NATURAL LANGUAGE PROCESSING

Certified Journal

Submitted in partial fulfilment of the

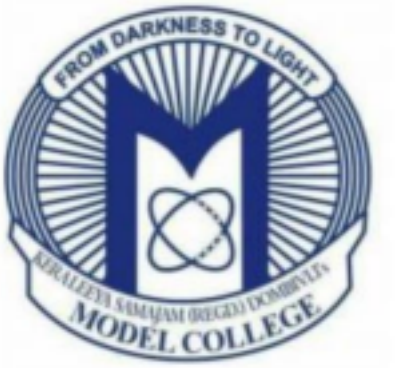
Requirements for the award of the Degree of

MASTER OF SCIENCE

(INFORMATION_TECHNOLOGY)

By

Anjali Rameshwar Nimje



DEPARTMENT OF INFORMATION TECHNOLOGY

KERALEEYA SAMAJAM (REGD.) DOMBIVLI'S

MODEL COLLEGE (AUTONOMOUS)

Re-Accredited 'A' Grade by NAAC

(Affiliated to University of Mumbai)

FOR THE YEAR

(2023-24)

Keraleeya Samajam(Regd.) Dombivli's

MODEL COLLEGE

Re-Accredited Grade "A" by NAAC

Kanchan Goan Village, Khambalpada, Thakurli East – 421201
Contact No – 7045682157, 7045682158. www.model-college.edu.in

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr. /Miss _____

*Studying in Class*_____ *Seat No.* _____

*Has completed the prescribed practicals in the subject*_____

*During the academic year*_____

Date : _____

External Examiner

Internal Examiner
M.Sc. Information Technology

INDEX

Sr No	Practical Name	Date	Signature
1A	Install NLTK		
1B	Convert the given text to speech		
1C	Convert audio file Speech to Text.		
2A	Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories,		
2C	Study Conditional frequency distributions		
2D	Study of tagged corpora with methods like tagged_sents, tagged_words.		
2E	Write a program to find the most frequent noun tags.		
2F	Map Words to Properties Using Python Dictionaries		
2G	Study DefaultTagger, Regular expression tagger, UnigramTagger		
3A	Study of Wordnet Dictionary with methods as synsets, definitions, examples,antonyms.		
3B	Study lemmas, hyponyms, hypernyms.		

3C	**Write a program using python to find synonym and antonym of word "active" using Wordnet.**		
3D	**Compare two nouns.**		
3E	**Handling stopword**		
4A	**Tokenization using Python's split() function.**		
4B	**Tokenization using Regular Expressions (RegEx)**		
4C	**Tokenization using NLTK**		
4D	**Tokenization using the spaCy library**		
4E	**Tokenization using Keras**		
4F	**Tokenization using Gensim**		
6A	**Illustrate part of speech tagging Part of speech Tagging and chunking of user defined text.**		

6B	**Named Entity recognition using user defined text.**		
6C	**Named Entity recognition with diagram using NLTK corpus – treebank.**		
7A	**Finite state automata Define grammar using nltk. Analyze a sentence using the same.**		
7B	**Accept the input string with Regular expression of Finite Automaton: 101+.**		
7C	**Accept the input string with Regular expression of FA: (a+b)*bba.**		
7D	**Implementation of Deductive Chart Parsing using context free grammar and a given sentence.**		
8	**Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer ,Study WordNetLemmatizer**		
9	**Implement Naive Bayes classifier**		

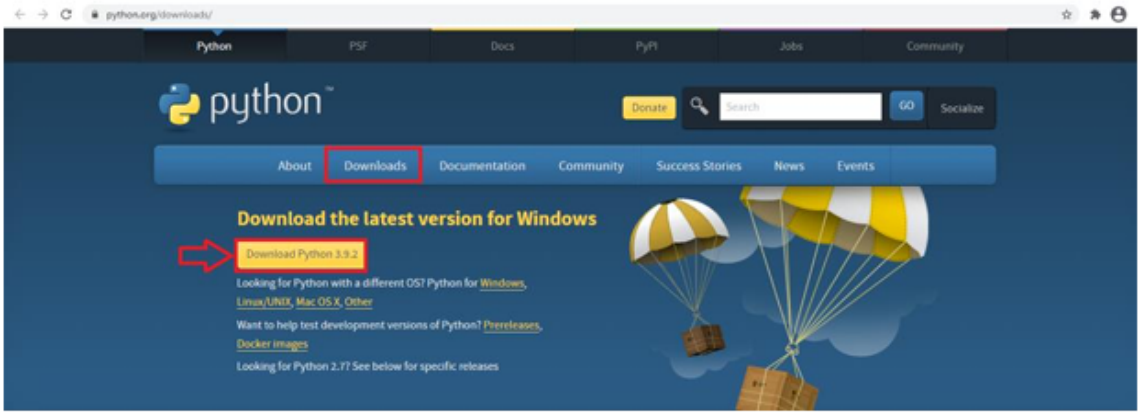
10 A.1	**Speech tagging using spacy**		
10 A.2	**Speech tagging using nltk**		
10 B.1	**Usage of Give and Gave in the Penn Treebank sample**		
10 B.2	**Probabilistic parser**		
10 C	**Malt Parsing**		
11 A	**Multiword Expressions in NLP**		
11 B	**Normalized Web Distance and Word Similarity**		
11 C	**Word Sense Disambiguation**		

PRACTICAL 1A

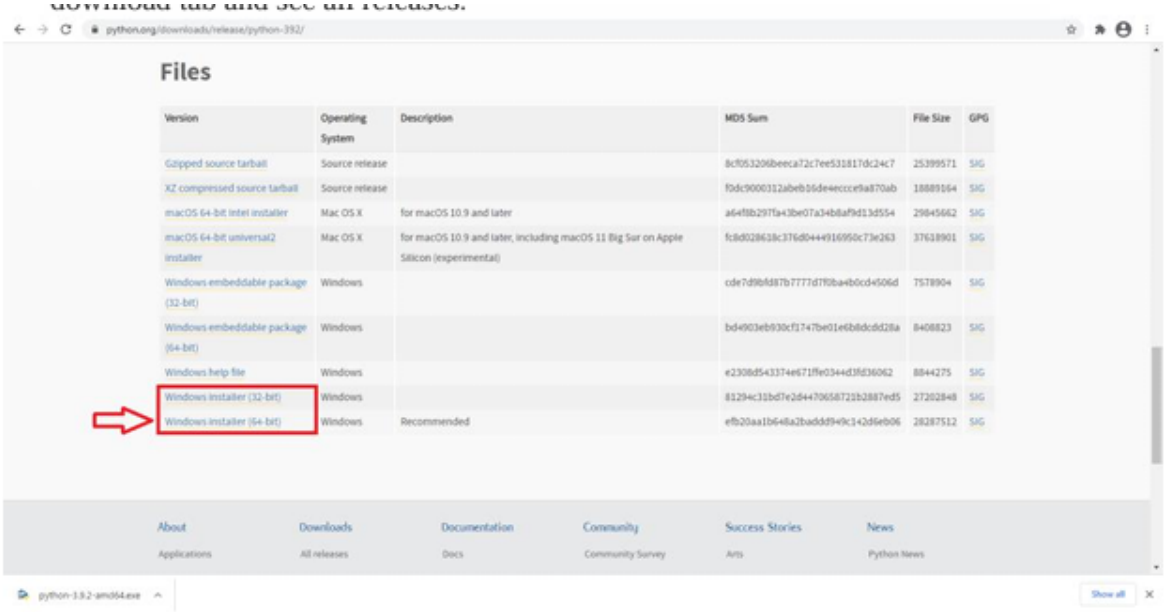
Install NLTK

Python 3.9.2 Installation on Windows

Step 1) Go to link https://www.python.org/downloads/, and select the latest version for windows.

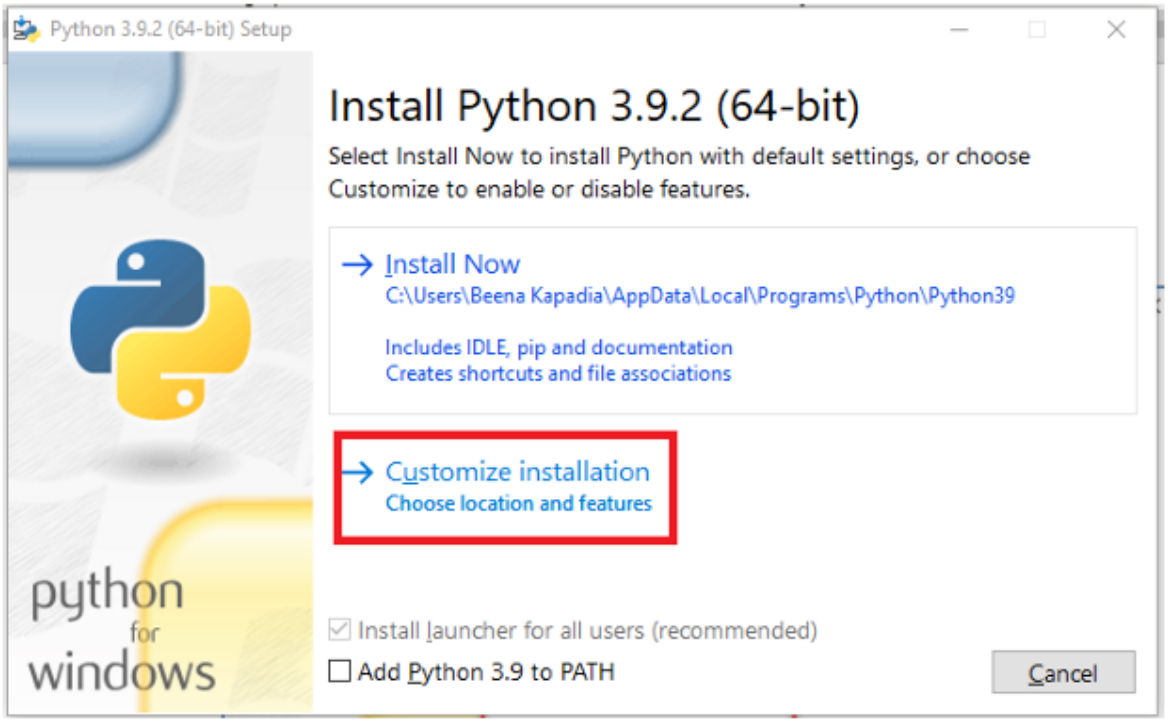


Note: If you don't want to download the latest version, you can visit the download tab and see all releases.

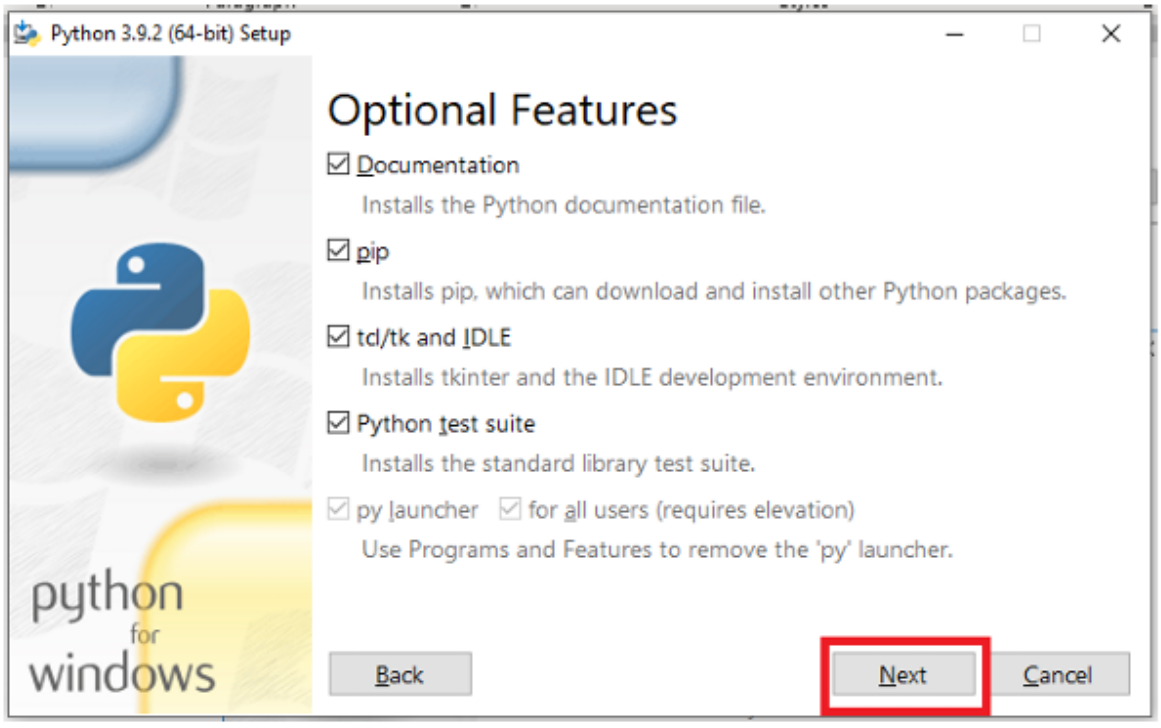


Step 2) Click on the Windows installer (64 bit)

Step 3) Select Customize Installation



Step 4) Click NEXT

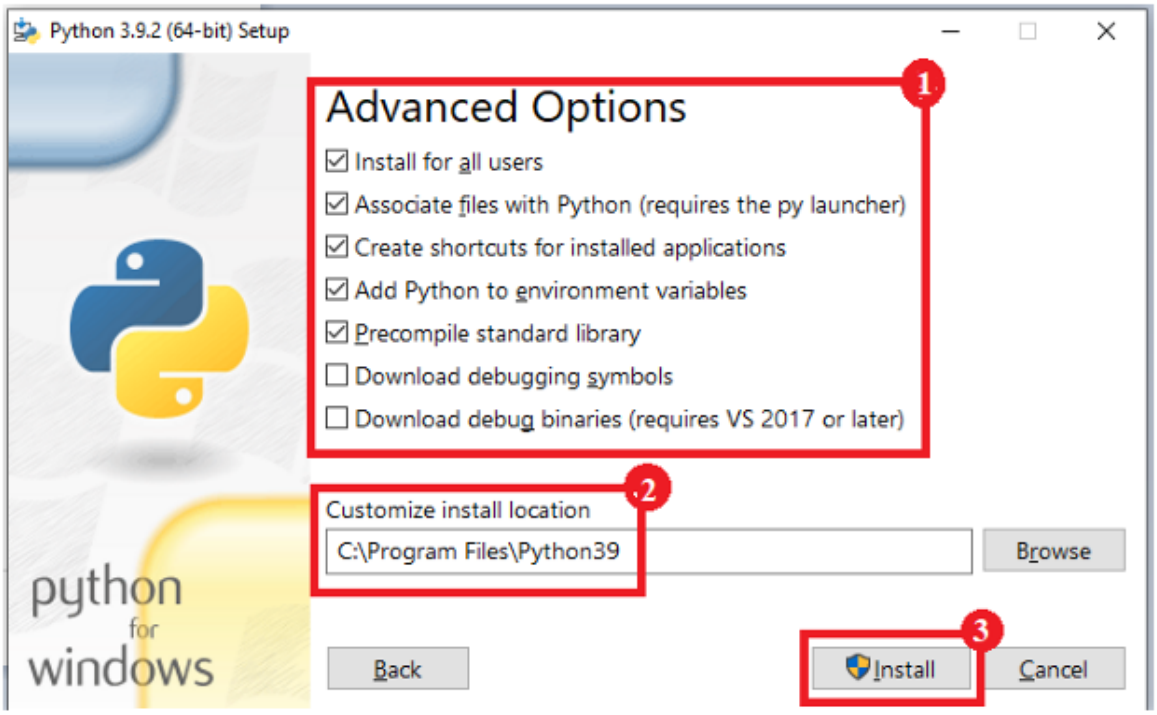


Step 5) In next screen

1. Select the advanced options

2. Give a Custom install location. Keep the default folder as c:\Program files\Python39

3. Click Install



Step 6) Click Close button once install is done.

Step 7) open command prompt window and run the following commands:

C:\Users\Beena Kapadia>pip install --upgrade pip

C:\Users\Beena Kapadia> pip install --user -U nltk

C:\Users\Beena Kapadia> >pip install --user -U numpy

C:\Users\Beena Kapadia>python

>>> import nltk

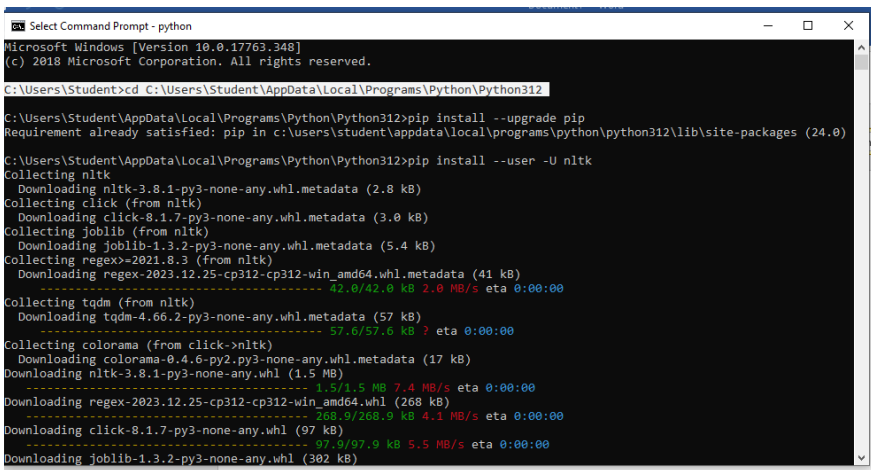
PRACTICAL 1B

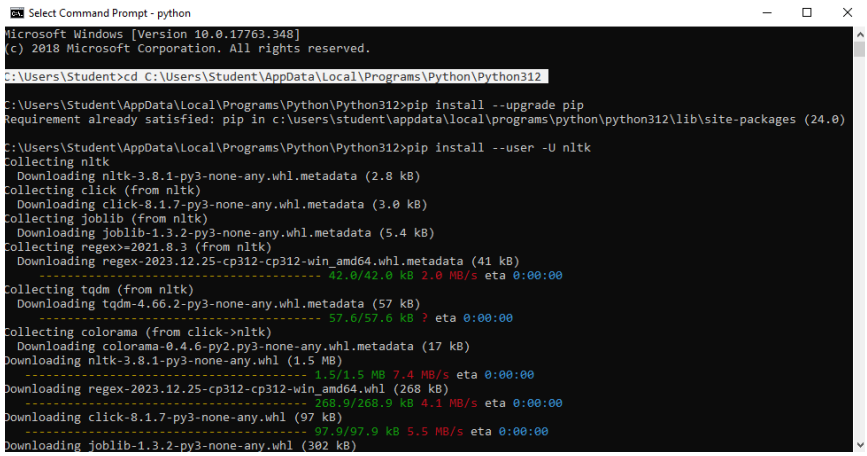
Convert the given text to speech.

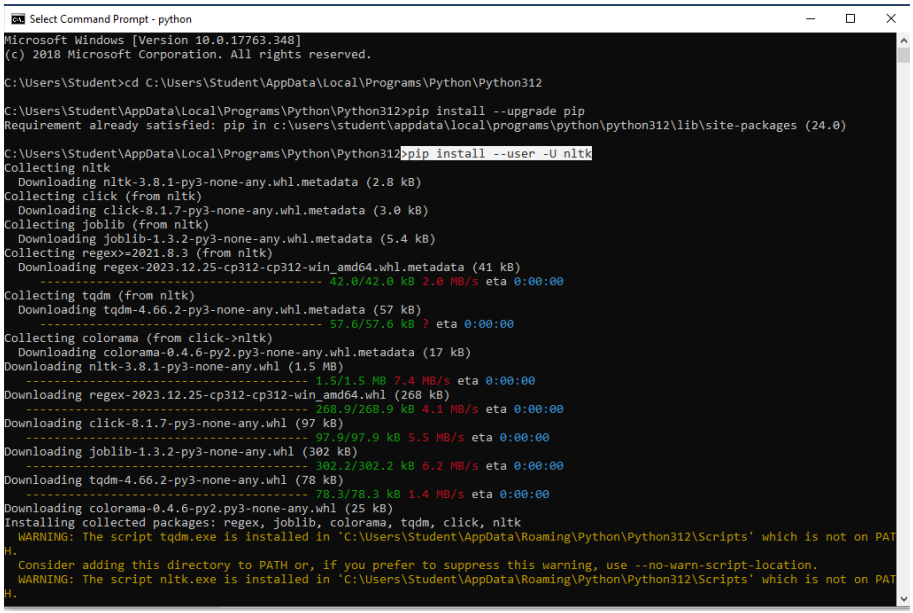
Code:

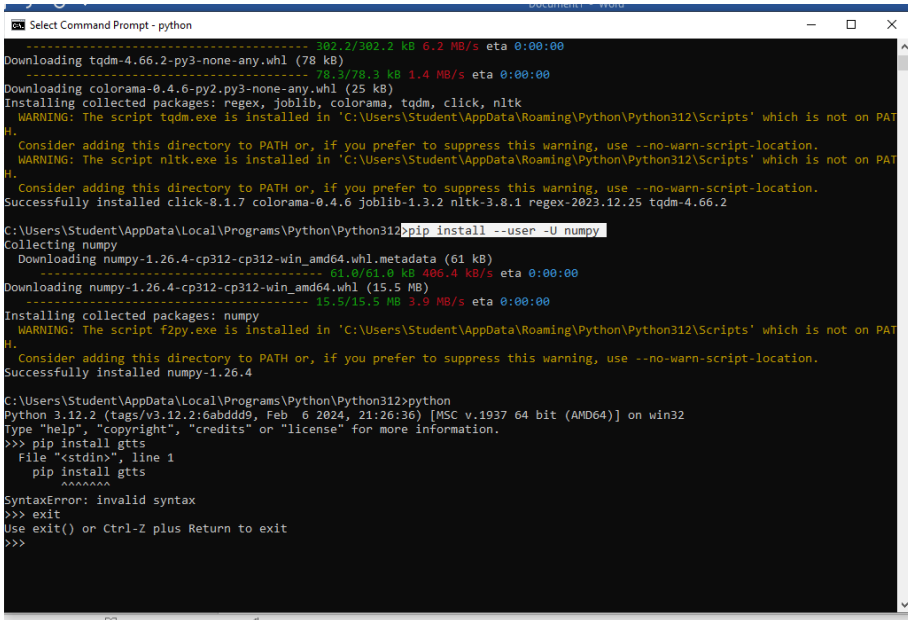
```
from playsound import playsound
from gtts import gTTS
mytext="Welcome to Natural Language Processing"
language="en"
myobj=gTTS(text=mytext,lang=language,slow=False)
myobj.save("myfile.mp3")
playsound("myfile.mp3")
```

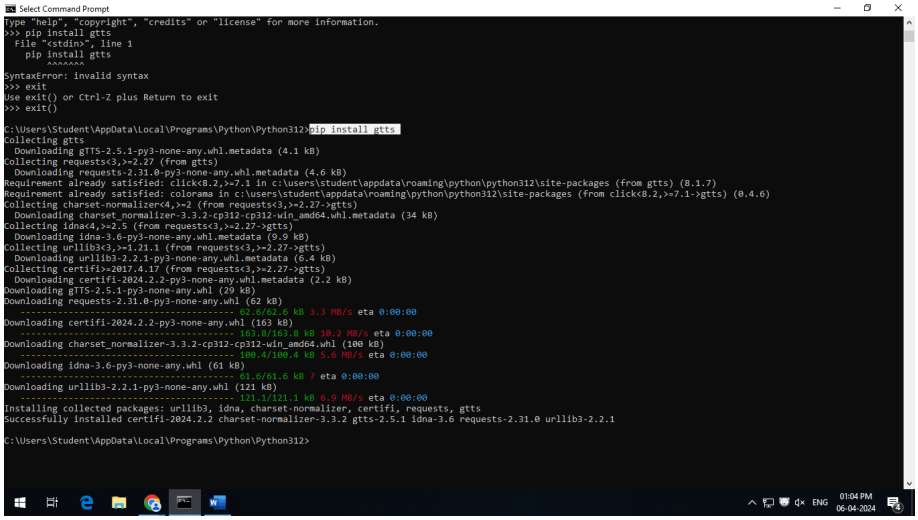

```
from playsound import playsound
from gtts import gTTS
mytext="Welcome to Natural Language Processing"
language="en"
myobj=gTTS(text=mytext,lang=language,slow=False)
myobj.save("myfile.mp3")
playsound("myfile.mp3")
```

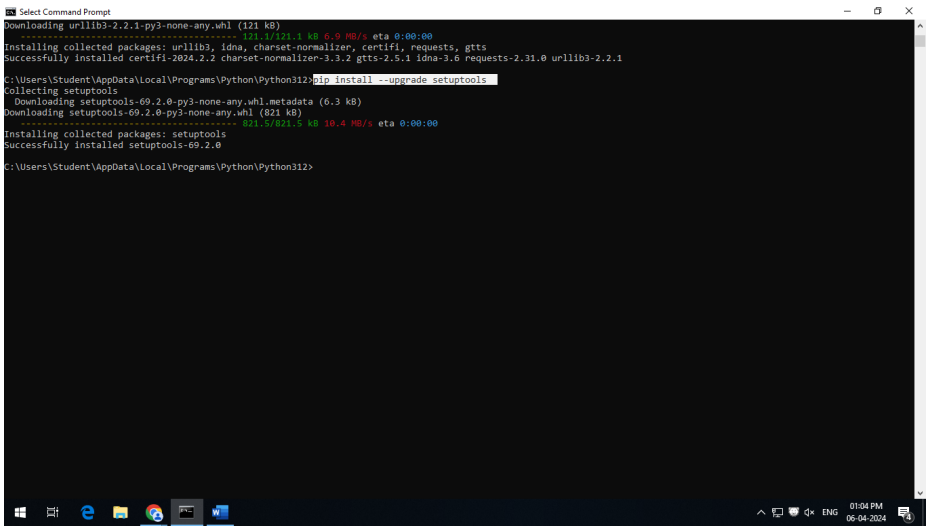



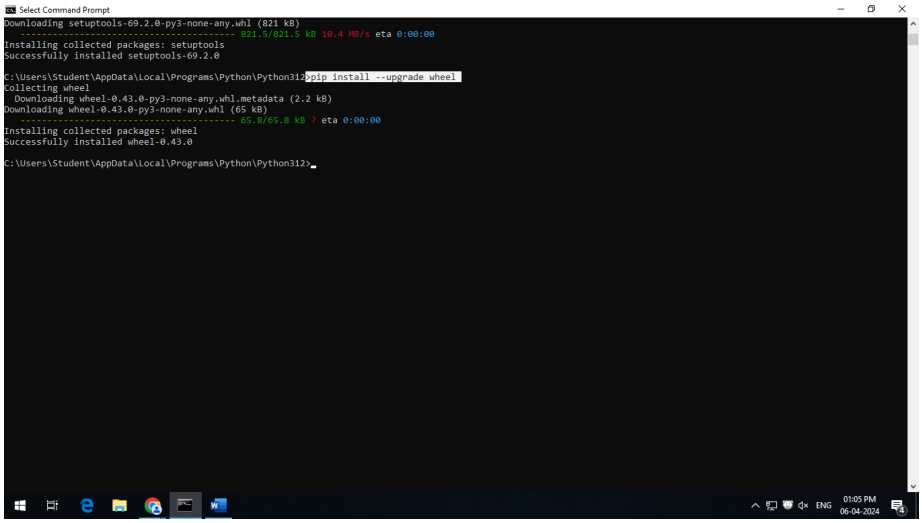

Select Command Prompt - python
Microsoft Windows [Version 10.0.17763.348]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Student>cd C:\Users\Student\AppData\Local\Programs\Python\Python312
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --upgrade pip
Requirement already satisfied: pip in c:\users\student\appdata\local\programs\python\python312\lib\site-packages (24.0)
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --user -U nltk
Collecting nltk
Downloading nltk-3.8.1-py3-none-any.whl.metadata (2.8 kB)
Collecting click (from nltk)
Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting joblib (from nltk)
Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
Collecting regex>=2021.8.3 (from nltk)
Downloading regex-2023.12.25-cp312-cp312-win_amd64.whl.metadata (41 kB)
42.0/42.0 kB 2.0 MB/s eta 0:00:00
Collecting tqdm (from nltk)
Downloading tqdm-4.66.2-py3-none-any.whl.metadata (57 kB)
57.6/57.6 kB ? eta 0:00:00
Collecting colorama (from click->nltk)
Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
1.5/1.5 MB 7.4 MB/s eta 0:00:00
Downloading regex-2023.12.25-cp312-cp312-win_amd64.whl (268 kB)
268.9/268.9 kB 4.1 MB/s eta 0:00:00
Downloading click-8.1.7-py3-none-any.whl (97 kB)
97.9/97.9 kB 5.5 MB/s eta 0:00:00
Downloading joblib-1.3.2-py3-none-any.whl (302 kB)

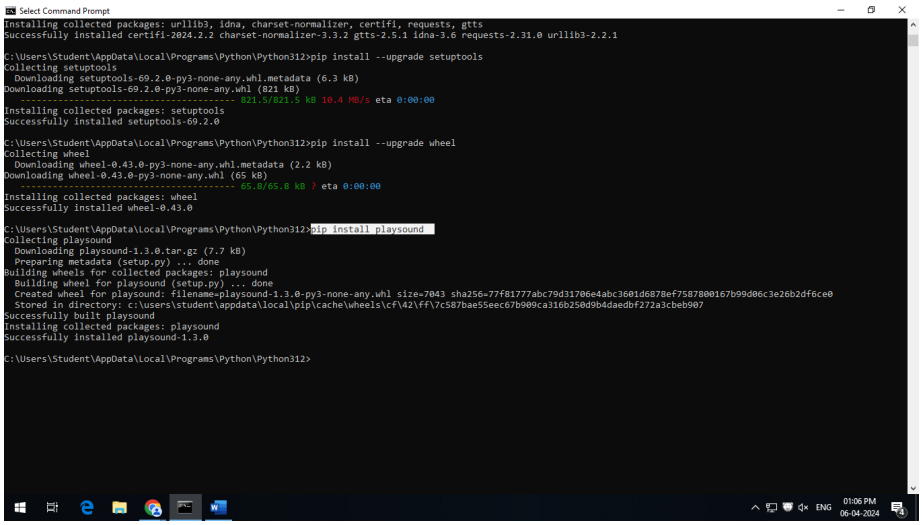

Select Command Prompt - python
Microsoft Windows [Version 10.0.17763.348]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Student>cd C:\Users\Student\AppData\Local\Programs\Python\Python312
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --upgrade pip
Requirement already satisfied: pip in c:\users\student\appdata\local\programs\python\python312\lib\site-packages (24.0)
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --user -U nltk
Collecting nltk
Downloading nltk-3.8.1-py3-none-any.whl.metadata (2.8 kB)
Collecting click (from nltk)
Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting joblib (from nltk)
Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
Collecting regex>=2021.8.3 (from nltk)
Downloading regex-2023.12.25-cp312-cp312-win_amd64.whl.metadata (41 kB)
42.0/42.0 kB 2.0 MB/s eta 0:00:00
Collecting tqdm (from nltk)
Downloading tqdm-4.66.2-py3-none-any.whl.metadata (57 kB)
57.6/57.6 kB ? eta 0:00:00
Collecting colorama (from click->nltk)
Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
1.5/1.5 MB 7.4 MB/s eta 0:00:00
Downloading regex-2023.12.25-cp312-cp312-win_amd64.whl (268 kB)
268.9/268.9 kB 4.1 MB/s eta 0:00:00
Downloading click-8.1.7-py3-none-any.whl (97 kB)
97.9/97.9 kB 5.5 MB/s eta 0:00:00
Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
302.2/302.2 kB 6.2 MB/s eta 0:00:00
Downloading tqdm-4.66.2-py3-none-any.whl (78 kB)
78.3/78.3 kB 1.4 MB/s eta 0:00:00
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: regex, joblib, colorama, tqdm, click, nltk
WARNING: The script tqdm.exe is installed in 'C:\Users\Student\AppData\Roaming\Python\Python312\Scripts' which is not on PAT
H.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script nltk.exe is installed in 'C:\Users\Student\AppData\Roaming\Python\Python312\Scripts' which is not on PAT
H.


Select Command Prompt - python
Downloading tqdm-4.66.2-py3-none-any.whl (78 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: regex, joblib, colorama, tqdm, click, nltk
WARNING: The script tqdm.exe is installed in 'C:\Users\Student\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script nltk.exe is installed in 'C:\Users\Student\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed click-8.1.7 colorama-0.4.6 joblib-1.3.2 nltk-3.8.1 regex-2023.12.25 tqdm-4.66.2
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --user -U numpy
Collecting numpy
Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl.metadata (61 kB)
Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl (15.5 MB)
Installing collected packages: numpy
WARNING: The script f2py.exe is installed in 'C:\Users\Student\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.26.4
C:\Users\Student\AppData\Local\Programs\Python\Python312>python
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> pip install gtts
File "<stdin>", line 1
pip install gtts
^^^^^^^
SyntaxError: invalid syntax
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>>

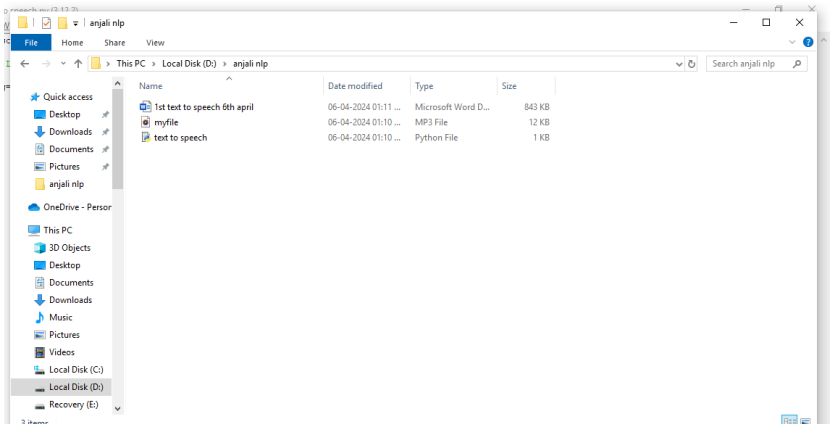

Select Command Prompt
Type "help", "copyright", "credits" or "license" for more information.
>>> pip install gtts
File "<stdin>", line 1
pip install gtts
^^^^^^^
SyntaxError: invalid syntax
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> exit()
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install gtts
Collecting gtts
Downloading gTTS-2.5.1-py3-none-any.whl.metadata (4.1 kB)
Collecting requests<3,>=2.27 (from gtts)
Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: click<8.2,>=7.1 in c:\users\student\appdata\roaming\python\python312\site-packages (from gtts) (8.1.7)
Requirement already satisfied: colorama in c:\users\student\appdata\roaming\python\python312\site-packages (from click<8.2,>=7.1->gtts) (0.4.6)
Collecting charset-normalizer<4,>=2 (from requests<3,>=2.27->gtts)
Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl.metadata (34 kB)
Collecting idna<4,>=2.5 (from requests<3,>=2.27->gtts)
Downloading idna-3.6-py3-none-any.whl.metadata (9.9 kB)
Collecting urllib3<3,>=1.21.1 (from requests<3,>=2.27->gtts)
Downloading urllib3-2.2.1-py3-none-any.whl.metadata (6.4 kB)
Collecting certifi>=2017.4.17 (from requests<3,>=2.27->gtts)
Downloading certifi-2024.2.2-py3-none-any.whl.metadata (2.2 kB)
Downloading gTTS-2.5.1-py3-none-any.whl (29 kB)
Downloading requests-2.31.0-py3-none-any.whl (62 kB)
Downloading certifi-2024.2.2-py3-none-any.whl (163 kB)
Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl (100 kB)
Downloading idna-3.6-py3-none-any.whl (61 kB)
Downloading urllib3-2.2.1-py3-none-any.whl (121 kB)
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests, gtts
Successfully installed certifi-2024.2.2 charset-normalizer-3.3.2 gtts-2.5.1 idna-3.6 requests-2.31.0 urllib3-2.2.1
C:\Users\Student\AppData\Local\Programs\Python\Python312>
ENG
01:04 PM
06-04-2024


Select Command Prompt
Downloading urllib3-2.2.1-py3-none-any.whl (121 kB)
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests, gtts
Successfully installed certifi-2024.2.2 charset-normalizer-3.3.2 gtts-2.5.1 idna-3.6 requests-2.31.0 urllib3-2.2.1
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --upgrade setuptools
Collecting setuptools
Downloading setuptools-69.2.0-py3-none-any.whl.metadata (6.3 kB)
Downloading setuptools-69.2.0-py3-none-any.whl (821 kB)
Installing collected packages: setuptools
Successfully installed setuptools-69.2.0
C:\Users\Student\AppData\Local\Programs\Python\Python312>
ENG
01:04 PM
06-04-2024


Select Command Prompt
Downloading setuptools-69.2.0-py3-none-any.whl (821 kB)
821.5/821.5 kB 10.4 MB/s eta 0:00:00
Installing collected packages: setuptools
Successfully installed setuptools-69.2.0
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --upgrade wheel
Collecting wheel
Downloading wheel-0.43.0-py3-none-any.whl.metadata (2.2 kB)
Downloading wheel-0.43.0-py3-none-any.whl (65 kB)
65.8/65.8 kB ? eta 0:00:00
Installing collected packages: wheel
Successfully installed wheel-0.43.0
C:\Users\Student\AppData\Local\Programs\Python\Python312>
ENG
01:05 PM
06-04-2024


Select Command Prompt
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests, gtts
Successfully installed certifi-2024.2.2 charset-normalizer-3.3.2 gtts-2.5.1 idna-3.6 requests-2.31.0 urllib3-2.2.1
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --upgrade setuptools
Collecting setuptools
Downloading setuptools-69.2.0-py3-none-any.whl.metadata (6.3 kB)
Downloading setuptools-69.2.0-py3-none-any.whl (821 kB)
821.5/821.5 kB 10.4 MB/s eta 0:00:00
Installing collected packages: setuptools
Successfully installed setuptools-69.2.0
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install --upgrade wheel
Collecting wheel
Downloading wheel-0.43.0-py3-none-any.whl.metadata (2.2 kB)
Downloading wheel-0.43.0-py3-none-any.whl (65 kB)
65.8/65.8 kB ? eta 0:00:00
Installing collected packages: wheel
Successfully installed wheel-0.43.0
C:\Users\Student\AppData\Local\Programs\Python\Python312>pip install playsound
Collecting playsound
Downloading playsound-1.3.0.tar.gz (7.7 kB)
Preparing metadata (setup.py) ... done
Building wheels for collected packages: playsound
Building wheel for playsound (setup.py) ... done
Created wheel for playsound: filename=playsound-1.3.0-py3-none-any.whl size=7043 sha256=77f81777abc79d31706e4abc3601d6878ef758780167b99d06c3e26b2df6ce0
Stored in directory: c:\users\student\appdata\local\pip\cache\wheels\cf\42\ff\7c587bae55eec67b909ca316b250d9b4daedbf272a3cbeb907
Successfully built playsound
Installing collected packages: playsound
Successfully installed playsound-1.3.0
C:\Users\Student\AppData\Local\Programs\Python\Python312>
ENG
01:06 PM
06-04-2024

Output:


anjali nlp
File
Home
Share
View
This PC
Local Disk (D:)
anjali nlp
Search anjali nlp
Quick access
Desktop
Downloads
Documents
Pictures
anjali nlp
OneDrive - Persor
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Recovery (E:)
Name
Date modified
Type
Size
1st text to speech 6th april
06-04-2024 01:11 ...
Microsoft Word D...
843 KB
myfile
06-04-2024 01:10 ...
MP3 File
12 KB
text to speech
06-04-2024 01:10 ...
Python File
1 KB

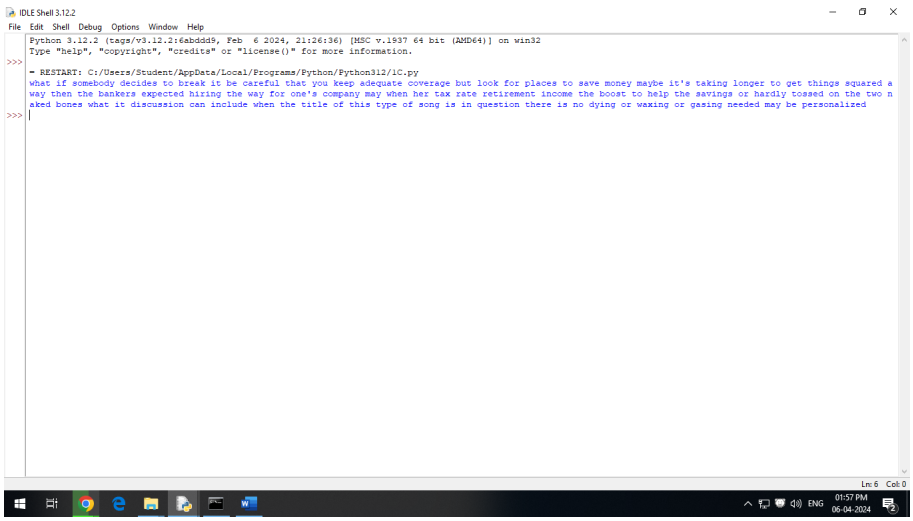
PRACTICAL 1C

Convert audio file Speech to Text.

Code:

```
import speech_recognition as sr
filename="male.wav"
r=sr.Recognizer()
with sr.AudioFile(filename)as source:
    audio_data=r.record(source)
    text=r.recognize_google(audio_data)
    print(text)
```

Output:



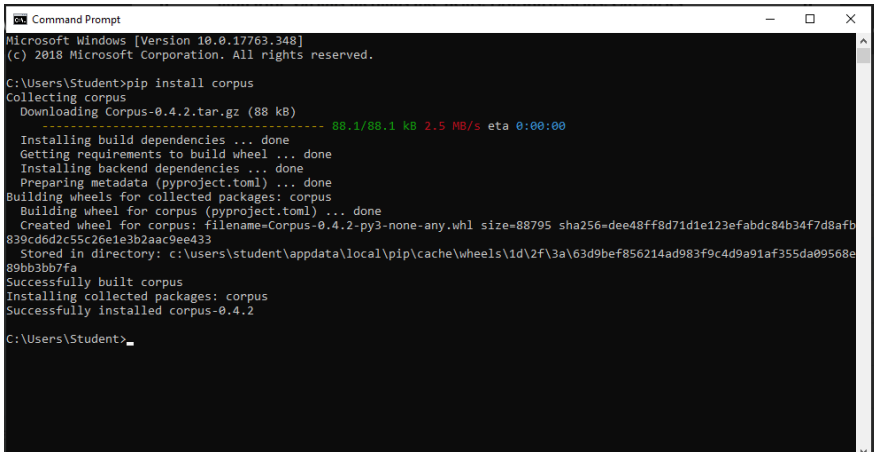
PRACTICAL 2A

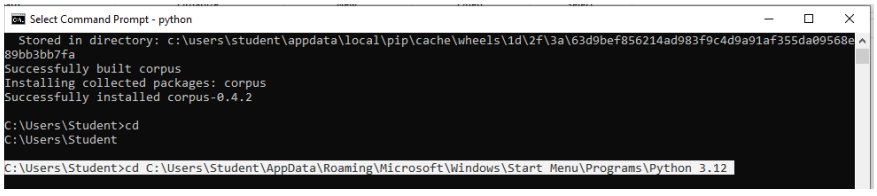
Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories,

Code:

```
import nltk

from nltk.corpus import brown

print ('File ids of brown corpus\n',brown.fileids())

'''Let's pick out the first of these texts — Emma by Jane Austen — and give it a short

name, emma, then find out how many words it contains:'''

ca01 = brown.words('ca01')

# display first few words

print('\nca01 has following words:\n',ca01)

# total number of words in ca01

print('\nca01 has',len(ca01),'words')


#categories or files

print ('\n\nCategories or file in brown corpus:\n')

print (brown.categories())

'''display other information about each text, by looping over all the values of fileid

corresponding to the brown file identifiers listed earlier and then computing statistics

for each text.'''

print ('\n\nStatistics for each text:\n')

print

('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFileName')
```

```
for fileid in brown.fileids():
    num_chars = len(brown.raw(fileid))
    num_words = len(brown.words(fileid))
    num_sents = len(brown.sents(fileid))
    num_vocab = len(set([w.lower() for w in brown.words(fileid)]))
    print (int(num_chars/num_words),'\t\t\t', int(num_words/num_sents),'\t\t\t',
int(num_words/num_vocab),'\t\t\t', fileid)
```



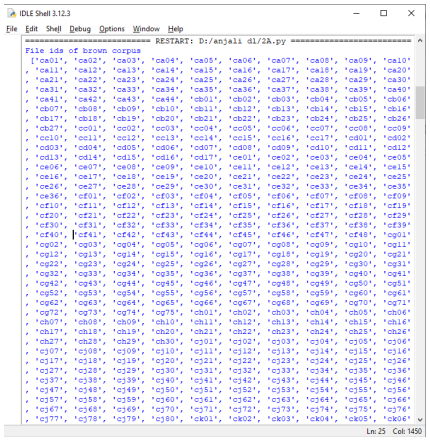


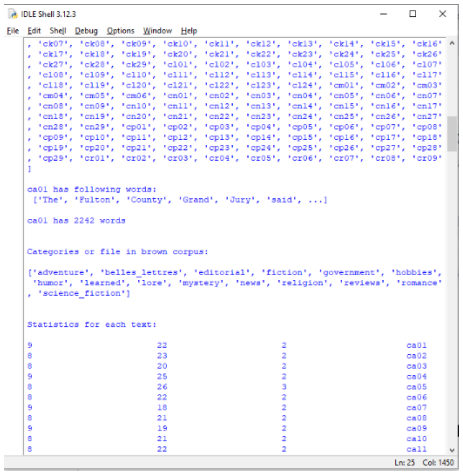
```
C:\Users\Student\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.12>pip install nltk
Collecting nltk
  Downloading nltk-3.8.1-py3-none-any.whl.metadata (2.8 kB)
Collecting click (from nltk)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting joblib (from nltk)
  Downloading joblib-1.4.0-py3-none-any.whl.metadata (5.4 kB)
Collecting regex>=2021.8.3 (from nltk)
  Downloading regex-2024.4.16-cp312-cp312-win_amd64.whl.metadata (41 kB)
     ---------------------------------------- 42.0/42.0 kB 2.1 MB/s eta 0:00:00
Collecting tqdm (from nltk)
  Downloading tqdm-4.66.2-py3-none-any.whl.metadata (57 kB)
     ---------------------------------------- 57.6/57.6 kB 3.0 MB/s eta 0:00:00
Collecting colorama (from click->nltk)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
   ---------------------------------------- 1.5/1.5 MB 6.0 MB/s eta 0:00:00
Downloading regex-2024.4.16-cp312-cp312-win_amd64.whl (268 kB)
```

```
C:\Users\Student\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.12>python
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
```

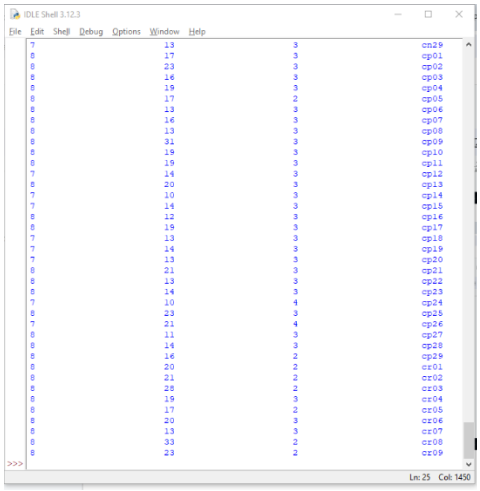
```
>>> nltk.download('brown')
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\brown.zip.
True
>>>
```

Output:




IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
, 'ck07', 'ck08', 'ck09', 'ck10', 'ck11', 'ck12', 'ck13', 'ck14', 'ck15', 'ck16'
, 'ck17', 'ck18', 'ck19', 'ck20', 'ck21', 'ck22', 'ck23', 'ck24', 'ck25', 'ck26'
, 'ck27', 'ck28', 'ck29', 'cl01', 'cl02', 'cl03', 'cl04', 'cl05', 'cl06', 'cl07'
, 'cl08', 'cl09', 'cl10', 'cl11', 'cl12', 'cl13', 'cl14', 'cl15', 'cl16', 'cl17'
, 'cl18', 'cl19', 'cl20', 'cl21', 'cl22', 'cl23', 'cl24', 'cm01', 'cm02', 'cm03'
, 'cm04', 'cm05', 'cm06', 'cn01', 'cn02', 'cn03', 'cn04', 'cn05', 'cn06', 'cn07'
, 'cn08', 'cn09', 'cn10', 'cn11', 'cn12', 'cn13', 'cn14', 'cn15', 'cn16', 'cn17'
, 'cn18', 'cn19', 'cn20', 'cn21', 'cn22', 'cn23', 'cn24', 'cn25', 'cn26', 'cn27'
, 'cn28', 'cn29', 'cp01', 'cp02', 'cp03', 'cp04', 'cp05', 'cp06', 'cp07', 'cp08'
, 'cp09', 'cp10', 'cp11', 'cp12', 'cp13', 'cp14', 'cp15', 'cp16', 'cp17', 'cp18'
, 'cp19', 'cp20', 'cp21', 'cp22', 'cp23', 'cp24', 'cp25', 'cp26', 'cp27', 'cp28'
, 'cp29', 'cr01', 'cr02', 'cr03', 'cr04', 'cr05', 'cr06', 'cr07', 'cr08', 'cr09'
]
ca01 has following words:
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
ca01 has 2242 words
Categories or file in brown corpus:
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies',
'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance'
, 'science_fiction']
Statistics for each text:
9 22 2 ca01
8 23 2 ca02
8 20 2 ca03
9 25 2 ca04
8 26 3 ca05
8 22 2 ca06
9 18 2 ca07
8 21 2 ca08
9 19 2 ca09
8 21 2 ca10
8 22 2 ca11
Ln: 25 Col: 1450


IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
8 23 2 ca02
8 20 2 ca03
9 25 2 ca04
8 26 3 ca05
8 22 2 ca06
9 18 2 ca07
8 21 2 ca08
9 19 2 ca09
8 21 2 ca10
8 22 2 ca11
8 22 2 ca12
8 20 2 ca13
8 17 2 ca14
8 21 2 ca15
8 20 2 ca16
8 22 2 ca17
8 22 2 ca18
8 20 2 ca19
8 23 2 ca20
9 20 2 ca21
8 21 2 ca22
8 20 2 ca23
8 22 2 ca24
8 21 2 ca25
8 21 2 ca26
8 19 2 ca27
8 23 2 ca28
8 19 2 ca29
8 19 2 ca30
8 20 2 ca31
8 18 2 ca32
8 24 2 ca33
8 23 3 ca34
8 29 3 ca35
9 25 2 ca36
8 24 2 ca37
8 23 3 ca38
8 19 2 ca39
8 24 2 ca40
8 22 2 ca41
Ln: 25 Col: 1450


IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
7 13 3 cn29
8 17 3 cp01
8 23 3 cp02
8 16 3 cp03
8 19 3 cp04
8 17 2 cp05
8 13 3 cp06
8 16 3 cp07
8 13 3 cp08
8 31 3 cp09
8 19 3 cp10
8 19 3 cp11
7 14 3 cp12
8 20 3 cp13
7 10 3 cp14
7 14 3 cp15
8 12 3 cp16
8 19 3 cp17
7 13 3 cp18
7 14 3 cp19
7 13 3 cp20
8 21 3 cp21
8 13 3 cp22
8 14 3 cp23
7 10 4 cp24
8 23 3 cp25
7 21 4 cp26
8 11 3 cp27
8 14 3 cp28
8 16 2 cp29
8 20 2 cr01
8 21 2 cr02
8 28 2 cr03
8 19 3 cr04
8 17 2 cr05
8 20 3 cr06
8 13 3 cr07
8 33 2 cr08
8 23 2 cr09
>>>
Ln: 25 Col: 1450

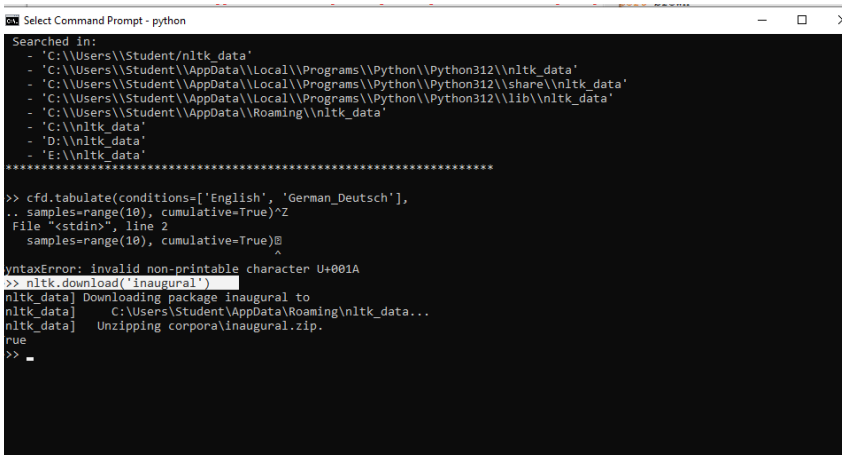
PRACTICAL 2C

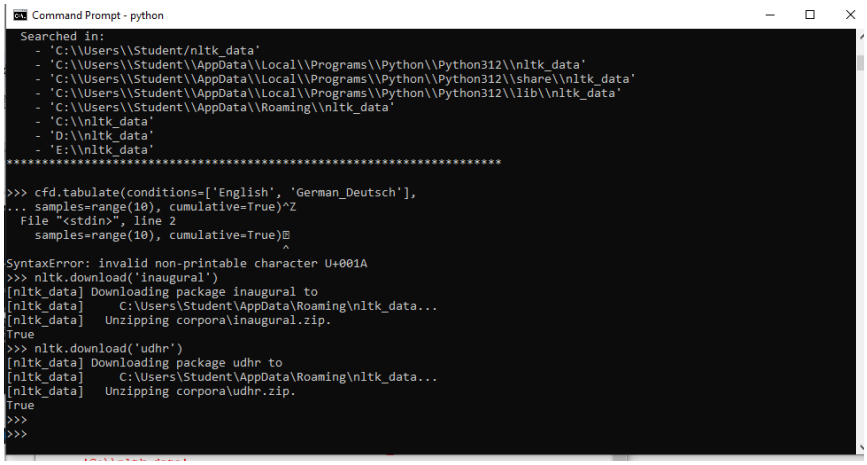
Study Conditional frequency distributions

Code:

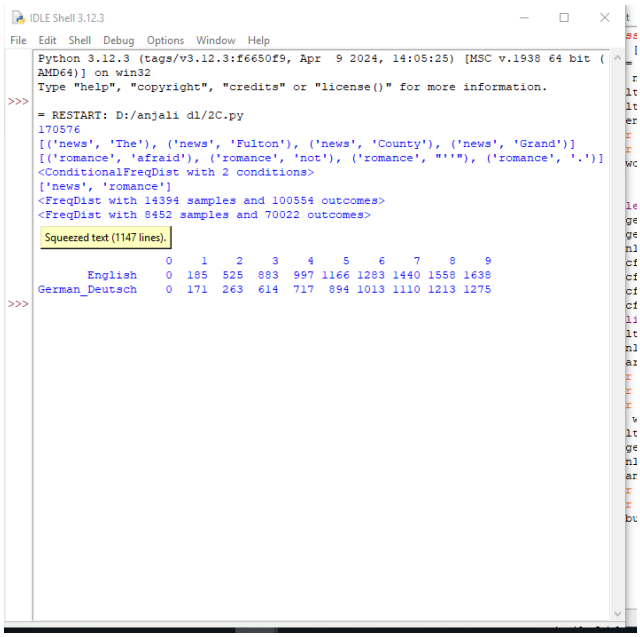
```
#process a sequence of pairs
text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
pairs = [('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ...]
import nltk
from nltk.corpus import brown
fd = nltk.ConditionalFreqDist(
   (genre, word)
   for genre in brown.categories()
   for word in brown.words(categories=genre))
genre_word = [(genre, word)
            for genre in ['news', 'romance']
            for word in brown.words(categories=genre)]
print(len(genre_word))
print(genre_word[:4])
print(genre_word[-4:])
cfd = nltk.ConditionalFreqDist(genre_word)
print(cfd)
print(cfd.conditions())
print(cfd['news'])
print(cfd['romance'])
print(list(cfd['romance']))
from nltk.corpus import inaugural
cfd = nltk.ConditionalFreqDist(
   (target, fileid[:4])
   for fileid in inaugural.fileids()
```

```
    for w in inaugural.words(fileid)
    for target in ['america', 'citizen']
    if w.lower().startswith(target))
from nltk.corpus import udhr
languages = ['Chickasaw', 'English', 'German_Deutsch','Greenlandic_Inuktikut',
'Hungarian_Magyar', 'Ibibio_Efik']
cfd = nltk.ConditionalFreqDist(
    (lang, len(word))
    for lang in languages
    for word in udhr.words(lang + '-Latin1'))
cfd.tabulate(conditions=['English', 'German_Deutsch'],
            samples=range(10), cumulative=True)
```




Command Prompt - python
Searched in:
- 'C:\\Users\\Student/nltk_data'
- 'C:\\Users\\Student\\AppData\\Local\\Programs\\Python\\Python312\\nltk_data'
- 'C:\\Users\\Student\\AppData\\Local\\Programs\\Python\\Python312\\share\\nltk_data'
- 'C:\\Users\\Student\\AppData\\Local\\Programs\\Python\\Python312\\lib\\nltk_data'
- 'C:\\Users\\Student\\AppData\\Roaming\\nltk_data'
- 'C:\\nltk_data'
- 'D:\\nltk_data'
- 'E:\\nltk_data'
****
>>> cfd.tabulate(conditions=['English', 'German_Deutsch'],
... samples=range(10), cumulative=True)^Z
File "<stdin>", line 2
samples=range(10), cumulative=True)
^
SyntaxError: invalid non-printable character U+001A
>>> nltk.download('inaugural')
[nltk_data] Downloading package inaugural to
[nltk_data] C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\inaugural.zip.
True
>>> nltk.download('udhr')
[nltk_data] Downloading package udhr to
[nltk_data] C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\udhr.zip.
True
>>>
>>>

Output:


IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/anjali dl/2C.py
170576
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]
[('romance', 'afraid'), ('romance', 'not'), ('romance', "'"), ('romance', '.')]
<ConditionalFreqDist with 2 conditions>
['news', 'romance']
<FreqDist with 14394 samples and 100554 outcomes>
<FreqDist with 8452 samples and 70022 outcomes>
Squeezed text (1147 lines).
0 1 2 3 4 5 6 7 8 9
English 0 185 525 883 997 1166 1283 1440 1558 1638
German_Deutsch 0 171 263 614 717 894 1013 1110 1213 1275
>>>

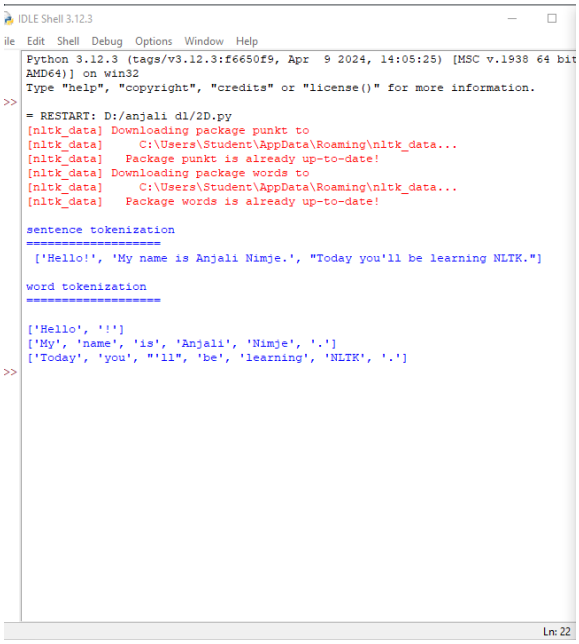
PRACTICAL 2D

Study of tagged corpora with methods like tagged_sents, tagged_words.

Code:

```
import nltk
from nltk import tokenize
nltk.download('punkt')
nltk.download('words')
para = "Hello! My name is Anjali Nimje. Today you'll be learning NLTK."
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n==================\n",sents)
# word tokenization
print("\nword tokenization\n==================\n")
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)
```

Output:

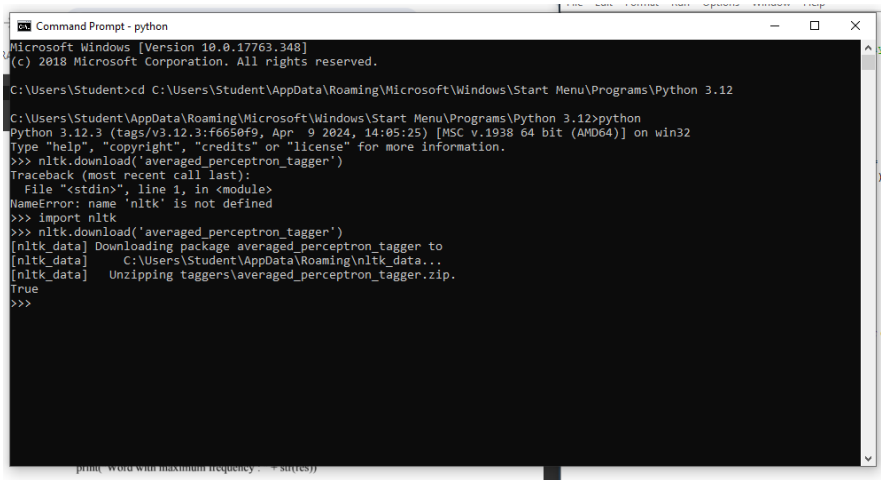


PRACTICAL 2E

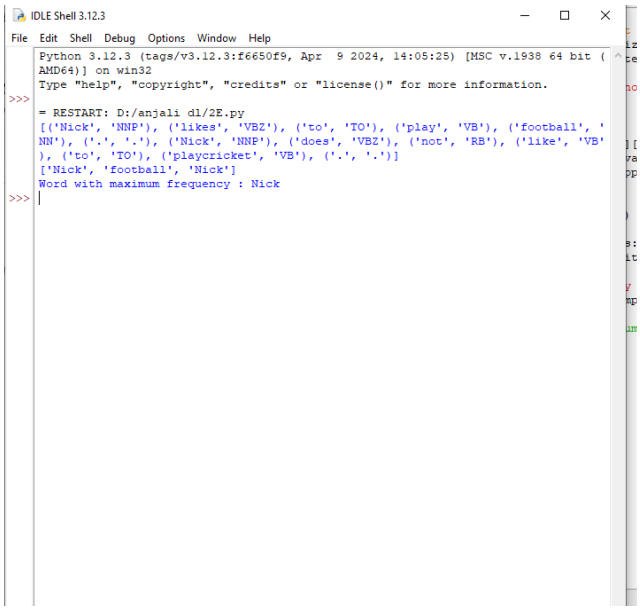
Write a program to find the most frequent noun tags.

Code:

```
import nltk
from collections import defaultdict
text = nltk.word_tokenize("Nick likes to play football. Nick does not like to playcricket.")
tagged = nltk.pos_tag(text)
print(tagged)
# checking if it is a noun or not
addNounWords = []
count=0
for words in tagged:
    val = tagged[count][1]
    if(val == 'NN' or val == 'NNS' or val == 'NNPS' or val == 'NNP'):
        addNounWords.append(tagged[count][0])
    count+=1
print (addNounWords)
temp = defaultdict(int)
# memoizing count
for sub in addNounWords:
    for wrd in sub.split():
        temp[wrd] += 1
# getting max frequency
res = max(temp, key=temp.get)
# printing result
print("Word with maximum frequency : " + str(res))
```


Command Prompt - python
Microsoft Windows [Version 10.0.17763.348]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Student>cd C:\Users\Student\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.12
C:\Users\Student\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.12>python
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> nltk.download('averaged_perceptron_tagger')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'nltk' is not defined
>>> import nltk
>>> nltk.download('averaged_perceptron_tagger')
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data] Unzipping taggers\averaged_perceptron_tagger.zip.
True
>>>

Output:


IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/anjali dl/2E.py
[('Nick', 'NNP'), ('likes', 'VBZ'), ('to', 'TO'), ('play', 'VB'), ('football', 'NN'), ('.', '.'), ('Nick', 'NNP'), ('does', 'VBZ'), ('not', 'RB'), ('like', 'VB'), ('to', 'TO'), ('playcricket', 'VB'), ('.', '.')]
['Nick', 'football', 'Nick']
Word with maximum frequency : Nick
>>>

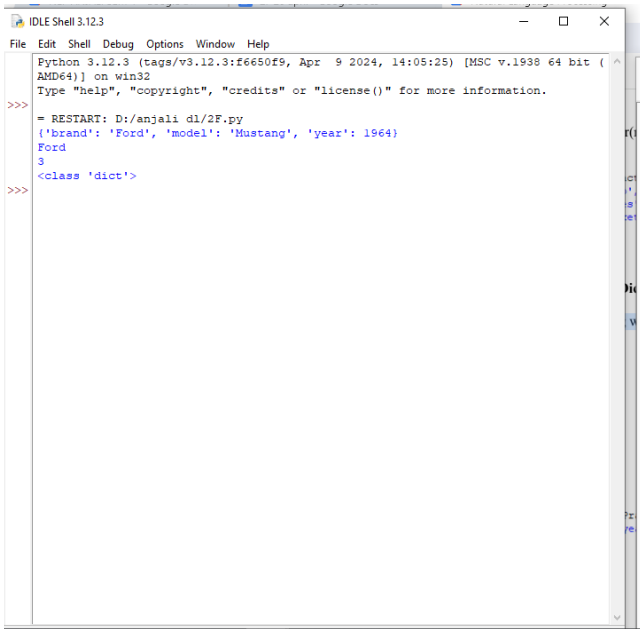
PRACTICAL 2F

Map Words to Properties Using Python Dictionaries

Code:

```
#creating and printing a dictionay by mapping word with its properties
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```

Output



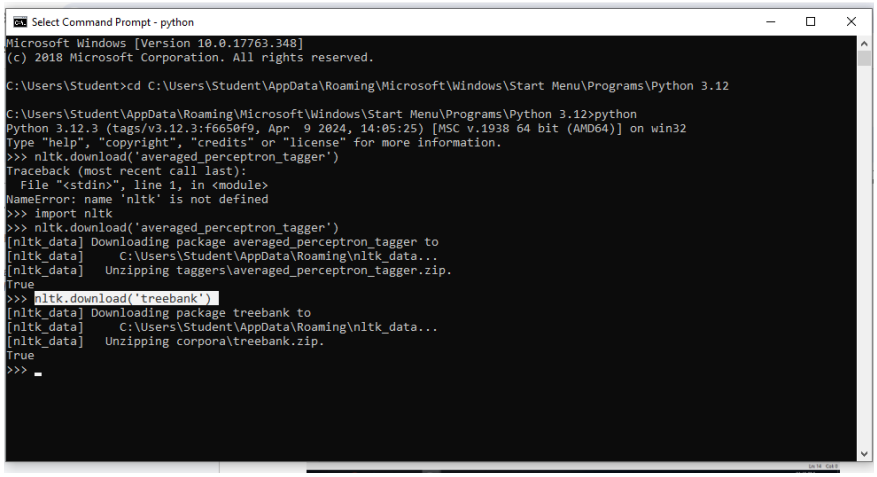
PRACTICAL 2G

Study DefaultTagger, Regular expression tagger, UnigramTagger

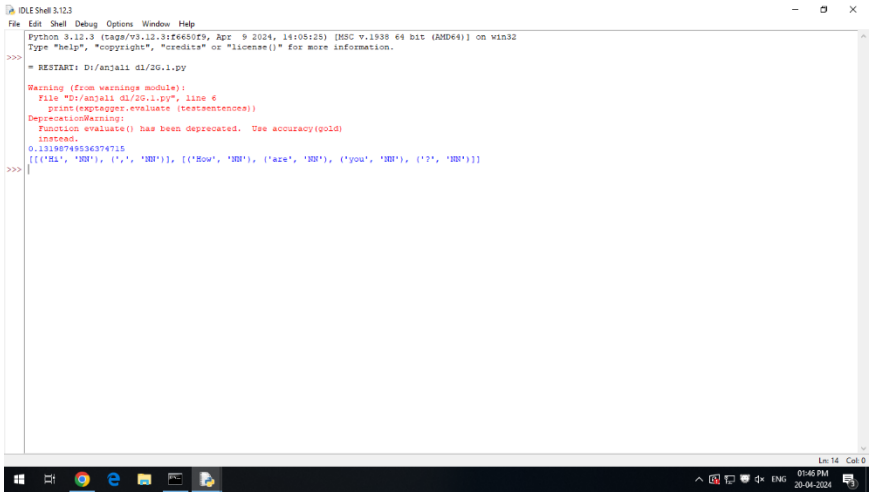
A)Default Tagger

Code:

```
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
from nltk.corpus import treebank
testsentences = treebank.tagged_sents() [1000:]
print(exptagger.evaluate (testsentences))
#Tagging a list of sentences
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
print(exptagger.tag_sents([['Hi', ','], ['How', 'are', 'you', '?']]))
```



Output:

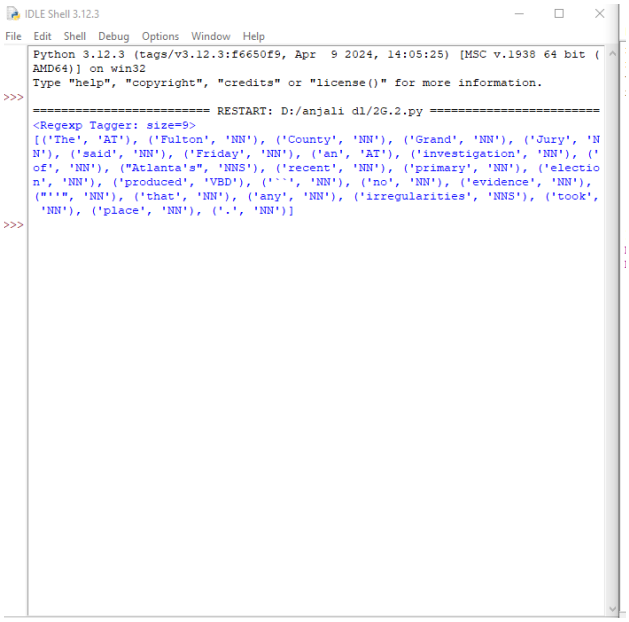

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/anjali dl/2G.1.py
Warning (from warnings module):
File "D:/anjali dl/2G.1.py", line 6
print(exptagger.evaluate(testsentences))
DeprecationWarning:
Function evaluate() has been deprecated. Use accuracy(gold)
instead.
0.13190749536374715
[[('Hi', 'NN'), (',', 'NN')], [('How', 'NN'), ('are', 'NN'), ('you', 'NN'), ('?', 'NN')]]
>>>
Ln: 14 Col: 0
ENG
01:46 PM
20-04-2024

B)Regular Expressions

Code:

```
from nltk.corpus import brown
from nltk.tag import RegexpTagger
test_sent = brown.sents(categories='news')[0]
regexp_tagger = RegexpTagger(
    [(r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers
    (r'(The|the|A|a|An|an)$', 'AT'), # articles
    (r'.*able$', 'JJ'), # adjectives
    (r'.*ness$', 'NN'), # nouns formed from adjectives
    (r'.*ly$', 'RB'), # adverbs
    (r'.*s$', 'NNS'), # plural nouns
    (r'.*ing$', 'VBG'), # gerunds
    (r'.*ed$', 'VBD'), # past tense verbs
    (r'.*', 'NN') # nouns (default)
])
print(regexp_tagger)
print(regexp_tagger.tag(test_sent))
```

Output:

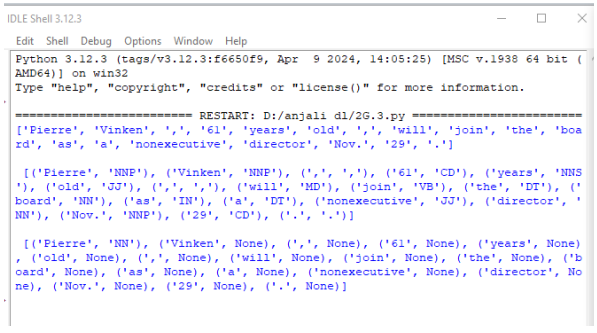

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/anjali dl/2G.2.py ==========
<Regexp Tagger: size=9>
[('The', 'AT'), ('Fulton', 'NN'), ('County', 'NN'), ('Grand', 'NN'), ('Jury', 'N
N'), ('said', 'NN'), ('Friday', 'NN'), ('an', 'AT'), ('investigation', 'NN'), ('
of', 'NN'), ("Atlanta's", 'NNS'), ('recent', 'NN'), ('primary', 'NN'), ('electio
n', 'NN'), ('produced', 'VBD'), ('``', 'NN'), ('no', 'NN'), ('evidence', 'NN'),
("''", 'NN'), ('that', 'NN'), ('any', 'NN'), ('irregularities', 'NNS'), ('took',
'NN'), ('place', 'NN'), ('.', 'NN')]
>>>

C) Unigram Tagger

Code:

```
# Loading Libraries
from nltk.tag import UnigramTagger
from nltk.corpus import treebank
# Training using first 10 tagged sentences of the treebank corpus as data.
# Using data
train_sents = treebank.tagged_sents()[:10]
# Initializing
tagger = UnigramTagger(train_sents)
# Lets see the first sentence
# (of the treebank corpus) as list
print(treebank.sents()[0])
print('\n',tagger.tag(treebank.sents()[0]))
#Finding the tagged results after training.
tagger.tag(treebank.sents()[0])
#Overriding the context model
tagger = UnigramTagger(model ={'Pierre': 'NN'})
print('\n',tagger.tag(treebank.sents()[0]))
```

Output:



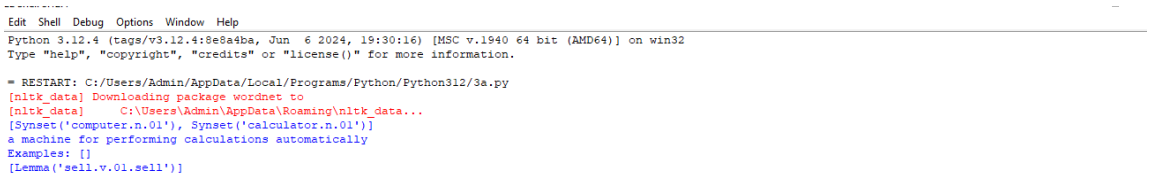
PRACTICAL 3A

Study of Wordnet Dictionary with methods as synsets, definitions, examples,antonyms.

Code:

```
import nltk

nltk.download('wordnet')

from nltk.corpus import wordnet

print(wordnet.synsets("computer"))

print(wordnet.synset("computer.n.01").definition())

print("Examples:", wordnet.synset("computer.n.01").examples())

print(wordnet.lemma('buy.v.01.buy').antonyms())
```

Output:



PRACTICAL 3B

Study lemmas, hyponyms, hypernyms.

Code:

```
import nltk
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
print(wordnet.synset("computer.n.01").lemma_names())
for e in wordnet.synsets("computer"):
print(f'{e} --> {e.lemma_names()}')
print(wordnet.synset("computer.n.01").lemmas())
print(wordnet.lemma('computer.n.01.computing_device').synset())
print(wordnet.lemma('computer.n.01.computing_device').name())
syn = wordnet.synset('computer.n.01')
print(syn.hyponyms)
print([lemma.name() for synset in syn.hyponyms() for lemma in
synset.lemmas()])
vehicle = wordnet.synset('vehicle.n.01')car = wordnet.synset('car.n.01')
print(car.lowest_common_hypernyms(vehicle))
```

Output:

```
>>>
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/3b.py ==========
[Synset('computer.n.01'), Synset('calculator.n.01')]
['computer', 'computing_machine', 'computing_device', 'data_processor', 'electronic_computer', 'information_processing_system']
Synset('computer.n.01') --> ['computer', 'computing_machine', 'computing_device', 'data_processor', 'electronic_computer', 'information_processing_system']
Synset('calculator.n.01') --> ['calculator', 'reckoner', 'figurer', 'estimator', 'computer']
[Lemma('computer.n.01.computer'), Lemma('computer.n.01.computing_machine'), Lemma('computer.n.01.computing_device'), Lemma('computer.n.01.data_processor'), Lemma('
computer.n.01.electronic_computer'), Lemma('computer.n.01.information_processing_system')]
Synset('computer.n.01')
computing_device
<bound method _WordNetObject.hyponyms of Synset('computer.n.01')>
['analog_computer', 'analogue_computer', 'digital_computer', 'home_computer', 'node', 'client', 'guest', 'number_cruncher', 'pari-mutuel_machine', 'totalizer', 'to
taliser', 'totalizator', 'totalisator', 'predictor', 'server', 'host', 'Turing_machine', 'web_site', 'website', 'internet_site', 'site']
[Synset('vehicle.n.01')]
>>>
```

PRACTICAL 3C

Write a program using python to find synonym and antonym of word "active" using Wordnet.

Code:

```
from nltk.corpus import wordnet

print(wordnet.synsets("active"))

print(wordnet.lemma('active.a.01.active').antonyms())
```

Output:

```
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/3c.py ==========
[Synset('active_agent.n.01'), Synset('active_voice.n.01'), Synset('active.n.03'), Synset('active.a.01'), Synset('active.s.02'), Synset('active.a.03'), Synset('acti
ve.s.04'), Synset('active.a.05'), Synset('active.a.06'), Synset('active.a.07'), Synset('active.s.08'), Synset('active.a.09'), Synset('active.a.10'), Synset('active
.a.11'), Synset('active.a.12'), Synset('active.a.13'), Synset('active.a.14')]
[Lemma('inactive.a.02.inactive')]
>>>
```

PRACTICAL 3D

Compare two nouns.

Code:

```
import nltk
from nltk.corpus import wordnet
syn1 = wordnet.synsets('football')
syn2 = wordnet.synsets('soccer')
for s1 in syn1:
    for s2 in syn2:
        print("path similarity of:")
        print(s1,'(',s1.pos(),')','[',s1.definition(),']')
        print(s2,'(',s2.pos(),')','[',s2.definition(),']')
        print("is", s1.path_similarity(s2))
        print()
```

Output:

```
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/3d.py ==========
path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with a ball (round or oval) in which two teams try to kick or carry or propel the ball into each other's goal ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players try to kick or head a ball into the opponents' goal ]
is 0.5

path similarity of:
Synset('football.n.02') ( n ) [ the inflated oblong ball used in playing American football ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players try to kick or head a ball into the opponents' goal ]
is 0.05

>>>
```

PRACTICAL 3E

Handling stopword

A) Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List

Code:

```
import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

from nltk.tokenize import word_tokenize

text = "Yashesh like to play football, however he is not too fond of tennis."

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]

print(tokens_without_sw)

all_stopwords = stopwords.words('english')all_stopwords.append('play')

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]

print(tokens_without_sw)

all_stopwords.remove('not')

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]

print(tokens_without_sw)
```

Output:

```
>>>
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/3e1.py ==========
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['Yashesh', 'play', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'like', 'football', ',', 'however', 'fond', 'tennis', '.']
['Yashesh', 'like', 'football', ',', 'however', 'not', 'fond', 'tennis', '.']
>>>
                                                                                Ln: 5
```

B) Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List

Code:

```
import gensim
from gensim.parsing.preprocessing import remove_stopwords
from nltk.tokenize import word_tokenize
text = "Yashesh likes to play football, however he is not too fond of tennis."
filtered_sentense = remove_stopwords(text)
print(filtered_sentense)
all_stpwords = gensim.parsing.preprocessing.STOPWORDS
print( all_stpwords)
from gensim.parsing.preprocessing import STOPWORDS
all_stpwords_genism = STOPWORDS.union(set(['likes','play']))
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stpwords]print(tokens_without_sw)
print("==========================================")
all_stpwords_genism = STOPWORDS
sw_list = {"not"}
all_stpwords_genism = STOPWORDS.difference(sw_list)
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stpwords_genism]
print(all_stpwords_genism)
print(tokens_without_sw)
```

```
C:\Windows\system32\cmd.exe

C:\Users\Admin>pip install gensim
Collecting gensim
  Downloading gensim-4.3.3-cp312-cp312-win_amd64.whl.metadata (8.2 kB)
Collecting numpy<2.0,>=1.18.5 (from gensim)
  Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl.metadata (61 kB)
     ---------------------------------------- 61.0/61.0 kB 651.3 kB/s eta 0:00:00
Collecting scipy<1.14.0,>=1.7.0 (from gensim)
  Downloading scipy-1.13.1-cp312-cp312-win_amd64.whl.metadata (60 kB)
     ---------------------------------------- 60.6/60.6 kB 799.0 kB/s eta 0:00:00
Collecting smart-open>=1.8.1 (from gensim)
  Downloading smart_open-7.0.4-py3-none-any.whl.metadata (23 kB)
Collecting wrapt (from smart-open>=1.8.1->gensim)
  Downloading wrapt-1.16.0-cp312-cp312-win_amd64.whl.metadata (6.8 kB)
Downloading gensim-4.3.3-cp312-cp312-win_amd64.whl (24.0 MB)
     ---------------------------------------- 24.0/24.0 MB 637.9 kB/s eta 0:00:00
Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl (15.5 MB)
     ---------------------------------------- 15.5/15.5 MB 681.6 kB/s eta 0:00:00
Downloading scipy-1.13.1-cp312-cp312-win_amd64.whl (45.9 MB)
     ---------------------------------------- 45.9/45.9 MB 815.6 kB/s eta 0:00:00
Downloading smart_open-7.0.4-py3-none-any.whl (61 kB)
     ---------------------------------------- 61.2/61.2 kB 542.3 kB/s eta 0:00:00
Downloading wrapt-1.16.0-cp312-cp312-win_amd64.whl (37 kB)
Installing collected packages: wrapt, numpy, smart-open, scipy, gensim
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.1
    Uninstalling numpy-2.0.1:
      Successfully uninstalled numpy-2.0.1
  WARNING: Failed to remove contents in a temporary directory 'C:\Users\Admin\AppData\Roaming\Python\Python312\site-packages\~umpy.libs'.
  You can safely remove it manually.
  WARNING: Failed to remove contents in a temporary directory 'C:\Users\Admin\AppData\Roaming\Python\Python312\site-packages\~umpy'.
  You can safely remove it manually.
Successfully installed gensim-4.3.3 numpy-1.26.4 scipy-1.13.1 smart-open-7.0.4 wrapt-1.16.0

C:\Users\Admin>
```

Output:

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/3e2.py ==========
Yashesh likes play football, fond tennis.
frozenset({'we', 'your', 'though', 'such', 'yourselves', 'very', 'but', 'computer', 'whoever', 'itself', 'is', 'moreover', 'four', 'other', 'the', 'wherein', 'that
', 'thick', 'many', 'might', 'within', 'hundred', 'please', 'eg', 'from', 'seeming', 'fifteen', 'thereafter', 'five', 'herein', 'about', 'own', 'serious', 'namely'
, 'onto', 'least', 'she', 'neither', 'if', 'during', 'everyone', 'for', 'even', 'under', 'sixty', 'whatever', 'am', 'de', 'either', 'i', 'found', 'back', 'another'
, 'doing', 'system', 'can', 'should', 'thereby', 'just', 'sometime', 'latter', 'each', 'ours', 'nobody', 'how', 'you', 'do', 'whenever', 'yourself', 'whereas', 'fi
fty', 'thence', 'and', 'detail', 'into', 'regarding', 'put', 'most', 'via', 'also', 'often', 'empty', 'un', 'one', 'get', 'than', 'among', 'however', 'none', 'here
after', 'anywhere', 'anyway', 'against', 'eight', 'although', 'km', 'nowhere', 'due', 'above', 'her', 'could', 'call', 'whereafter', 'twelve', 'it', 'keep', 'forme
rly', 'make', 'cry', 'become', 'along', 'once', 'becoming', 'whole', 'amougst', 'thereupon', 'take', 'without', 'some', 'now', 'herself', 'using', 'there', 'us',
'being', 'rather', 'not', 'seemed', 'became', 'enough', 'mill', 'almost', 'thus', 'afterwards', 'well', 'except', 'after', 'over', 'everywhere', 'bottom', 'by', 'e
verything', 'perhaps', 'hereupon', 'further', 'both', 'whether', 'him', 'thin', 'otherwise', 'last', 'myself', 'same', 'hereby', 'hers', 'used', 'ourselves', 'alre
ady', 'thru', 'never', 'part', 'done', 'an', 'so', 'first', 'alone', 'go', 'may', 'must', 'side', 'less', 'cn', 'then', 'somehow', 'don', 'were', 'be', 'inc', 'no'
, 'beforehand', 'whither', 'next', 'their', 'seem', 'while', 'nevertheless', 'his', 'around', 'besides', 'at', 'when', 'various', 'say', 'together', 'why', 'move',
'others', 'therefore', 'upon', 'too', 'toward', 'who', 'my', 'fire', 'anyone', 'hasnt', 'since', 'sincere', 'wherever', 'bill', 'every', 'whereby', 'third', 'does'
, 'has', 'towards', 'throughout', 'until', 'whence', 'two', 'beyond', 'becomes', 'anyhow', 'down', 'someone', 'as', 'much', 'con', 'forty', 'quite', 'find', 'himse
lf', 'twenty', 'a', 'whereupon', 'give', 'would', 'mostly', 'here', 'yet', 'name', 'its', 'nine', 'what', 'six', 'this', 'or', 'because', 'had', 'was', 'eleven', '
beside', 'former', 'only', 'in', 'front', 're', 'whose', 'out', 'unless', 'be', 'these', 'meanwhile', 'below', 'describe', 'are', 'elsewhere', 'amongst', 'more', '
always', 'behind', 'of', 'cannot', 'to', 'show', 'been', 'off', 'up', 'interest', 'latterly', 'will', 'mine', 'anything', 'really', 'ie', 'again', 'me', 'didn', 't
hey', 'fill', 'noone', 'our', 'etc', 'them', 'hence', 'full', 'kg', 'else', 'several', 'somewhere', 'those', 'with', 'co', 'sometimes', 'indeed', 'nor', 'did', 'it
d', 'see', 'amount', 'through', 'where', 'whom', 'still', 'all', 'themselves', 'ten', 'yours', 'ever', 'something', 'before', 'therein', 'per', 'doesn', 'which', '
between', 'cant', 'couldnt', 'seems', 'top', 'made', 'few', 'any', 'nothing', across', 'three', 'have'})
['Yashesh', 'likes', 'play', 'football', ',', 'fond', 'tennis', '.']
>>> ========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/3e2.py ==========
frozenset({'we', 'your', 'don', 'though', 'were', 'he', 'such', 'yourselves', 'inc', 'very', 'but', 'no', 'beforehand', 'whither', 'next', 'computer', 'whoever', '
itself', 'is', 'their', 'seem', 'moreover', 'while', 'four', 'other', 'nevertheless', 'the', 'his', 'around', 'wherein', 'that', 'besides', 'at', 'when', 'thick',
'many', 'various', 'might', 'within', 'hundred', 'say', 'together', 'why', 'move', 'others', 'therefore', 'please', 'eg', 'from', 'upon', 'seeming', 'fifteen', 'th
ereafter', 'five', 'too', 'toward', 'who', 'herein', 'about', 'my', 'fire', 'own', 'serious', 'anyone', 'namely', 'onto', 'hasnt', 'least', 'since', 'she', 'sincer
e', 'neither', 'wherever', 'if', 'during', 'everyone', 'for', 'bill', 'even', 'every', 'whereby', 'third', 'under', 'does', 'has', 'sixty', 'whatever', 'am', 'de',
'either', 'towards', 'throughout', 'until', 'whence', 'two', 'beyond', 'becomes', 'anyhow', 'down', 'i', 'found', 'someone', 'back', 'another', 'doing', 'as', 'sys
tem', 'can', 'much', 'con', 'forty', 'should', 'thereby', 'just', 'quite', 'sometime', 'find', 'latter', 'each', 'ours', 'himself', 'twenty', 'nobody', 'how', 'a',
'you', 'whereupon', 'give', 'do', 'whenever', 'would', 'mostly', 'here', 'yourself', 'yet', 'name', 'whereas', 'its', 'nine', 'what', 'six', 'fifty', 'thence', 'th
is', 'and', 'detail', 'or', 'into', 'regarding', 'because', 'had', 'was', 'put', 'most', 'eleven', 'via', 'beside', 'also', 'often', 'empty', 'former', 'only', 'un
', 'in', 'front', 're', 'whose', 'out', 'one', 'unless', 'get', 'than', 'among', 'be', 'these', 'meanwhile', 'below', 'describe', 'however', 'none', 'are', 'hereaf
ter', 'anywhere', 'anyway', 'elsewhere', 'against', 'eight', 'although', 'km', 'amongst', 'nowhere', 'more', 'always', 'due', 'behind', 'above', 'of', 'her', 'cann
ot', 'to', 'show', 'could', 'been', 'call', 'off', 'whereafter', 'up', 'twelve', 'interest', 'it', 'keep', 'formerly', 'make', 'cry', 'become', 'along', 'once', 'b
ecoming', 'whole', 'latterly', 'amougst', 'will', 'mine', 'anything', 'really', 'ie', 'thereupon', 'again', 'take', 'me', 'without', 'some', 'didn', 'they', 'now'
, 'fill', 'noone', 'our', 'etc', 'them', 'hence', 'full', 'kg', 'herself', 'else', 'using', 'several', 'there', 'us', 'being', 'rather', 'seemed', 'somewhere', 'be
came', 'those', 'with', 'co', 'enough', 'sometimes', 'mill', 'almost', 'indeed', 'not', 'thus', 'afterwards', 'well', 'did', 'except', 'after', 'over', 'ltd', 'see
', 'amount', 'through', 'everywhere', 'bottom', 'where', 'whom', 'by', 'still', 'everything', 'perhaps', 'hereupon', 'all', 'further', 'both', 'whether', 'themselv
es', 'ten', 'him', 'thin', 'yours', 'otherwise', 'last', 'myself', 'ever', 'same', 'hereby', 'something', 'before', 'therein', 'per', 'doesn', 'hers', 'used', 'whi
ch', 'between', 'ourselves', 'cant', 'already', 'thru', 'couldnt', 'never', 'part', 'done', 'an', 'so', 'seems', 'top', 'first', 'made', 'few', 'alone', 'any', 'no
thing', 'go', 'may', 'must', 'side', 'across', 'less', 'on', 'three', 'then', 'somehow', 'have'})
['Yashesh', 'likes', 'play', 'football', ',', 'not', 'fond', 'tennis', '.']
>>>
                                                                                                                                                  Ln: 69   Col: 0
```

C) Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List.

Code:

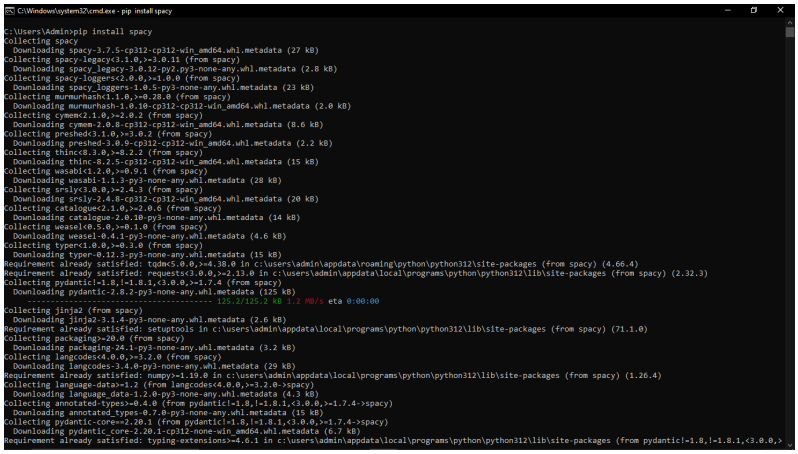
```
import spacy

import nltk

from nltk.tokenize import word_tokenize

sp = spacy.load('en_core_web_sm')

all_stopwords = sp.Defaults.stop_words

all_stopwords.add('play')

text = "Yashesh like to play football, however he is not too fond of tennis."

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in
all_stopwords]

print(tokens_without_sw)

all_stopwords.remove('not')

tokens_without_sw = [word for word in text_tokens if not word in
all_stopwords]

print(tokens_without_sw)
```

Commands:

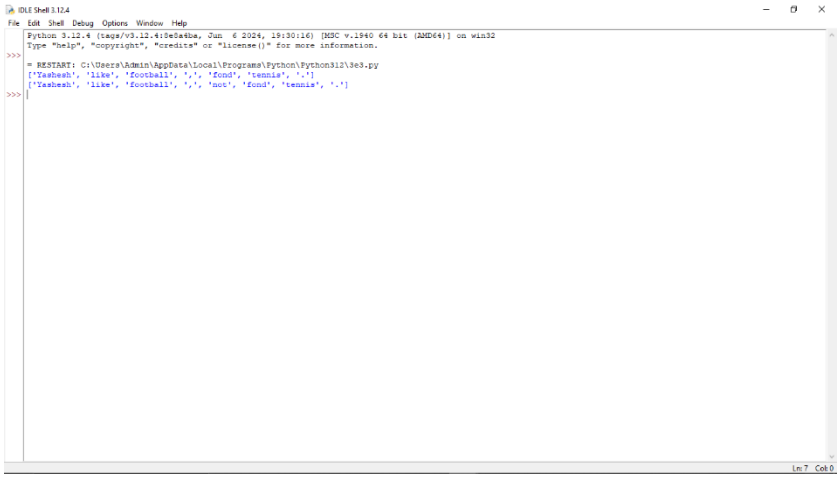
#pip install spacy

#python -m spacy download en_core_web_sm

#python -m spacy download en


C:\Windows\system32\cmd.exe - pip install spacy
C:\Users\Admin>pip install spacy
Collecting spacy
Downloading spacy-3.7.5-cp312-cp312-win_amd64.whl.metadata (27 kB)
Collecting spacy-legacy<3.1.0,>=3.0.11 (from spacy)
Downloading spacy_legacy-3.0.12-py2.py3-none-any.whl.metadata (2.8 kB)
Collecting spacy-loggers<2.0.0,>=1.0.0 (from spacy)
Downloading spacy_loggers-1.0.5-py3-none-any.whl.metadata (23 kB)
Collecting murmurhash<1.1.0,>=0.28.0 (from spacy)
Downloading murmurhash-1.0.10-cp312-cp312-win_amd64.whl.metadata (2.0 kB)
Collecting cymem<2.1.0,>=2.0.2 (from spacy)
Downloading cymem-2.0.8-cp312-cp312-win_amd64.whl.metadata (8.6 kB)
Collecting preshed<3.1.0,>=3.0.2 (from spacy)
Downloading preshed-3.0.9-cp312-cp312-win_amd64.whl.metadata (2.2 kB)
Collecting thinc<8.3.0,>=8.2.2 (from spacy)
Downloading thinc-8.2.5-cp312-cp312-win_amd64.whl.metadata (15 kB)
Collecting wasabi<1.2.0,>=0.9.1 (from spacy)
Downloading wasabi-1.1.3-py3-none-any.whl.metadata (28 kB)
Collecting srsly<3.0.0,>=2.4.3 (from spacy)
Downloading srsly-2.4.8-cp312-cp312-win_amd64.whl.metadata (20 kB)
Collecting catalogue<2.1.0,>=2.0.6 (from spacy)
Downloading catalogue-2.0.10-py3-none-any.whl.metadata (14 kB)
Collecting weasel<0.5.0,>=0.1.0 (from spacy)
Downloading weasel-0.4.1-py3-none-any.whl.metadata (4.6 kB)
Collecting typer<1.0.0,>=0.3.0 (from spacy)
Downloading typer-0.12.3-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\admin\appdata\roaming\python\python312\site-packages (from spacy) (4.66.4)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from spacy) (2.32.3)
Collecting pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 (from spacy)
Downloading pydantic-2.8.2-py3-none-any.whl.metadata (125 kB)
125.2/125.2 kB 1.2 MB/s eta 0:00:00
Collecting jinja2 (from spacy)
Downloading jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: setuptools in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from spacy) (71.1.0)
Collecting packaging>=20.0 (from spacy)
Downloading packaging-24.1-py3-none-any.whl.metadata (3.2 kB)
Collecting langcodes<4.0.0,>=3.2.0 (from spacy)
Downloading langcodes-3.4.0-py3-none-any.whl.metadata (29 kB)
Requirement already satisfied: numpy>=1.19.0 in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from spacy) (1.26.4)
Collecting language-data>=1.2 (from langcodes<4.0.0,>=3.2.0->spacy)
Downloading language_data-1.2.0-py3-none-any.whl.metadata (4.3 kB)
Collecting annotated-types>=0.4.0 (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy)
Downloading annotated_types-0.7.0-py3-none-any.whl.metadata (15 kB)
Collecting pydantic-core==2.20.1 (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy)
Downloading pydantic_core-2.20.1-cp312-none-win_amd64.whl.metadata (6.7 kB)
Requirement already satisfied: typing-extensions>=4.6.1 in c:\users\admin\appdata\local\programs\python\python312\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>

Output:


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4baa, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Admin\AppData\Local\Programs\Python\Python312\3e3.py
['Yasheah', 'like', 'football', ',', 'fond', 'tennis', '.']
['Yasheah', 'like', 'football', ',', 'not', 'fond', 'tennis', '.']
>>>
Ln: 7 Col: 0

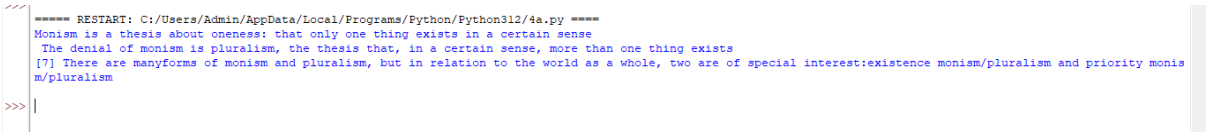
PRACTICAL 4A

Tokenization using Python's split() function.

Code:

```
text = "Monism is a thesis about oneness: that only one thing exists in a certain sense. The denial of monism is pluralism, the thesis that, in a certain sense, more than one thing exists.[7] There are manyforms of monism and pluralism, but in relation to the world as a whole, two are of special interest:existence monism/pluralism and priority monism/pluralism."

data = text.split('.')

for i in data:
    print(i)
```

Output:



PRACTICAL 4B

Tokenization using Regular Expressions (RegEx)

Code:

```
import nltk
from nltk.tokenize import RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps=True)
str = "I love to study Natrual Language processing in python"
tokkens = tk.tokenize(str)
print(tokkens)
```

Output:

```
''
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/4b.py ==========
['I', 'love', 'to', 'study', 'Natrual', 'Language', 'processing', 'in', 'python']
>>>
```

PRACTICAL 4C

Tokenization using NLTK

Code:

```
import nltk
from nltk.tokenize import word_tokenize
str = "I love to study Natrual Language processing in python"
print(word_tokenize(str))
```

Output

```
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/4c.py ==========
['I', 'love', 'to', 'study', 'Natrual', 'Language', 'processing', 'in', 'python']
>>>
```

PRACTICAL 4D

Tokenization using the spaCy library

Code:

```
import spacy
nlp = spacy.blank("en")
str = "I love to study Natrual Language processing in python"
doc = nlp(str)
words = [ word.text for word in doc]
print(words)
```

Output

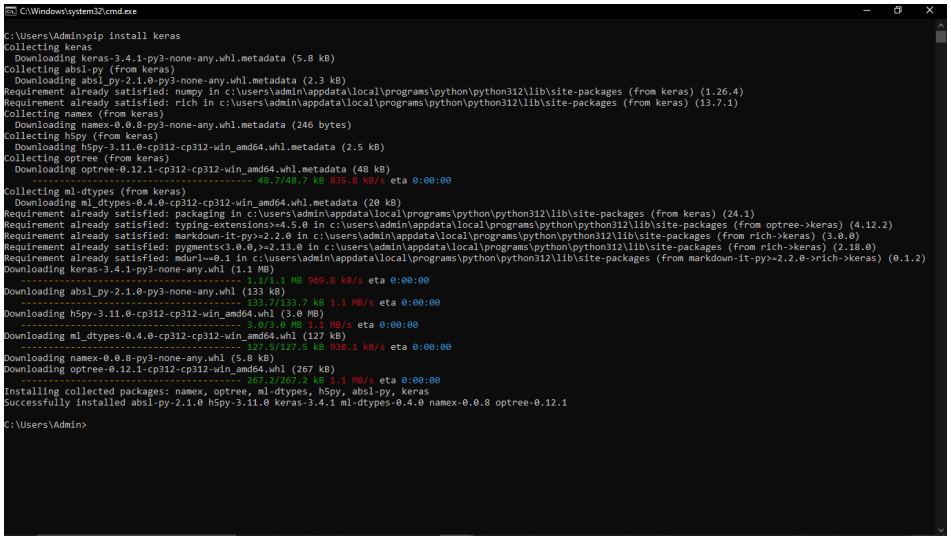
```
>>>
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/4d.py =======
['I', 'love', 'to', 'study', 'Natrual', 'Language', 'processing', 'in', 'python']
>>>
```

PRACTICAL 4E

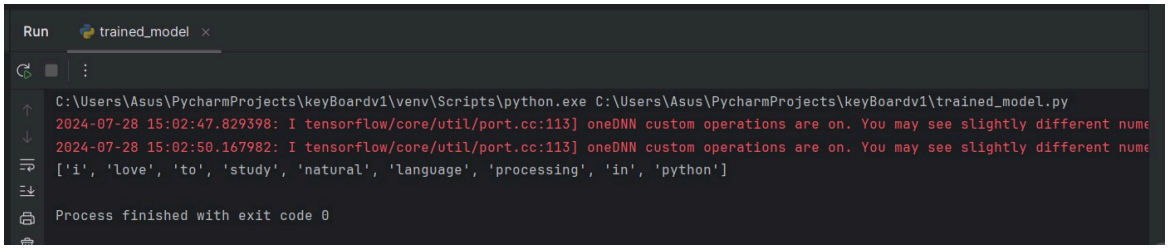
Tokenization using Keras

Code:

```
import tensorflow
import keras
from tensorflow.keras.preprocessing.text import text_to_word_sequence
 # Create a string input
str = "I love to study Natural Language Processing in Python"
 # tokenizing the text
tokens = text_to_word_sequence(str)
print(tokens)
```



Output:



PRACTICAL 4F

Tokenization using Gensim

Code:

```
from gensim.utils import tokenize
str = "I love to study Natrual Language processing in python"
print(list(tokenize(str)))
```

Output:

```
>>>
===== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/4f.py ====
['I', 'love', 'to', 'study', 'Natrual', 'Language', 'processing', 'in', 'python']
>>>
```

PRACTICAL 6A

Illustrate part of speech tagging

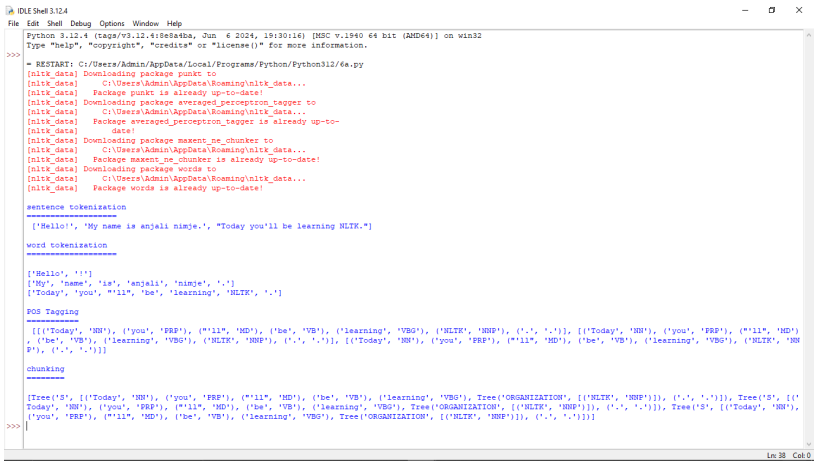
Part of speech Tagging and chunking of user defined text.

Code:

```
import nltk
from nltk import tokenize
nltk.download('punkt')
from nltk import tag
from nltk import chunk
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')

para = "Hello! My name is anjali nimje . Today you'll be learning NLTK."
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n==================\n",sents)

# word tokenization
print("\nword tokenization\n==================\n")
for index in range(len(sents)):
        words = tokenize.word_tokenize(sents[index])
        print(words)

# POS Tagging
tagged_words = []
for index in range(len(sents)):
        tagged_words.append(tag.pos_tag(words))
```

```
print("\nPOS Tagging\n==========\n",tagged_words)


# chunking
tree = []
for index in range(len(sents)):
    tree.append(chunk.ne_chunk(tagged_words[index]))
print("\nchunking\n========\n")
print(tree)
```

Output:



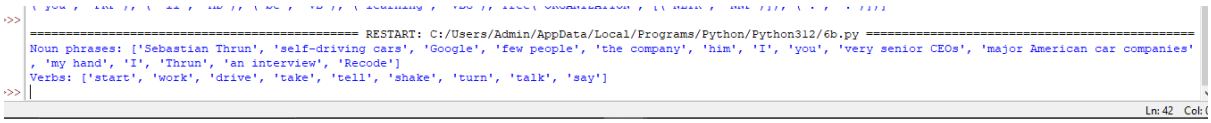
PRACTICAL 6B

Named Entity recognition using user defined text.

Code:

```
import spacy

nlp = spacy.load("en_core_web_sm")

text = ("when seabastian Thrun started working on self-driving cars at"

"Google in 2007, few people oustside of the company took him"

"Seroiusly,I can tell you very senior CEO's of major Amercan"

"car companies would shake my hand and turn away beacause I wasn't"

"worth talking to,said Thurn , in an interview with recorder earlier")doc =
nlp(text)

print("nouns:\n",[chunk.text for chunk in doc.noun_chunks])

print("verbs",[token.lemma_ for token in doc if token.pos_ =="VERB"])
```

Output:



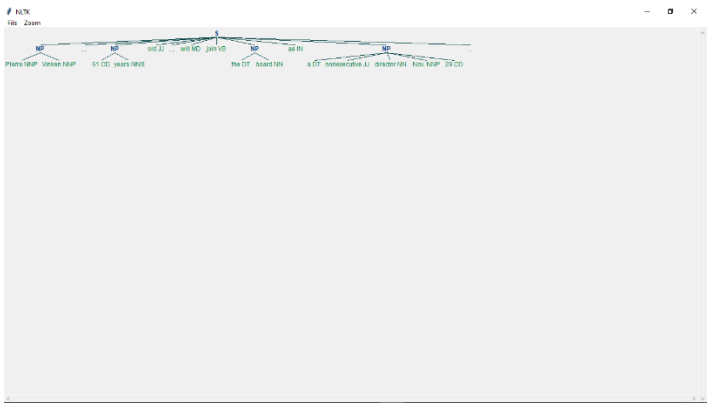
PRACTICAL 6C

Named Entity recognition with diagram using NLTK corpus – treebank.

Code:

```
import nltk
nltk.download('treebank')
from nltk.corpus import treebank_chunk
treebank_chunk.tagged_sents()[0]
treebank_chunk.chunked_sents()[0]
treebank_chunk.chunked_sents()[0].draw()
```

Output:



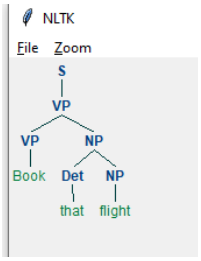
PRACTICAL 7A

Define grammar using nltk. Analyze a sentence using the same.

Code:

```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
        S -> VP
        VP -> VP NP
        NP -> Det NP
        Det -> 'that'
        NP -> singular Noun
        NP -> 'flight'
        VP -> 'Book'
        """)
sentence = "Book that flight"
for index in range(len(sentence)):
        all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
        print(tree)
        tree.draw()
```

Output:



PRACTICAL 7B

Accept the input string with Regular expression of Finite Automaton: 101

Code:

```
def FA(s):
#if the length is less than 3 then it can't be accepted, Therefore end the process.
    if len(s)<3:
        return "Rejected"
#first three characters are fixed. Therefore, checking them using index
    if s[0]=='1':
        if s[1]=='0':
            if s[2]=='1':
# After index 2 only "1" can appear. Therefore break the process if any other character is detected
                for i in range(3,len(s)):
                    if s[i]!='1':
                        return "Rejected"
                    return "Accepted" # if all 4 nested if true
                return "Rejected" # else of 3rd if
            return "Rejected" # else of 2nd if
        return "Rejected" # else of 1st if
inputs=['1','10101','101','10111','01010','100','','10111101','1011111']
for i in inputs:
    print(FA(i))
```

Output:

```
>>>
========== RESTART: D:/nlp/7b.py ==========
Rejected
Rejected
Rejected
Accepted
None
Rejected
Rejected
Accepted
Accepted
>>>
```

PRACTICAL 7C

Accept the input string with Regular expression of FA: (a+b)*bba.

Code:

```
def FA(s):
    size=0
#scan complete string and make sure that it contains only 'a' & 'b'
    for i in s:
        if i=='a' or i=='b':
            size+=1
        else:
            return "Rejected"
#After checking that it contains only 'a' & 'b'
#check it's length it should be 3 atleast
    if size>=3:
#check the last 3 elements
        if s[size-3]=='b':
            if s[size-2]=='b':
                if s[size-1]=='a':
                    return "Accepted" # if all 4 if true
                return "Rejected" # else of 4th if
            return "Rejected" # else of 3rd if
        return "Rejected" # else of 2nd if
    return "Rejected" # else of 1st if
inputs=['bba', 'ababbba', 'abba','abb', 'baba','bbb','']
for i in inputs:
    print(FA(i))
```

Output:

```
>>>
========== RESTART: D:/nlp/7c.py ==========
Accepted
Accepted
Accepted
Rejected
Rejected
Rejected
Rejected
>>>
```

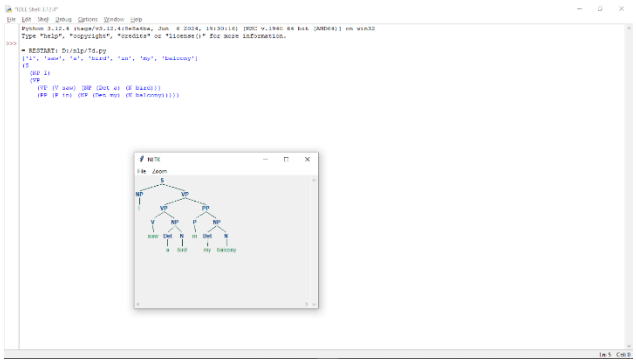
PRACTICAL 7D

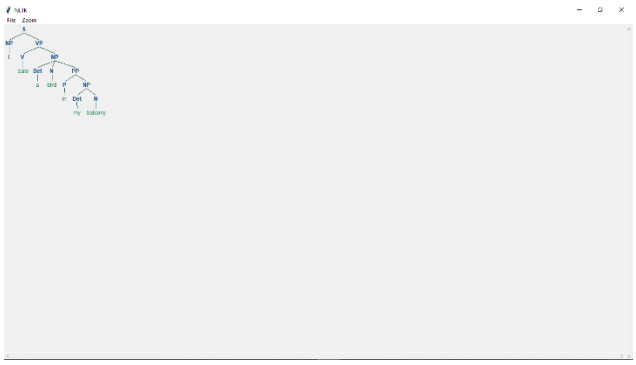
Implementation of Deductive Chart Parsing using context free grammar and a given sentence.

Code:

```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
   S -> NP VP
   PP -> P NP
   NP -> Det N | Det N PP | 'I'
   VP -> V NP | VP PP
   Det -> 'a' | 'my'
   N -> 'bird' | 'balcony'
   V -> 'saw'
   P -> 'in'
   """)
sentence = "I saw a bird in my balcony"

for index in range(len(sentence)):
   all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)

# all_tokens = ['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
   print(tree)
   tree.draw()
```

Output:


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/nlp/7d.py
['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
(S
(NP I)
(VP
(VP (V saw) (NP (Det a) (N bird)))
(PP (P in) (NP (Det my) (N balcony))))
Ln: 5 Col: 0
NLTK
File Zoom
S
NP
VP
I
VP
PP
V
NP
P
NP
saw
Det
N
in
Det
N
a
bird
my
balcony


NLTK
File Zoom
S
NP
VP
I
V
NP
PP
saw
Det
N
P
NP
a
bird
in
Det
N
my
balcony

PRACTICAL 8

Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer ,Study WordNetLemmatizer

Code:

```
import nltk
from nltk.stem import PorterStemmer
word_stemm = PorterStemmer()
print(word_stemm.stem('writing'))
import nltk
from nltk.stem import LancasterStemmer
lanc_stemm = LancasterStemmer()
print(lanc_stemm.stem('writing'))
import nltk
from nltk.stem import RegexpStemmer
Reg_stemm = RegexpStemmer('ing$|s$|e$|able$', min=4)
print(Reg_stemm.stem('writing'))
import nltk
from nltk.stem import SnowballStemmer
english_stemm = SnowballStemmer('english')
print(english_stemm.stem('writing'))
import nltk
from nltk.stem import WordNetLemmatizer
lemetizer = WordNetLemmatizer()
print("word:\t lemma")
print("rocks:", lemetizer.lemmatize("rocks"))
print("corpa:",lemetizer.lemmatize("corpora"))
```

Output:

```
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/8.py ==========
PorterStemmer:
write

LancasterStemmer:
writ

RegexpStemmer:
writ

SnowballStemmer:
write

WordNetLemmatizer:
word :  lemma
rocks : rock
corpora : corpus
better : good
>>>
```

PRACTICAL 9

Implement Naive Bayes classifier

Code:

```
#pip install pandas

#pip install sklearn

import pandas as pd

import numpy as np

sms_data =
pd.read_csv('C:\\Users\\Student\\Desktop\\Mayuri\\spam.csv',encoding='latin-
1')

import re

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

stemming = PorterStemmer()

corpus = []

for i in range (0,len(sms_data)):

        s1 = re.sub('[^a-zA-Z]',repl='',string=sms_data['v2'][i])

        s1.lower()

        s1 = s1.split()

        s1 = [stemming.stem(word) for word in s1 if word not in
set(stopwords.words('english'))]

        s1 = ' '.join(s1)

        corpus.append(s1)


from sklearn.feature_extraction.text import CountVectorizer

countvectorizer =CountVectorizer()

x = countvectorizer.fit_transform(corpus).toarray()
```

```
print(x)
y = sms_data['v1'].values
print(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,
stratify=y,random_state=2)

#Multinomial Naïve Bayes.
from sklearn.naive_bayes import MultinomialNB
multinomialnb = MultinomialNB()
multinomialnb.fit(x_train,y_train)

# Predicting on test data:
y_pred = multinomialnb.predict(x_test)
print(y_pred)

#Results of our Models
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
print(classification_report(y_test,y_pred))
print("accuracy_score: ",accuracy_score(y_test,y_pred))
```

Output:

```
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/9.py ====
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
['ham' 'ham' 'spam' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
              precision    recall  f1-score   support

         ham       0.87      1.00      0.93      1448
        spam       1.00      0.02      0.04       224

    accuracy                           0.87      1672
   macro avg       0.93      0.51      0.49      1672
weighted avg       0.89      0.87      0.81      1672

accuracy_score:  0.8690191387559809
>>>
```

PRACTICAL 10 A.1

Speech tagging using spacy

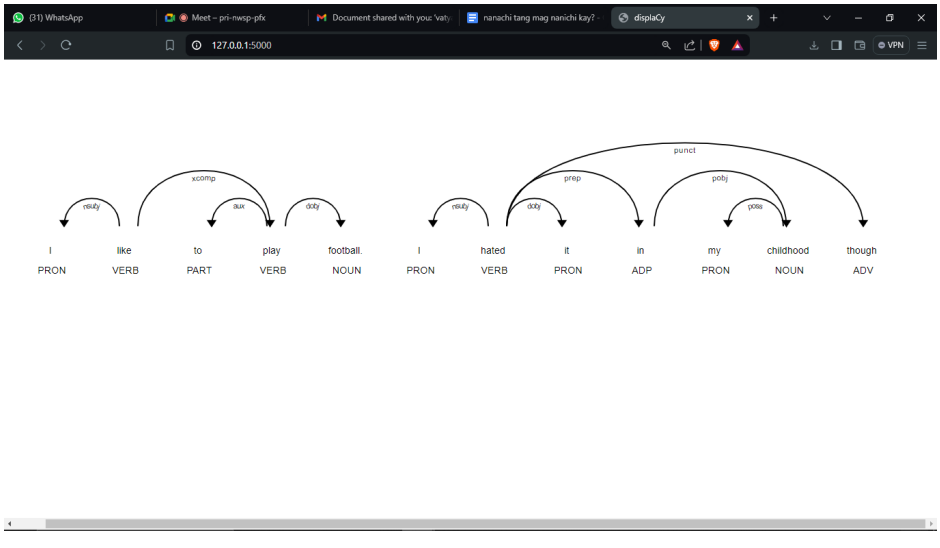
Code:

```
import spacy

sp = spacy.load('en_core_web_sm')

sen = sp(u"I like to play football. I hated it in my childhood though")

print(sen.text)

print(sen[7].pos_)

print(sen[7].tag_)

print(spacy.explain(sen[7].tag_))

for word in sen:

        print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}}
{spacy.explain(word.tag_)}')


sen = sp(u'Can you google it?')

word = sen[2]

print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}}
{spacy.explain(word.tag_)}')

sen = sp(u'Can you search it on google?')

word = sen[5]

print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}}
{spacy.explain(word.tag_)}')

#Finding the Number of POS Tags

sen = sp(u"I like to play football. I hated it in my childhood though")

num_pos = sen.count_by(spacy.attrs.POS)

num_pos

for k,v in sorted(num_pos.items()):

        print(f'{k}. {sen.vocab[k].text:{8}}: {v}')

#Visualizing Parts of Speech Tags
```

```
from spacy import displacy
sen = sp(u"I like to play football. I hated it in my childhood though")
displacy.serve(sen, style='dep', options={'distance': 120})
```

Output:

```
========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/10a1.py ==========
I like to play football. I hated it in my childhood though
VERB
VBD
verb, past tense
I           PRON       PRP      pronoun, personal
like        VERB       VBP      verb, non-3rd person singular present
to          PART       TO       infinitival "to"
play        VERB       VB       verb, base form
football    NOUN       NN       noun, singular or mass
.           PUNCT      .        punctuation mark, sentence closer
I           PRON       PRP      pronoun, personal
hated       VERB       VBD      verb, past tense
it          PRON       PRP      pronoun, personal
in          ADP        IN       conjunction, subordinating or preposition
my          PRON       PRP$     pronoun, possessive
childhood   NOUN       NN       noun, singular or mass
though      ADV        RB       adverb
google      VERB       VB       verb, base form
google      PROPN      NNP      noun, proper singular
85. ADP     : 1
86. ADV     : 1
92. NOUN    : 2
94. PART    : 1
95. PRON    : 4
97. PUNCT   : 1
100. VERB    : 3

Using the 'dep' visualizer
Serving on http://0.0.0.0:5000 ...
```



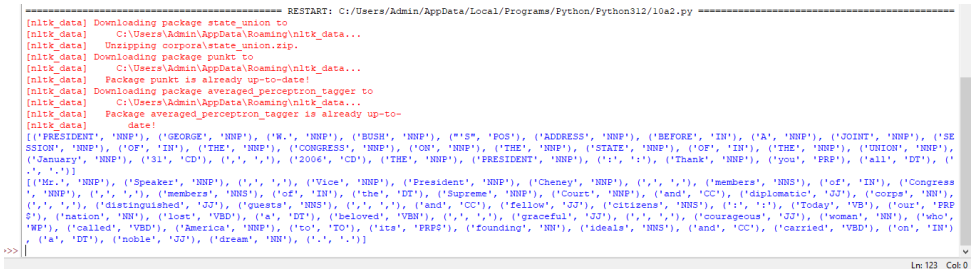
PRACTICAL 10 A.2

Speech tagging using nltk

Code:

```
import nltk
nltk.download('state_union')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer

#create our training and testing data:
train_text = state_union.raw("2005-GWBush.txt")
sample_text = state_union.raw("2006-GWBush.txt")

#train the Punkt tokenizer like:
custom_sent_tokenizer = PunktSentenceTokenizer(train_text)
# tokenize:
tokenized = custom_sent_tokenizer.tokenize(sample_text)
def process_content():
        try:
        for i in tokenized[:2]:
        words = nltk.word_tokenize(i)
        tagged = nltk.pos_tag(words)
        print(tagged)
        except Exception as e:
        print(str(e))
process_content()
```

Output:


========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/10a2.py ==========
[nltk_data] Downloading package state_union to
[nltk_data] C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\state_union.zip.
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data] date!
[('PRESIDENT', 'NNP'), ('GEORGE', 'NNP'), ('W.', 'NNP'), ('BUSH', 'NNP'), ("'S", 'POS'), ('ADDRESS', 'NNP'), ('BEFORE', 'IN'), ('A', 'NNP'), ('JOINT', 'NNP'), ('SESSION', 'NNP'), ('OF', 'IN'), ('THE', 'NNP'), ('CONGRESS', 'NNP'), ('ON', 'NNP'), ('THE', 'NNP'), ('STATE', 'NNP'), ('OF', 'IN'), ('THE', 'NNP'), ('UNION', 'NNP'), ('January', 'NNP'), ('31', 'CD'), (',', ','), ('2006', 'CD'), ('THE', 'NNP'), ('PRESIDENT', 'NNP'), (':', ':'), ('Thank', 'NNP'), ('you', 'PRP'), ('all', 'DT'), ('.', '.')]
[('Mr.', 'NNP'), ('Speaker', 'NNP'), (',', ','), ('Vice', 'NNP'), ('President', 'NNP'), ('Cheney', 'NNP'), (',', ','), ('members', 'NNS'), ('of', 'IN'), ('Congress', 'NNP'), (',', ','), ('members', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('Supreme', 'NNP'), ('Court', 'NNP'), ('and', 'CC'), ('diplomatic', 'JJ'), ('corps', 'NN'), (',', ','), ('distinguished', 'JJ'), ('guests', 'NNS'), (',', ','), ('and', 'CC'), ('fellow', 'JJ'), ('citizens', 'NNS'), (':', ':'), ('Today', 'VB'), ('our', 'PRP$'), ('nation', 'NN'), ('lost', 'VBD'), ('a', 'DT'), ('beloved', 'VBN'), (',', ','), ('graceful', 'JJ'), (',', ','), ('courageous', 'JJ'), ('woman', 'NN'), ('who', 'WP'), ('called', 'VBD'), ('America', 'NNP'), ('to', 'TO'), ('its', 'PRP$'), ('founding', 'NN'), ('ideals', 'NNS'), ('and', 'CC'), ('carried', 'VBD'), ('on', 'IN'), ('a', 'DT'), ('noble', 'JJ'), ('dream', 'NN'), ('.', '.')]
>>>
Ln: 123 Col: 0

PRACTICAL 10 B.1

Usage of Give and Gave in the Penn Treebank sample

Code:

```
import nltk
import nltk.parse.viterbi
import nltk.parse.pchart
def give(t):
 return t.label() == 'VP' and len(t) > 2 and t[1].label() == 'NP'\
 and (t[2].label() == 'PP-DTV' or t[2].label() == 'NP')\
 and ('give' in t[0].leaves() or 'gave' in t[0].leaves())
def sent(t):
 return ' '.join(token for token in t.leaves() if token[0] not in '*-0')
def print_node(t, width):
 output = "%s %s: %s / %s: %s" %\
        (sent(t[0]), t[1].label(), sent(t[1]), t[2].label(), sent(t[2]))
 if len(output) > width:
        output = output[:width] + "..."
 print (output)
for tree in nltk.corpus.treebank.parsed_sents():
 for t in tree.subtrees(give):
        print_node(t, 72)
```

Output:

```
>>> ========== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/10b1.py ==========
gave NP: the chefs / NP: a standing ovation
give NP: advertisers / NP: discounts for maintaining or increasing ad sp...
give NP: it / PP-DTV: to the politicians
gave NP: them / NP: similar help
give NP: them / NP:
give NP: only French history questions / PP-DTV: to students in a Europe...
give NP: federal judges / NP: a raise
give NP: consumers / NP: the straight scoop on the U.S. waste crisis
gave NP: Mitsui / NP: access to a high-tech medical product
give NP: Mitsubishi / NP: a window on the U.S. glass industry
give NP: much thought / PP-DTV: to the rates she was receiving , nor to ...
give NP: your Foster Savings Institution / NP: the gift of hope and free...
give NP: market operators / NP: the authority to suspend trading in futu...
gave NP: quick approval / PP-DTV: to $ 3.18 billion in supplemental appr...
give NP: the Transportation Department / NP: up to 50 days to review any...
give NP: the president / NP: such power
give NP: me / NP: the heebie-jeebies
give NP: holders / NP: the right , but not the obligation , to buy a cal...
gave NP: Mr. Thomas / NP: only a ``qualified'' rating , rather than ``...
give NP: the president / NP: line-item veto power
>>>
```

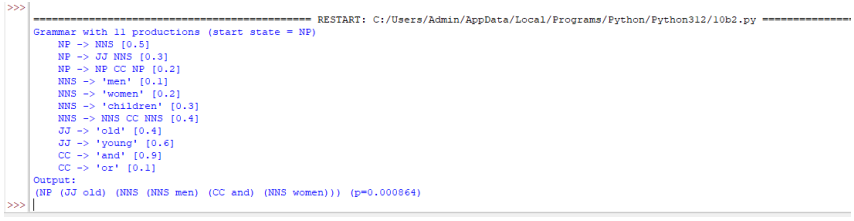
PRACTICAL 10 B.2

Probabilistic parser

Code:

```
import nltk
from nltk import PCFG
grammar = PCFG.fromstring('''
NP -> NNS [0.5] | JJ NNS [0.3] | NP CC NP [0.2]
NNS -> "men" [0.1] | "women" [0.2] | "children" [0.3] | NNS CC NNS [0.4]
JJ -> "old" [0.4] | "young" [0.6]
CC -> "and" [0.9] | "or" [0.1]
''')
print(grammar)
viterbi_parser = nltk.ViterbiParser(grammar)
token = "old men and women".split()
obj = viterbi_parser.parse(token)
print("Output: ")
for x in obj:
        print(x)
```

Output:



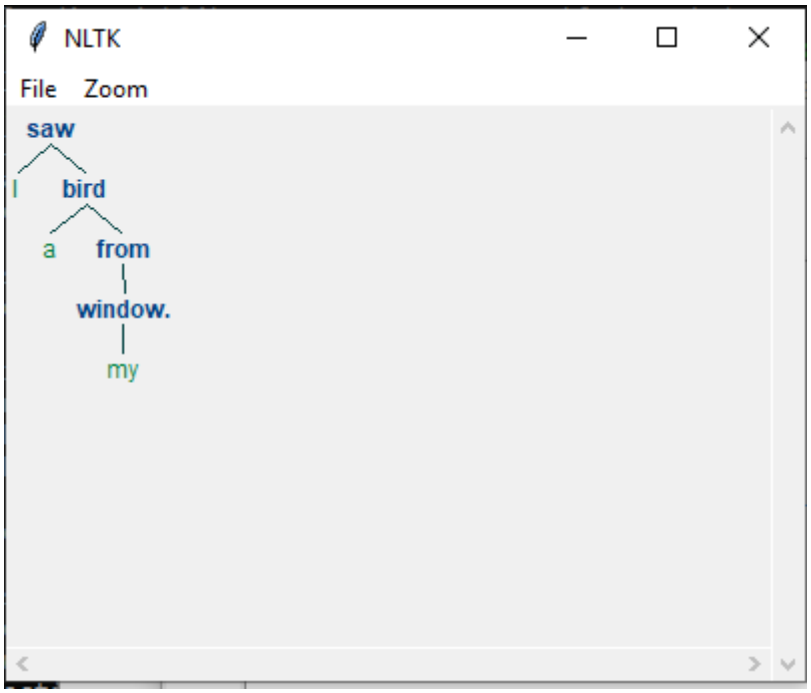
PRACTICAL 10C

Malt Parsing

Code:

```
from nltk.parse import malt
mp = malt.MaltParser('maltparser-1.9.2', 'engmalt.linear-1.7.mco')#file
t = mp.parse_one('I saw a bird from my window.'.split()).tree()
print(t)
t.draw()
```

Output:



PRACTICAL 11A

Multiword Expressions in NLP

Code:

```
from nltk.tokenize import MWETokenizer

from nltk import sent_tokenize, word_tokenize

s = '''Good cake cost Rs.1500\kg in Mumbai. Please buy me one of them.\n\nThanks.'''

mwe = MWETokenizer([('New', 'York'), ('Hong', 'Kong')], separator='_')

for sent in sent_tokenize(s):

        print(mwe.tokenize(word_tokenize(sent)))
```

Output:

```
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/11a.py
['Good', 'cake', 'cost', 'Rs.1500\\kg', 'in', 'Mumbai', '.']
['Please', 'buy', 'me', 'one', 'of', 'them', '.']
['Thanks', '.']
>>>
```

PRACTICAL 11B

Normalized Web Distance and Word Similarity

Code:

```
import numpy as np
import re
import textdistance # pip install textdistance
# we will need scikit-learn>=0.21
import sklearn #pip install sklearn
from sklearn.cluster import AgglomerativeClustering
texts = [
'Reliance supermarket', 'Reliance hypermarket', 'Reliance', 'Reliance',
'Reliancedowntown', 'Relianc market',
'Mumbai', 'Mumbai Hyper', 'Mumbai dxb', 'mumbai airport',
'k.m trading', 'KM Trading', 'KM trade', 'K.M. Trading', 'KM.Trading'
]
def normalize(text):
    """ Keep only lower-cased text and numbers"""
    return re.sub('[^a-z0-9]+', ' ', text.lower())
def group_texts(texts, threshold=0.4):
    """ Replace each text with the representative of its cluster"""
    normalized_texts = np.array([normalize(text) for text in texts])
    distances = 1 - np.array([
    [textdistance.jaro_winkler(one, another) for one in normalized_texts]
    for another in normalized_texts
    ])
    clustering = AgglomerativeClustering(
    distance_threshold=threshold, # this parameter needs to be tuned
carefully
```

```
        metric="precomputed", linkage="complete", n_clusters=None
        ).fit(distances)
        centers = dict()
        for cluster_id in set(clustering.labels_):
        index = clustering.labels_ == cluster_id
        centrality = distances[:, index][index].sum(axis=1)
        centers[cluster_id] = normalized_texts[index][centrality.argmin()]
        return [centers[i] for i in clustering.labels_]
print(group_texts(texts))
```

Output:

PRACTICAL 11C

Word Sense Disambiguation

Code:

```
from nltk.corpus import wordnet as wn
def get_first_sense(word, pos=None):
    if pos:
    synsets = wn.synsets(word,pos)
    else:
    synsets = wn.synsets(word)
    return synsets[0]

best_synset = get_first_sense('bank')
print ('%s: %s' % (best_synset.name, best_synset.definition))
best_synset = get_first_sense('set','n')
print ('%s: %s' % (best_synset.name, best_synset.definition))
best_synset = get_first_sense('#set','v')
print ('%s: %s' % (best_synset.name, best_synset.definition))
```

Output:

```
>>>
========== RESTART: C:\Users\Admin\AppData\Local\Programs\Python\Python312\11c.py ==========
<bound method Synset.name of Synset('bank.n.01')>: <bound method Synset.definition of Synset('bank.n.01')>
<bound method Synset.name of Synset('set.n.01')>: <bound method Synset.definition of Synset('set.n.01')>
<bound method Synset.name of Synset('put.v.01')>: <bound method Synset.definition of Synset('put.v.01')>
>>>
```