In [2]:

```
%%html
<style>
table {display: block;}
td {
  font-size: 18px
}
.rendered_html { font-size: 18px; }
*{ line-height: 200%; }
</style>
```

Natural Language Processing and the Web WS 2022/23 - Practice Class - Tutorial 2

Content

In this practice class, we will discuss the following main points

- How to read/write text data from/to a file system and web page
- Text Segmentation, lemmatization, stemming
- Part of speech tagging using NLTK

To run this notebook, login to Moodle, click nlp4web tutorial2. This will download ch2.zip that you have to unzip to your preferred location. Start Anaconda navigation, open Jupyter notebook and navigate to the ch2 folder you have unzipped and open ch2.ipynb file

How to read/read data from a file system

To open an existing file, you can do it as follows

```
file = open("filename", "r")
```

You can then get the content as

```
file.read()
```

If you want to handle some special characters encodings, you need to open the file with the given encodings. For example, to open the file with UTF-8 encoding

```
import io
file = io.open("filename", mode="r", encoding=
"utf-8")
```

If you want to read the file line by line, you can do it as follows

```
file = open("filename", "r")
for line in file:
    print (line)
```

To read multiple files from a folder or sub folders, you can do it as follows

```
import os
files = os.lisdir("foldername")
```

Or using os.scandir

```
import os
with os.scandir("foldername") as files
for file in files:
  print(file.name)
```

In [1]:

```
# Install chardet library to detect the encoding of a text
!pip install chardet
```

```
Requirement already satisfied: chardet in /Library/Frameworks/Python.f
ramework/Versions/3.8/lib/python3.8/site-packages (5.0.0)

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: pip install --upgrade pip
```

In [2]:

```
#Checking the encoding of a file
import chardet
def detect_encoding(file):
    with open(file,'rb') as raw_data:
        result = chardet.detect(raw_data.read())
        return(result)
```

In [3]:

```
print(detect_encoding("data/news/news.txt"))
print(detect_encoding("data/news/news4.txt"))
```

```
{'encoding': 'ISO-8859-1', 'confidence': 0.73, 'language': ''}
{'encoding': 'utf-8', 'confidence': 0.99, 'language': ''}
```

In [4]:

```python

# Open file for reading ("r)"
file = open("data/test.txt","r")
# Print all the content
print("All content:", file.read())
file.close()
# Iterate line by line
file = open("data/test.txt","r")
print("Prinitng line by line ...")
for line in file:
   print(line)
file.close()
```

```
All content: This is line one.
And this is line two.
Prinitng line by line ...
This is line one.

And this is line two.
```

In [5]:

```python
import io
file = io.open("data/news/news4.txt", mode="r", encoding="utf-8")
file.read()
```

Out[5]:

'የፋይናንስ ኢንዱስትሪውን በ500 ሚሊዮን ብር የተከፈለ ካፒታል የሚቀላቀለው አማራ ባን
ክ\n21 August 2019\nዳዊት ታዬ\nበፋይናንስ ኢንዱስትሪው ውስጥ 2011 ዓ.ም. በተለ
የ የሚታይበትን ክስተት አስተናግዷል። ይህም የኢትዮጵያ ብሔራዊ ባንክ ለማቋቋም የሚጠይ
ቀውን የተከፈለ ካፒታል መጠን ከ100 ሚሊዮን ብር ወደ 500 ሚሊዮን ብር ካሳደገ ወዲህ፣
ላለፉት ሰባት ዓመታት አንድም ባንክ አሳይቋቋም ቆይቶ ዘንድሮ ግን ሰባት ያህል ባንኮች ለ
ምሥረታ መዘጋጀታቸው ነው። እነዚህ ባንኮች በአዲሱ ዓመት ወደ ሥራ እንደሚገቡ በማስታ
ወቅ የአክሲዮን ሽያጭ ውስጥ ገብተዋል። በዕቅዳቸው መሠረት ከተጓዙ በቀጣዩ ዓመት የአገ
ሪቱ ባንኮች ቁጥር ወደ 25 ያድጋል። እስከ ቀጣዩ ዓመት አጋማሽ ድረስ ወደ ሥራ ለመግባት
ካቀዱትና በወራት ውስጥ የአክሲዮን ሽያጫቸውን ጨርሰው የባንክ ኢንዱስትሪውን እንደሚ
ቀላቀሉ ካስታወቁት ውስጥ አንዱ፣ አማራ ባንክ አክሲዮን ማኅበር ነው። በሁለት ቢሊዮን ብ
ር የተፈቀደ ካፒታልና በ500 ሚሊዮን ብር የተከፈለ ካፒታል ወደ ሥራ እገባለሁ ያለውን አማ
ራ ባንክ አደራጅ ኮሚቴውን በሰብሳቢነት የሚመሩት፣ የገቢዎችና ጉምሩክ ባለሥልጣን የቀድ
ሞ ዋና ዳይሬክተር አቶ መላኩ ፈንታ ናቸው። የባንኩ የፕሮጀክት አስተባባሪ በመሆን የተሰየ
ሙት ደግሞ የቀድሞ የልማት ባንክ ምክትል ፕሬዚዳንት፣ ከዚያም የዓባይ ባንክ የመጀመርያ
ዋ ፕሬዚዳንት ወ/ሮ መሰንበት ሽንቁጥ ናቸው። ወ/ሮ መሰንበት የአዲስ አበባ ንግድና ዘር
ፍ ማኅበራት ምክር ቤት ፕሬዚዳንት በመሆንም እያገለገሉ ነው። አማራ ባንክ ቅዳሜ ነሐሴ
11 ቀን 2011 ዓ.ም. የአክሲዮን ሽያጩን በይፋ የጀመረበት ሥነ ሥርዓት ላይ በክብር እንግ

In [6]:

```
import os
folder = "./data/"
with os.scandir(folder) as files:
    for file in files:
        print(file.name)
```

```
.DS_Store
names2.csv
simple-documents
names.csv
news
testout.txt
test.txt
```

Writing to a file

You can write text data to a file as follows:

```
file = open("filename","w")
file.write("content")
file.close()
```

If you want to append to an existing file,

```
file = open("filename", "a")
file.write("content to append")
```

In [8]:

```
# Write to a file "testout.txt", if the file does not exist, create it in the cu
file = open("testout.txt", "w")
file.write("This is the first line\n")
file.close()

# Append content to the "testout.txt" file
file = open("testout.txt", "a")
file.write("This is the second appended line\n")
file.close()

```

How to read data from web page content

The standard requests library can be used to retrieve content from a web page. Lets see some of the methods for the GET HTTP action. A better way of retrieving web content will be discussed later, mainly using different packages.

In [10]:

```
# requests is the standard html library to make HTTP requests
import requests
link1 = "https://en.wikipedia.org/wiki/Natural_language_processing"

# get is an HTTP method to retrieve data from a specified resource
# response is an HTTP "Response" object to inspect the request
response = requests.get(link1)
# HTTP status code, example 200 is an ok status, 404 is not found status code
status = response.status_code
print(status)
```

```
200
```

In [11]:

```
# the information obtained from the resource, a payload in the form of bytes
content = response.content
print (type(content))
print("====")
print (content)
```

```
<class 'bytes'>
====
b'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<h
ead>\n<meta charset="UTF-8"/>\n<title>Natural language processing - W
ikipedia</title>\n<script>document.documentElement.className="client-
js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":
["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dm
y","wgMonthNames":["","January","February","March","April","May","Jun
e","July","August","September","October","November","December"],"wgRe
questId":"612780b3-202e-491b-bd00-172fa9fb7437","wgCSPNonce":false,"w
gCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespa
ceNumber":0,"wgPageName":"Natural_language_processing","wgTitle":"Nat
ural language processing","wgCurRevisionId":1112038569,"wgRevisionI
d":1112038569,"wgArticleId":21652,"wgIsArticle":true,"wgIsRedirect":f
alse,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCate
gories":["All accuracy disputes","Accuracy disputes from December 201
3","CS1 maint: location","Articles with short description","Short des
cription is different from Wikidata","Commons category link from Wiki
data","Articles with J9U identifiers","Articles with LCCN identifier
```

In [12]:

```
# the information obtained from the resource, a payload in the form of string
content = response.text

print (type(content))
print("====")
print (content)
```

```
<class 'str'>
====
<!DOCTYPE html>
<html class="client-nojs" lang="en" dir="ltr">
<head>
<meta charset="UTF-8"/>
<title>Natural language processing - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBre
akFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransform
Table":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","Januar
y","February","March","April","May","June","July","August","Septembe
r","October","November","December"],"wgRequestId":"612780b3-202e-491b
-bd00-172fa9fb7437","wgCSPNonce":false,"wgCanonicalNamespace":"","wgC
anonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"Na
tural_language_processing","wgTitle":"Natural language processing","w
gCurRevisionId":1112038569,"wgRevisionId":1112038569,"wgArticleId":21
652,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUser
Name":null,"wgUserGroups":["*"],"wgCategories":["All accuracy dispute
s","Accuracy disputes from December 2013","CS1 maint: location","Arti
```

In [13]:

```
# Examine which encoding the request determines for the response
encoding = response.encoding
print(encoding)
```

```
UTF-8
```

In [14]:

```
# Spacify the encoding of the response manually
response.encoding = "utf-8"

# Examine all the header information
headers = response.headers
print (headers)
```

```
{'date': 'Tue, 25 Oct 2022 04:56:49 GMT', 'vary': 'Accept-Encoding,Coo
kie,Authorization', 'server': 'ATS/9.1.3', 'x-content-type-options':
'nosniff', 'content-language': 'en', 'last-modified': 'Sun, 23 Oct 202
2 04:18:20 GMT', 'content-type': 'text/html; charset=UTF-8', 'content-
encoding': 'gzip', 'age': '11566', 'x-cache': 'cp3052 hit, cp3056 hit/
10', 'x-cache-status': 'hit-front', 'server-timing': 'cache;desc="hit-
front", host;desc="cp3056"', 'strict-transport-security': 'max-age=106
384710; includeSubDomains; preload', 'report-to': '{ "group": "wm_ne
l", "max_age": 86400, "endpoints": [{ "url": "https://intake-logging.w
ikimedia.org/v1/events?stream=w3c.reportingapi.network_error&schema_ur
i=/w3c/reportingapi/network_error/1.0.0" }] }', 'nel': '{ "report_to":
"wm_nel", "max_age": 86400, "failure_fraction": 0.05, "success_fractio
n": 0.0}', 'set-cookie': 'WMF-Last-Access=25-Oct-2022;Path=/;HttpOnly;
secure;Expires=Sat, 26 Nov 2022 00:00:00 GMT, WMF-Last-Access-Global=2
5-Oct-2022;Path=/;Domain=.wikipedia.org;HttpOnly;secure;Expires=Sat, 2
6 Nov 2022 00:00:00 GMT, GeoIP=DE:HH:Hamburg:53.55:9.99:v4; Path=/; se
cure; Domain=.wikipedia.org', 'accept-ch': 'Sec-CH-UA-Arch,Sec-CH-UA-B
itness,Sec-CH-UA-Full-Version-List,Sec-CH-UA-Model,Sec-CH-UA-Platform-
Version', 'permissions-policy': 'interest-cohort=(),ch-ua-arch=(self
"intake-analytics.wikimedia.org"),ch-ua-bitness=(self "intake-analytic
s.wikimedia.org"),ch-ua-full-version-list=(self "intake-analytics.wiki
media.org"),ch-ua-model=(self "intake-analytics.wikimedia.org"),ch-ua-
platform-version=(self "intake-analytics.wikimedia.org")', 'x-client-i
p': '134.100.15.81', 'cache-control': 'private, s-maxage=0, max-age=0,
must-revalidate', 'accept-ranges': 'bytes', 'content-length': '49334'}
```

In [15]:

```
# determine the type of the content. For example, if you chose link1, it is an h
contentType = response.headers.get('content-type')
print(contentType)
```

```
text/html; charset=UTF-8
```

In [16]:

```
link2 = "https://api.github.com"

# get is an HTTP method to retrieve data from a specified resource
# response is an HTTP "Response" object to inspect the request
response = requests.get(link2)
# HTTP status code, example 200 is an ok status, 404 is not found status code
status = response.status_code
print(status)
```

```
200
```

In [17]:

```
# the information obtained from the resource, a payload in the form of bytes
content = response.content
print (content)
```


```
b'{\n  "current_user_url": "https://api.github.com/user",\n  "current
_user_authorizations_html_url": "https://github.com/settings/connecti
ons/applications{/client_id}",\n  "authorizations_url": "https://api.
github.com/authorizations",\n  "code_search_url": "https://api.githu
b.com/search/code?q={query}{&page,per_page,sort,order}",\n  "commit_s
earch_url": "https://api.github.com/search/commits?q={query}{&page,pe
r_page,sort,order}",\n  "emails_url": "https://api.github.com/user/em
ails",\n  "emojis_url": "https://api.github.com/emojis",\n  "events_u
rl": "https://api.github.com/events",\n  "feeds_url": "https://api.gi
thub.com/feeds",\n  "followers_url": "https://api.github.com/user/fol
lowers",\n  "following_url": "https://api.github.com/user/following{/
target}",\n  "gists_url": "https://api.github.com/gists{/gist_id}",\n
"hub_url": "https://api.github.com/hub",\n  "issue_search_url": "http
s://api.github.com/search/issues?q={query}{&page,per_page,sort,orde
r}",\n  "issues_url": "https://api.github.com/issues",\n  "keys_url":
"https://api.github.com/user/keys",\n  "label_search_url": "https://a
pi.github.com/search/labels?q={query}&repository_id={repository_id}{&
page,per_page}",\n  "notifications_url": "https://api.github.com/noti
fications",\n  "organization_url": "https://api.github.com/orgs/{or
```

In [18]:

```
headers = response.headers
print(headers)
print("====")
contentType = response.headers.get('content-type')
print(contentType)
```

```
{'Server': 'GitHub.com', 'Date': 'Tue, 25 Oct 2022 08:12:04 GMT', 'Cac
he-Control': 'public, max-age=60, s-maxage=60', 'Vary': 'Accept, Accep
t-Encoding, Accept, X-Requested-With', 'ETag': '"4f825cc84e1c733059d46
e76e6df9db557ae5254f9625dfe8e1b09499c449438"', 'Access-Control-Expose-
Headers': 'ETag, Link, Location, Retry-After, X-GitHub-OTP, X-RateLimi
t-Limit, X-RateLimit-Remaining, X-RateLimit-Used, X-RateLimit-Resourc
e, X-RateLimit-Reset, X-OAuth-Scopes, X-Accepted-OAuth-Scopes, X-Poll-
Interval, X-GitHub-Media-Type, X-GitHub-SSO, X-GitHub-Request-Id, Depr
ecation, Sunset', 'Access-Control-Allow-Origin': '*', 'Strict-Transpor
t-Security': 'max-age=31536000; includeSubdomains; preload', 'X-Frame-
Options': 'deny', 'X-Content-Type-Options': 'nosniff', 'X-XSS-Protecti
on': '0', 'Referrer-Policy': 'origin-when-cross-origin, strict-origin-
when-cross-origin', 'Content-Security-Policy': "default-src 'none'",
'Content-Type': 'application/json; charset=utf-8', 'X-GitHub-Media-Typ
e': 'github.v3; format=json', 'Content-Encoding': 'gzip', 'X-RateLimit
-Limit': '60', 'X-RateLimit-Remaining': '59', 'X-RateLimit-Reset': '16
66689126', 'X-RateLimit-Resource': 'core', 'X-RateLimit-Used': '1', 'A
ccept-Ranges': 'bytes', 'Content-Length': '530', 'X-GitHub-Request-I
d': 'E7A2:F493:9B67D5:9DD7F4:63579A56'}
====
application/json; charset=utf-8
```

print a formated content using `pprint`

In [20]:

```
import pprint
json = response.json()
pprint.pprint(json)
```

```
{'authorizations_url': 'https://api.github.com/authorizations',
 'code_search_url': 'https://api.github.com/search/code?q={query}{&pag
e,per_page,sort,order}',
 'commit_search_url': 'https://api.github.com/search/commits?q={query}
{&page,per_page,sort,order}',
 'current_user_authorizations_html_url': 'https://github.com/settings/
connections/applications{/client_id}',
 'current_user_repositories_url': 'https://api.github.com/user/repos{?
type,page,per_page,sort}',
 'current_user_url': 'https://api.github.com/user',
 'emails_url': 'https://api.github.com/user/emails',
 'emojis_url': 'https://api.github.com/emojis',
 'events_url': 'https://api.github.com/events',
 'feeds_url': 'https://api.github.com/feeds',
 'followers_url': 'https://api.github.com/user/followers',
 'following_url': 'https://api.github.com/user/following{/target}',
 'gists_url': 'https://api.github.com/gists{/gist_id}',
 'hub_url': 'https://api.github.com/hub',
 'issue_search_url': 'https://api.github.com/search/issues?q={query}{&
page,per_page,sort,order}',
 'issues_url': 'https://api.github.com/issues',
 'keys_url': 'https://api.github.com/user/keys',
 'label_search_url': 'https://api.github.com/search/labels?q={query}&r
epository_id={repository_id}{&page,per_page}',
 'notifications_url': 'https://api.github.com/notifications',
 'organization_repositories_url': 'https://api.github.com/orgs/{org}/r
epos{?type,page,per_page,sort}',
 'organization_teams_url': 'https://api.github.com/orgs/{org}/teams',
 'organization_url': 'https://api.github.com/orgs/{org}',
 'public_gists_url': 'https://api.github.com/gists/public',
 'rate_limit_url': 'https://api.github.com/rate_limit',
 'repository_search_url': 'https://api.github.com/search/repositories?
q={query}{&page,per_page,sort,order}',
 'repository_url': 'https://api.github.com/repos/{owner}/{repo}',
 'starred_gists_url': 'https://api.github.com/gists/starred',
 'starred_url': 'https://api.github.com/user/starred{/owner}{/repo}',
 'topic_search_url': 'https://api.github.com/search/topics?q={query}{&
page,per_page}',
 'user_organizations_url': 'https://api.github.com/user/orgs',
 'user_repositories_url': 'https://api.github.com/users/{user}/repos{?
type,page,per_page,sort}',
 'user_search_url': 'https://api.github.com/search/users?q={query}{&pa
ge,per_page,sort,order}',
 'user_url': 'https://api.github.com/users/{user}'}
```

File/Webpage processing using BeautifulSoup

library.

This library can be used to process HTML and XML files.

In [21]:

```
# Install bs4 package
!pip install beautifulsoup4
```

```
Requirement already satisfied: beautifulsoup4 in /Library/Frameworks/P
ython.framework/Versions/3.8/lib/python3.8/site-packages (4.8.1)
Requirement already satisfied: soupsieve>=1.2 in /Library/Frameworks/P
ython.framework/Versions/3.8/lib/python3.8/site-packages (from beautif
ulsoup4) (1.9.5)

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: pip install --upgrade pip
```

In [22]:

```
# Reading web content using BeautifulSoup
from bs4 import BeautifulSoup
page = requests.get("https://www.crummy.com/software/BeautifulSoup/")
soup = BeautifulSoup(page.content, 'html.parser')
# Display the content of the page - html
print(soup.prettify())
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/transitional.dtd">
<html>
 <head>
  <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/
>
  <title>
   Beautiful Soup: We called him Tortoise because he taught us.
  </title>
  <link href="mailto:leonardr@segfault.org" rev="made"/>
  <link href="/nb/themes/Default/nb.css" rel="stylesheet" type="text/
css"/>
  <meta content="Beautiful Soup: a library designed for screen-scrapi
ng HTML and XML." name="Description"/>
  <meta content="Markov Approximation 1.4 (module: leonardr)" name="g
enerator"/>
  <meta content="Leonard Richardson" name="author"/>
 </head>
 <body alink="red" bgcolor="white" link="blue" text="black" vlink="66
```

In [23]:

```
# Get only the text content of the page
print(soup.text)
```

```
Beautiful Soup: We called him Tortoise because he taught us.



[ Download | Documentation | Hall of Fame | For enterprise | Source |
Changelog | Discussion group  | Zine ]

Beautiful Soup
```

In [24]:

```
# list all the available elements in the page
[type(item) for item in list(soup.children)]
```

Out[24]:

```
[bs4.element.Doctype,
 bs4.element.NavigableString,
 bs4.element.Tag,
 bs4.element.NavigableString,
 bs4.element.Tag,
 bs4.element.NavigableString,
 bs4.element.Tag,
 bs4.element.Comment,
 bs4.element.Tag,
 bs4.element.NavigableString,
 bs4.element.NavigableString,
 bs4.element.NavigableString,
 bs4.element.NavigableString,
 bs4.element.NavigableString]
```

In [25]:

```
# The third (index 2) element is the HTML tag
print(list(soup.children)[2].name)
# get the sub-tags unde the HTML tag
html = list(soup.children)[2]
list(html.children) # contains the head and body sub tags
```

```
html
```

Out[25]:

```
['\n', <head>
 <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
 <title>Beautiful Soup: We called him Tortoise because he taught us.
</title>
 <link href="mailto:leonardr@segfault.org" rev="made"/>
 <link href="/nb/themes/Default/nb.css" rel="stylesheet" type="text/c
ss"/>
 <meta content="Beautiful Soup: a library designed for screen-scrapin
g HTML and XML." name="Description"/>
 <meta content="Markov Approximation 1.4 (module: leonardr)" name="ge
nerator"/>
 <meta content="Leonard Richardson" name="author"/>
 </head>, '\n', <body alink="red" bgcolor="white" link="blue" text="b
lack" vlink="660066">
 <style>
 #tidelift { }

 #tidelift a {
  border: 1px solid #666666;
  margin-left: auto;
  padding: 10px;
  text-decoration: none;
 }

 #tidelift .cta {
  background: url("tidelift.svg") no-repeat;
  padding-left: 30px;
 }
 </style>
 <img align="right" src="10.1.jpg" width="250"/><br/>
 <p>[ <a href="#Download">Download</a> | <a href="bs4/doc/">Documenta
tion</a> | <a href="#HallOfFame">Hall of Fame</a> | <a href="enterpri
se.html">For enterprise</a> | <a href="https://code.launchpad.net/bea
utifulsoup">Source</a> | <a href="https://bazaar.launchpad.net/%7Eleo
nardr/beautifulsoup/bs4/view/head:/CHANGELOG">Changelog</a> | <a href
="https://groups.google.com/forum/?fromgroups#!forum/beautifulsoup">D
iscussion group</a>  | <a href="zine/">Zine</a> ]</p>
 <div align="center">
 <a href="bs4/download/"><h1>Beautiful Soup</h1></a>
 </div>
 <p>You didn't write that awful page. You're just trying to get some
 data out of it. Beautiful Soup is here to help. Since 2004, it's bee
n
 saving programmers hours or days of work on quick-turnaround
```

```
 screen scraping projects.</p>
 <p>Beautiful Soup is a Python library designed for quick turnaround
 projects like screen-scraping. Three features make it powerful:

 <ol>
 <li>Beautiful Soup provides a few simple methods and Pythonic idioms
 for navigating, searching, and modifying a parse tree: a toolkit for
 dissecting a document and extracting what you need. It doesn't take
 much code to write an application

 <li>Beautiful Soup automatically converts incoming documents to
 Unicode and outgoing documents to UTF-8. You don't have to think
 about encodings, unless the document doesn't specify an encoding and
 Beautiful Soup can't detect one. Then you just have to specify the
 original encoding.

 <li>Beautiful Soup sits on top of popular Python parsers like <a hre
f="http://lxml.de/">lxml</a> and <a href="http://code.google.com/p/ht
ml5lib/">html5lib</a>, allowing you
 to try out different parsing strategies or trade speed for
 flexibility.

 </li></li></li></ol>
 <p>Beautiful Soup parses anything you give it, and does the tree
 traversal stuff for you. You can tell it "Find all the links", or
 "Find all the links of class <tt>externalLink</tt>", or "Find all th
e
 links whose urls match "foo.com", or "Find the table heading that's
 got bold text, then give me that text."

 <p>Valuable data that was once locked up in poorly-designed websites
 is now within your reach. Projects that would have taken hours take
 only minutes with Beautiful Soup.

 <p>Interested? <a href="bs4/doc/">Read more.</a>
 <h3>Getting and giving support</h3>
 <div align="center" id="tidelift">
 <a href="https://tidelift.com/subscription/pkg/pypi-beautifulsoup4?u
tm_source=pypi-beautifulsoup4&amp;utm_medium=referral&amp;utm_campaig
n=enterprise" target="_blank">
 <span class="cta">
   Beautiful Soup for enterprise available via Tidelift
  </span>
 </a>
 </div>
 <p>If you have questions, send them to <a href="https://groups.googl
e.com/forum/?fromgroups#!forum/beautifulsoup">the discussion
 group</a>. If you find a bug, <a href="https://bugs.launchpad.net/be
autifulsoup/">file it on Launchpad</a>. If it's a security vulnerabil
ity, report it confidentially through <a href="https://tidelift.com/s
ecurity">Tidelift</a>.</p>
 <p>If you use Beautiful Soup as part of your work, please consider a
<a href="https://tidelift.com/subscription/pkg/pypi-beautifulsoup4?ut
m_source=pypi-beautifulsoup4&amp;utm_medium=referral&amp;utm_campaign
=website">Tidelift subscription</a>. This will support many of the fr
ee software projects your organization depends on, not just Beautiful
Soup.


 <p>If Beautiful Soup is useful to you on a personal level, you might
like to read <a href="zine/"><i>Tool Safety</i></a>, a short zine I w
```

```
rote about what I learned about software development from working on
Beautiful Soup. Thanks!</p>
 </p></p></p></p></p></body>]
```

In [26]:

```
body = list(html.children)[3]
```

In [27]:

```
# Get all the sub-tags under body
for tag in list(body.children):
    if tag.name:
        print (tag.name)
```

```
style
img
br
p
div
p
p
```

In [28]:

```
# Get all the text under the p tags
for tag in list(body.children):
    if tag.name and tag.name =='p':
        #pass  # uncomment the following line
        print (tag.text)
# OR use findall by the tag name
print("=======USing find all======")
#for p in soup.find_all('p'):
#    print(p.text)
```

```
[ Download | Documentation | Hall of Fame | For enterprise | Source |
Changelog | Discussion group  | Zine ]
You didn't write that awful page. You're just trying to get some
data out of it. Beautiful Soup is here to help. Since 2004, it's been
saving programmers hours or days of work on quick-turnaround
screen scraping projects.
Beautiful Soup is a Python library designed for quick turnaround
projects like screen-scraping. Three features make it powerful:


Beautiful Soup provides a few simple methods and Pythonic idioms
for navigating, searching, and modifying a parse tree: a toolkit for
dissecting a document and extracting what you need. It doesn't take
much code to write an application

Beautiful Soup automatically converts incoming documents to
Unicode and outgoing documents to UTF-8. You don't have to think
about encodings, unless the document doesn't specify an encoding and
Beautiful Soup can't detect one. Then you just have to specify the
```

Reading and writing CSV files.

CSV files are comma delimited files that can be used to store structured files in the form of lists. It might contain headers, and the header as well as individual records are separated by new lines. In general delimiters can be also user defined.

You can use the csv library to process CSV files.

```
import csv
```

In [29]:

```
# Read personal information: name.csv is obtained from https://github.com/CoreyM
import csv
with open('data/names.csv','r') as csv_names:
    name_reader = csv.reader(csv_names,delimiter=',')
    # skip the header
    next(name_reader)
    for name in name_reader:
        print(name)
```

```
['John', 'Doe', 'john-doe@bogusemail.com']
['Mary', 'Smith-Robinson', 'maryjacobs@bogusemail.com']
['Dave', 'Smith', 'davesmith@bogusemail.com']
['Jane', 'Stuart', 'janestuart@bogusemail.com']
['Tom', 'Wright', 'tomwright@bogusemail.com']
['Steve', 'Robinson', 'steverobinson@bogusemail.com']
['Nicole', 'Jacobs', 'nicolejacobs@bogusemail.com']
['Jane', 'Wright', 'janewright@bogusemail.com']
['Jane', 'Doe', 'janedoe@bogusemail.com']
['Kurt', 'Wright', 'kurtwright@bogusemail.com']
['Kurt', 'Robinson', 'kurtrobinson@bogusemail.com']
['Jane', 'Jenkins', 'janejenkins@bogusemail.com']
['Neil', 'Robinson', 'neilrobinson@bogusemail.com']
['Tom', 'Patterson', 'tompatterson@bogusemail.com']
['Sam', 'Jenkins', 'samjenkins@bogusemail.com']
['Steve', 'Stuart', 'stevestuart@bogusemail.com']
['Maggie', 'Patterson', 'maggiepatterson@bogusemail.com']
['Maggie', 'Stuart', 'maggiestuart@bogusemail.com']
['Jane', 'Doe', 'janedoe@bogusemail.com']
['Steve', 'Patterson', 'stevepatterson@bogusemail.com']
['Dave', 'Smith', 'davesmith@bogusemail.com']
['Sam', 'Wilks', 'samwilks@bogusemail.com']
['Kurt', 'Jefferson', 'kurtjefferson@bogusemail.com']
['Sam', 'Stuart', 'samstuart@bogusemail.com']
['Jane', 'Stuart', 'janestuart@bogusemail.com']
['Dave', 'Davis', 'davedavis@bogusemail.com']
['Sam', 'Patterson', 'sampatterson@bogusemail.com']
['Tom', 'Jefferson', 'tomjefferson@bogusemail.com']
['Jane', 'Stuart', 'janestuart@bogusemail.com']
['Maggie', 'Jefferson', 'maggiejefferson@bogusemail.com']
['Mary', 'Wilks', 'marywilks@bogusemail.com']
['Neil', 'Patterson', 'neilpatterson@bogusemail.com']
['Corey', 'Davis', 'coreydavis@bogusemail.com']
['Steve', 'Jacobs', 'stevejacobs@bogusemail.com']
['Jane', 'Jenkins', 'janejenkins@bogusemail.com']
['John', 'Jacobs', 'johnjacobs@bogusemail.com']
['Neil', 'Smith', 'neilsmith@bogusemail.com']
['Corey', 'Wilks', 'coreywilks@bogusemail.com']
['Corey', 'Smith', 'coreysmith@bogusemail.com']
['Mary', 'Patterson', 'marypatterson@bogusemail.com']
['Jane', 'Stuart', 'janestuart@bogusemail.com']
['Travis', 'Arnold', 'travisarnold@bogusemail.com']
```

```
['John', 'Robinson', 'johnrobinson@bogusemail.com']
['Travis', 'Arnold', 'travisarnold@bogusemail.com']
```

In [30]:

```python
# Re-write the csv file with a different delimiter
with open('data/names.csv','r') as csv_names:
    name_reader = csv.reader(csv_names,delimiter=',')
    with open('data/names2.csv','w') as csv_names:
        name_writer = csv.writer(csv_names, delimiter="\t")
        for name in name_reader:
            name_writer.writerow(name)

```

Excercise_1 (3pt)

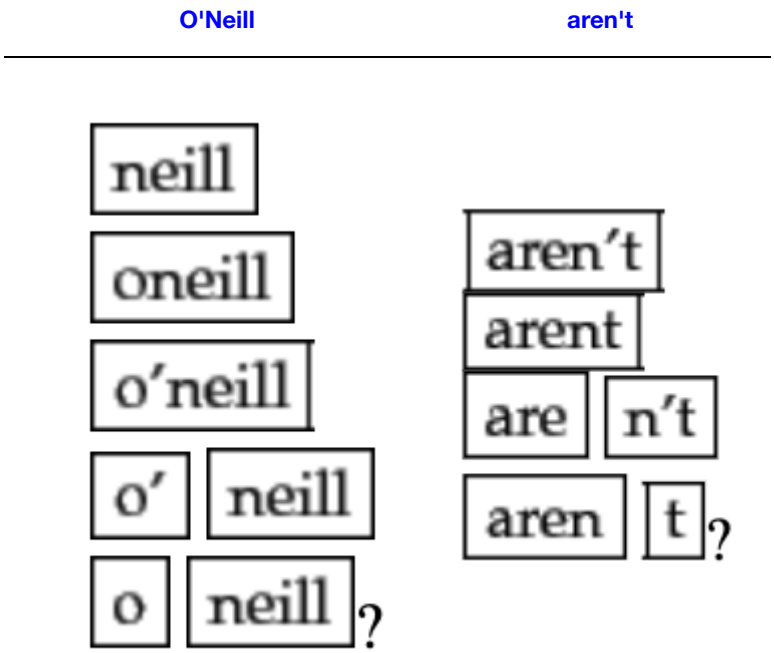
1. List all the files in the subidirectories of the data folder that you have downloaded from Moodle. (0.5 pt)
2. Merge all the news content into a single file and save into a different file named allnews.txt using the ISO-8859-1 encoding. (0.5 pt)
3. Read the country name and capital city from the this (https://geographyfieldwork.com/WorldCapitalCities.htm) page, which lists the world capital cities with their country. Save the result as a comma separated value (csv) file format.(2 pts)

Tokenization

Tokenization is the first step in almost all NLP applications. It is an important prerequisite for a number of downstream NLP tasks. Consider the following example:

```
Mr. O'Neill thinks that the boys' stories abou
t Chile's capital aren't amusing.```
```

For O'Neill and aren't, which of the following are the desired tokenization?



Tokenizing with regex

Sentence segmentation

Lets assume that the text is properly written English documents. We can build a simple segmenter, that will split the sentence by a full stop.

```
m = re.split('(\.)', text)
```

However, if the text contains `.` which is not the end of the sentence by itself, it will wrongly split them.

Better solution is to use lookahead and lookbehind regex patters.

Lookahead mechanism asserts the presence or absence of certain expressions without consuming the pattern itself. Positive lookahead is defined with (?=REGEX) while negative lookahead is defined as (?!REGEX)

Example: Match a `q` not followed by a `u`.

```
q(?!u)
```

Matche a `q` that is followed by a `u`

```
q(?=u)
```

Positive lookabehind is defined with (?<=REGEX) while negative lookabehind is defined as (?<!REGEX)

In [31]:

```
import re
#Split sentence with .
text = """Fruits like apple, orange, and mango are healthy. But they are expensi
($1.5/killo), i.e Mrs. Bean can't afford them! However, One can order some onlir
"""
sents = re.split(r"\.", text) # Add space here and see the effect
for sent in sents:
    print (sent) # What is the problem here?
print("===")
# Find all words that are followed by , or .
# \w+ --> one or more words
#(?=,|\.) --> followed by , or . or !
p = re.compile("\w+(?=,|\.|!)")
print("Lookahead matches", p.findall(text))
```

```
Fruits like apple, orange, and mango are healthy
 But they are expensive
($1
5/killo), i
e Mrs
 Bean can't afford them! However, One can order some online from www
rewe
de


===
Lookahead matches ['apple', 'orange', 'healthy', '1', 'i', 'Mrs', 'the
m', 'However', 'www', 'rewe', 'de']
```

In [32]:

```
# Sentence splitting using lookbehinde and lookahead
print("unsegmented text:", text)
print("===")
# lookahead/lookbehind matchs
# (?<![A-Z][a-z]\.) --> If . is Not behind two characters (first cap, second sma
# (?<=[\.|?|\!])\s+ --> If . is before one or more spaces
# After the space, the first character should be Capitalized
sents = re.split("(?<![A-Z][a-z]\.)(?<=[\.|?|\!])\s+(?=[A-Z])", text)
for sent in sents:
    print (sent)
```

```
unsegmented text: Fruits like apple, orange, and mango are healthy. Bu
t they are expensive
($1.5/killo), i.e Mrs. Bean can't afford them! However, One can order
some online from www.rewe.de.

===
Fruits like apple, orange, and mango are healthy.
But they are expensive
($1.5/killo), i.e Mrs.
Bean can't afford them!
However, One can order some online from www.rewe.de.
```

Excercise_2 (3pt)

Modify the regex above for sentence segmentation so that the following text are split into correct sentences.

```
Fruits like apple, orange, and mango are health. But
they are expensive, i.e Mr. Bean can't afford them!
One can order some online from www.rewe.de. Prof.
Karl, Dep. of Plant Science. Email:
karl@plant.science.de. Regards!
```

Word tokenization with regex

Word tokenization for English languages can be attained using white space characters. However, there are still special cases that need to be addressed.

In [33]:

```
# English tokenization,
tokens = re.split("(\w+|\$[\d\.]+|\S+)", text)
for token in tokens:
    if token.strip():
        print (token) # Can you fix some of the issues?
```

```
Fruits
like
apple
,
orange
,
and
mango
are
healthy
.
But
they
are
expensive
($1.5/killo),
i
.e
Mrs
.
Bean
can
't
afford
them
!
However
,
One
can
order
some
online
from
www
.rewe.de.
```

Excercise_2(3pt)

Modify/re-write the word tokenization pattern given above so that you can achieve near `ideal` tokenization for the following text

```
"I said, 'what're you? Crazy?'" said Sandowsky. "I can't afford to do that."
```

See the ideal tokenization result from the `Exercise_2 - Ideal tokenization - file` in Moodle.

Tokenization with NLTK

NLTK includes tokenizers for different languages

In [34]:

```
import nltk
sentence = "These are simply sentences obtained from the treminals."
tokens = nltk.word_tokenize(sentence, language='english')
print(tokens)
sentence2 = """
Die Brände in Brasilien setzen erhebliche Mengen an klimaschädlichen Treibhausga
Die Nasa hat nun simuliert, wie sich Kohlenmonoxid über Südamerika ausbreitet. A
das Gas der Gesundheit erheblich."""
tokens2 = nltk.word_tokenize(sentence2, language='german')
print (tokens2)
```

```
['These', 'are', 'simply', 'sentences', 'obtained', 'from', 'the', 'tr
eminals', '.']
['Die', 'Brände', 'in', 'Brasilien', 'setzen', 'erhebliche', 'Mengen',
'an', 'klimaschädlichen', 'Treibhausgasen', 'frei', '.', 'Die', 'Nas
a', 'hat', 'nun', 'simuliert', ',', 'wie', 'sich', 'Kohlenmonoxid', 'ü
ber', 'Südamerika', 'ausbreitet', '.', 'Am', 'Boden', 'schadet', 'da
s', 'Gas', 'der', 'Gesundheit', 'erheblich', '.']
```

Stemmer in NLTK

Stemming tries to remove or chops off the end of a word to reduce the word to its base form, somehow a crude approach.

Different stemmers use different rules.

In [35]:

```
porter = nltk.PorterStemmer()
lancaster = nltk.LancasterStemmer()
[porter.stem(token) for token in tokens]

```

Out[35]:

```
['these', 'are', 'simpli', 'sentenc', 'obtain', 'from', 'the', 'tremi
n', '.']
```

In [36]:

```
[lancaster.stem(token) for token in tokens]
```

Out[36]:

```
['thes', 'ar', 'simply', 'sent', 'obtain', 'from', 'the', 'tremin',
'.']
```

In [37]:

```
#Example from: https://www.datacamp.com/community/tutorials/stemming-lemmatizati
#A list of words to be stemmed
word_list = ["friend", "friendship", "friends", "friendships","stabil","destabil
             "railroad","moonlight","football","ill","illness","sick","sickness"
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","lancaster Stemmer"))
for word in word_list:
    print("{0:20}{1:20}{2:20}".format(word,porter.stem(word),lancaster.stem(word
```

```
Word                 Porter Stemmer      lancaster Stemmer
friend               friend              friend
friendship           friendship          friend
friends              friend              friend
friendships          friendship          friend
stabil               stabil              stabl
destabilize          destabil            dest
misunderstanding     misunderstand       misunderstand
railroad             railroad            railroad
moonlight            moonlight           moonlight
football             footbal             footbal
ill                  ill                 il
illness              ill                 il
sick                 sick                sick
sickness             sick                sick
hope                 hope                hop
```

In [38]:

```
#CISTEM Stemmer for German
de_stem = nltk.stem.cistem.Cistem()
#Show the top 6 stems from the list
[de_stem.stem(t) for t in tokens2][:6]
```

Out[38]:

```
['die', 'brand', 'in', 'brasilie', 'setz', 'erheblich']
```

Tokenization spaCy

In [39]:

```
import spacy
spacy = spacy.load('en_core_web_sm')
doc = spacy(text)
print("{0:15}{1:15}".format("Token","Lemma"))
for token in doc:
    # print the token and its lemma
    print("{0:15}{1:15}".format(token.text, token.lemma_)) #How does it compare
```

```
Token          Lemma
Fruits         fruit
like           like
apple          apple
,              ,
orange         orange
,              ,
and            and
mango          mango
are            be
healthy        healthy
.              .
But            but
they           they
are            be
expensive      expensive



(              (
$              $
1.5            1.5
/              /
killo          killo
)              )
,              ,
i.e            i.e
Mrs.           Mrs.
Bean           Bean
ca             ca
n't            n't
afford         afford
them           they
!              !
However        however
,              ,
One            one
can            can
order          order
some           some
online         online
from           from
www.rewe.de    www.rewe.de
.              .
```

In [40]:

```
for sent in doc.sents:
    print (sent) #How does it compare with the sentences in the regex and NLTK?
```

```
Fruits like apple, orange, and mango are healthy.
But they are expensive
($1.5/killo), i.e Mrs. Bean can't afford them!
However, One can order some online from www.rewe.de.
```

Lemmatization

A lemma is the canonical, `uninflected` or `dictionary form` of a word. For example, the lemma of `smallest` is `small,` and the lemma of `eating` is `eat.` In many languages, the lemma for nouns is the `nominative singula` r form, the lemma for adjectives is the `nominative singular` positive form, and the lemma for verbs is the `infinitive.` But given an inflected form, finding the lemma (a process called `lemmatization`) is not always as easy. Words often undergo regular spelling changes when inflected for example, in English, verbs and adjectives ending in -e often drop this letter when inflecting: `bake → baking` . Sometimes final consonants are doubled, as in (British) English `travel → travelling` .

An accurate algorithm for lemmatization must be aware of these sorts of inflectional rules and know how to undo them to arrive at the `base form` of the word. It must also know about completely irregular cases, such as `go → went` , `mouse → mice` , and `good → better` . Lemmatization is a difficult task for computers, and requires some basic understanding of the grammatical context and properties of the word. For example, the lemma of `dove` depends on whether the word is being used as a noun

(as in the `bird`) or a verb (as in the past tense of `dive`). However, lemmatization is an important task because, as with part-of-speech tagging, many NLP applications rely on lemmatized text.

Examples of lemmatization:

```
corpora : corpus
better : good
```

NLTK Lemmatizer

In [41]:

```python
import nltk
nltk.download('wordnet')
# Lemmatize using WordNet's buil-in morphy function
# Returns the input unchanged if it cannot be found in WordNet
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
#Give the POS tag as a context to the tager,  a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos ="a"))
#Lemmatizing sentence
sentence = "The striped bats are hanging on their feet for best"
word_list = nltk.word_tokenize(sentence)
print("words:",word_list)
# Lemmatize list of words and join
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in word_list])
print("lemma:",lemmatized_output)
```

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

rocks : rock
corpora : corpus
better : good
words: ['The', 'striped', 'bats', 'are', 'hanging', 'on', 'their', 'fe
et', 'for', 'best']
lemma: The striped bat are hanging on their foot for best
```

spaCy Lemmatizzer

In [42]:

```
import spacy
# Initialize spacy 'en_core_web_sm' model, keeping only tagger component needed
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
sentence = "The striped bats are hanging on their feet for best"
# Parse the sentence using the loaded 'English' model object `nlp`
doc = nlp(sentence)
# Extract the lemma for each token and join
" ".join(["[" + token.lemma_ + "]" for token in doc])
```

Out[42]:

```
'[the] [striped] [bat] [be] [hang] [on] [their] [foot] [for] [good]'
```

TextBlob Lemmatizer

In [43]:

```
!pip install textblob
```

```
Requirement already satisfied: textblob in /Library/Frameworks/Python.
framework/Versions/3.8/lib/python3.8/site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.8/lib/python3.8/site-packages (from textblob)
(3.5)
Requirement already satisfied: click in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->textb
lob) (8.0.3)
Requirement already satisfied: regex in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->textb
lob) (2020.11.13)
Requirement already satisfied: tqdm in /Library/Frameworks/Python.fram
ework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->textbl
ob) (4.54.1)
Requirement already satisfied: joblib in /Library/Frameworks/Python.fr
amework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->text
blob) (1.0.0)

[notice] A new release of pip available: 22.2.2 -> 22.3

[notice] To update, run: pip install --upgrade pip
```

In [44]:

```
from textblob import TextBlob, Word
# Lemmatize a word, use the WordNet's morphy function
word = 'stripes'
w = Word(word)
w.lemmatize()
```

Out[44]:

```
'stripe'
```

In [45]:

```
# Lemmatize a sentence
sentence = "The striped bats are hanging on their feet for best"
sent = TextBlob(sentence)
" ". join(["["+ w.lemmatize()+"]" for w in sent.words])
```

Out[45]:

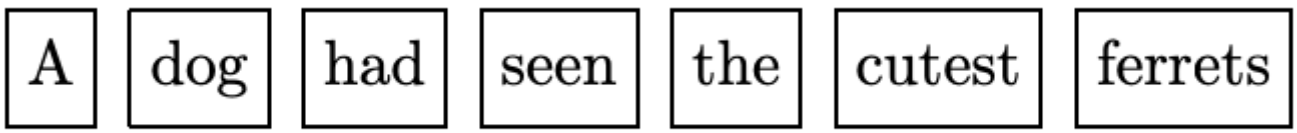
```
'[The] [striped] [bat] [are] [hanging] [on] [their] [foot] [for] [bes
t]'
```

Parts of speech tagging with NLTK

Part-of-speech tagging (POS tagging) is the process of marking up the words in a text with their corresponding part of speech (e.g., `noun`, `verb`, `adjective`). For example, take the following sentence:

A dog had seen the cutest ferrets.`

A tokenizer would split it into the following tokens:



A part-of-speech tagger could then assign labels, or tags, to the tokens according to their respective parts of speech:

A DT dog NN had VBD seen VBN the DT cutest JJS ferrets NNS

The `Penn Treebank tags` used here are as follows: `DT` determiner `NN` noun, singular or mass `VBD` verb, past tense `JJS` adjective, superlative `NNS` noun, plural `VBN` verb, past participle

The inventory from which these POS tags are drawn varies from `language to language`, and from `application to application`.

NLTK includes a Part-of-speech tagger, which assign a tag, or word class, or lexical category for a given token in a text. The default POS tagset for English is based on PennTreebank tagset (https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html).

NLTK also include the Universal POS tagset (https://universaldependencies.org/u/pos/)

In [46]:

```python
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.tag import pos_tag
import nltk
nltk.download('tagsets')
text = "I saw a man sawing the tree with a saw. He can't finish it ontime."
sentences = sent_tokenize(text)
print("tagged", pos_tag(word_tokenize(sentences[0])))
print("=====")
for sentence in sentences:
    for token, pos in pos_tag(word_tokenize(sentence)):
        print(token, pos)
# to get information about a given tag
print("=====")
nltk.help.upenn_tagset("VB")
```

```
tagged [('I', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('man', 'NN'), ('sa
wing', 'VBG'), ('the', 'DT'), ('tree', 'NN'), ('with', 'IN'), ('a', 'D
T'), ('saw', 'NN'), ('.', '.')]
=====
I PRP
saw VBD
a DT
man NN
sawing VBG
the DT
tree NN
with IN
a DT
saw NN
. .
He PRP
ca MD
n't RB
finish VB
it PRP
ontime RB
. .
=====
VB: verb, base form
    ask assemble assess assign assume atone attention avoid bake balka
nize
    bank begin behold believe bend benefit bevel beware bless boil bom
b
    boost brace break bring broil brush build ...

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

In [47]:

```
# you can also decide to use the Universal POS tagset
nltk.download('universal_tagset')
print("{0:15}{1:15}".format("Token","POS tag"))
for sentence in sentences:
    for token, pos in pos_tag(word_tokenize(sentence), tagset='universal'):
        print("{0:15}{1:15}".format(token, pos) )
```

```
Token           POS tag
I               PRON
saw             VERB
a               DET
man             NOUN
sawing          VERB
the             DET
tree            NOUN
with            ADP
a               DET
saw             NOUN
.               .
He              PRON
ca              VERB
n't             ADV
finish          VERB
it              PRON
ontime          ADV
.               .

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

WordNet Lemmatizer with appropriate POS tag

wordnet pos tags are: noun (n), verb(v) , adj(a) and adv (r)

In [48]:

```
# Lemmatize with POS Tag
from nltk.corpus import wordnet
# wordnet pos tags are: noun (n), verb(v) , adj(a) and adv (r)
def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

# 1. Init Lemmatizer
lemmatizer = WordNetLemmatizer()
# 2. Lemmatize Single Word with the appropriate POS tag
word = 'feet'
print(lemmatizer.lemmatize(word, get_wordnet_pos(word)))
# 3. Lemmatize a Sentence with the appropriate POS tag
sentence = "The striped bats are hanging on their feet for best"
print([lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in nltk.word_tokenize(s

```

```
foot
['The', 'strip', 'bat', 'be', 'hang', 'on', 'their', 'foot', 'for', 'b
est']
```

Part of speech tagging with spaCy

In [47]:

```
import spacy
import pprint
# Load English tokenizer, tagger,
# parser, NER and word vectors
nlp = spacy.load("en_core_web_sm")
text = ("I saw a man sawing the tree with a saw. He can't finish it ontime!")
doc = nlp(text)
# Print token and Tag
for token in doc:
    print(token, token.pos_)
# Example list of Verb tokens
print("Verbs:", [token.text for token in doc if token.pos_ == "VERB"])
```

```
I PRON
saw VERB
a DET
man NOUN
sawing VERB
the DET
tree NOUN
with ADP
a DET
saw NOUN
. PUNCT
He PRON
ca AUX
n't PART
finish VERB
it PRON
ontime ADV
! PUNCT
Verbs: ['saw', 'sawing', 'finish']
```

Part of speech tagging with TextBlob

In [49]:

```
!pip install -U textblob
```

```
Requirement already satisfied: textblob in /Library/Frameworks/Python.
framework/Versions/3.8/lib/python3.8/site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.8/lib/python3.8/site-packages (from textblob)
(3.5)
Requirement already satisfied: tqdm in /Library/Frameworks/Python.fram
ework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->textbl
ob) (4.54.1)
Requirement already satisfied: click in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->textb
lob) (8.0.3)
Requirement already satisfied: regex in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->textb
lob) (2020.11.13)
Requirement already satisfied: joblib in /Library/Frameworks/Python.fr
amework/Versions/3.8/lib/python3.8/site-packages (from nltk>=3.1->text
blob) (1.0.0)

[notice] A new release of pip available: 22.2.2 -> 22.3

[notice] To update, run: pip install --upgrade pip
```

In [50]:

```
!python -m textblob.download_corpora
```

```
[nltk_data] Downloading package brown to /Users/abhikjana/nltk_data...
[nltk_data]   Package brown is already up-to-date!
[nltk_data] Downloading package punkt to /Users/abhikjana/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/abhikjana/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/abhikjana/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package conll2000 to
[nltk_data]     /Users/abhikjana/nltk_data...
[nltk_data]   Package conll2000 is already up-to-date!
[nltk_data] Downloading package movie_reviews to
[nltk_data]     /Users/abhikjana/nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
Finished.
```

In [51]:

```
from textblob import TextBlob
text = ("I saw a man sawing the tree with a saw. He can't finish it ontime!")
# create a textblob object
blob_object = TextBlob(text)
# print word with pos tag.
print(blob_object.tags)
```

```
[('I', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('man', 'NN'), ('sawing',
'VBG'), ('the', 'DT'), ('tree', 'NN'), ('with', 'IN'), ('a', 'DT'),
('saw', 'NN'), ('He', 'PRP'), ('ca', 'MD'), ("n't", 'RB'), ('finish',
'VB'), ('it', 'PRP'), ('ontime', 'RB')]
```

TextBlob Lemmatizer with appropriate POS tag

In [52]:

```
#Define function to lemmatize each word with its POS tag
def lemmatize_with_postag(sentence):
    sent = TextBlob(sentence)
    tag_dict = {"J": 'a',
                "N": 'n',
                "V": 'v',
                "R": 'r'}
    words_and_tags = [(w, tag_dict.get(pos[0], 'n')) for w, pos in sent.tags]
    lemmatized_list = ["["+wd.lemmatize(tag) +"]" for wd, tag in words_and_tags]
    return " ".join(lemmatized_list)
# Lemmatize
sentence = "The striped bats are hanging on their feet for best"
lemmatize_with_postag(sentence)
```

Out[52]:

```
'[The] [striped] [bat] [be] [hang] [on] [their] [foot] [for] [best]'
```

Excercise_3 (3pt)

Lemmatization for German

There is no lemmatization library in NLTK for German. However, the GermaLemma (https://github.com/WZBSocialScienceCenter/germalemma) (https://github.com/WZBSocialScienceCenter/germalemma (https://github.com/WZBSocialScienceCenter/germalemma)) library is an open source lemmatizer for German. To lemmatize a word, you need to pass the POS tag as a secondary argument. In this exercise, you can use the POS tagger for German from pattern.de but then you have to convert tags into `N`, `V`, `ADJ`, or `ADV`. So your task is, when the word category is in one of the four tags, map them and pass to the lematizer. If the POS tag is not in the four categories, return the word itself as the lemma. See the cells below on how to execute the lemmatizer and pos tager for German.

You can install GermaLemma as

```
pip install -U germalemma
```

Also make sure mysql and related packages are installed

In [52]:

```
#uncomment the following for Mysql cleint dev in Linux
# !sudo apt install default-libmysqlclient-dev
# Installing GermaLemma
!pip install -U germalemma
```

```
Collecting germalemma
  Using cached germalemma-0.1.3-py3-none-any.whl (2.3 MB)
Collecting PatternLite>=3.6
  Using cached PatternLite-3.6-py3-none-any.whl (22.1 MB)
Collecting Pyphen>=0.9.5
  Downloading pyphen-0.13.0-py3-none-any.whl (2.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 20.5 MB/s eta

0:00:0000:0100:01
Requirement already satisfied: numpy in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from PatternLite>=3.6
->germalemma) (1.17.4)
Requirement already satisfied: scipy in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from PatternLite>=3.6
->germalemma) (1.4.1)
Requirement already satisfied: nltk in /Library/Frameworks/Python.fram
ework/Versions/3.8/lib/python3.8/site-packages (from PatternLite>=3.6-
>germalemma) (3.5)
Requirement already satisfied: regex in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from nltk->PatternLit
e>=3.6->germalemma) (2020.11.13)
Requirement already satisfied: click in /Library/Frameworks/Python.fra
mework/Versions/3.8/lib/python3.8/site-packages (from nltk->PatternLit
e>=3.6->germalemma) (8.0.3)
Requirement already satisfied: tqdm in /Library/Frameworks/Python.fram
ework/Versions/3.8/lib/python3.8/site-packages (from nltk->PatternLite
>=3.6->germalemma) (4.54.1)
Requirement already satisfied: joblib in /Library/Frameworks/Python.fr
amework/Versions/3.8/lib/python3.8/site-packages (from nltk->PatternLi
te>=3.6->germalemma) (1.0.0)
Installing collected packages: Pyphen, PatternLite, germalemma
Successfully installed PatternLite-3.6 Pyphen-0.13.0 germalemma-0.1.3

[notice] A new release of pip available: 22.2.2 -> 22.3

[notice] To update, run: pip install --upgrade pip
```

In [53]:

```
# POS taggin for German (ISSUE - run this cell multiple times if it raises excep
from germalemma import GermaLemma
from pattern.de import parse, split
s = parse('Die Katze liegt auf der Matte.')
for sentence in split(s):
    for token in sentence:
        print (token, token.pos)
```

```
Die DT
Katze NN
liegt VB
auf IN
der DT
Matte NN
. .
```

Address the `TODO` part below to complete your exercise

In [54]:

```
# Lemmatizer for German using GermaLemma lematizer -
#(ISSUE - run this cell multiple times, it may happen that it raises exceptions)

sentence2 = """Die Brände in Brasilien setzen erhebliche Mengen an klimaschädlic
Die Nasa hat nun simuliert, wie sich Kohlenmonoxid über Südamerika ausbreitet.
Am Boden schadet das Gas der Gesundheit erheblich."""
de_lemma = GermaLemma()
# POS tagger
poses = parse(sentence2)
for sentence in split(poses):
    print ("===")
    for token in sentence:
        print (token, token.pos)
        #TODO: Here map the POS tag to V, N, ADJ, or ADV as MAPD_POS. MAPD_POS =
        # If the POS tag is not in  V, N, ADJ, or ADV,  no need to lemmatize
        #  Print the lemma here
        # Print([de_lemma.find_lemma(token, MAPED_POS) for token in split(poses)
    print ("===")
#
```

```
===
Die DT
Brände NNS
in IN
Brasilien NNP
setzen VB
erhebliche JJ
Mengen NN
an IN
klimaschädlichen JJ
Treibhausgasen NN
frei JJ
. .
===
===
Die DT
Nasa NN
hat VB
nun IN
simuliert NN
, ,
wie IN
sich PRP
Kohlenmonoxid NN
über IN
```

```
Südamerika NNP
ausbreitet NNP
. .
===
===
Am IN
Boden NN
schadet VB
das DT
Gas NN
der DT
Gesundheit NN
erheblich JJ
. .
===
```

Resources

- HTTP request (https://realpython.com/python-requests/)
- Regex (https://www.programiz.com/python-programming/regex)
- Regex lookahead and lookbehind (https://www.regular-expressions.info/lookaround.html)
- BeautifulSoup tutorial (https://www.dataquest.io/blog/web-scraping-tutorial-python/)
- Online Regex editor (https://regex101.com/r/nG1gU7/)
- Processing Raw Text NLTK (https://www.nltk.org/book/ch03.html)
- Different Stemmers in NLTK (https://www.nltk.org/api/nltk.stem.html)
- German lemmatizer (https://github.com/WZBSocialScienceCenter/germalemma)

In [ ]:

```
1
```