Practical no 7

Tutorial

In this practical we will Extract sample document and apply following document preprocessing methods:

- **Tokenization**
- **POS Tagging**
- **Stop words removal**
- **Stemming**
- **Lemmatization**

We don't need a database for this practical. The practical might be confusing at times, so here are two video links if you have any problems:

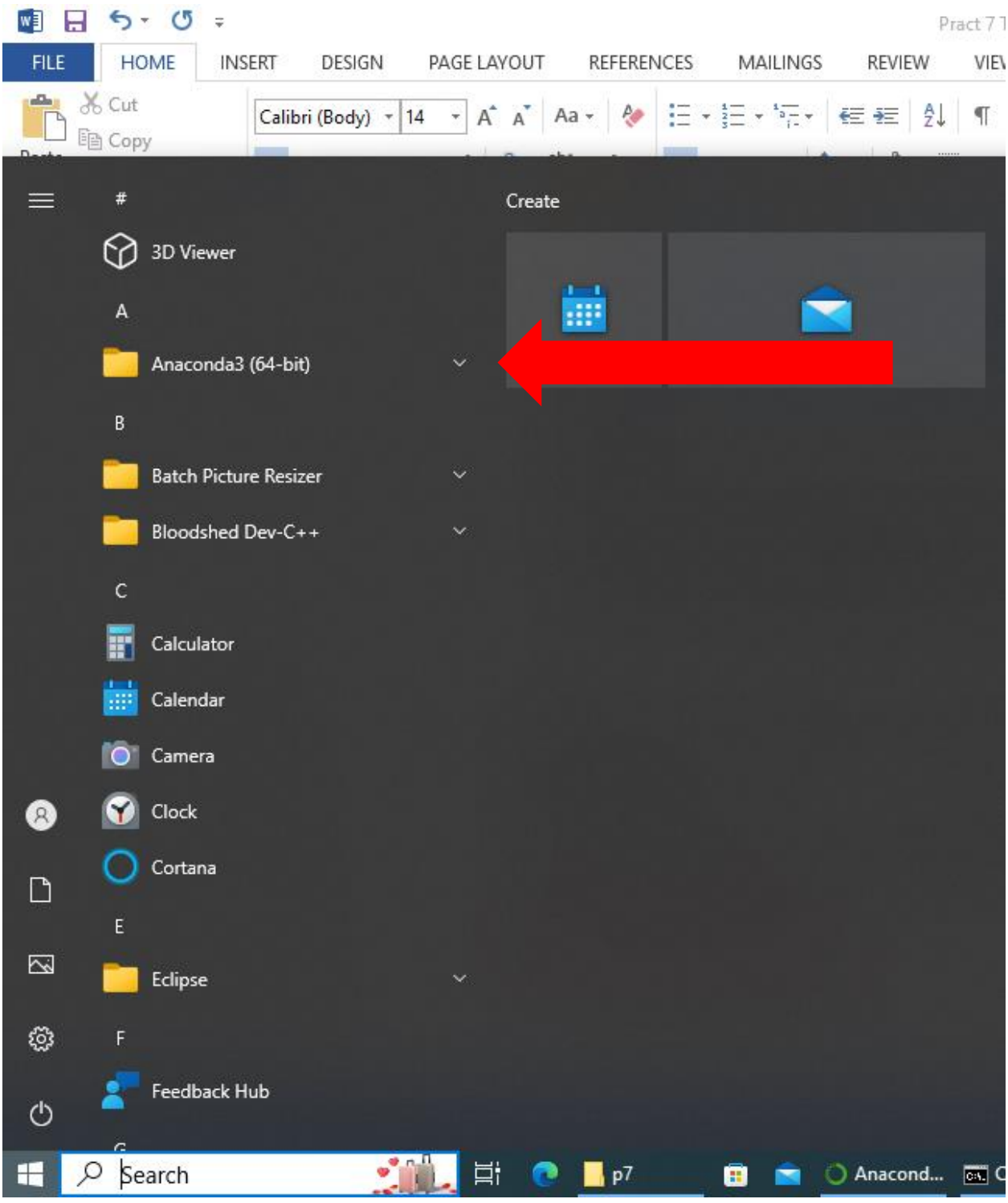
Part 1: https://www.youtube.com/watch?v=h31iQxwODXw

Part 2: https://www.youtube.com/watch?v=oUG0tZAen9k

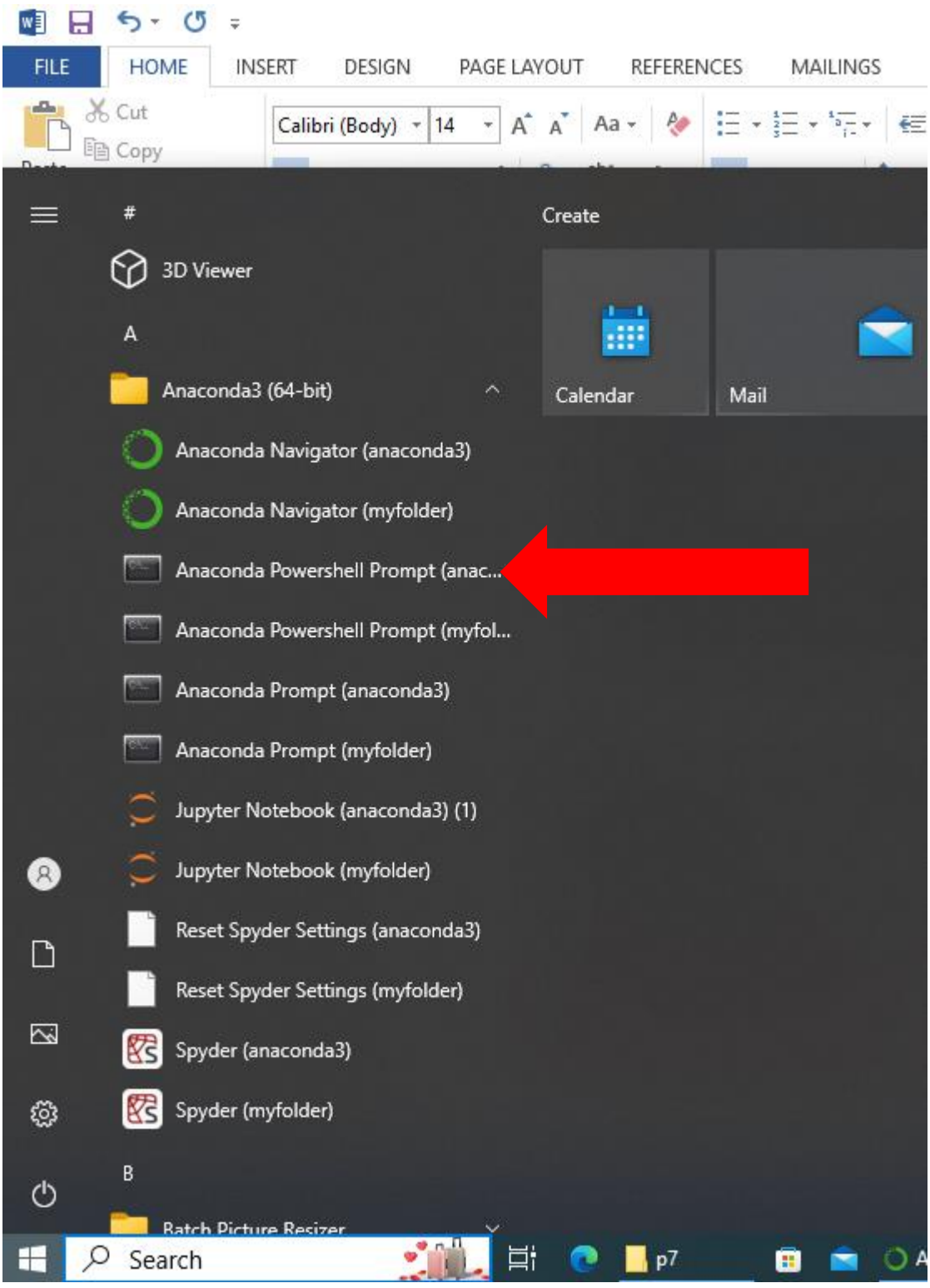
You can refer them if you want but the tutorial will explain things more clearly.

First we need to configure and download NLTK library. For that we need to install it in the anaconda mainframe.

Head over to start and click on the anaconda folder.



In the anaconda subfolder open Anaconda prompt.

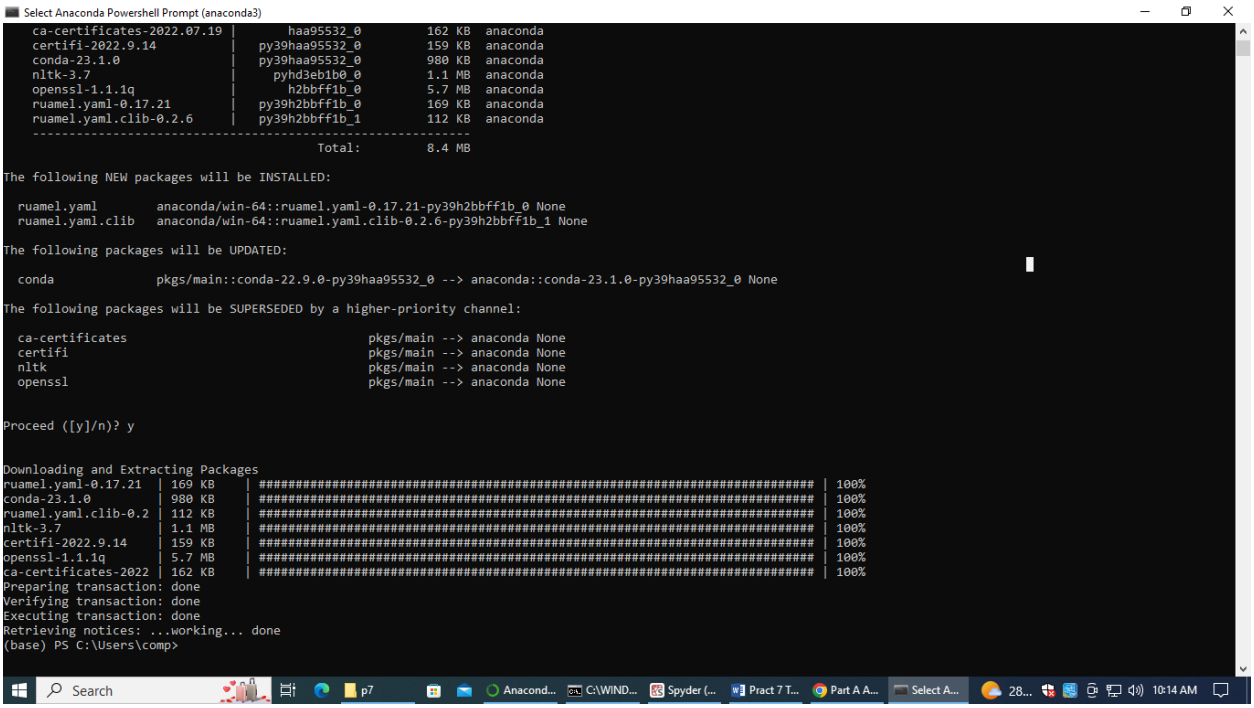

FILE
HOME
INSERT
DESIGN
PAGE LAYOUT
REFERENCES
MAILINGS
Cut
Copy
Calibri (Body)
14
Aa
#
3D Viewer
A
Anaconda3 (64-bit)
Anaconda Navigator (anaconda3)
Anaconda Navigator (myfolder)
Anaconda Powershell Prompt (anac...
Anaconda Powershell Prompt (myfol...
Anaconda Prompt (anaconda3)
Anaconda Prompt (myfolder)
Jupyter Notebook (anaconda3) (1)
Jupyter Notebook (myfolder)
Reset Spyder Settings (anaconda3)
Reset Spyder Settings (myfolder)
Spyder (anaconda3)
Spyder (myfolder)
B
Batch Picture Resizer
Create
Calendar
Mail
Search
p7

After opening the prompt type the command:

conda install -c anaconda nltk

Press enter and let the process run. After a few sec you will get a message seeking permission to install : (y/n).

Type y and hit enter. NLTK will be downloaded. The prompt will look like this:



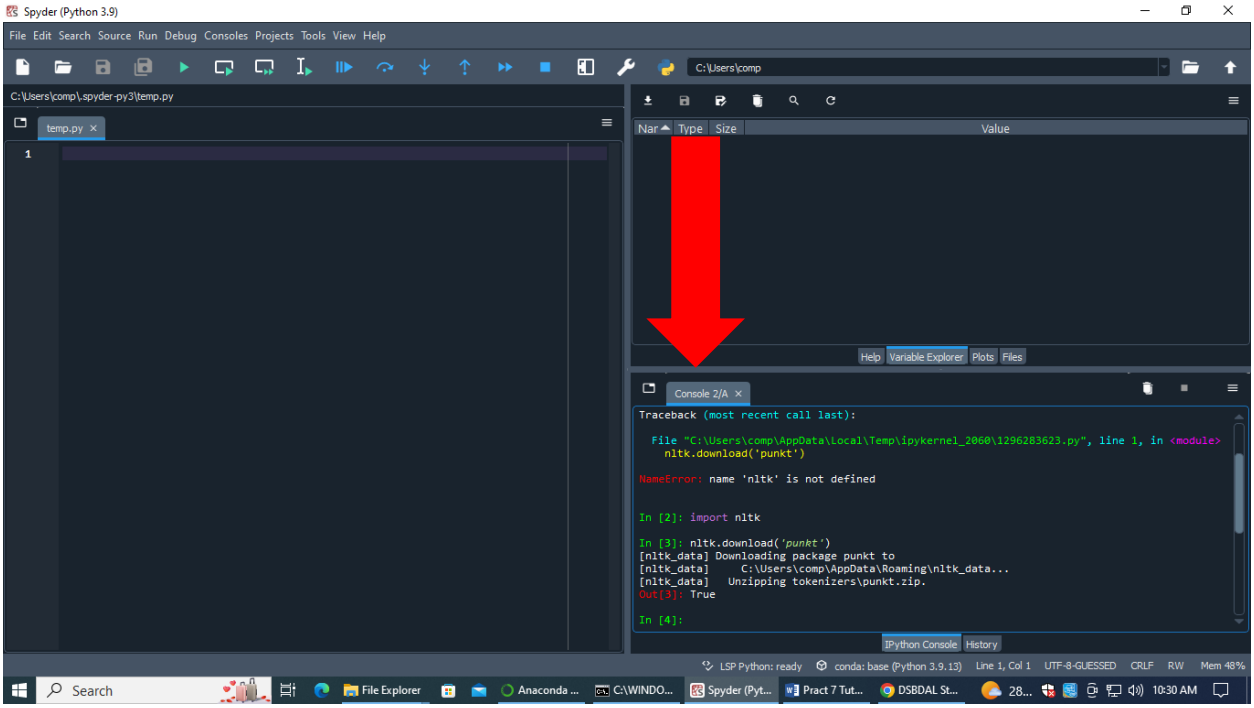
After the installation is done, head over to anaconda navigator and run spyder.

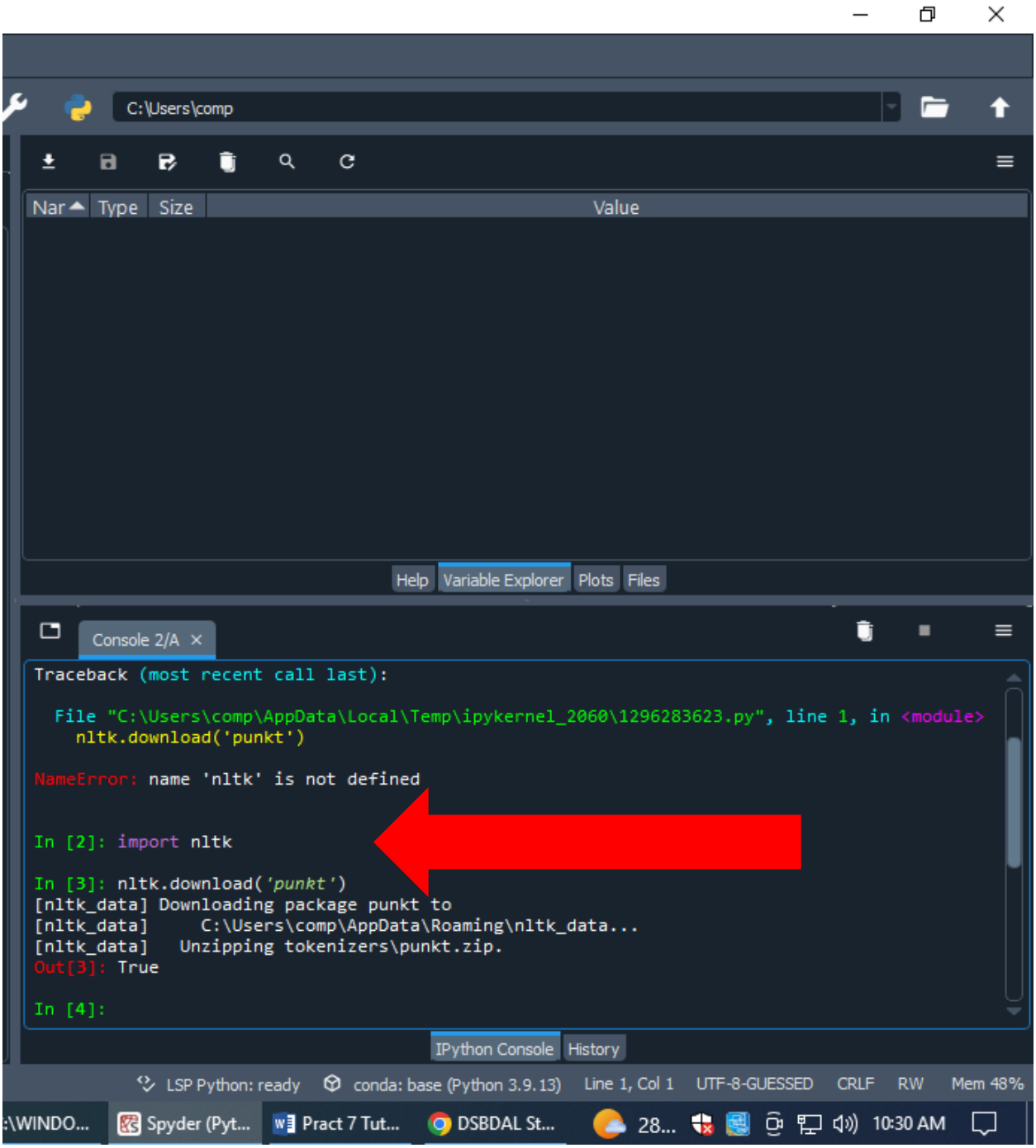
Create a new file and follow the given algorithm:

Algorithm for Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization:

Step 1: Download the required packages

First type : import nltk in the console of the spyder gui and hit enter


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\comp
C:\Users\comp\.spyder-py3\temp.py
temp.py
1
Nar
Type
Size
Value
Help
Variable Explorer
Plots
Files
Console 2/A
Traceback (most recent call last):
File "C:\Users\comp\AppData\Local\Temp\ipykernel_2060\1296283623.py", line 1, in <module>
nltk.download('punkt')
NameError: name 'nltk' is not defined
In [2]: import nltk
In [3]: nltk.download('punkt')
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\comp\AppData\Roaming\nltk_data...
[nltk_data] Unzipping tokenizers\punkt.zip.
Out[3]: True
In [4]:
IPython Console
History
LSP Python: ready
conda: base (Python 3.9.13)
Line 1, Col 1
UTF-8-GUESSED
CRLF
RW
Mem 48%
Search
File Explorer
Anaconda ...
C:\WINDO...
Spyder (Pyt...
Pract 7 Tut...
DSBDAL St...
28...
10:30 AM


C:\Users\comp
Nar
Type
Size
Value
Help
Variable Explorer
Plots
Files
Console 2/A
Traceback (most recent call last):
File "C:\Users\comp\AppData\Local\Temp\ipykernel_2060\1296283623.py", line 1, in <module>
nltk.download('punkt')
NameError: name 'nltk' is not defined
In [2]: import nltk
In [3]: nltk.download('punkt')
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\comp\AppData\Roaming\nltk_data...
[nltk_data] Unzipping tokenizers\punkt.zip.
Out[3]: True
In [4]:
IPython Console
History
LSP Python: ready
conda: base (Python 3.9.13)
Line 1, Col 1
UTF-8-GUESSED
CRLF
RW
Mem 48%
:\WINDO...
Spyder (Pyt...
Pract 7 Tut...
DSBDAL St...
28...
10:30 AM

After that, type and download the following packages one by one in the console below the import nltk command :

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

Step 2: Initialize the text.

Now head over to the main editing area and type the following code :

```
text= "Tokenization is the first step in text analytics. The
process of breaking down a text paragraph into smaller chunks
such as words or sentences is called Tokenization."
```

Step 3: Perform Tokenization

```
#Sentence Tokenization
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
#Word Tokenization
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

Step 4: Removing Punctuations and Stop Word

```
# print stop words of English
from nltk import re
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
```

```
print(stop_words)
text= "How to remove stop words with NLTK library in Python?"
text= re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
if w not in stop_words:
filtered_text.append(w)
print("Tokenized Sentence:",tokens)
print("Filterd Sentence:",filtered_text)
```

Step 5 : Perform Stemming

```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
rootWord=ps.stem(w)
print(rootWord)
```

Step 6: Perform Lemmatization

```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
```

```
print("Lemma for {} is {}".format(w,
wordnet_lemmatizer.lemmatize(w)))
```

Step 7: Apply POS Tagging to text

```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
print(nltk.pos_tag([word]))
```

Now for the next part:

Algorithm to Create a representation of document by calculating TFIDF

Step 1: Import the necessary libraries.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

Step 2: Initialize the Documents.

```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

Step 3: Create BagofWords (BoW) for Document A and B.

```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

Step 4: Create Collection of Unique words from Document A and B.

```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

Step 5: Create a dictionary of words and their occurrence for each document in the corpus

```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
numOfWordsA[word] += 1
numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
numOfWordsB[word] += 1
```

Step 6: Compute the term frequency for each of our documents.

```
def computeTF(wordDict, bagOfWords):
tfDict = {}
bagOfWordsCount = len(bagOfWords)
for word, count in wordDict.items():
tfDict[word] = count / float(bagOfWordsCount)
return tfDict
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

Step 7: Compute the term Inverse Document Frequency.

```
def computeIDF(documents):
import math
N = len(documents)
```

```
idfDict = dict.fromkeys(documents[0].keys(), 0)
for document in documents:
for word, val in document.items():
if val > 0:

idfDict[word] += 1
for word, val in idfDict.items():
idfDict[word] = math.log(N / float(val))
return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])idfs
```

Step 8: Compute the term TF/IDF for all words.

```
def computeTFIDF(tfBagOfWords, idfs):
tfidf = {}
for word, val in tfBagOfWords.items():
tfidf[word] = val * idfs[word]
return tfidf
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
df
```

Conclusion:

In this way we have done text data analysis using TF IDF algorithm