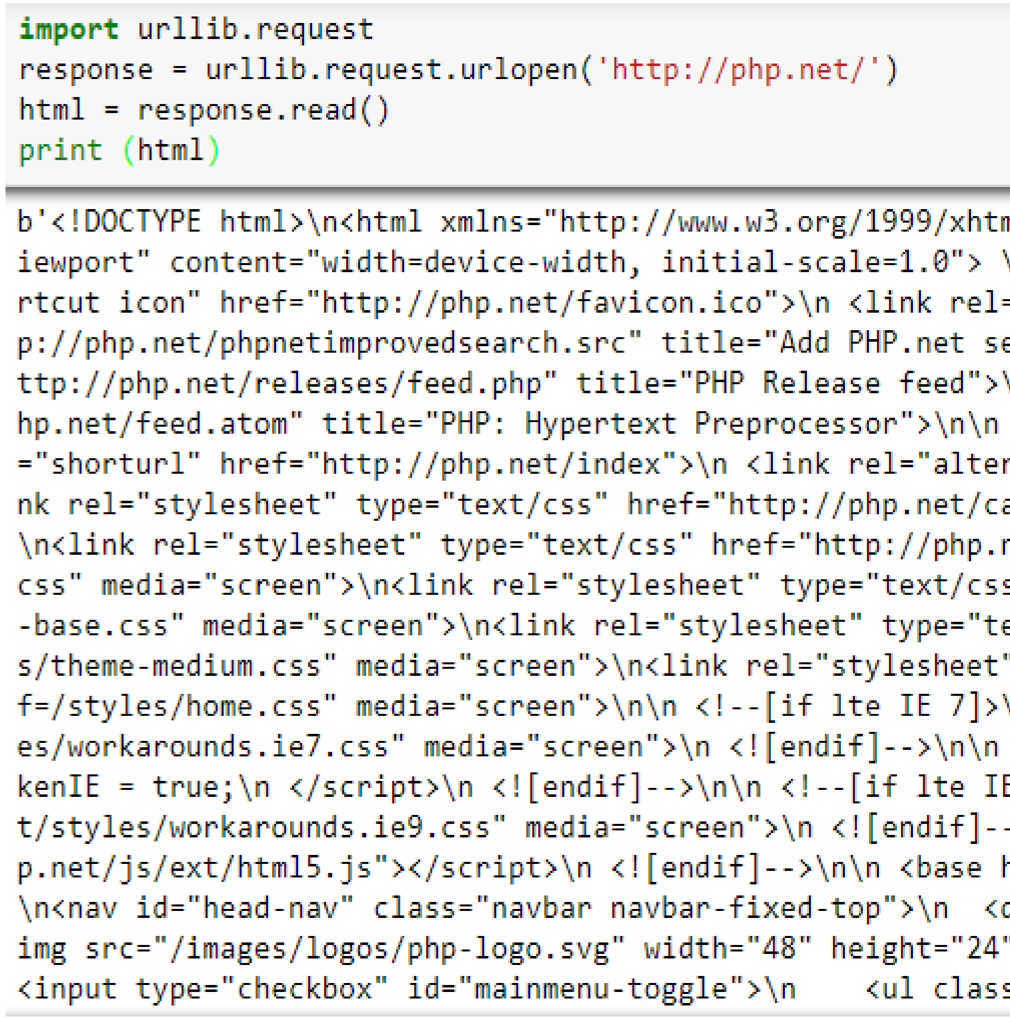
```python
from nltk.tokenize import  word_tokenize
from nltk.corpus import stopwords

stopWords = set(stopwords.words('english'))
words = word_tokenize(data)
wordsFiltered = []

for w in words:
    if w not in stopWords:
        wordsFiltered.append(w)

print(wordsFiltered)
```

```
['Love', 'defined', 'intense', 'feeling', 'affection', ',', 'warmth', ',', 'fondness', 'regard', 'towa
rds', 'person', 'thing', '.', 'Devotion', 'defined', 'strong', 'feeling', 'love', 'loyalty', '.', 'I
t', 'loyal', 'cause', 'duty', '.']
```

# Tokenize Text Using Pure Python

- First, access web page content.
- Then, analyze the text to see what the page is about, using urllib module
- The output contains lot of HTML tags that need to be cleaned.

```
import urllib.request
response = urllib.request.urlopen
                                    ('http://php.net/')
html = response.read()
print (html)
```

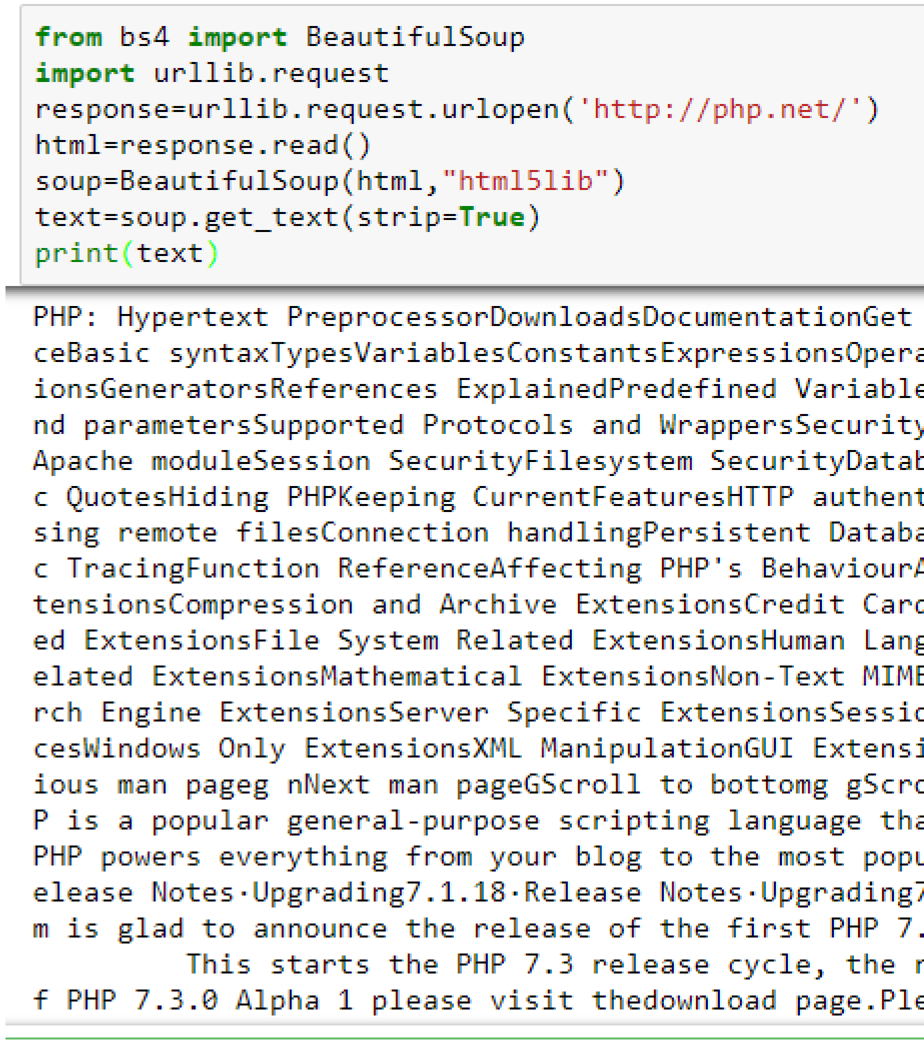


```
import urllib.request
response = urllib.request.urlopen('http://php.net/')
html = response.read()
print (html)
```

```
b'<!DOCTYPE html>\n<html xmlns="http://www.w3.org/1999/xhtm
iewport" content="width=device-width, initial-scale=1.0"> \
rtcut icon" href="http://php.net/favicon.ico">\n <link rel=
p://php.net/phpnetimprovedsearch.src" title="Add PHP.net se
ttp://php.net/releases/feed.php" title="PHP Release feed">\
hp.net/feed.atom" title="PHP: Hypertext Preprocessor">\n\n
="shorturl" href="http://php.net/index">\n <link rel="alter
nk rel="stylesheet" type="text/css" href="http://php.net/ca
\n<link rel="stylesheet" type="text/css" href="http://php.r
css" media="screen">\n<link rel="stylesheet" type="text/css
-base.css" media="screen">\n<link rel="stylesheet" type="te
s/theme-medium.css" media="screen">\n<link rel="stylesheet"
f=/styles/home.css" media="screen">\n\n <!--[if lte IE 7]>\
es/workarounds.ie7.css" media="screen">\n <![endif]-->\n\n
kenIE = true;\n </script>\n <![endif]-->\n\n <!--[if lte IE
t/styles/workarounds.ie9.css" media="screen">\n <![endif]--
p.net/js/ext/html5.js"></script>\n <![endif]-->\n\n <base h
\n<nav id="head-nav" class="navbar navbar-fixed-top">\n  <c
img src="/images/logos/php-logo.svg" width="48" height="24"
<input type="checkbox" id="mainmenu-toggle">\n    <ul class
```

# Cleaning HTML Text

```
from bs4 import BeautifulSoup

import urllib.request

response=urllib.request.urlopen('http://php.net/')

html=response.read()

soup=BeautifulSoup(html,"html5lib")

text=soup.get_text(strip=True)

print(text)
```



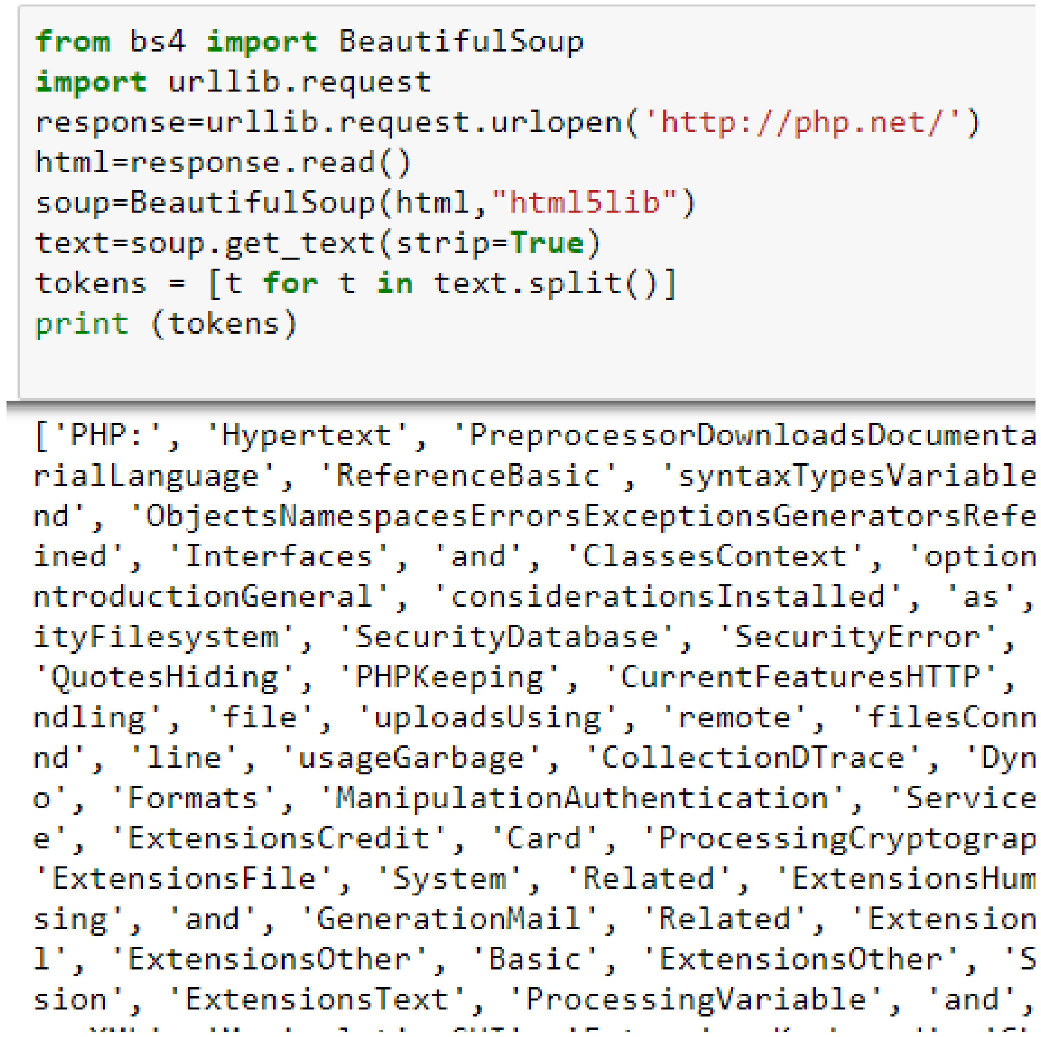
```
from bs4 import BeautifulSoup
import urllib.request
response=urllib.request.urlopen('http://php.net/')
html=response.read()
soup=BeautifulSoup(html,"html5lib")
text=soup.get_text(strip=True)
print(text)
```

```
PHP: Hypertext PreprocessorDownloadsDocumentationGet
ceBasic syntaxTypesVariablesConstantsExpressionsOpera
ionsGeneratorsReferences ExplainedPredefined Variable
nd parametersSupported Protocols and WrappersSecurity
Apache moduleSession SecurityFilesystem SecurityDatab
c QuotesHiding PHPKeeping CurrentFeaturesHTTP authent
sing remote filesConnection handlingPersistent Databa
c TracingFunction ReferenceAffecting PHP's BehaviourA
tensionsCompression and Archive ExtensionsCredit Card
ed ExtensionsFile System Related ExtensionsHuman Lang
elated ExtensionsMathematical ExtensionsNon-Text MIME
rch Engine ExtensionsServer Specific ExtensionsSessio
cesWindows Only ExtensionsXML ManipulationGUI Extensi
ious man pageg nNext man pageGScroll to bottomg gScro
P is a popular general-purpose scripting language tha
PHP powers everything from your blog to the most popu
elease Notes·Upgrading7.1.18·Release Notes·Upgrading7
m is glad to announce the release of the first PHP 7.
          This starts the PHP 7.3 release cycle, the r
f PHP 7.3.0 Alpha 1 please visit thedownload page.Ple
```

# Splitting Text

- convert the text into tokens by splitting the text

```
text=soup.get_text(strip=True)
tokens = [t for t in text.split()]
print (tokens)
```

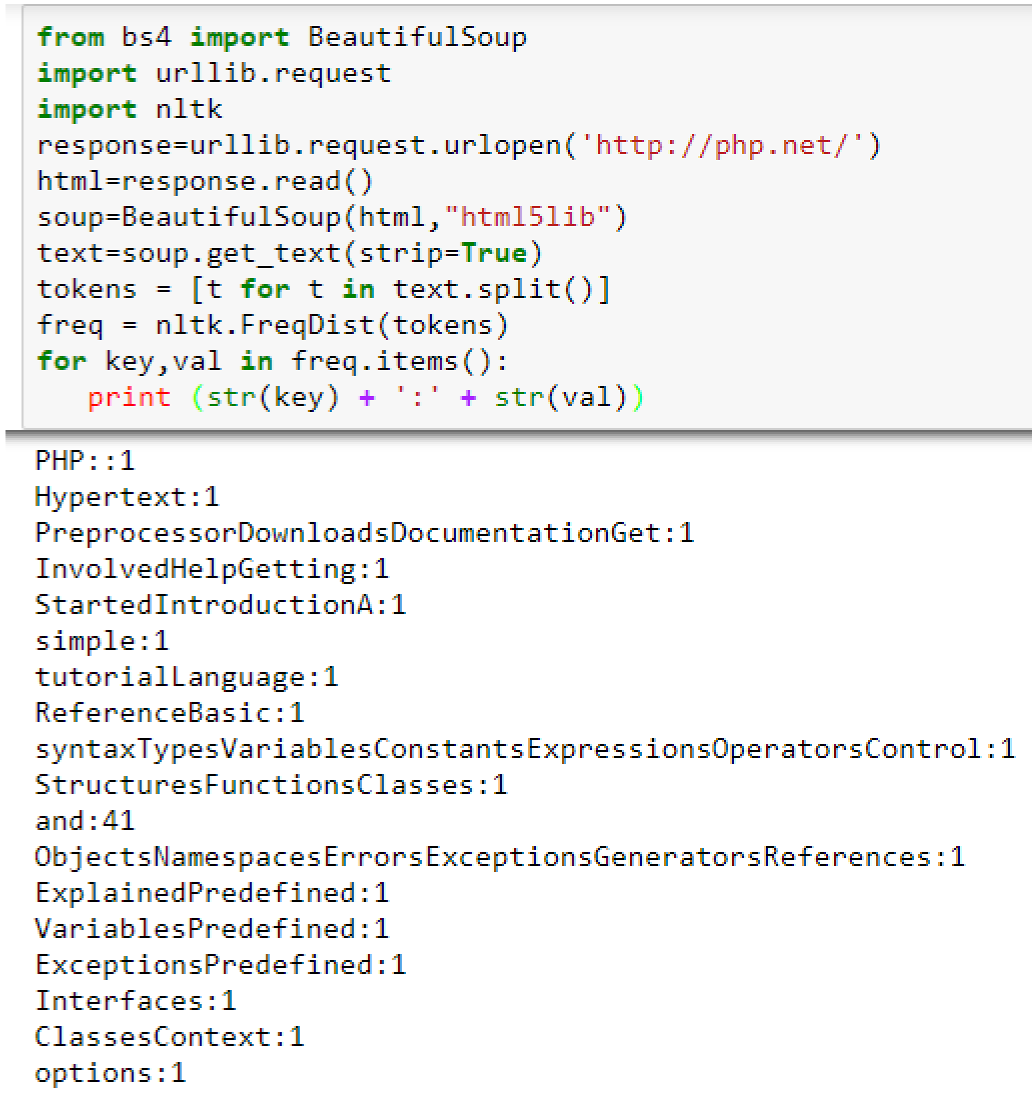


```
from bs4 import BeautifulSoup
import urllib.request
response=urllib.request.urlopen('http://php.net/')
html=response.read()
soup=BeautifulSoup(html,"html5lib")
text=soup.get_text(strip=True)
tokens = [t for t in text.split()]
print (tokens)
```

```
['PHP:', 'Hypertext', 'PreprocessorDownloadsDocumenta
rialLanguage', 'ReferenceBasic', 'syntaxTypesVariable
nd', 'ObjectsNamespacesErrorsExceptionsGeneratorsRefe
ined', 'Interfaces', 'and', 'ClassesContext', 'option
ntroductionGeneral', 'considerationsInstalled', 'as',
ityFilesystem', 'SecurityDatabase', 'SecurityError',
'QuotesHiding', 'PHPKeeping', 'CurrentFeaturesHTTP',
ndling', 'file', 'uploadsUsing', 'remote', 'filesConn
nd', 'line', 'usageGarbage', 'CollectionDTrace', 'Dyn
o', 'Formats', 'ManipulationAuthentication', 'Service
e', 'ExtensionsCredit', 'Card', 'ProcessingCryptograp
'ExtensionsFile', 'System', 'Related', 'ExtensionsHum
sing', 'and', 'GenerationMail', 'Related', 'Extension
l', 'ExtensionsOther', 'Basic', 'ExtensionsOther', 'S
sion', 'ExtensionsText', 'ProcessingVariable', 'and',
```

# Counting word Frequency

- calculate the frequency distribution of those tokens using Python NLTK
- Using FreqDist() function in NLTK



```python
from bs4 import BeautifulSoup
import urllib.request
import nltk
response=urllib.request.urlopen('http://php.net/')
html=response.read()
soup=BeautifulSoup(html,"html5lib")
text=soup.get_text(strip=True)
tokens = [t for t in text.split()]
freq = nltk.FreqDist(tokens)
for key,val in freq.items():
   print (str(key) + ':' + str(val))
```

```
PHP::1
Hypertext:1
PreprocessorDownloadsDocumentationGet:1
InvolvedHelpGetting:1
StartedIntroductionA:1
simple:1
tutorialLanguage:1
ReferenceBasic:1
syntaxTypesVariablesConstantsExpressionsOperatorsControl:1
StructuresFunctionsClasses:1
and:41
ObjectsNamespacesErrorsExceptionsGeneratorsReferences:1
ExplainedPredefined:1
VariablesPredefined:1
ExceptionsPredefined:1
Interfaces:1
ClassesContext:1
options:1
```

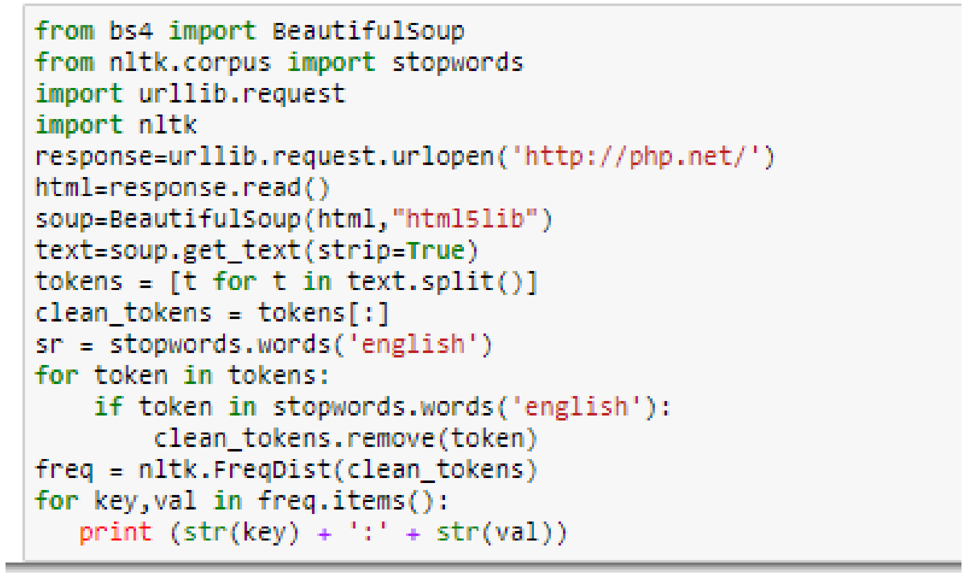
# Remove Stop Words Using NLTK

- NLTK is shipped with stop words lists for most languages.

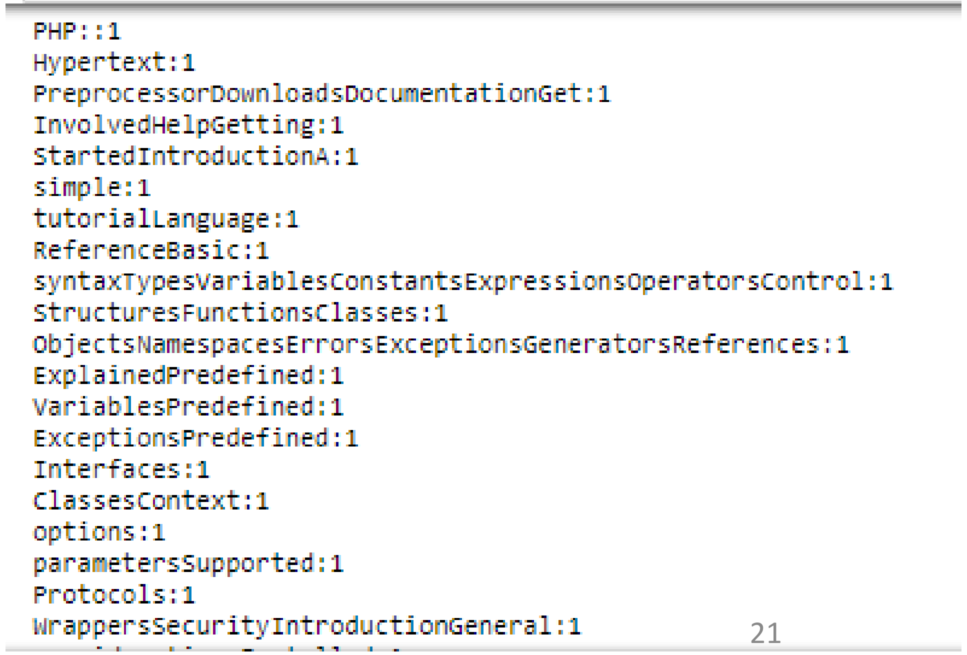
  from nltk.corpus import stopwords
  stopwords.words('english')

- Iterate over the tokens and remove the stop words

```
tokens = [t for t in text.split()]
clean_tokens = tokens[:]
sr = stopwords.words('english')
for token in tokens:
    if token in sr:
        clean_tokens.remove(token)
freq = nltk.FreqDist(clean_tokens)
for key,val in freq.items():
   print (str(key) + ':' + str(val))
```



```
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
import urllib.request
import nltk
response=urllib.request.urlopen('http://php.net/')
html=response.read()
soup=BeautifulSoup(html,"html5lib")
text=soup.get_text(strip=True)
tokens = [t for t in text.split()]
clean_tokens = tokens[:]
sr = stopwords.words('english')
for token in tokens:
    if token in stopwords.words('english'):
        clean_tokens.remove(token)
freq = nltk.FreqDist(clean_tokens)
for key,val in freq.items():
   print (str(key) + ':' + str(val))
```

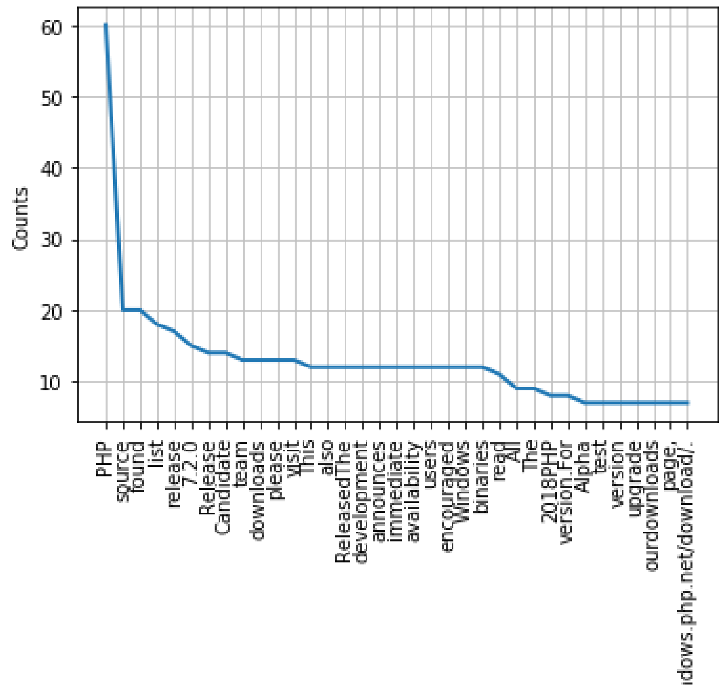


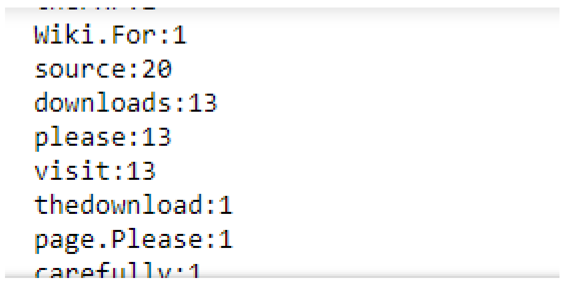
```
PHP::1
Hypertext:1
PreprocessorDownloadsDocumentationGet:1
InvolvedHelpGetting:1
StartedIntroductionA:1
simple:1
tutorialLanguage:1
ReferenceBasic:1
syntaxTypesVariablesConstantsExpressionsOperatorsControl:1
StructuresFunctionsClasses:1
ObjectsNamespacesErrorsExceptionsGeneratorsReferences:1
ExplainedPredefined:1
VariablesPredefined:1
ExceptionsPredefined:1
Interfaces:1
ClassesContext:1
options:1
parametersSupported:1
Protocols:1
WrappersSecurityIntroductionGeneral:1
```

- Printing graph from the output

freq.plot(20)

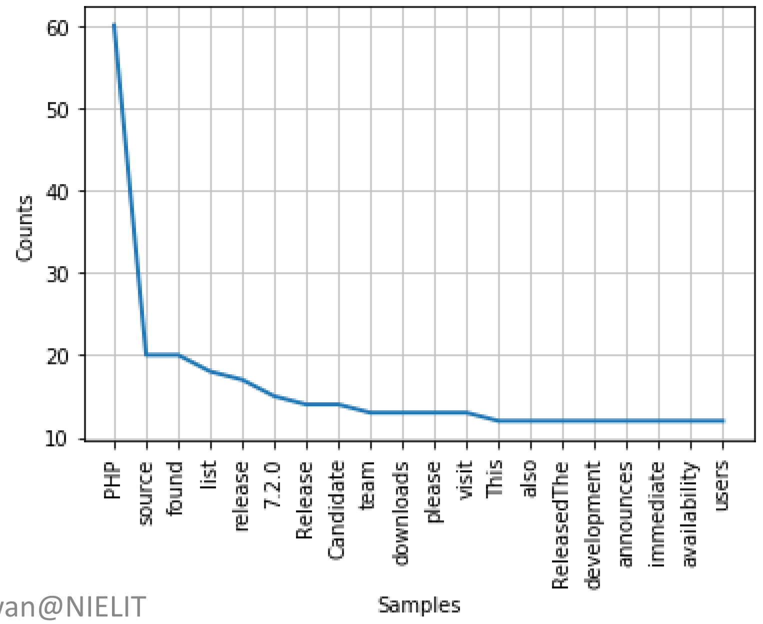
```
freq.plot(35)
```





```
Wiki.For:1
source:20
downloads:13
please:13
visit:13
thedownload:1
page.Please:1
carefully:1
```

```
freq.plot(20, cumulative=False)
```

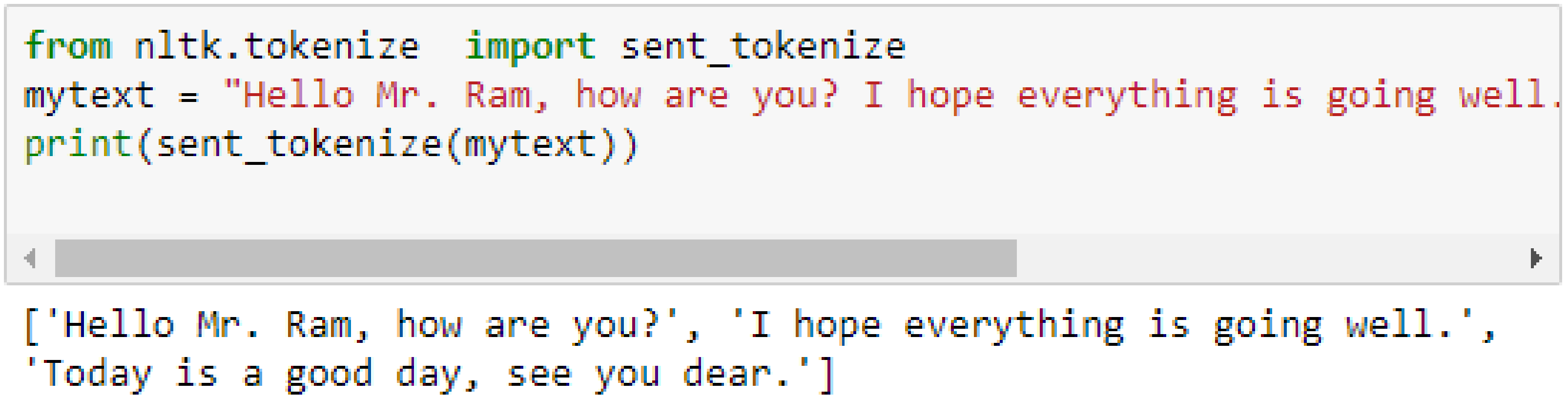


## tokenize text to sentences

from nltk.tokenize  import sent_tokenize

mytext = "Hello Mr. Ram, how are you? I hope everything is going well. Today is a good day, see you dear."

print(sent_tokenize(mytext))

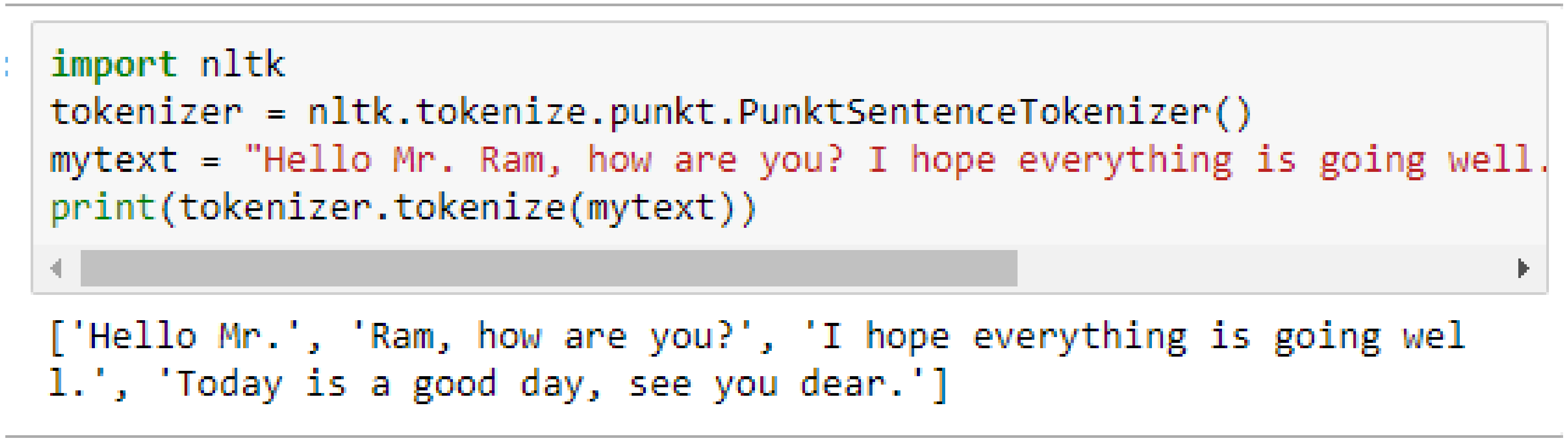


```
from nltk.tokenize  import sent_tokenize
mytext = "Hello Mr. Ram, how are you? I hope everything is going well.
print(sent_tokenize(mytext))
```

```
['Hello Mr. Ram, how are you?', 'I hope everything is going well.',
'Today is a good day, see you dear.']
```

# using PunktSentenceTokenizer

```
from nltk.tokenize import word_tokenize
mytext = "Hello Mr. Ram, how are you? I hope everything is going well. Today is a good day, see you dear."
print(word_tokenize(mytext))
```

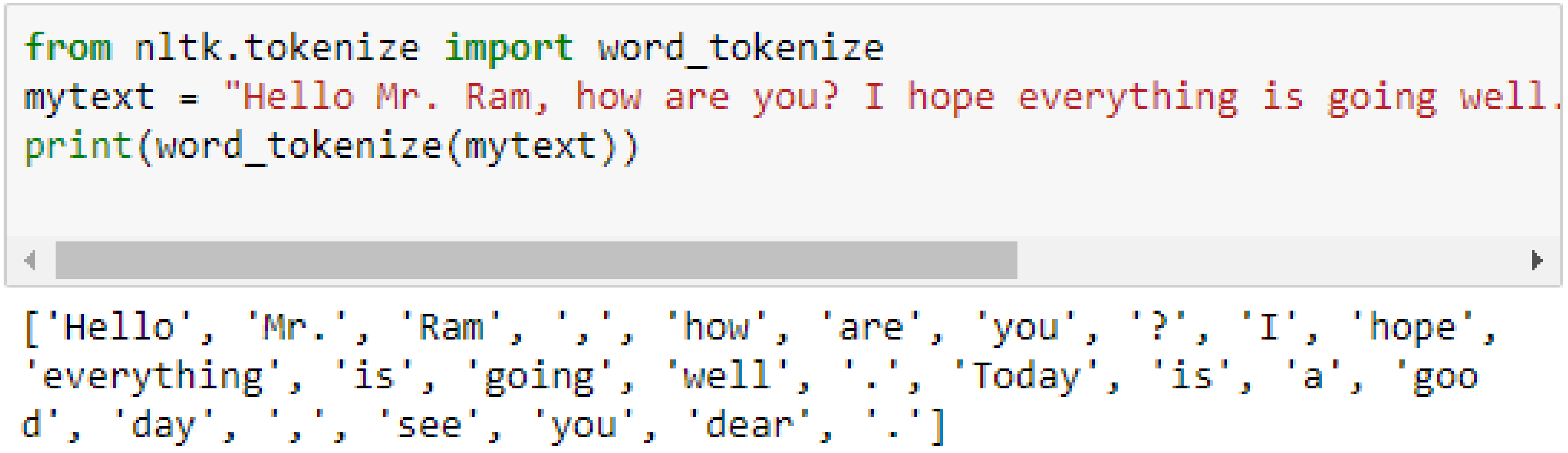


```
import nltk
tokenizer = nltk.tokenize.punkt.PunktSentenceTokenizer()
mytext = "Hello Mr. Ram, how are you? I hope everything is going well.
print(tokenizer.tokenize(mytext))
```

```
['Hello Mr.', 'Ram, how are you?', 'I hope everything is going wel
l.', 'Today is a good day, see you dear.']
```

# Tokenizing words from the sentence

```
from nltk.tokenize import word_tokenize

mytext = "Hello Mr. Ram, how are you? I hope everything is going well. Today is a good day, see you dear."

print(word_tokenize(mytext))
```

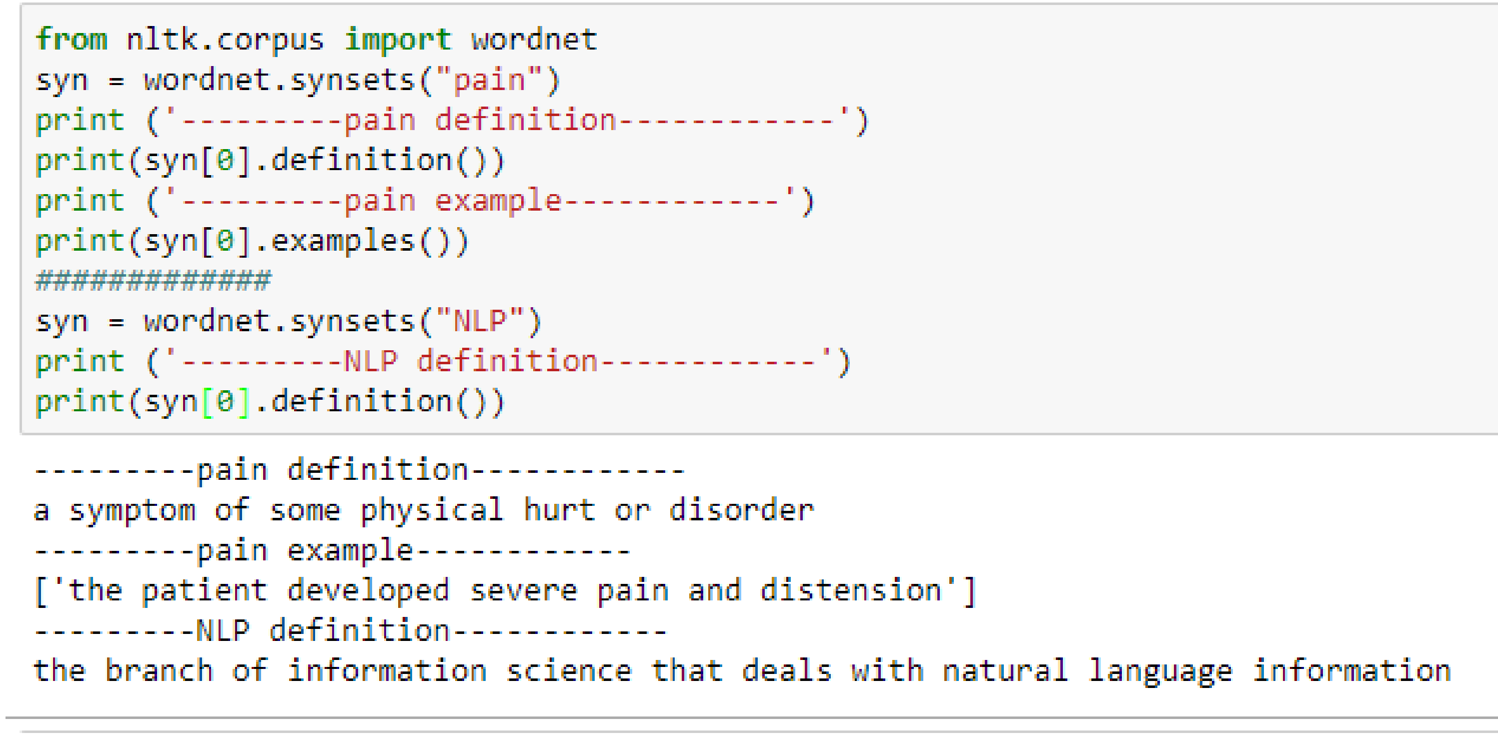


```
from nltk.tokenize import word_tokenize
mytext = "Hello Mr. Ram, how are you? I hope everything is going well.
print(word_tokenize(mytext))
```

```
['Hello', 'Mr.', 'Ram', ',', 'how', 'are', 'you', '?', 'I', 'hope',
'everything', 'is', 'going', 'well', '.', 'Today', 'is', 'a', 'goo
d', 'day', ',', 'see', 'you', 'dear', '.']
```

# Get Synonyms From WordNet

- NLTK has one of the packages - WordNet.
- WordNet is a database built for natural language processing.
- It includes groups of synonyms and a brief definition.

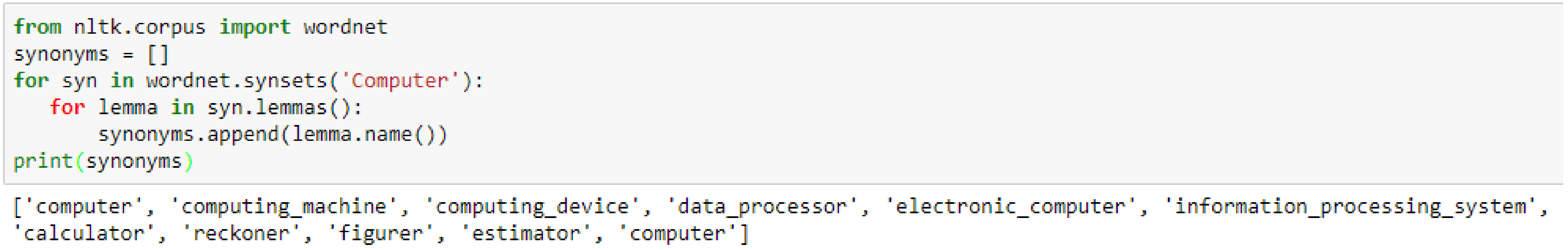


```
from nltk.corpus import wordnet
syn = wordnet.synsets("pain")
print ('---------pain definition-----------')
print(syn[0].definition())
print ('---------pain example-----------')
print(syn[0].examples())
#############
syn = wordnet.synsets("NLP")
print ('---------NLP definition-----------')
print(syn[0].definition())
```

```
---------pain definition-----------
a symptom of some physical hurt or disorder
---------pain example-----------
['the patient developed severe pain and distension']
---------NLP definition-----------
the branch of information science that deals with natural language information
```

# Using WordNet to get synonymous words

```
from nltk.corpus import wordnet
synonyms = []
for syn in wordnet.synsets('Computer'):
   for lemma in syn.lemmas():
       synonyms.append(lemma.name())
print(synonyms)
```

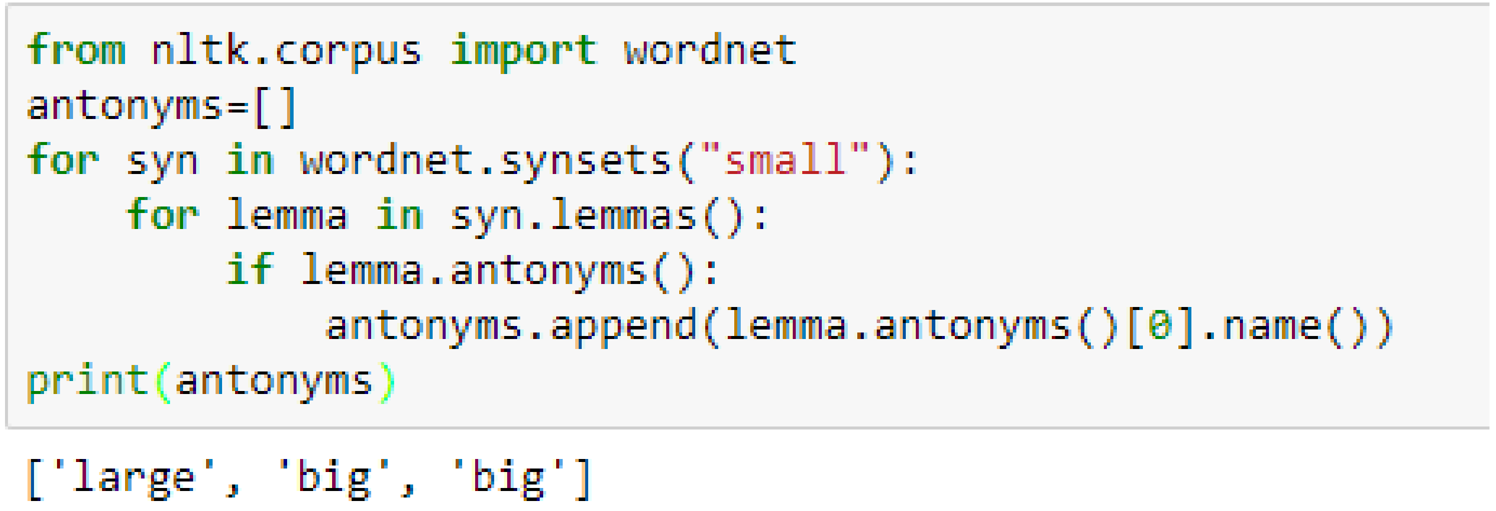


```
from nltk.corpus import wordnet
synonyms = []
for syn in wordnet.synsets('Computer'):
   for lemma in syn.lemmas():
       synonyms.append(lemma.name())
print(synonyms)
```

```
['computer', 'computing_machine', 'computing_device', 'data_processor', 'electronic_computer', 'information_processing_system',
'calculator', 'reckoner', 'figurer', 'estimator', 'computer']
```

# Using WordNet to get antonyms

```
from nltk.corpus import wordnet
antonyms=[]
for syn in wordnet.synsets("small"):
    for lemma in syn.lemmas():
        if lemma.antonyms():
            antonyms.append(lemma.antonyms()[0].name())
print(antonyms)
```



```
from nltk.corpus import wordnet
antonyms=[]
for syn in wordnet.synsets("small"):
    for lemma in syn.lemmas():
        if lemma.antonyms():
            antonyms.append(lemma.antonyms()[0].name())
print(antonyms)
```

```
['large', 'big', 'big']
```

## NLTK Word Stemming

- Word stemming means removing affixes from words and returning the root word. (The stem of the word working is *work*.)
- Search engines use this technique when indexing pages, so many people write different versions for the same word and all of them are stemmed to the root word.
- There are many algorithms for stemming, but the most used algorithm is the Porter stemming algorithm.
- NLTK has a class called PorterStemmer that uses this algorithm.

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
print(stemmer.stem('increses'))
```

- Results in : increas

## Lemmatizing Words Using WordNet

- Word lemmatizing is similar to stemming, but the difference is the result of lemmatizing is a real word.
- lemmatize the same word using NLTK WordNet

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize('increases'))
```

- Results in : increase

## Lemmatize to extract verb, noun,…

- lemmatize a word like the word playing, it will end up with the same word.
- This is because the default part of speech is nouns.

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize('playing', pos="v"))
print(lemmatizer.lemmatize('playing', pos="n"))
print(lemmatizer.lemmatize('playing', pos="a"))
print(lemmatizer.lemmatize('playing', pos="r"))
```