

In [1]:

```
1
2 %%html
3 <style>
4 table {display: block;}
5 td {
6     font-size: 18px
7 }
8 .rendered_html { font-size: 18px; }
9 *{ line-height: 200%; }
10 </style>
```

# Welcome to the Natural Language Processing and the Web WS22/23 course

## Dr. Seid Muhie Yimam

- Email: [seid.muhi.yimam@uni-hamburg.de](mailto:seid.muhi.yimam@uni-hamburg.de) (<mailto:seid.muhi.yimam@uni-hamburg.de>)
- Office: Informatikum, F-415
- Tel: 040 / 42883 - 2418

## Dr. Abhik Jana

- Email [abhik.jana@uni-hamburg.de](mailto:abhik.jana@uni-hamburg.de) (<mailto:abhik.jana@uni-hamburg.de>)
- Office: Informatikum, F-403

- Tel: 040 / 42883 - 2320

## Practice Class

### When:

- Group-A: Tuesday 16:15 - 17:45 D-125/129:
- Group-B: Wednesday 16:15 -- 17:45 G-203:

### Who:

1. Dr. Seid Muhie Yimam
2. Dr. Abhik Jana

## Exercises

- Exercises (some can be done in groups) should be submitted before the next tutorial
- Machine learning and final projects will have separate deadlines
- Machine learning projects and final projects can be done in groups

---

### To pass the class:

- Obtain at least **50%** in the exercise points and Machine learning projects **AND**
- Obtain at least **50%** in the final project points

- 
- You must use your own computer to do the exercises and projects

- All exercises will use Python as programming language with different NLP libraries(example [NLTK](#), [spaCy](#)), visualization (example [matplotlib](#) or [seaborn](#)) , and machine learning (example [scikit learn](#), [fast.ai](#) and [PyTorch](#)) frameworks

In [3]:

```
1 y = 6
2 print(y)
```

6

## Assignment 1 solution

### Course Outline

Week	Topic	Lecturer	When
1 - 3	NLTK and different NLP processing components	Seid/Abhik	Oct 18/19 -- Nov 1/2
4	Introductions to ML using Scikit learn and Feature engineering for ML projects	Seid/Abhik	Nov 8/9, Distribute ML Project
5	Deep learning using PyTorch	Seid/Abhik	Nov 15/16
6	Crossmodal Learning using PyTorch	Seid/Abhik	Nov 22/23
7	ML project presentation	Seid/Abhik	Nov 29/20
8	Distribute Project idea	Seid/Abhik	Dec 6/7
9	Project proposal presentation/discussion	Seid/Abhik	Dec 13/14
10 - 13	Project development consultation	Seid/Abhik	Dec 20/21/ -- Jan 24/25
14	Final project presentation	Seid/Abhik	Jan 31/Feb 1

# How to install Python

The Practice class will be conducted using [Python 3](#). We will rely on [Anaconda3](#) (<https://www.anaconda.com/distribution/> (<https://www.anaconda.com/distribution/>)), specifically version [3.7+](#), a [Python](#) (and [R](#)) distribution which include most of the NLP and machine learning components. It also includes [Anaconda Navigator](#), which is desktop graphical user interface that allows to launch different applications (Jupyter notebook, Spider GUI, GraphViz...) easily.

## How to install Anaconda

Choose Python 3.7+ and install it to a location you prefer. Currently, Anaconda installs around [200 packages](#), which include [NLTK](#) and the [scikit learn](#) machine learning packages. See [here](http://docs.anaconda.com/anaconda) (<http://docs.anaconda.com/anaconda>) for the list of included packages.

## Accessing Jupyter Notebook

For most of the practice classes, we will use the [Jupyter Notebook](#) web application to run and demonstrate examples. Except in the case of developing complex assignments and projects, you will also use [Jupyter notebook](#) to submit your assignments and in-class exercises.

## How to open Jupyter Notebook:

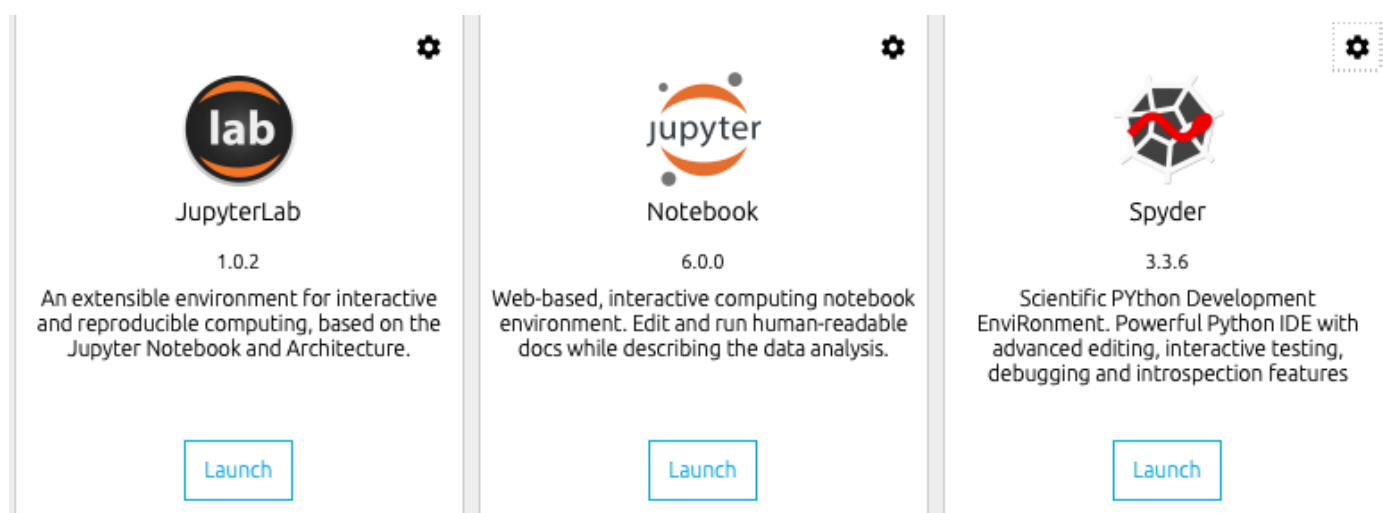
### Using Anaconda Navigation

To open [Jupyter Notebook](#) using Anaconda navigator:

Once Anaconda is installed, open [Anaconda Navigator](#) from your program/application lists for [MacOS](#) and [Windows](#).

For Linux, open a terminal and type `anaconda-navigator`

Then you can choose either [JupyterLab](#) or [Jupyter Notebook](#) to open Jupyter notebook. You can also open it via Spyder but you need to install a notebook plugin



## Open Jupyter using command line

From the command line, type:

```
jupyter notebook
```

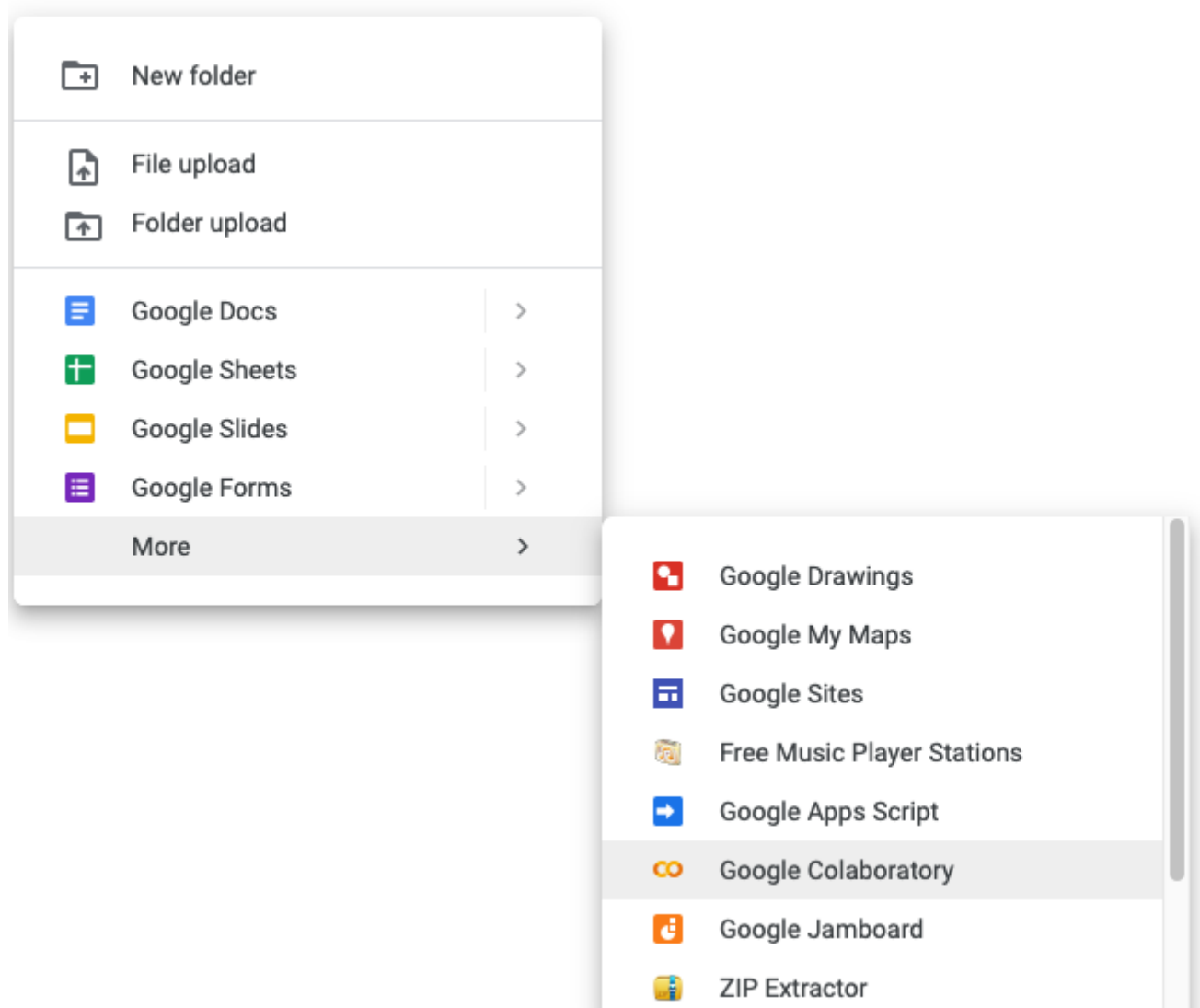
This will open the Jupyter Notebook in the browser.

## Google Colab

Google provides a free cloud service based on Jupyter Notebooks that support free GPU, allows developing deep learning applications using PyTorch, TensorFlow, Keras, and so on. Using Colab, you can share your Jupyter notebooks, mount your Google Drive, etc.

# Creating a folder for your notebooks

Create a folder in your Google Drive and create Google Colaboratory



Or you can directly go to Colab: <https://colab.research.google.com/> and create a new notebook.

From there you can create a new notebook and set up your environment, such as selecting a GPU.

## Mounting your Google Drive

Mounting your Google Drive allows you to persist models, files, and logs on your Google Drive folder as the Colab session is limited.

Once in Colab, you can run the following code to mount to your google drive

```
from google.colab import drive  
drive.mount('/content/gdrive')
```

It will generate a link, click the link, copy the code and paste it in the text box, press Enter and your drive will be mounted.

To see the contents in your Google Drive

```
!ls "/content/gdrive/My Drive/"
```

## How to open the associated notebooks

1. From Moodle, unzip the attachment [ch1.zip](#)
2. Once Jupyter notebook is open, navigate to the extracted folder and open the file [ch1.ipynb](#)

## Notebook cheatsheet

See [here \(https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/pdf\\_bw/\)](https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/pdf_bw/) to get the basic shortcuts and commands to execute a Jupyter notebook cell or display information

Example: Inside a [cell](#), The following are important shortcuts:

1. `shift-Tab-Tab` -- Show documentation of methods or classes
2. `shift-Enter` --- Execute cell and move/create the next cell
3. `Ctrl-Enter` -- Execute a cell and remain the focus on the same cell

---

## ~ Hands on Session ~

# Install Anaconda, Open Notebooks, Colab Environment **15 minutes**

---

## Python preprocessing and NLP pipeline

In this practice class, we will install and discuss two packages/libraries that are important for preprocessing and NLP pipeline, namely [NLTK](#) and [spaCy](#). We will use these two libraries extensively for the practice classes and assignments throughout the course.

## Installing NLTK components

[NLTK](#) is installed automatically with the Anaconda distribution. However, different modules, datasets, and packages are left for the user to download [on demand](#). Try running the following code to install some or all the required resources. If you already have the required resources, the program should run without any problem. Otherwise, it will give an error with hints on how to install the missing resources.

Example of installing missed resources:

```
import nltk  
nltk.download('punkt')
```



In [2]:

```
1 import nltk
2 nltk.download('punkt')
```

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

Out[2]:

True

If you want to choose from a graphical user interface, run the following from the command line

```
nltk.download()
```

Then you can choose what you like to download. You can download everything but it will take more time and storage.

To install all the packages/resources, run the following

```
nltk.download('all')
```

In [3]:

```
1 import nltk
2 #Download specific resources
3 nltk.download('gutenberg')
4 nltk.download('genesis')
5 nltk.download('inaugural')
6 nltk.download('nps_chat')
7 nltk.download('webtext')
8 nltk.download('treebank')
9 # If you want to download everything from nltk, uncomment the following
10 # nltk.download('all')
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]       /Users/seidmuhieyimam/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package genesis to
[nltk_data]       /Users/seidmuhieyimam/nltk_data...
[nltk_data]   Package genesis is already up-to-date!
[nltk_data] Downloading package inaugural to
[nltk_data]       /Users/seidmuhieyimam/nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
[nltk_data] Downloading package nps_chat to
[nltk_data]       /Users/seidmuhieyimam/nltk_data...
[nltk_data]   Package nps_chat is already up-to-date!
[nltk_data] Downloading package webtext to
[nltk_data]       /Users/seidmuhieyimam/nltk_data...
[nltk_data]   Package webtext is already up-to-date!
[nltk_data] Downloading package treebank to
[nltk_data]       /Users/seidmuhieyimam/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
```

Out[3]:

True

The NLTK resource contains dataset that are annotated for different linguistic annotation tasks such as POS tagging, dependency parsing, NER...

In [4]:

```
1 # Show all books from nltk
2 from nltk.book import *
3 #the books are available as text1---text9
4 print (text1.name)
5 print ("----")
6 print (text1.concordance("monstrous"))
7 print ("----")
```

\*\*\* 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

Moby Dick by Herman Melville 1851

----

Displaying 11 of 11 matches:

ong the former , one was of a most monstrous size . ... This came towa  
rds us ,

ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork  
we have r

ll over with a heathenish array of monstrous clubs and spears . Some w  
ere thick

d as you gazed , and wondered what monstrous cannibal and savage could  
ever hav

that has survived the flood ; most monstrous and most mountainous ! Th  
at Himmal

they might scout at Moby Dick as a monstrous fable , or still worse an  
d more de

th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I sh  
all ere l

ing Scenes . In connexion with the monstrous pictures of whales , I am  
strongly

ere to enter upon those still more monstrous stories of them which are  
to be fo

ght have been rummaged out of this monstrous cabinet there is no telli  
ng . But

of Whale - Bones ; for Whales of a monstrous size are oftentimes cast  
up dead u

None

----

Individual items can be downloaded from NLTK corpus library

For example, the brown corpus can be downloaded and imported as follows

```
nltk.download('brown')
```

```
from nltk.corpus import brown
```

The brown corpus includes different genres of texts such as fictions, reviews, hobbies, news, and so on.

To view the categories:

```
brown.categories()
```

In [5]:

```
1 nltk.download('brown')
2 from nltk.corpus import brown
3 brown.categories()
```

```
[nltk_data] Downloading package brown to
[nltk_data]   /Users/seidmuhieyimam/nltk_data...
[nltk_data]   Package brown is already up-to-date!
```

Out[5]:

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

In [6]:

```
1 #List the first 10 words in the corpus
2 print(brown.words(categories='fiction')[:10])
3 # How many words are there in the corpus
4 print("All:", len(brown.words()))
5 # How many words in the adventure category
6 print("adventure:", len(brown.words(categories='adventure')))
7 # Print the last 10 sentences from the romance categories
8 print("sents from romance:", brown.sents(categories="romance")[-10:])
9 #How many sentences are there in total
10 print("All Sentences:", len(brown.sents()))
```

```
['Thirty-three', 'Scotty', 'did', 'not', 'go', 'back', 'to', 'school',
'.', 'His']
All: 1161192
adventure: 69342
sents from romance: [['I'm', 'not', 'giving', 'you', 'a', 'chance',
',', 'Bill', ',', 'but', 'availing', 'myself', 'of', 'your', 'generou
s', 'offer', 'of', 'assistance', '.'], ['Good', 'luck', 'to', 'you',
'', '.'], ['`', 'All', 'the', 'in-laws', 'have', 'got', 'to', 'hav
e', 'their', 'day', '', 'Adam', 'said', ',', 'and', 'glared',
'at', 'William', 'and', 'Freddy', 'in', 'turn', '.'], ['Sweat', 'start
ed', 'out', 'on', "William's", 'forehead', ',', 'whether', 'from', 're
lief', 'or', 'disquietude', 'he', 'could', 'not', 'tell', '.'], ['Acro
ss', 'the', 'table', ',', 'Hamrick', 'saluted', 'him', 'jubilantly',
'with', 'an', 'encircled', 'thumb', 'and', 'forefinger', '.'], ['Nobod
y', 'else', 'showed', 'pleasure', '.'], ['Spike-haired', ',', 'burly',
',', 'red-faced', ',', 'decked', 'with', 'horn-rimmed', 'glasses', 'an
d', 'an', 'Ivy', 'League', 'suit', ',', 'Jack', 'Hamrick', 'awaited',
'William', 'at', 'the', "officers'", 'club', '.'], ['`', 'Hello',
',', 'boss', '', 'he', 'said', ',', 'and', 'grinned', '.'], ['`
', 'I', 'suppose', 'I', 'can', 'never', 'expect', 'to', 'call', 'yo
u', '', 'General', '', 'after', 'that', 'Washington', 'episode',
'', '.'], ['`', 'I'm', 'afraid', 'not', '', '.']]
All Sentences: 57340
```

NLTK also includes Web and chat texts from Firefox discussion forum, the movie script from Pirates of the Caribbean, wine reviews, ...

To import the Web and chat corpus:

```
from nltk.corpus import webtext
```

In [7]:

```
1 from nltk.corpus import webtext
2
3 for id in webtext.fileids():
4     #display some parts of the text in the document
5     print(id, webtext.raw(id)[:30], "...")
```

```
firefox.txt Cookie Manager: "Don't allow s ...
grail.txt SCENE 1: [wind] [clap clap clo ...
overheard.txt White guy: So, do you have any ...
pirates.txt PIRATES OF THE CARRIBEAN: DEAD ...
singles.txt 25 SEXY MALE, seeks attrac old ...
wine.txt Lovely delicate, fragrant Rhon ...
```

NLTK also include inaugural speeches of US presidents.

You can import these corpora as follows

```
from nltk.corpus import inaugural
```

In [8]:

```
1 from nltk.corpus import inaugural
2 # print all inaugural
3 print("Inaugral Texts:", inaugural.fileids())
4 inaugural.raw("2017-Trump.txt")[:1000]
```

```
Inaugral Texts: ['1789-Washington.txt', '1793-Washington.txt', '1797-A
dams.txt', '1801-Jefferson.txt', '1805-Jefferson.txt', '1809-Madison.t
xt', '1813-Madison.txt', '1817-Monroe.txt', '1821-Monroe.txt', '1825-A
dams.txt', '1829-Jackson.txt', '1833-Jackson.txt', '1837-VanBuren.tx
t', '1841-Harrison.txt', '1845-Polk.txt', '1849-Taylor.txt', '1853-Pie
rce.txt', '1857-Buchanan.txt', '1861-Lincoln.txt', '1865-Lincoln.txt',
'1869-Grant.txt', '1873-Grant.txt', '1877-Hayes.txt', '1881-Garfield.t
xt', '1885-Cleveland.txt', '1889-Harrison.txt', '1893-Cleveland.txt',
'1897-McKinley.txt', '1901-McKinley.txt', '1905-Roosevelt.txt', '1909-
Taft.txt', '1913-Wilson.txt', '1917-Wilson.txt', '1921-Harding.txt',
'1925-Coolidge.txt', '1929-Hoover.txt', '1933-Roosevelt.txt', '1937-Ro
osevelt.txt', '1941-Roosevelt.txt', '1945-Roosevelt.txt', '1949-Truma
n.txt', '1953-Eisenhower.txt', '1957-Eisenhower.txt', '1961-Kennedy.tx
t', '1965-Johnson.txt', '1969-Nixon.txt', '1973-Nixon.txt', '1977-Cart
er.txt', '1981-Reagan.txt', '1985-Reagan.txt', '1989-Bush.txt', '1993-
Clinton.txt', '1997-Clinton.txt', '2001-Bush.txt', '2005-Bush.txt', '2
009-Obama.txt', '2013-Obama.txt', '2017-Trump.txt']
```

Out[8]:

```
"Chief Justice Roberts, President Carter, President Clinton, President
Bush, President Obama, fellow Americans, and people of the world: Than
k you.\n\nWe, the citizens of America, are now joined in a great natio
nal effort to rebuild our country and restore its promise for all of o
ur people. Together, we will determine the course of America and the w
orld for many, many years to come. We will face challenges, we will co
nfront hardships, but we will get the job done.\n\nEvery 4 years, we g
ather on these steps to carry out the orderly and peaceful transfer of
power, and we are grateful to President Obama and First Lady Michelle
Obama for their gracious aid throughout this transition. They have bee
n magnificent. Thank you.\n\nToday's ceremony, however, has very speci
al meaning. Because today we are not merely transferring power from on
e administration to another or from one party to another, but we are t
ransferring power from Washington, DC, and giving it back to you, the
people.\n\nFor too long, a "
```

**NLTK contains tagged and parsed corpora that we will discuss in the upcoming practice classes**

In [9]:

```
1 import nltk
2 nltk.download('maxent_ne_chunker')
3 nltk.download('averaged_perceptron_tagger')
4 nltk.download('words')
5
6 texts = [
7     """
8     ONCE UPON A TIME a girl named Cinderella lived with her stepmother and two st
9     Poor Cinderella had to work hard all day long so the others could rest.
10    It was she who had to wake up each morning when it was still dark and cold to
11    It was she who cooked the meals. It was she who kept the fire going.
12    The poor girl could not stay clean, from all the ashes and cinders by the fir
13    """
14 ]
15 for text in texts:
16     sentences = nltk.sent_tokenize(text)
17     for sentence in sentences:
18         words = nltk.word_tokenize(sentence)
19         tagged_words = nltk.pos_tag(words)
20         ne_tagged_words = nltk.ne_chunk(tagged_words)
21         print (ne_tagged_words)
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data] /Users/seidmuhieyimam/nltk_data...
[nltk_data] Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] /Users/seidmuhieyimam/nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data] date!
[nltk_data] Downloading package words to
[nltk_data] /Users/seidmuhieyimam/nltk_data...
[nltk_data] Package words is already up-to-date!
```

```
(S
  ONCE/RB
  UPON/IN
  A/NNP
  TIME/NNP
  a/DT
  girl/NN
  named/VBN
  (PERSON Cinderella/NNP)
  lived/VBD
  with/IN
```



her/PRP\$  
stepmother/NN  
and/CC  
two/CD  
stepsisters/NNS  
./.)  
(S  
  (PERSON Poor/NNP)  
  (PERSON Cinderella/NNP)  
  had/VBD  
  to/TO  
  work/VB  
  hard/RB  
  all/DT  
  day/NN  
  long/RB  
  so/RB  
  the/DT  
  others/NNS  
  could/MD  
  rest/VB  
  ./.)  
(S  
  It/PRP  
  was/VBD  
  she/PRP  
  who/WP  
  had/VBD  
  to/TO  
  wake/VB  
  up/RP  
  each/DT  
  morning/NN  
  when/WRB  
  it/PRP  
  was/VBD  
  still/RB  
  dark/JJ  
  and/CC  
  cold/JJ  
  to/TO  
  start/VB  
  the/DT  
  fire/NN  
  ./.)  
(S It/PRP was/VBD she/PRP who/WP cooked/VBD the/DT meals/NNS ./.)  
(S  
  It/PRP  
  was/VBD  
  she/PRP  
  who/WP  
  kept/VBD  
  the/DT  
  fire/NN  
  going/VBG  
  ./.)  
(S  
  The/DT  
  poor/JJ  
  girl/NN  
  could/MD

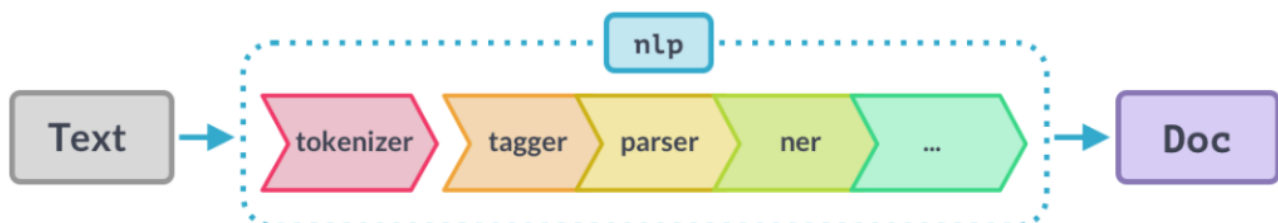
```
not/RB
stay/VB
clean/JJ
,/,
from/IN
all/PDT
the/DT
ashes/NNS
and/CC
cinders/NNS
by/IN
the/DT
fire/NN
./.)
```

## ~ Hands on Session ~

## Experiment with NLTK 15 minutes

### spaCy

[spaCy](#) is a free, open-source library for advanced Natural Language Processing (NLP) in Python. It is a package, with a support for a number of [nlp](#) tasks such as tokenization, lemmatisation, part-of-speech tagging, entity recognition and so on.



### How to Install spaCy

To install spaCy, visit the documentation page [here \(https://spacy.io/usage\)](https://spacy.io/usage).

## Using pip

```
pip install -U spacy
```

**If you want to install it from the notebook, append the `!` before the command**

```
!pip install -U spacy
```

To install some models, run the following from the command line (make sure to use the correct python version, that is python3)

```
python3 -m spacy download en_core_web_sm
```

or

```
python3 -m spacy download en
```

`en_core_web_sm`: (11 MB) English multi-task CNN trained on [OntoNotes](#). Assigns context-specific token vectors, POS tags, dependency parse and named entities.

`en_core_web_md`: (91 MB) English multi-task CNN trained on [OntoNotes](#), with GloVe vectors trained on Common Crawl. Assigns word vectors, context-specific token vectors, POS tags, dependency parse and named entities.

`en_core_web_lg`: (789 MB) English multi-task CNN trained on [OntoNotes](#), with GloVe vectors trained on Common Crawl. Assigns word vectors, context-specific token vectors, POS tags, dependency parse and named entities.

In [10]:

```
1 !pip install -U spacy
```

```
Requirement already satisfied: spacy in /opt/anaconda3/lib/python3.8/s
ite-packages (3.1.3)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /opt/anaconda3/l
ib/python3.8/site-packages (from spacy) (4.59.0)
Requirement already satisfied: pathy>=0.3.5 in /opt/anaconda3/lib/pyth
on3.8/site-packages (from spacy) (0.6.0)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /opt/anacond
a3/lib/python3.8/site-packages (from spacy) (2.0.6)
Requirement already satisfied: Jinja2 in /opt/anaconda3/lib/python3.8/
site-packages (from spacy) (2.11.3)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /opt/anaco
nda3/lib/python3.8/site-packages (from spacy) (1.0.5)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /opt/anaconda3/li
b/python3.8/site-packages (from spacy) (0.7.4)
Requirement already satisfied: numpy>=1.15.0 in /opt/anaconda3/lib/pyt
hon3.8/site-packages (from spacy) (1.20.1)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /opt/anaconda3/l
ib/python3.8/site-packages (from spacy) (2.0.5)
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in /opt/anaconda3/l
ib/python3.8/site-packages (from spacy) (2.4.1)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/p
ython3.8/site-packages (from spacy) (20.9)
Requirement already satisfied: thinc<8.1.0,>=8.0.9 in /opt/anaconda3/l
ib/python3.8/site-packages (from spacy) (8.0.10)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.8 in /opt/anac
onda3/lib/python3.8/site-packages (from spacy) (3.0.8)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /opt/anaconda3/
lib/python3.8/site-packages (from spacy) (0.8.2)
Requirement already satisfied: pydantic!=1.8,!1.8.1,<1.9.0,>=1.7.4 in
/opt/anaconda3/lib/python3.8/site-packages (from spacy) (1.8.2)
Requirement already satisfied: typer<0.5.0,>=0.3.0 in /opt/anaconda3/l
ib/python3.8/site-packages (from spacy) (0.4.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /opt/anacond
a3/lib/python3.8/site-packages (from spacy) (2.25.1)
Requirement already satisfied: setuptools in /opt/anaconda3/lib/python
3.8/site-packages (from spacy) (52.0.0.post20210125)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /opt/anaconda
3/lib/python3.8/site-packages (from spacy) (3.0.5)
Requirement already satisfied: pyparsing>=2.0.2 in /opt/anaconda3/lib/
python3.8/site-packages (from packaging>=20.0->spacy) (2.4.7)
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /opt/anacod
a3/lib/python3.8/site-packages (from pathy>=0.3.5->spacy) (5.2.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /opt/anac
onda3/lib/python3.8/site-packages (from pydantic!=1.8,!1.8.1,<1.9.0,>
=1.7.4->spacy) (3.7.4.3)
Requirement already satisfied: idna<3,>=2.5 in /opt/anaconda3/lib/pyth
on3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/li
b/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy) (2020.
12.5)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/anaconda3/li
b/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy) (4.0.
0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/anaconda
```

```
3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy)
(1.26.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /opt/anaconda3/lib/python3.8/site-packages (from typer<0.5.0,>=0.3.0->spacy) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /opt/anaconda3/lib/python3.8/site-packages (from jinja2->spacy) (1.1.1)
```

In [11]:

```
1 !python3 -m spacy download en_core_web_sm
```

Collecting en-core-web-sm==3.1.0

Downloading [https://github.com/explosion/spacy-models/releases/download/en\\_core\\_web\\_sm-3.1.0/en\\_core\\_web\\_sm-3.1.0-py3-none-any.whl](https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.1.0/en_core_web_sm-3.1.0-py3-none-any.whl) ([https://github.com/explosion/spacy-models/releases/download/en\\_core\\_web\\_sm-3.1.0/en\\_core\\_web\\_sm-3.1.0-py3-none-any.whl](https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.1.0/en_core_web_sm-3.1.0-py3-none-any.whl)) (13.6 MB)

|██| 13.6 MB 5.9 MB/s eta 0:00:01  
Requirement already satisfied: spacy<3.2.0,>=3.1.0 in /opt/anaconda3/lib/python3.8/site-packages (from en-core-web-sm==3.1.0) (3.1.3)  
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.0.5)  
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.8 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (3.0.8)  
Requirement already satisfied: Jinja2 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.11.3)  
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.0.5)  
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.8.2)  
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.4.1)  
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (4.59.0)  
Requirement already satisfied: numpy>=1.15.0 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.20.1)  
Requirement already satisfied: Pydantic!=1.8,!1.8.1,<1.9.0,>=1.7.4 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.8.2)  
Requirement already satisfied: setuptools in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (52.0.0.post20210125)  
Requirement already satisfied: thinc<8.1.0,>=8.0.9 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (8.0.10)  
Requirement already satisfied: pathy>=0.3.5 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.6.0)  
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (20.9)  
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.0.6)  
Requirement already satisfied: typer<0.5.0,>=0.3.0 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.4.0)

Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (3.0.5)

Requirement already satisfied: requests<3.0.0,>=2.13.0 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.25.1)

Requirement already satisfied: blis<0.8.0,>=0.4.0 in /opt/anaconda3/lib/python3.8/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (0.7.4)

Requirement already satisfied: pyparsing>=2.0.2 in /opt/anaconda3/lib/python3.8/site-packages (from packaging>=20.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.4.7)

Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /opt/anaconda3/lib/python3.8/site-packages (from pathy>=0.3.5->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (5.2.1)

Requirement already satisfied: typing-extensions>=3.7.4.3 in /opt/anaconda3/lib/python3.8/site-packages (from pydantic!=1.8,!1.8.1,<1.9.0,>=1.7.4->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (3.7.4.3)

Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2020.12.5)

Requirement already satisfied: idna<3,>=2.5 in /opt/anaconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (2.10)

Requirement already satisfied: chardet<5,>=3.0.2 in /opt/anaconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (4.0.0)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/anaconda3/lib/python3.8/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.26.4)

Requirement already satisfied: click<9.0.0,>=7.1.1 in /opt/anaconda3/lib/python3.8/site-packages (from typer<0.5.0,>=0.3.0->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (7.1.2)

Requirement already satisfied: MarkupSafe>=0.23 in /opt/anaconda3/lib/python3.8/site-packages (from jinja2->spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0) (1.1.1)

✓ Download and installation successful

You can now load the package via `spacy.load('en_core_web_sm')`

In [12]:

```
1 # start spaCy's pipeline, which can load different spaCy models
2 import spacy
3 nlp = spacy.load("en_core_web_sm")
4 doc = nlp("The big grey dog ate all of the chocolate, but fortunately the he was sick!")
5 #Tokenization
6 print("Using split", doc.text.split())
7 #Using spaCy, only orth_ method
8 print("Using spaCy", [token.orth_ for token in doc])
9 # using spacy, including the integer representation of tokens
10 print("spaCy integer", [(token, token.orth_, token.orth) for token in doc] )
```

```
Using split ['The', 'big', 'grey', 'dog', 'ate', 'all', 'of', 'the', 'chocolate,', 'but', 'fortunately', 'the', 'he', 'wasn't', 'sick!']
Using spaCy ['The', 'big', 'grey', 'dog', 'ate', 'all', 'of', 'the', 'chocolate', ',', 'but', 'fortunately', 'the', 'he', 'was', 'n't', 'sick', '!']
spaCy integer [(The, 'The', 5059648917813135842), (big, 'big', 15511632813958231649), (grey, 'grey', 10475807793332549289), (dog, 'dog', 7562983679033046312), (ate, 'ate', 10806788082624814911), (all, 'all', 13409319323822384369), (of, 'of', 886050111519832510), (the, 'the', 7425985699627899538), (chocolate, 'chocolate', 10946593968795032542), (,, ', ', 2593208677638477497), (but, 'but', 14560795576765492085), (fortunately, 'fortunately', 13851269277375979931), (the, 'the', 7425985699627899538), (he, 'he', 1655312771067108281), (was, 'was', 9921686513378912864), (n't, 'n't', 2043519015752540944), (sick, 'sick', 14841597609857081305), (!, '!', 17494803046312582752)]
```

# Important functions for language processing

## 1. Referencing, Equality, Conditionals, Sequences, Operations on sequences, Rearranging, Zipping, enumerating



In [13]:

```
1 a = 'English'
2 b = a
3 a = 'French'
4 # b is Still English
5 print (b)
6 print("----")
```

English

----

In [14]:

```
1 a = ["English", "Spanish"]
2 b = a
3 a[0] = "French"
4 # Now b is changed ==> access by reference
5 print (b)
6 print("----")
```

['French', 'Spanish']

----

In [15]:

```
1 # the same reference for all 5 entries
2 a = ["German"] * 5
3 print ([a[i] for i, _ in enumerate(a)])
4 # Are they the same? use the 'is' keyword
5 print (a[0] is a[1] is a[2] is a[3] is a[4])
6 print("----")
```

['German', 'German', 'German', 'German', 'German']

True

----

In [16]:

```
1 # change value of one of the entry (the forth item)
2 a[3] = "French"
3 print ([a[i] for i,_ in enumerate(a)])
4 print (a[0] is a[1] is a[2] is a[3] is a[4])
5 print("----")
```

```
['German', 'German', 'German', 'French', 'German']
False
---
```

In [17]:

```
1 # conditioning
2 words = ["the", "quick", "brown", "fox", ",", "jumps", "over", "the", "lazy", "dog"]
3 punct = "!.,;"
4 for word in words:
5     if word in punct:
6         print (word + "is a punctuation", end="; ")# print single line until end of line
7
```

```
,is a punctuation; !is a punctuation;
```

In [18]:

```
1 # Zip (concatenate lists). Example words, frequencies, POS Tag
2 print()
3 print("----")
4 words = ['I', 'turned', 'off', 'the', 'spectroroute']
5 freq = [15, 3, 6, 23, 1]
6 tags = ['noun', 'verb', 'prep', 'det', 'noun']
7 print ("Zipped:", list(zip(words, tags, freq)))
8 print("----")
```

```
---
Zipped: [('I', 'noun', 15), ('turned', 'verb', 3), ('off', 'prep', 6),
('the', 'det', 23), ('spectroroute', 'noun', 1)]
---
```

In [19]:

```
1 # Sorting, reversing, differences
2 words_on = ['I', 'turned', 'on', 'the', 'light']
3 print ("sorted:", sorted(words))
4 print ("reversed:", list(reversed(words)))
5 print ("difference:", set(words).difference(words_on))
6 print ("difference:", set(words_on).difference(words))
7 print ("---")
```

```
sorted: ['I', 'off', 'spectroroute', 'the', 'turned']
reversed: ['spectroroute', 'the', 'off', 'turned', 'I']
difference: {'off', 'spectroroute'}
difference: {'light', 'on'}
---
```

In [20]:

```
1 # rearranging lists
2 words_on[2], words_on[3], words_on[4] = words_on[3], words_on[4], words_on[2]
3 print ("rearranged:", words_on)
```

```
rearranged: ['I', 'turned', 'the', 'light', 'on']
```

## 2. Splitting datasets:

We will discuss in upcoming practice class on how to properly split datasets into training, development, and test set. There are frameworks that help splitting and shuffling.

In [21]:

```
1 text = """
2 This is a simple training example.
3 We split by the text into training and test sets.
4 Most of the data goes to training.
5 Small parts for testing
6 """
7 split = int(0.9 * len(text.split()))
8 train, test = text.split()[:split], text.split()[split:]
9 print ("train set",train)
10 print("test set", test)
```

```
train set ['This', 'is', 'a', 'simple', 'training', 'example.', 'We',
'split', 'by', 'the', 'text', 'into', 'training', 'and', 'test', 'set
s.', 'Most', 'of', 'the', 'data', 'goes', 'to', 'training.', 'Small']
test set ['parts', 'for', 'testing']
```

### 3. Generator

Assume you are searching a given word in a corpus

```
def search1(substring, words):
    result = []
    for word in words:
        if substring in word:
            result.append(word)
    return result
```

Here in `search1`, the list is `accumulated` and the final result is returned once the search is over. What if the file is `very large`...?

```
def search2(substring, words):  
    for word in words:  
        if substring in word:  
            yield word
```

Here, `search2` uses a `yield` statement, which is a `generator`. It will pause when it get the first word and the caller function will do whatever with the search word before search continues.

In [22]:

```
1 import nltk
2 def search1(substring, words):
3     result = []
4     for word in words:
5         if substring in word:
6             result.append(word)
7     return result
8
9 def search2(substring, words):
10     for word in words:
11         if substring in word:
12             yield word
13
14
15 for item in search1('zzo', nltk.corpus.brown.words()):
16     print(item, end=" ")
17 print("\n----")
18
19 for item in search2('zzo', nltk.corpus.brown.words()):
20     print(item, end=" ")
```

Palazzo Palazzo Palazzo Palazzo palazzo palazzo Palazzo Palazzo Palazz  
o palazzo Palazzo palazzo Piazzo mezzo palazzos  
----  
Palazzo Palazzo Palazzo Palazzo palazzo palazzo Palazzo Palazzo Palazz  
o palazzo Palazzo palazzo Piazzo mezzo palazzos

## 4. Procedural and declarative style

Example, count the average length of a token

In [23]:

```
1 #Procedural - average word length
2 tokens = nltk.corpus.brown.words(categories='news')
3 count = 0
4 total = 0
5 for token in tokens:
6     count += 1
7     total += len(token)
8 print(total / count)
9
```

4.401545438271973

In [24]:

```
1 #Declarative - average word length
2 tokens = nltk.corpus.brown.words(categories='news')
3 print (sum(len(t) for t in tokens)/len(tokens))
```

4.401545438271973

In [25]:

```
1 # Procedural - longest word
2 text = nltk.corpus.gutenberg.words('milton-paradise.txt')
3 longest = ''
4 for word in text:
5     if len(word) > len(longest):
6         longest = word
7 print(longest) # Print the longest word (ONLY one).
```

unextinguishable

In [26]:

```
1 # Procedural - longest word
2 maxlen = max(len(word) for word in text)# find longest word length
3 print([word for word in text if len(word) == maxlen]) # print those words
```

['unextinguishable', 'transubstantiate', 'inextinguishable', 'incomprehensible']

## 5. Using the **map** functions

In [27]:

```
1 # maps - applies a function to every item in a sequence
2 # Example - average lengths of a sentence in the news Brown corpus
3 lengths = list(map(len, nltk.corpus.brown.sents(categories='news')))
4 print (sum(lengths) / len(lengths))
5 # the same as the following
6 lengths = [len(sent) for sent in nltk.corpus.brown.sents(categories='news')]
7 print(sum(lengths) / len(lengths))
8
```

21.75081116158339

21.75081116158339

## 6. Using **lambda** functions in maps

In [28]:

```
1 # count the number of vowels in each word
2 sent = ['Take', 'care', 'of', 'the', 'sense', ',', 'and', 'the', 'sounds', 'will']
3 list(map(lambda w: len(list(filter(lambda c: c.lower() in "aeiou", w))), sent))
```

Out[28]:

[2, 2, 1, 1, 2, 0, 1, 1, 2, 1, 2, 2, 1, 3, 0]

## 7. Named arguments

When there are a lot of parameters for a function, it is possible to refer them by name, even using default values.



In [29]:

```
1 def pos_tag(word, tag="UNK"):
2     return {word: tag}
3 print(pos_tag("man", "NN"))
4 print(pos_tag("mann"))
5 # *args, **kwargs ==> variable number of unnamed and named/keywords arguments
6 def freq_words(file, min=1, num=10):
7     text = open(file).read()
8     tokens = nltk.word_tokenize(text)
9     freql = nltk.FreqDist(t for t in tokens if len(t) >= min)
10    return list(freql.keys())[:num]
11 print(freq_words('data/news/news1.txt', 14, 5))
12 print(freq_words('data/news/news1.txt', min=14, num=5))
13 print(freq_words('data/news/news1.txt', num=5, min=14))
```

```
{'man': 'NN'}
{'mann': 'UNK'}
['ethnic-majority', 'characteristic', 'underdeveloped', 'Party-approve
d', 'popularization']
['ethnic-majority', 'characteristic', 'underdeveloped', 'Party-approve
d', 'popularization']
['ethnic-majority', 'characteristic', 'underdeveloped', 'Party-approve
d', 'popularization']
```

## 8. Matplotlib

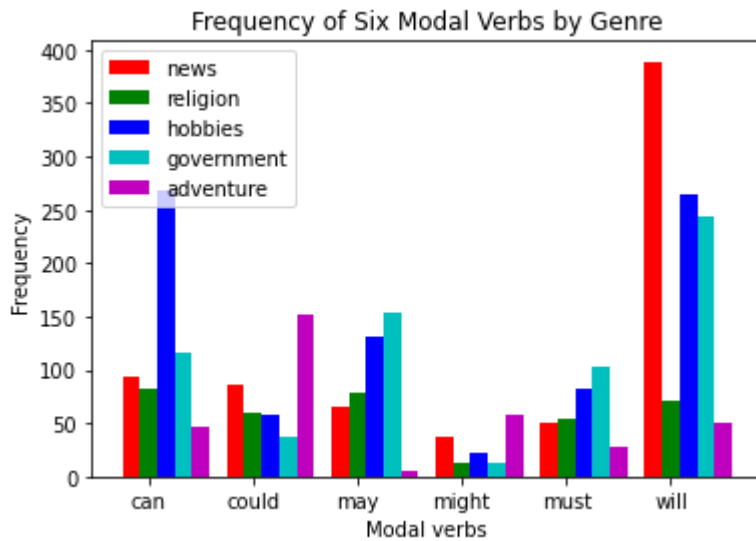
[Matplotlib](#) is a Python 2D plotting library which produces high quality figures in a variety of formats and interactive environments across platforms. It can be used in [Python scripts](#), the Python and IPython [shells](#), the [Jupyter notebook](#), web [application servers](#).

In [30]:

```
1 # Draw a barchart for the 6 modal verbs from the brown corpus
2 from numpy import arange
3 from matplotlib import pyplot
4 %matplotlib inline
5 colors = 'rgbcmk' # red, green, blue, cyan, magenta, yellow, black
6 def bar_chart(categories, words, counts):
7     "Plot a bar chart showing counts for each word by category"
8     ind = arange(len(words))
9     width = 1 / (len(categories) + 1)
10    bar_groups = []
11    for c in range(len(categories)):
12        bars = pyplot.bar(ind+c*width, counts[categories[c]], width,
13                          color=colors[c % len(colors)])
14        bar_groups.append(bars)
15    pyplot.xticks(ind+width, words)
16    pyplot.legend([b[0] for b in bar_groups], categories, loc='upper left')
17    pyplot.ylabel('Frequency');
18    pyplot.xlabel('Modal verbs');
19    pyplot.title('Frequency of Six Modal Verbs by Genre');
20    pyplot.show();
21 genres = ['news', 'religion', 'hobbies', 'government', 'adventure']
22 modals = ['can', 'could', 'may', 'might', 'must', 'will']
23 cfdist = nltk.ConditionalFreqDist(
24     (genre, word)
25     for genre in genres
26     for word in nltk.corpus.brown.words(categories=genre)
27     if word in modals)
28 counts = {}
29 for genre in genres:
30     counts[genre] = [cfdist[genre][word] for word in modals]
31 print(genres, modals, counts)
32 bar_chart(genres, modals, counts)
```

```
['news', 'religion', 'hobbies', 'government', 'adventure'] ['can', 'c
ould', 'may', 'might', 'must', 'will'] {'news': [93, 86, 66, 38, 50,
389], 'religion': [82, 59, 78, 12, 54, 71], 'hobbies': [268, 58, 131,
```

22, 83, 264], 'government': [117, 38, 153, 13, 102, 244], 'adventure': [46, 151, 5, 58, 27, 50]}



---

~ **Hands on Session** ~

**Experimenting with installing Spacy and important functions 10 minutes**

---

---

**Exercise 1 ( 5 points)**

# Submit befor the practice class of next week

1. Count the number of tokens and types in `data/news/news.txt` . Use the `split()` method to split the text with white space
  2. Count and show the words that are common between `data/news/news.txt` and `data/news/news1.txt` .
  3. Print the largest word (s) for each letter (a,b,c ...z) from the nltk books corpus using declarative style. Example ... i->  
`incomprehensible,inextinguishable`; t->`transubstantiate`, ...
  4. Print the most frequent starting and ending characters of words in the nltk books.
  5. Design an algorithm to find the `statistically improbable phrases` (SIP) of a document collection. You might just come up with a `Pseudocode` . SIPs are phrases that are less probably in a `universal` collection but common to a particular topic. For example, `dependency parsing` might be a common phrase that occur in computational linguistic journals but less frequent when compared with a global science journal indexes.
- 
- 

**my name**

**solution 1**

In [ ]:

1	
---	--

In [ ]:

1

## Materials

- [spaCy \(https://course.spacy.io/\)](https://course.spacy.io/)
- [NLTK book \(http://www.nltk.org/book/\)](http://www.nltk.org/book/)
- [Handbook of NLP](https://karczmarchuk.users.greyc.fr/TEACH/TAL/Doc/Handbook%20Of%20Natural%20Language%20Processing.pdf)  
(<https://karczmarchuk.users.greyc.fr/TEACH/TAL/Doc/Handbook%20Of%20Natural%20Language%20Processing.pdf>)