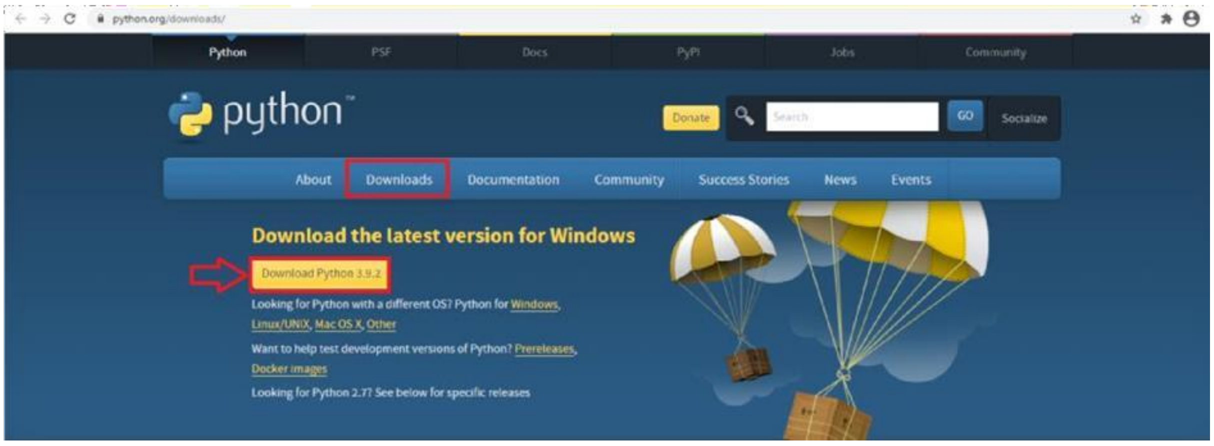
# Practical No. 1

## a) Install NLTK

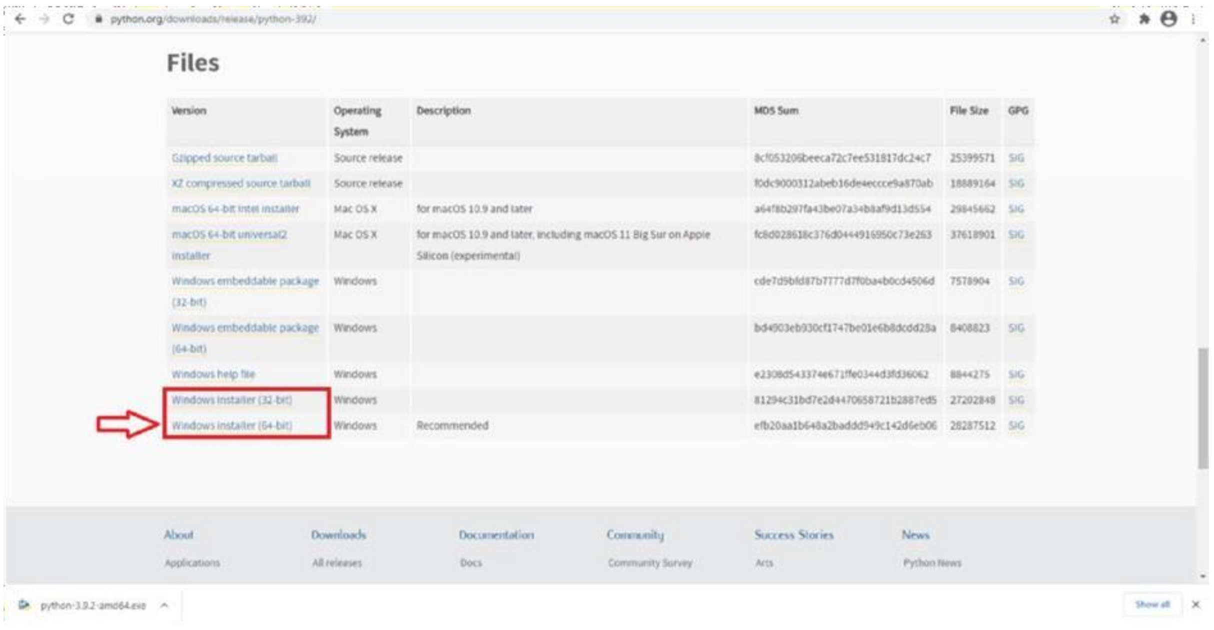
### Python 3.9.2 Installation on Windows

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


python.org/downloads/
Python
PSF
Docs
PyPI
Jobs
Community
python
Donate
Search
GO
Socialize
About
Downloads
Documentation
Community
Success Stories
News
Events
Download the latest version for Windows
Download Python 3.9.2
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other
Want to help test development versions of Python? Prereleases, Docker images
Looking for Python 2.7? See below for specific releases

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

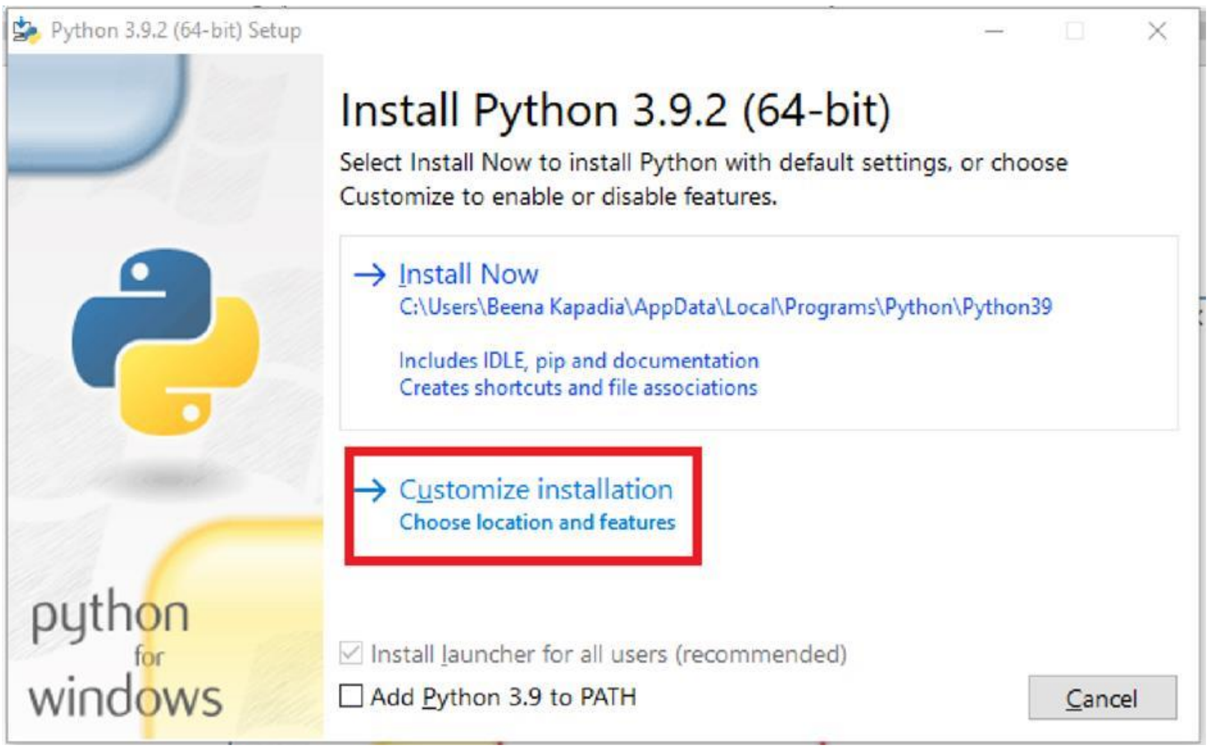
**Step 2) Click on the Windows installer (64 bit)**



Files

| Version | Operating System | Description | MD5 Sum | File Size | GPG |
|---|---|---|---|---|---|
| Gzipped source tarball | Source release | | 8cf053206beeca72c7ee531817dc24c7 | 25399571 | SIG |
| XZ compressed source tarball | Source release | | f0dc9000312abeb16de4eccce9a870ab | 18889164 | SIG |
| macOS 64-bit Intel installer | Mac OS X | for macOS 10.9 and later | a64f8b297fa43be07a34b8af9d13d554 | 29845662 | SIG |
| macOS 64-bit universal2 installer | Mac OS X | for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental) | fc8d028618c376d0444916950c73e263 | 37618901 | SIG |
| Windows embeddable package (32-bit) | Windows | | cde7d9bfd87b7777d7f0ba4b0cd4506d | 7578904 | SIG |
| Windows embeddable package (64-bit) | Windows | | bd4903eb930cf1747be01e6b8dcdd28a | 8408823 | SIG |
| Windows help file | Windows | | e2308d543374e671ffe0344d3fd36062 | 8844275 | SIG |
| Windows installer (32-bit) | Windows | | 81294c31bd7e2d4470658721b2887ed5 | 27202848 | SIG |
| Windows installer (64-bit) | Windows | Recommended | efb20aa1b648a2baddd949c142d6eb06 | 28287512 | SIG |

**Step 3) Select Customize Installation**


Python 3.9.2 (64-bit) Setup

Install Python 3.9.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

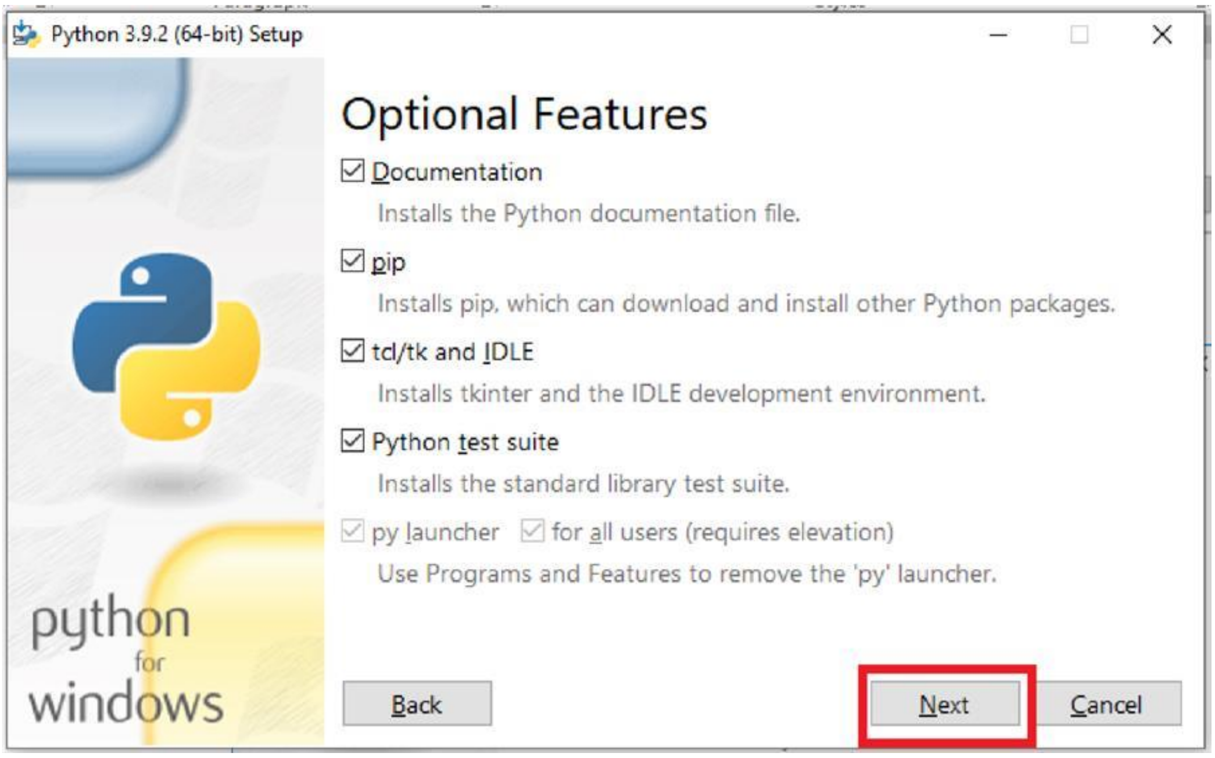
→ Install Now
C:\Users\Beena Kapadia\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.9 to PATH

**Step 4) Click NEXT**


Python 3.9.2 (64-bit) Setup

Optional Features

☑ Documentation
Installs the Python documentation file.

☑ pip
Installs pip, which can download and install other Python packages.

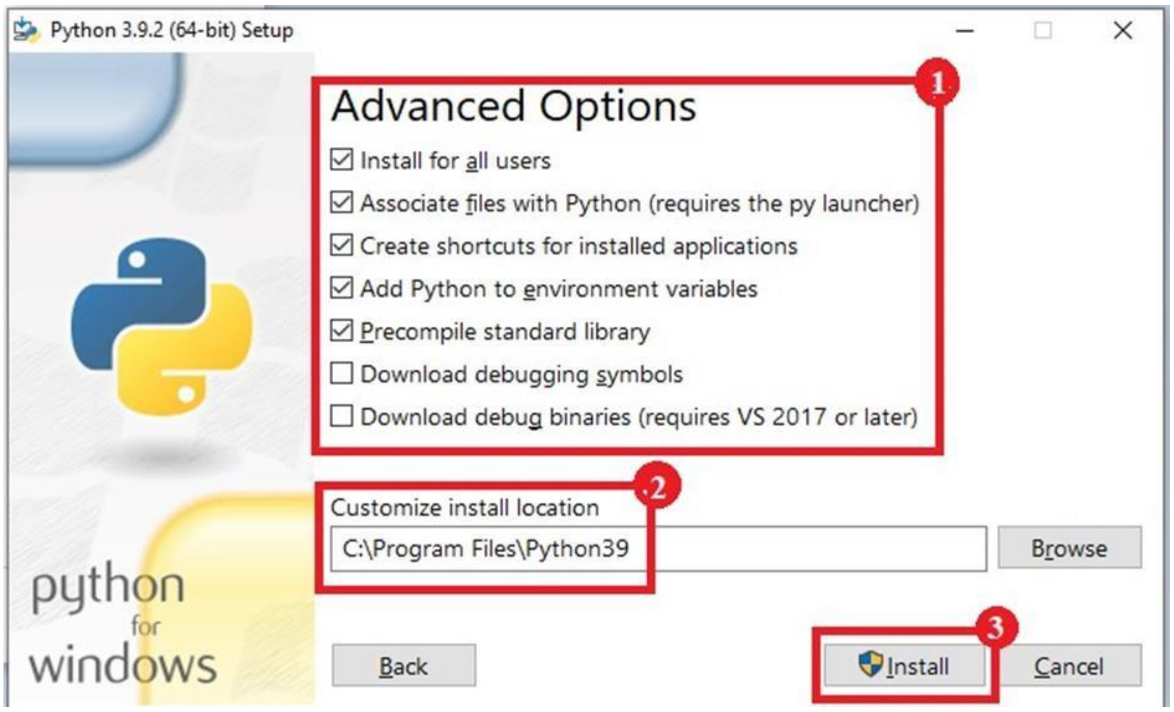
☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)
Use Programs and Features to remove the 'py' launcher.

**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



Python 3.9.2 (64-bit) Setup

Advanced Options

- [✓] Install for all users
- [✓] Associate files with Python (requires the py launcher)
- [✓] Create shortcuts for installed applications
- [✓] Add Python to environment variables
- [✓] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2017 or later)

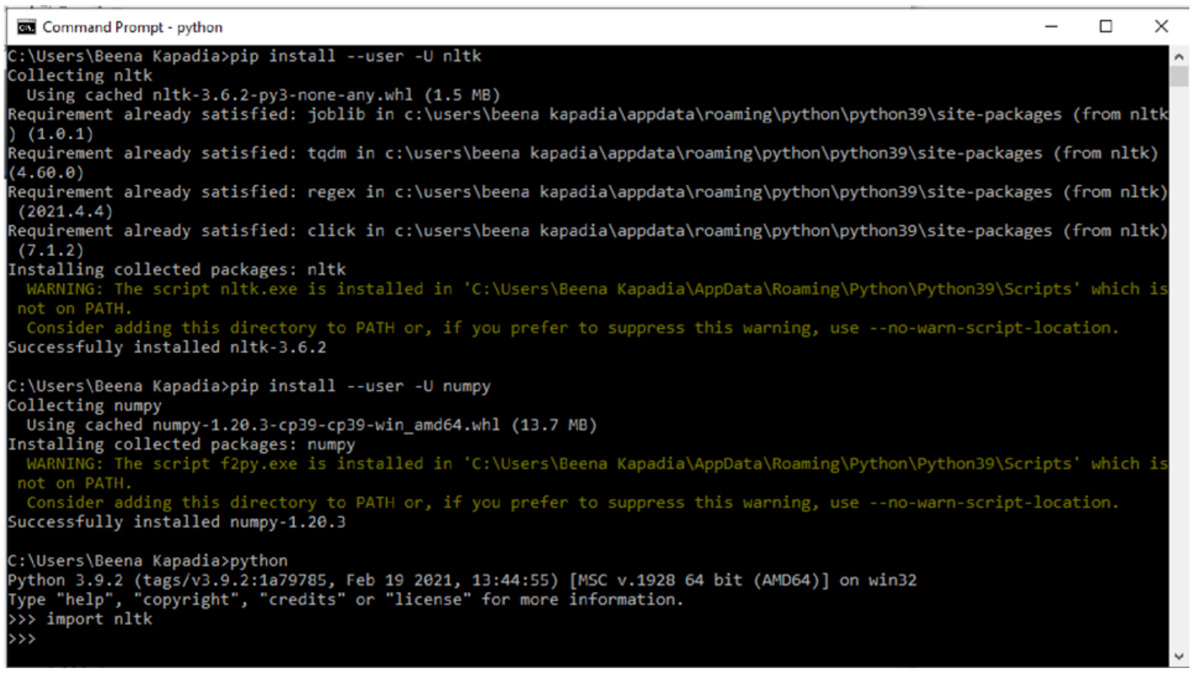
Customize install location

C:\Program Files\Python39

**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



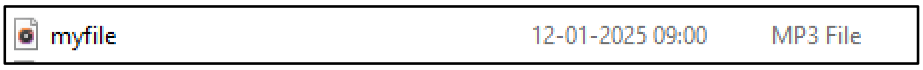
```
C:\Users\Beena Kapadia>pip install --user -U nltk
Collecting nltk
  Using cached nltk-3.6.2-py3-none-any.whl (1.5 MB)
Requirement already satisfied: joblib in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk
) (1.0.1)
Requirement already satisfied: tqdm in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk)
(4.60.0)
Requirement already satisfied: regex in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk)
 (2021.4.4)
Requirement already satisfied: click in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk)
 (7.1.2)
Installing collected packages: nltk
  WARNING: The script nltk.exe is installed in 'C:\Users\Beena Kapadia\AppData\Roaming\Python\Python39\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 nltk-3.6.2

C:\Users\Beena Kapadia>pip install --user -U numpy
Collecting numpy
  Using cached numpy-1.20.3-cp39-cp39-win_amd64.whl (13.7 MB)
Installing collected packages: numpy
  WARNING: The script f2py.exe is installed in 'C:\Users\Beena Kapadia\AppData\Roaming\Python\Python39\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-1.20.3

C:\Users\Beena Kapadia>python
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>>
```

## b) Convert the given text to speech.

**Source code:**

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

**Output:**



```
myfile                12-01-2025 09:00      MP3 File
```

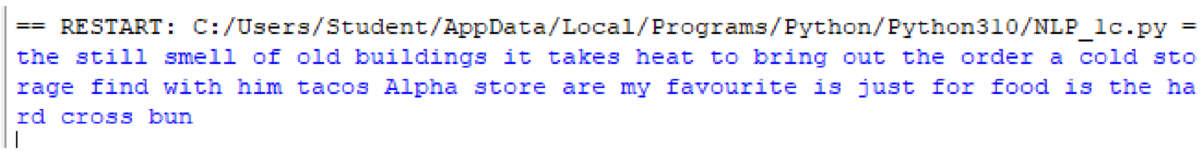
myfile.mp3 is getting created and it plays the file with playsound() method , while running to speech

## c) Convert audio file Speech to Text.

**Source code:**

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

**Output:**



```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/NLP_1c.py =
the still smell of old buildings it takes heat to bring out the order a cold sto
rage find with him tacos Alpha store are my favourite is just for food is the ha
rd cross bun
```

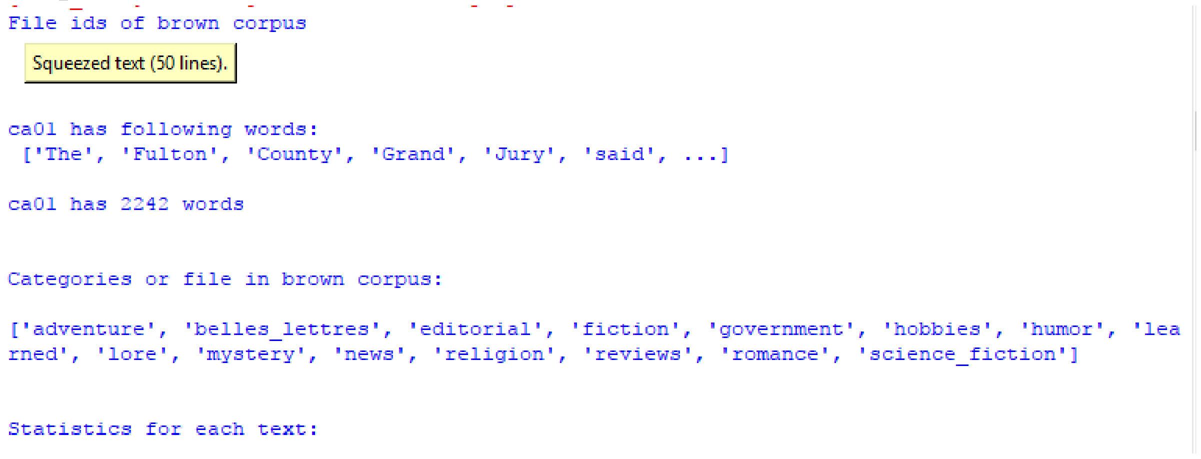
# Practical No. 2

**a) Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like filelds, raw, words, sents, categories.**

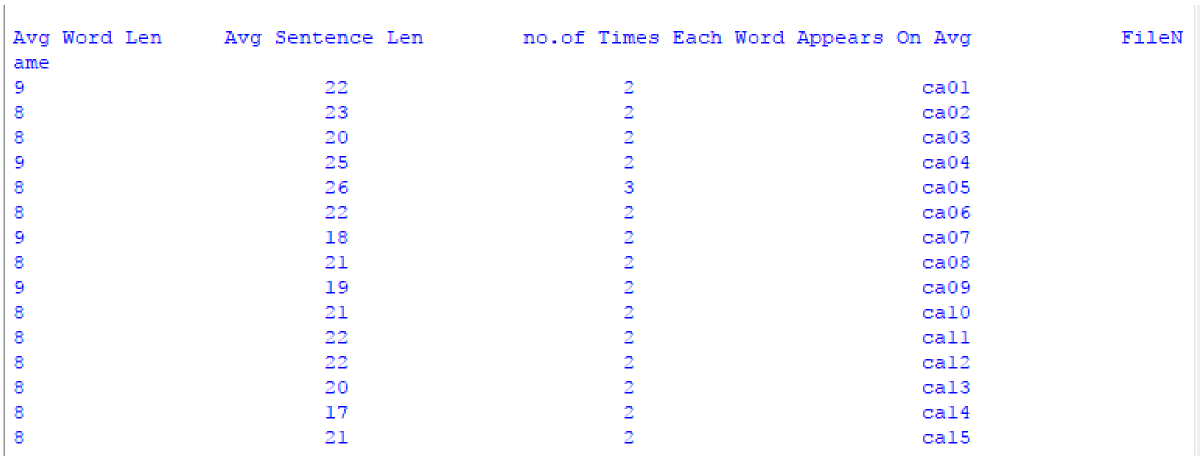
**Source Code:**

```
import nltk
nltk.download('brown')
from nltk.corpus import brown
print('File ids of brown corpus\n',brown.fileids())
ca01=brown.words('ca01')
print('\nca01 has following words:\n',ca01)
print('\nca01 has',len(ca01),'words')
print('\n\nCategories or file in brown corpus:\n')
print(brown.categories())
print('\n\nStatistics for each text:\n')
print('Avg Word Len \t Avg Sentence Len \t no.of Times Each Word Appears On Avg \t \t
FileName')
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)
```

**Output:**



```
File ids of brown corpus
 Squeezed text (50 lines).

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', 'lea
rned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']


Statistics for each text:
```



```
Avg Word Len     Avg Sentence Len         no.of Times Each Word Appears On Avg          FileN
ame
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
8                       22                      2                       ca12
8                       20                      2                       ca13
8                       17                      2                       ca14
8                       21                      2                       ca15
```

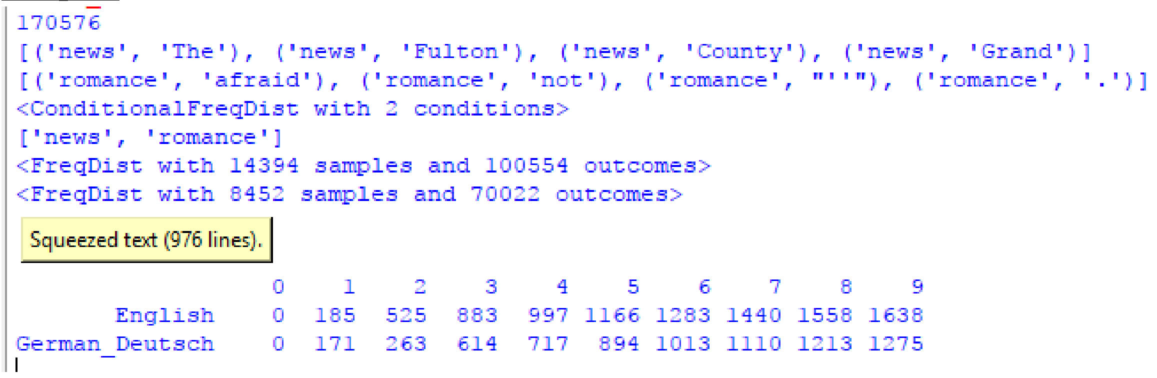
## c) Study Conditional frequency distributions

**source code:**

```
text=['The','Fulton','County','Grand','Jury','said',...]
pairs=[('news','The'),('news','Fulton'),('news','County'),...]
import nltk
nltk.download('inaugural')
nltk.download('udhr')
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:])
cdf=nltk.ConditionalFreqDist(genre_word)
print(cdf)
print(cdf.conditions())
print(cdf['news'])
print(cdf['romance'])
print(list(cdf['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)
```

**Output:**



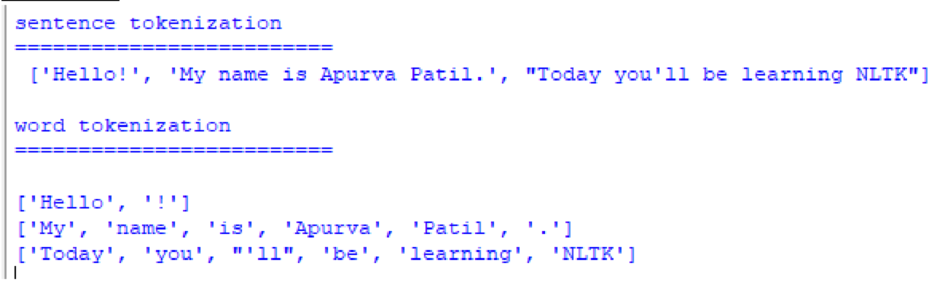
```
170576
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]
[('romance', 'afraid'), ('romance', 'not'), ('romance', "''"), ('romance', '.')]
<ConditionalFreqDist with 2 conditions>
['news', 'romance']
<FreqDist with 14394 samples and 100554 outcomes>
<FreqDist with 8452 samples and 70022 outcomes>
Squeezed text (976 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
```

## d) Study of tagged corpora with methods like tagged_sents, tagged_words.

**Source code:**

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

**Output:**



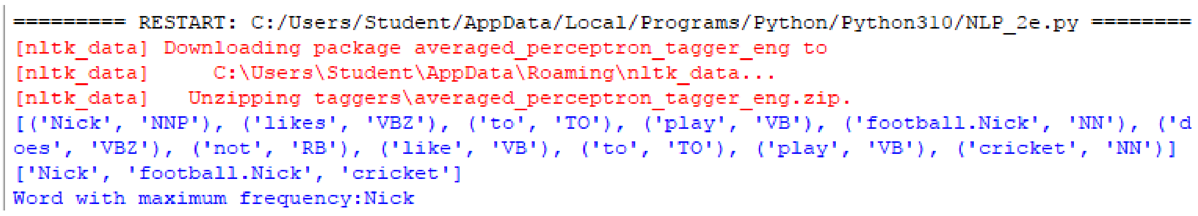
```
sentence tokenization
=========================
 ['Hello!', 'My name is Apurva Patil.', "Today you'll be learning NLTK"]

word tokenization
=========================

['Hello', '!']
['My', 'name', 'is', 'Apurva', 'Patil', '.']
['Today', 'you', "'ll", 'be', 'learning', 'NLTK']
```

## e) Write a program to find the most frequent noun tags.

**Source Code:**

```
import nltk
nltk.download('averaged_perceptron_tagger_eng')
from collections import defaultdict
text = nltk.word_tokenize("Nick likes to play football.Nick does not like to play cricket")
tagged = nltk.pos_tag(text)
print(tagged)
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)
for sub in addNounWords:
    for wrd in sub.split():
        temp[wrd]+=1
res=max(temp,key=temp.get)
print("Word with maximum frequency:"+str(res))
```

**Output:**



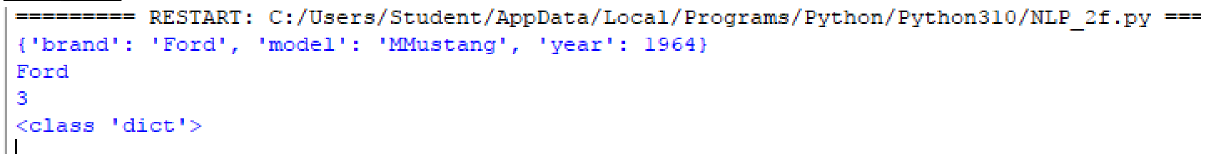
```
========= RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/NLP_2e.py ========
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger_eng.zip.
[('Nick', 'NNP'), ('likes', 'VBZ'), ('to', 'TO'), ('play', 'VB'), ('football.Nick', 'NN'), ('d
oes', 'VBZ'), ('not', 'RB'), ('like', 'VB'), ('to', 'TO'), ('play', 'VB'), ('cricket', 'NN')]
['Nick', 'football.Nick', 'cricket']
Word with maximum frequency:Nick
```

## f) Map Words to Properties Using Python Dictionaries

**Source code:**

```
thisdict={
    "brand":"Ford",
    "model":"MMustang",
    "year":1964
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```

**Output:**



```
========= RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/NLP_2f.py ===
{'brand': 'Ford', 'model': 'MMustang', 'year': 1964}
Ford
3
<class 'dict'>
```

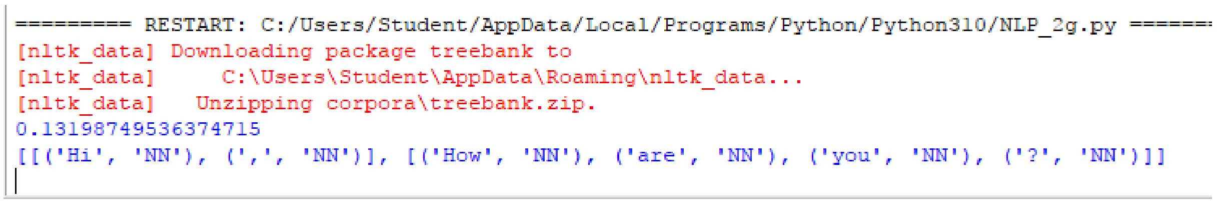
## g) Study i) DefaultTagger, ii) Regular expression tagger, iii) UnigramTagger

### i) DefaultTagger

**Source code:**

```
import nltk
nltk.download('treebank')
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
from nltk.corpus import treebank
testsentences = treebank.tagged_sents() [1000:]
print(exptagger.accuracy (testsentences))
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
print(exptagger.tag_sents([['Hi', ','], ['How', 'are', 'you', '?']]))
```

**Output:**



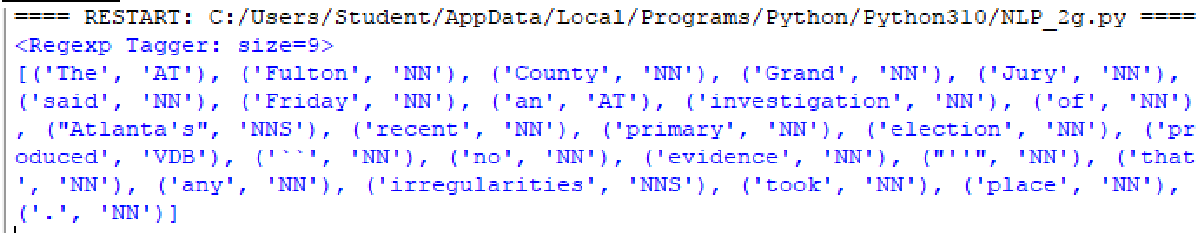
```
========= RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/NLP_2g.py ======
[nltk_data] Downloading package treebank to
[nltk_data]     C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\treebank.zip.
0.13198749536374715
[[('Hi', 'NN'), (',', 'NN')], [('How', 'NN'), ('are', 'NN'), ('you', 'NN'), ('?', 'NN')]]
```

### ii) Regular expression tagger

**Source 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]+)?$','CD'),
   (r'(The|the|A|a|An|an)$','AT'),
   (r'.*able$','JJ'),
   (r'.*ness$','NN'),
   (r'.*ly$','RB'),
   (r'.*s$','NNS'),
   (r'.*ing$','VBG'),
   (r'.*ed$','VDB'),
   (r'.*','NN')
  ])
print(regexp_tagger)
print(regexp_tagger.tag(test_sent))
```

**Output:**



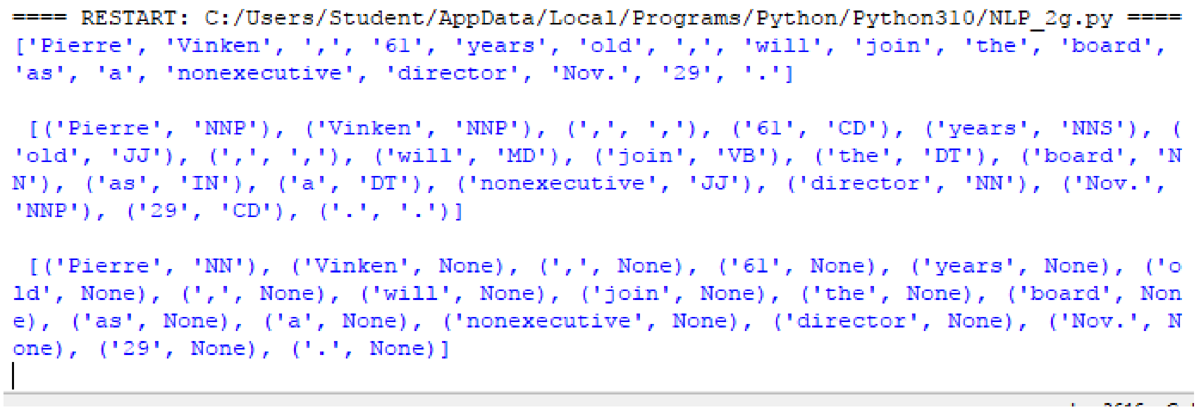
```
==== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/NLP_2g.py ====
<Regexp Tagger: size=9>
[('The', 'AT'), ('Fulton', 'NN'), ('County', 'NN'), ('Grand', 'NN'), ('Jury', 'NN'),
('said', 'NN'), ('Friday', 'NN'), ('an', 'AT'), ('investigation', 'NN'), ('of', 'NN')
, ("Atlanta's", 'NNS'), ('recent', 'NN'), ('primary', 'NN'), ('election', 'NN'), ('pr
oduced', 'VDB'), ('``', 'NN'), ('no', 'NN'), ('evidence', 'NN'), ("''", 'NN'), ('that
', 'NN'), ('any', 'NN'), ('irregularities', 'NNS'), ('took', 'NN'), ('place', 'NN'),
('.', 'NN')]
```

**iii) UnigramTagger**

**Source Code:**

```
from nltk.tag import UnigramTagger
from nltk.corpus import treebank
train_sents=treebank.tagged_sents()[:10]
tagger=UnigramTagger(train_sents)
print(treebank.sents()[0])
print('\n',tagger.tag(treebank.sents()[0]))
tagger.tag(treebank.sents()[0])
tagger=UnigramTagger(model={'Pierre':'NN'})
print('\n',tagger.tag(treebank.sents()[0]))
```

**Output:**



```
==== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/NLP_2g.py ====
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'board',
'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']

 [('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), (
'old', 'JJ'), (',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT'), ('board', 'N
N'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive', 'JJ'), ('director', 'NN'), ('Nov.',
'NNP'), ('29', 'CD'), ('.', '.')]

 [('Pierre', 'NN'), ('Vinken', None), (',', None), ('61', None), ('years', None), ('o
ld', None), (',', None), ('will', None), ('join', None), ('the', None), ('board', Non
e), ('as', None), ('a', None), ('nonexecutive', None), ('director', None), ('Nov.', N
one), ('29', None), ('.', None)]
```

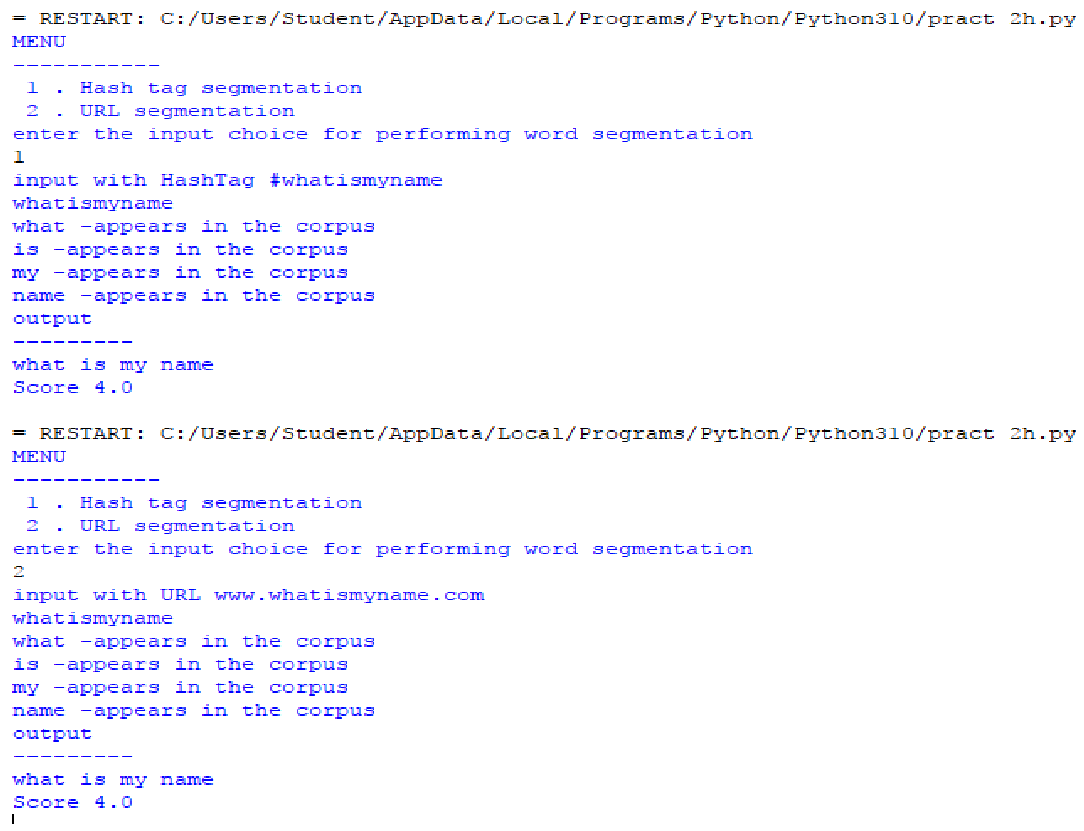
**h) Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find the score of words.**

**Source Code:**

```
from __future___import with_statement
import re
words = []
testword = []
ans = []
print("MENU")
print("-----------")
print(" 1 . Hash tag segmentation ")
print(" 2 . URL segmentation ")
print("enter the input choice for performing word segmentation")
choice = int(input())
if choice ==1:
   text="#whatismyname"
   print("input with HashTag",text)
   pattern=re.compile("[^\w']")
   a=pattern.sub('',text)
elif choice==2:
   text = "www.whatismyname.com"
   print("input with URL",text)
   a=re.split('\s|(?<!\d)[,.](?!\d)', text)
   splitwords = ["www","com","in"]
   a ="".join([each for each in a if each not in splitwords])
else:
   print("wrong choice...try again")
print(a)
for each in a:
   testword.append(each)
test_length = len(testword)
with open('Words.txt', 'r') as f:
   lines = f.readlines()
   words =[(e.strip()) for e in lines]
def Seg(a,length):
   ans=[]
   for k in range(0,length+1):
      if a[0:k] in words:
         print(a[0:k],"-appears in the corpus")
         ans.append(a[0:k])
         break
   if ans!=[]:
      g = max(ans,key=len)
```

```
        return g
test_tot_itr = 0
answer = []
Score = 0
N = 37
M = 0
C = 0
while test_tot_itr < test_length:
    ans_words = Seg(a,test_length)
    if ans_words != 0:
        test_itr = len(ans_words)
    answer.append(ans_words)
    a = a[test_itr:test_length]
    test_tot_itr += test_itr
Aft_Seg = " ".join([each for each in answer])
print("output")
print(" --------")
print(Aft_Seg)
C = len(answer)
score = C * N / N
print("Score",score)
```

**OUTPUT:**



```
= RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract 2h.py
MENU
-----------
 1 . Hash tag segmentation
 2 . URL segmentation
enter the input choice for performing word segmentation
1
input with HashTag #whatismyname
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
---------
what is my name
Score 4.0

= RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract 2h.py
MENU
-----------
 1 . Hash tag segmentation
 2 . URL segmentation
enter the input choice for performing word segmentation
2
input with URL www.whatismyname.com
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
---------
what is my name
Score 4.0
```

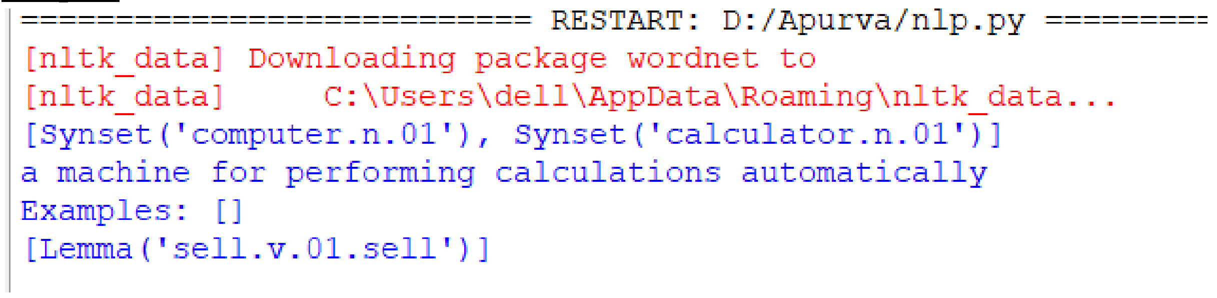
# Practical No.3

**a) Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms**

**Source code:**

```
import nltk
from nltk.corpus import wordnet
nltk.download('wordnet')
print(wordnet.synsets("computer"))
print(wordnet.synset("computer.n.01").definition())
print("Examples:", wordnet.synset("computer.n.01").examples())
#get Antonyms
print(wordnet.lemma('buy.v.01.buy').antonyms())
```

**Output:**



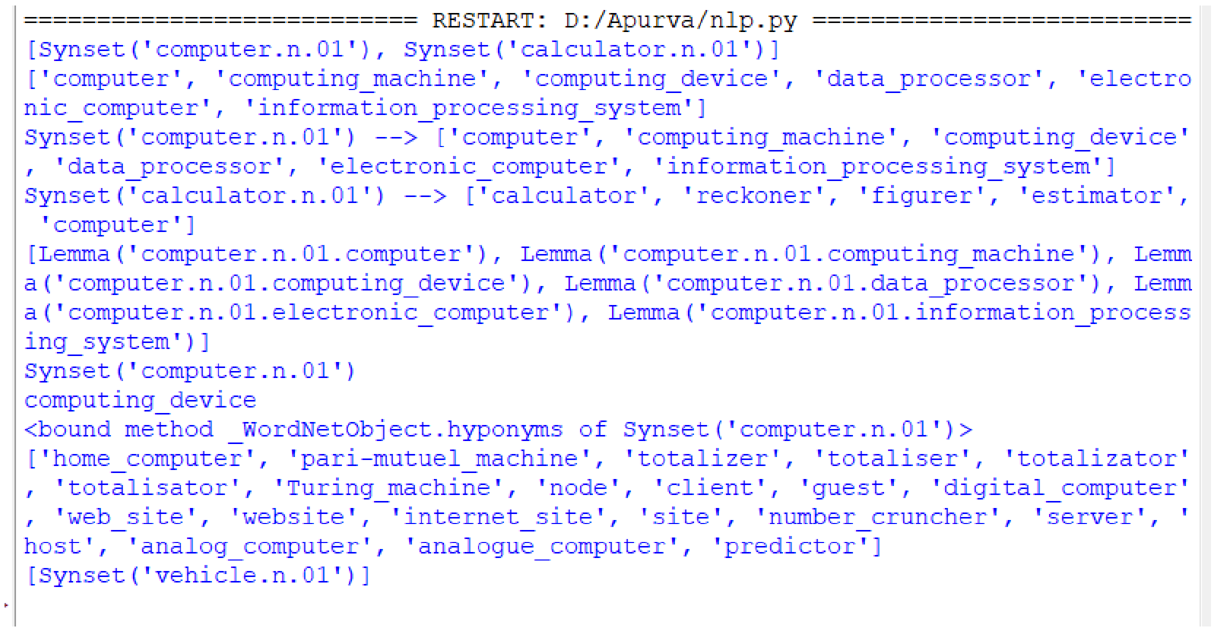
```
============================ RESTART: D:/Apurva/nlp.py =========
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\dell\AppData\Roaming\nltk_data...
[Synset('computer.n.01'), Synset('calculator.n.01')]
a machine for performing calculations automatically
Examples: []
[Lemma('sell.v.01.sell')]
```

**b) Study lemmas, hyponyms, hypernyms.**

**Source 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()])
#the semantic similarity in WordNet
vehicle = wordnet.synset('vehicle.n.01')
car = wordnet.synset('car.n.01')
print(car.lowest_common_hypernyms(vehicle))
```

**Output:**



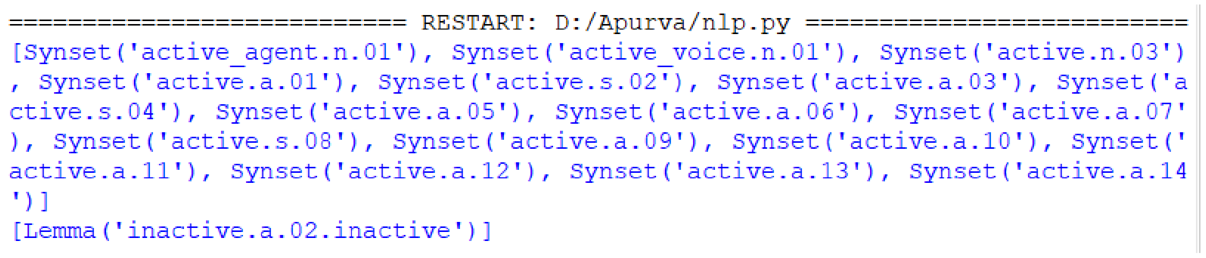
```
========================== RESTART: D:/Apurva/nlp.py ==========================
[Synset('computer.n.01'), Synset('calculator.n.01')]
['computer', 'computing_machine', 'computing_device', 'data_processor', 'electro
nic_computer', 'information_processing_system']
Synset('computer.n.01') --> ['computer', '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'), Lemm
a('computer.n.01.computing_device'), Lemma('computer.n.01.data_processor'), Lemm
a('computer.n.01.electronic_computer'), Lemma('computer.n.01.information_process
ing_system')]
Synset('computer.n.01')
computing_device
<bound method _WordNetObject.hyponyms of Synset('computer.n.01')>
['home_computer', 'pari-mutuel_machine', 'totalizer', 'totaliser', 'totalizator'
, 'totalisator', 'Turing_machine', 'node', 'client', 'guest', 'digital_computer'
, 'web_site', 'website', 'internet_site', 'site', 'number_cruncher', 'server', '
host', 'analog_computer', 'analogue_computer', 'predictor']
[Synset('vehicle.n.01')]
```

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

**Source code:**

```
import nltk
from nltk.corpus import wordnet
print(wordnet.synsets("active"))
print(wordnet.lemma('active.a.01.active').antonyms())
```

**Output:**



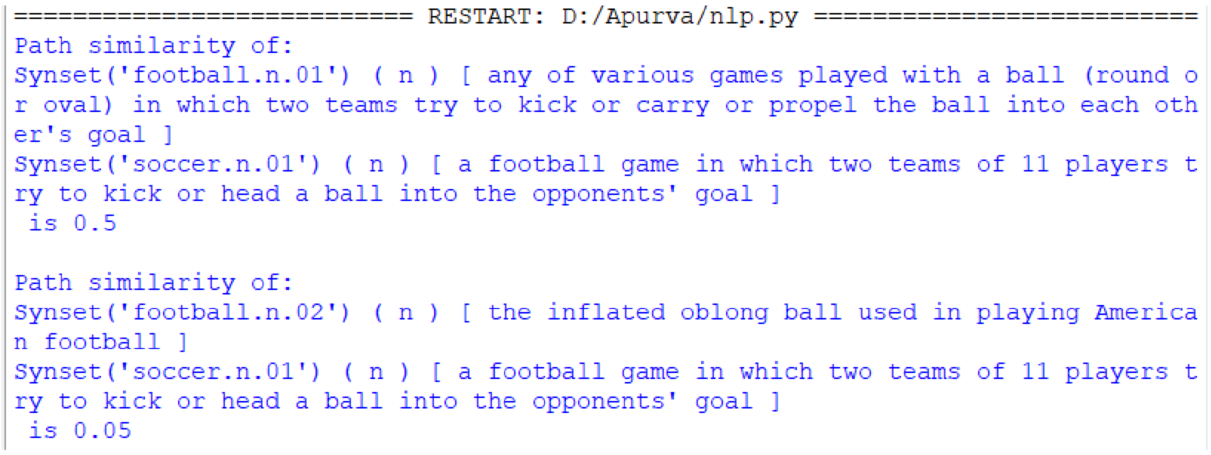
```
========================== RESTART: D:/Apurva/nlp.py ==========================
[Synset('active_agent.n.01'), Synset('active_voice.n.01'), Synset('active.n.03')
, Synset('active.a.01'), Synset('active.s.02'), Synset('active.a.03'), Synset('a
ctive.s.04'), Synset('active.a.05'), Synset('active.a.06'), 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')]
```

**d) Compare two nouns**

**Source code:**

```
import nltk
from nltk.corpus import wordnet
syn1 = wordnet.synsets('football')
syn2 = wordnet.synsets('soccer')
# A word may have multiple synsets, so need to compare each synset of word1 with synset of
word2
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: D:/Apurva/nlp.py ==========================
Path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with a ball (round o
r oval) in which two teams try to kick or carry or propel the ball into each oth
er's goal ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players t
ry 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 America
n football ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players t
ry to kick or head a ball into the opponents' goal ]
 is 0.05
```

## e)Handling stopword:

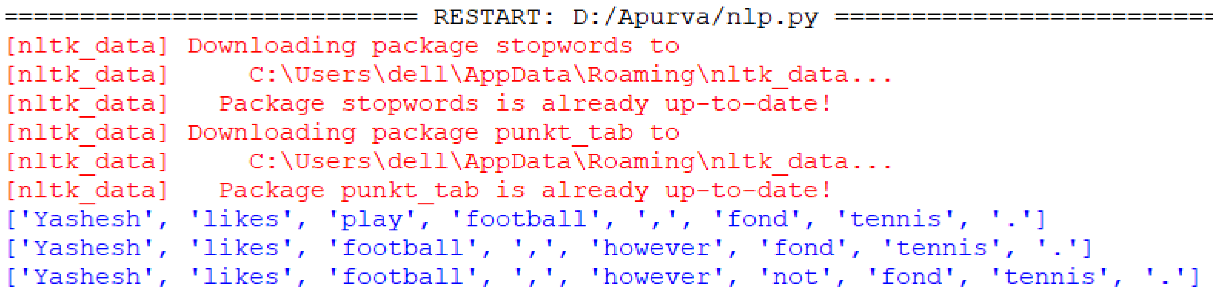
### i) Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List

**Source code:**

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
text = "Yashesh likes to play football, however he is not too fond of tennis."
nltk.download('punkt_tab')
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]
print(tokens_without_sw)
#add the word play to the NLTK stop word collection
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)
#remove ‘not’ from stop word collection
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)

**<u>Output:</u>**



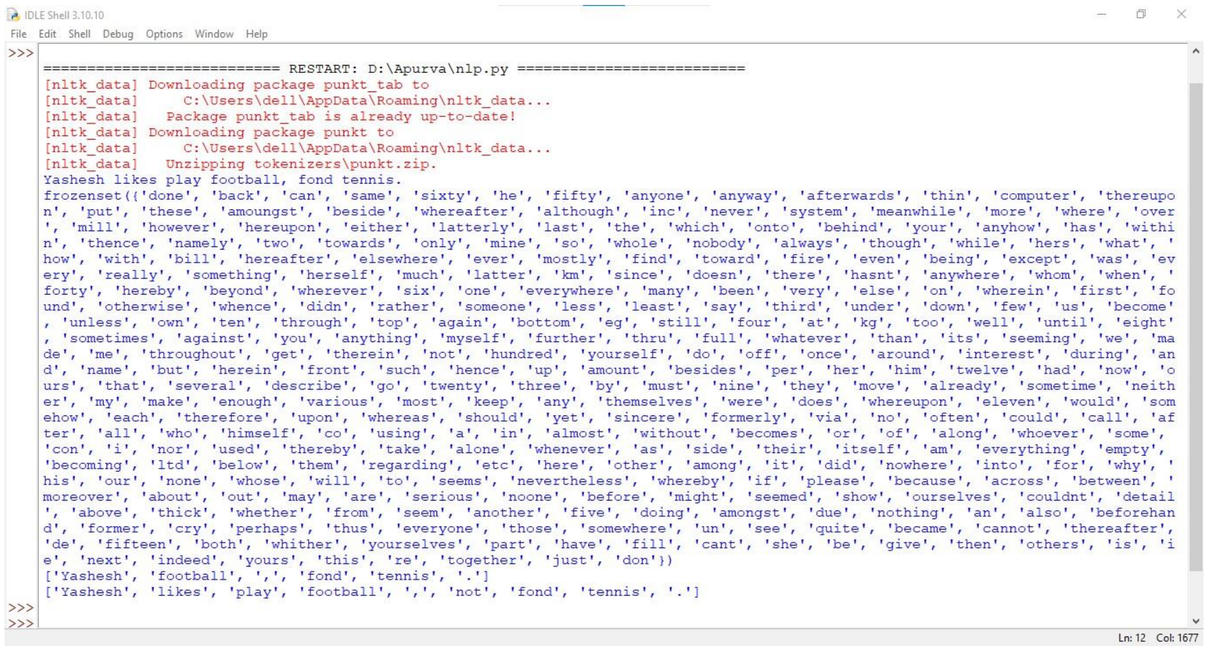
```
============================ RESTART: D:/Apurva/nlp.py ==========================
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\dell\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\dell\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
['Yashesh', 'likes', 'play', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'however', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'however', 'not', 'fond', 'tennis', '.']
```

## ii) Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List

**<u>Source code:</u>**

```
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt_tab')
nltk.download('punkt')
import gensim
from gensim.parsing.preprocessing import remove_stopwords
text = "Yashesh likes to play football, however he is not too fond of tennis."
filtered_sentence = remove_stopwords(text)
print(filtered_sentence)
all_stopwords = gensim.parsing.preprocessing.STOPWORDS
print(all_stopwords)

from gensim.parsing.preprocessing import STOPWORDS
all_stopwords_gensim = 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_stopwords_gensim]
print(tokens_without_sw)

from gensim.parsing.preprocessing import STOPWORDS
all_stopwords_gensim = STOPWORDS
sw_list = {"not"}
all_stopwords_gensim = 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_stopwords_gensim]
print(tokens_without_sw)
```

**Output:**



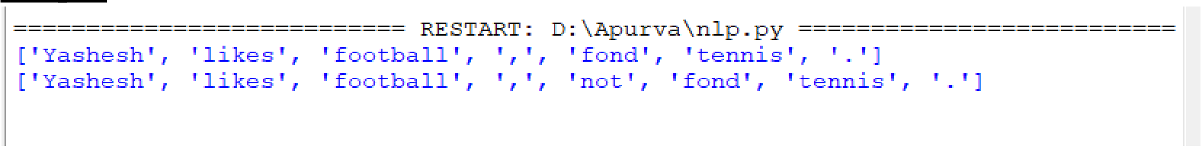
```
>>>
    ========================== RESTART: D:\Apurva\nlp.py ==========================
    [nltk_data] Downloading package punkt_tab to
    [nltk_data]     C:\Users\dell\AppData\Roaming\nltk_data...
    [nltk_data]   Package punkt_tab is already up-to-date!
    [nltk_data] Downloading package punkt to
    [nltk_data]     C:\Users\dell\AppData\Roaming\nltk_data...
    [nltk_data]   Unzipping tokenizers\punkt.zip.
    Yashesh likes play football, fond tennis.
    frozenset({'done', 'back', 'can', 'same', 'sixty', 'he', 'fifty', 'anyone', 'anyway', 'afterwards', 'thin', 'computer', 'thereupo
    n', 'put', 'these', 'amoungst', 'beside', 'whereafter', 'although', 'inc', 'never', 'system', 'meanwhile', 'more', 'where', 'over
    ', 'mill', 'however', 'hereupon', 'either', 'latterly', 'last', 'the', 'which', 'onto', 'behind', 'your', 'anyhow', 'has', 'withi
    n', 'thence', 'namely', 'two', 'towards', 'only', 'mine', 'so', 'whole', 'nobody', 'always', 'though', 'while', 'hers', 'what', '
    how', 'with', 'bill', 'hereafter', 'elsewhere', 'ever', 'mostly', 'find', 'toward', 'fire', 'even', 'being', 'except', 'was', 'ev
    ery', 'really', 'something', 'herself', 'much', 'latter', 'km', 'since', 'doesn', 'there', 'hasnt', 'anywhere', 'whom', 'when', '
    forty', 'hereby', 'beyond', 'wherever', 'six', 'one', 'everywhere', 'many', 'been', 'very', 'else', 'on', 'wherein', 'first', 'fo
    und', 'otherwise', 'whence', 'didn', 'rather', 'someone', 'less', 'least', 'say', 'third', 'under', 'down', 'few', 'us', 'become'
    , 'unless', 'own', 'ten', 'through', 'top', 'again', 'bottom', 'eg', 'still', 'four', 'at', 'kg', 'too', 'well', 'until', 'eight'
    , 'sometimes', 'against', 'you', 'anything', 'myself', 'further', 'thru', 'full', 'whatever', 'than', 'its', 'seeming', 'we', 'ma
    de', 'me', 'throughout', 'get', 'therein', 'not', 'hundred', 'yourself', 'do', 'off', 'once', 'around', 'interest', 'during', 'an
    d', 'name', 'but', 'herein', 'front', 'such', 'hence', 'up', 'amount', 'besides', 'per', 'her', 'him', 'twelve', 'had', 'now', 'o
    urs', 'that', 'several', 'describe', 'go', 'twenty', 'three', 'by', 'must', 'nine', 'they', 'move', 'already', 'sometime', 'neith
    er', 'my', 'make', 'enough', 'various', 'most', 'keep', 'any', 'themselves', 'were', 'does', 'whereupon', 'eleven', 'would', 'som
    ehow', 'each', 'therefore', 'upon', 'whereas', 'should', 'yet', 'sincere', 'formerly', 'via', 'no', 'often', 'could', 'call', 'af
    ter', 'all', 'who', 'himself', 'co', 'using', 'a', 'in', 'almost', 'without', 'becomes', 'or', 'of', 'along', 'whoever', 'some',
    'con', 'i', 'nor', 'used', 'thereby', 'take', 'alone', 'whenever', 'as', 'side', 'their', 'itself', 'am', 'everything', 'empty',
    'becoming', 'ltd', 'below', 'them', 'regarding', 'etc', 'here', 'other', 'among', 'it', 'did', 'nowhere', 'into', 'for', 'why', '
    his', 'our', 'none', 'whose', 'will', 'to', 'seems', 'nevertheless', 'whereby', 'if', 'please', 'because', 'across', 'between', '
    moreover', 'about', 'out', 'may', 'are', 'serious', 'noone', 'before', 'might', 'seemed', 'show', 'ourselves', 'couldnt', 'detail
    ', 'above', 'thick', 'whether', 'from', 'seem', 'another', 'five', 'doing', 'amongst', 'due', 'nothing', 'an', 'also', 'beforehan
    d', 'former', 'cry', 'perhaps', 'thus', 'everyone', 'those', 'somewhere', 'un', 'see', 'quite', 'became', 'cannot', 'thereafter',
    'de', 'fifteen', 'both', 'whither', 'yourselves', 'part', 'have', 'fill', 'cant', 'she', 'be', 'give', 'then', 'others', 'is', 'i
    e', 'next', 'indeed', 'yours', 'this', 're', 'together', 'just', 'don'})
    ['Yashesh', 'football', ',', 'fond', 'tennis', '.']
    ['Yashesh', 'likes', 'play', 'football', ',', 'not', 'fond', 'tennis', '.']
>>>
>>>
```

## iii) Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List

**Source code:**

```
import spacy
import nltk
from nltk.tokenize import word_tokenize
sp = spacy.load('en_core_web_sm')
#add the word play to the NLTK stop word collection
all_stopwords = sp.Defaults.stop_words
all_stopwords.add("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_stopwords]
print(tokens_without_sw)
#remove 'not' from stop word collection
all_stopwords.remove('not')
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
```

**Output**



```
========================== RESTART: D:\Apurva\nlp.py ==========================
['Yashesh', 'likes', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'not', 'fond', 'tennis', '.']
```

# Practical No. 4

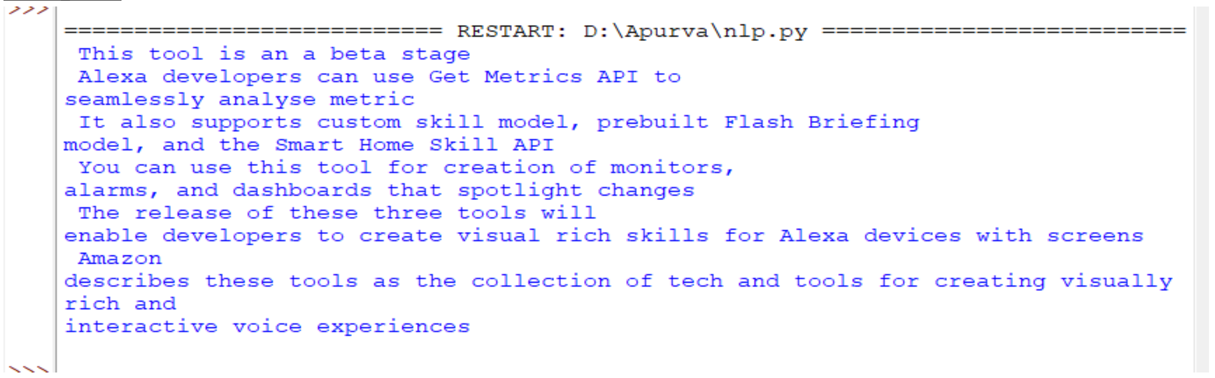
## Text Tokenization

### a)Tokenization using Python’s split() function

**Source code:**

```
text = """ This tool is an a beta stage. Alexa developers can use Get Metrics API to
seamlessly analyse metric. It also supports custom skill model, prebuilt Flash Briefing
model, and the Smart Home Skill API. You can use this tool for creation of monitors,
alarms, and dashboards that spotlight changes. The release of these three tools will
enable developers to create visual rich skills for Alexa devices with screens. Amazon
describes these tools as the collection of tech and tools for creating visually rich and
interactive voice experiences. """
data = text.split('.')
for i in data:
   print (i)
```

**Output:**



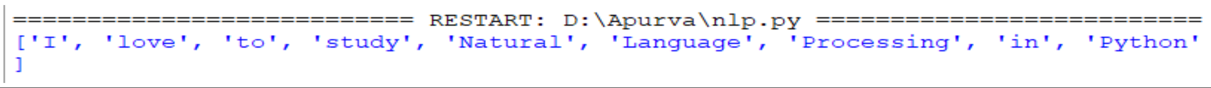
```
>>>
========================== RESTART: D:\Apurva\nlp.py ==========================
 This tool is an a beta stage
 Alexa developers can use Get Metrics API to
seamlessly analyse metric
 It also supports custom skill model, prebuilt Flash Briefing
model, and the Smart Home Skill API
 You can use this tool for creation of monitors,
alarms, and dashboards that spotlight changes
 The release of these three tools will
enable developers to create visual rich skills for Alexa devices with screens
 Amazon
describes these tools as the collection of tech and tools for creating visually
rich and
interactive voice experiences

>>>
```

### b) Tokenization using Regular Expressions (RegEx)

**Source code:**

```
import nltk
from nltk.tokenize import RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps = True)
str = "I love to study Natural Language Processing in Python"
tokens = tk.tokenize(str)
print(tokens)
```

**Output:**



```
=========================== RESTART: D:\Apurva\nlp.py ==========================
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python'
]
```

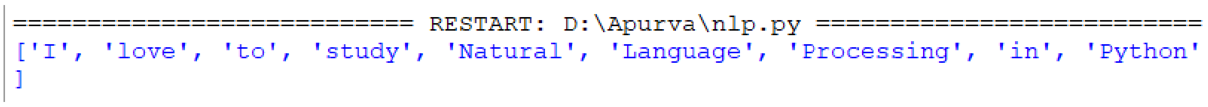
### c) Tokenization using NLTK

**Source code:**

```
import nltk
from nltk.tokenize import word_tokenize
```

```
str = "I love to study Natural Language Processing in Python"
print(word_tokenize(str))
```

**<u>Output:</u>**



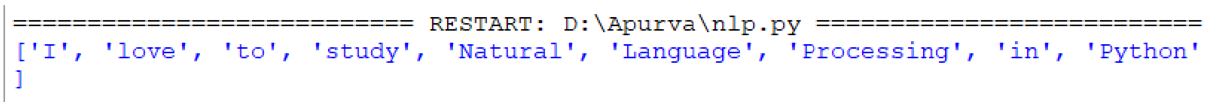
```
=========================== RESTART: D:\Apurva\nlp.py ==========================
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python'
]
```

## d) Tokenization using the spaCy library

**<u>Source code:</u>**

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

**<u>Output:</u>**



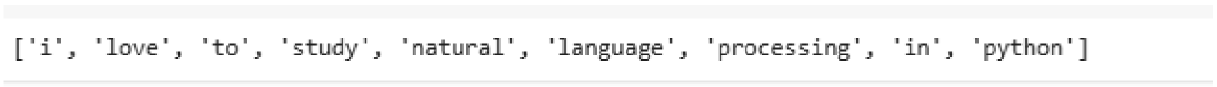
```
=========================== RESTART: D:\Apurva\nlp.py ==========================
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python'
]
```

## e)Tokenization using Keras

**<u>Source code:</u>**

```
#pip install keras tensorflow
from tensorflow.keras.preprocessing.text import text_to_word_sequence
text = "I love to study Natural Language Processing in Python"
tokens = text_to_word_sequence(text)
print(tokens)
```

**<u>Output:</u>**



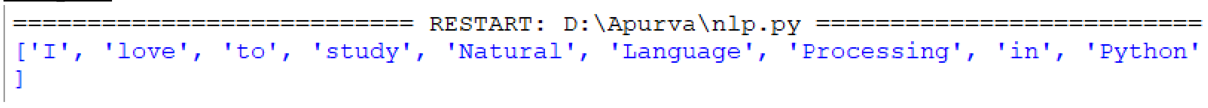
```
['i', 'love', 'to', 'study', 'natural', 'language', 'processing', 'in', 'python']
```

## f. Tokenization using Gensim

**<u>Source code:</u>**

```
from gensim.utils import tokenize
str = "I love to study Natural Language Processing in Python"
print(list(tokenize(str)))
```

**<u>Output:</u>**



```
=========================== RESTART: D:\Apurva\nlp.py ==========================
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python'
]
```

# Practical No. 5

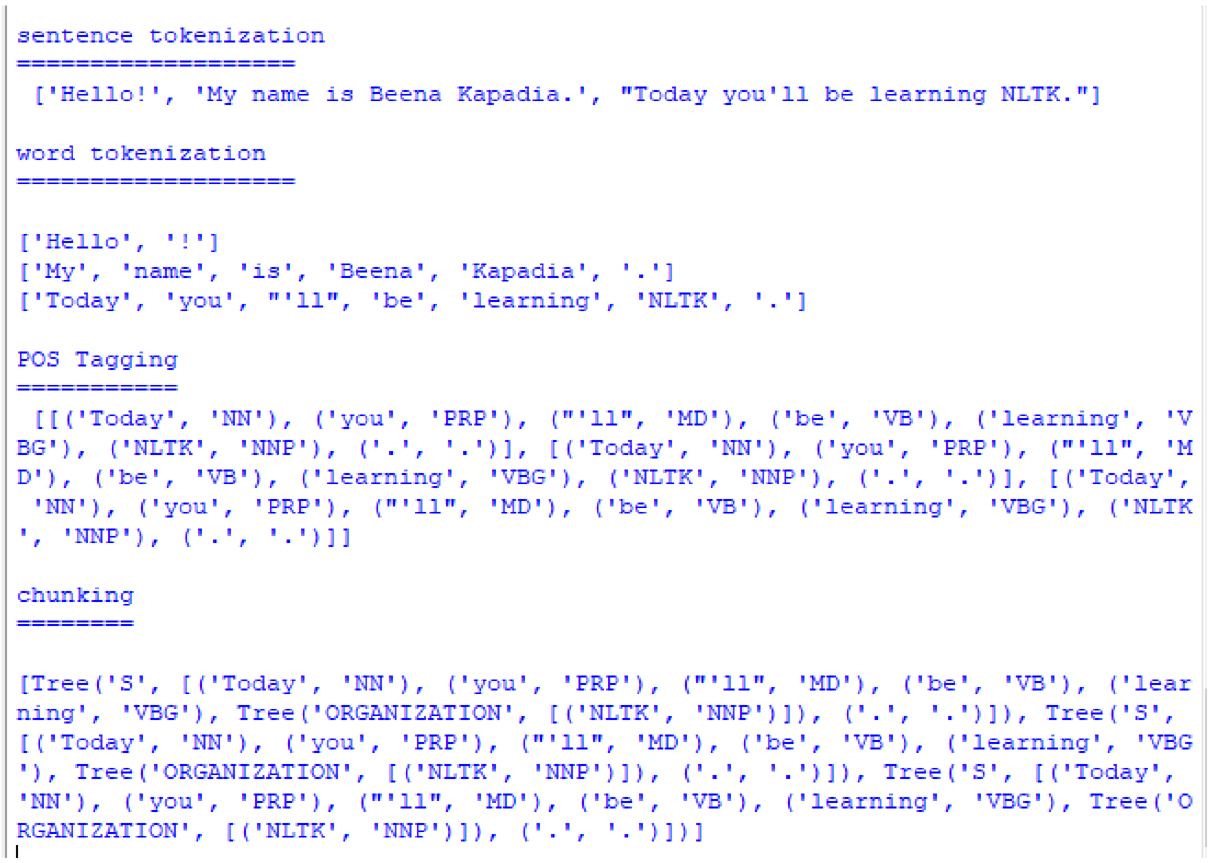
## Illustrate part of speech tagging.

**a) sentence tokenization, word tokenization, Part of speech Tagging and chunking of user defined text.**

**Source code:**

```
import nltk
from nltk import tokenize
nltk.download('punkt')
from nltk import tag
from nltk import chunk
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('maxent_ne_chunker_tab')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
para = "Hello! My name is Beena Kapadia. 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:**



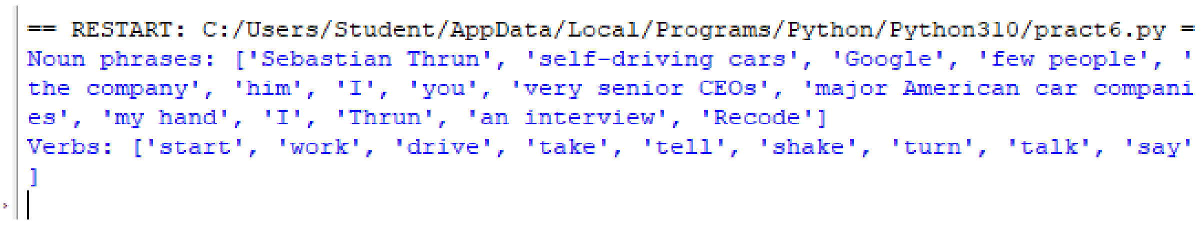
```
sentence tokenization
===================
 ['Hello!', 'My name is Beena Kapadia.', "Today you'll be learning NLTK."]

word tokenization
===================

['Hello', '!']
['My', 'name', 'is', 'Beena', 'Kapadia', '.']
['Today', 'you', "'ll", 'be', 'learning', 'NLTK', '.']

POS Tagging
===========
 [[('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'V
BG'), ('NLTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('you', 'PRP'), ("'ll", 'M
D'), ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')], [('Today',
 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'), ('NLTK
', 'NNP'), ('.', '.')]]

chunking
========

[Tree('S', [('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('lear
ning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')]), Tree('S',
[('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG
'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')]), Tree('S', [('Today',
'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'), Tree('O
RGANIZATION', [('NLTK', 'NNP')]), ('.', '.')])]
```

## b) Named Entity recognition using user defined text.

**Source code:**

```
import spacy
# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")
# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
"Google in 2007, few people outside of the company took him "
"seriously. “I can tell you very senior CEOs of major American "
"car companies would shake my hand and turn away because I wasn’t "
"worth talking to,” said Thrun, in an interview with Recode earlier "
"this week.")
doc = nlp(text)
# Analyse syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])
```

**Output:**



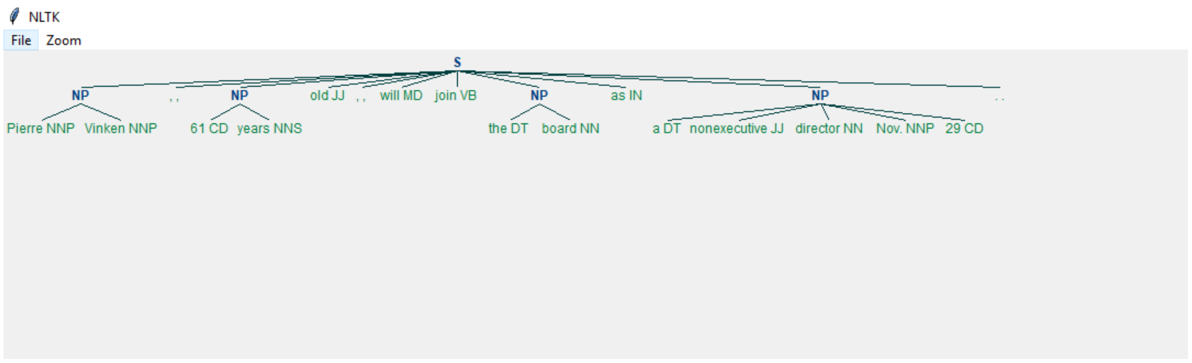
```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract6.py =
Noun phrases: ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people', '
the company', 'him', 'I', 'you', 'very senior CEOs', 'major American car compani
es', 'my hand', 'I', 'Thrun', 'an interview', 'Recode']
Verbs: ['start', 'work', 'drive', 'take', 'tell', 'shake', 'turn', 'talk', 'say'
]
>
```

**c) Named Entity recognition with diagram using NLTK corpus – treebank.**

**Source 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 No. 6

## Finite state automata

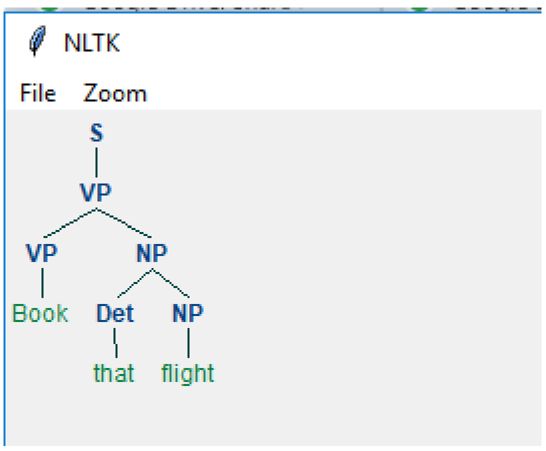
**a) Define grammar using nltk. Analyze a sentence using the same.**

**Source 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:**

```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract6.py =
['Book', 'that', 'flight']
(S (VP (VP Book) (NP (Det that) (NP flight))))
```

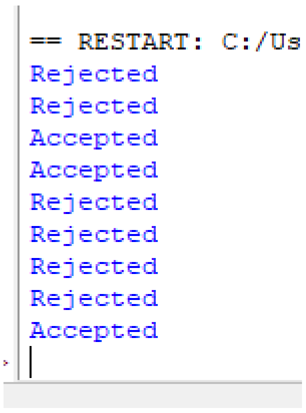


**b) Accept the input string with Regular expression of Finite Automaton: 101+.**

**Source 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':
                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:**



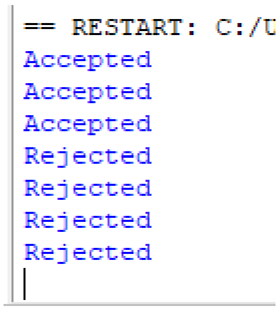
```
== RESTART: C:/Us
Rejected
Rejected
Accepted
Accepted
Rejected
Rejected
Rejected
Rejected
Accepted
```

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

**Source 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:**



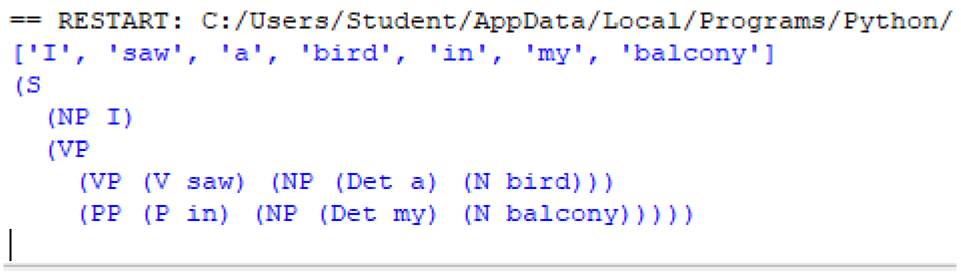
```
== RESTART: C:/U
Accepted
Accepted
Accepted
Rejected
Rejected
Rejected
Rejected
```

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

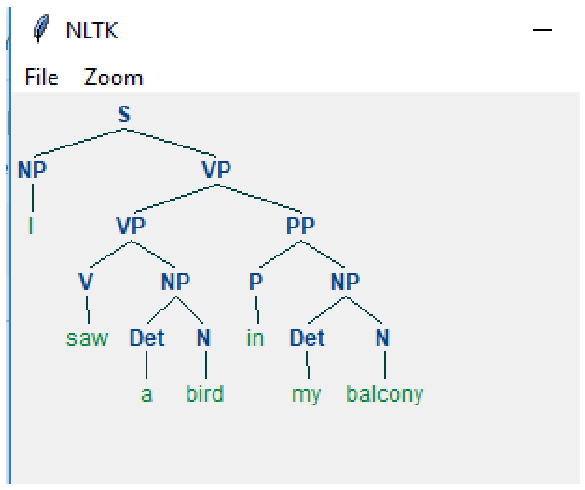
**Source code:**

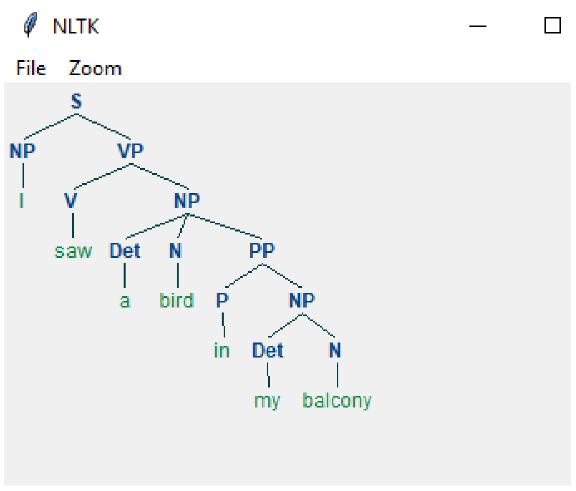
```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'a' | 'my'
N -> 'bird' | 'balcony'
V -> 'saw'
P -> 'in'
""")
sentence = "I saw a bird in my balcony"
for index in range(len(sentence)):
    all_tokens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



```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/
['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)))))
```





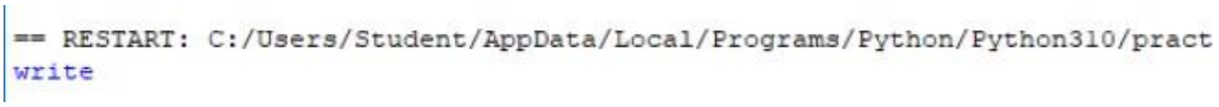
# Practical No.7

**Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer**
**Study WordNetLemmatizer**

**Source Code:**

```
# PorterStemmer
import nltk
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
print(word_stemmer.stem('writing'))
```

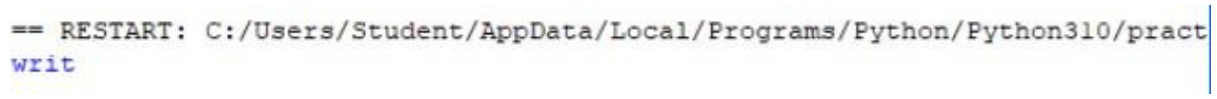
**Output:**



```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract
write
```

```
#LancasterStemmer
import nltk
from nltk.stem import LancasterStemmer
Lanc_stemmer = LancasterStemmer()
print(Lanc_stemmer.stem('writing'))
```

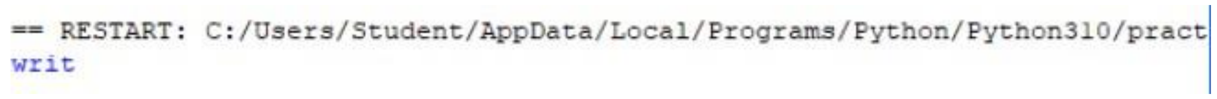
**Output:**



```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract
writ
```

```
#RegexpStemmer
import nltk
from nltk.stem import RegexpStemmer
Reg_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4)
print(Reg_stemmer.stem('writing'))
```

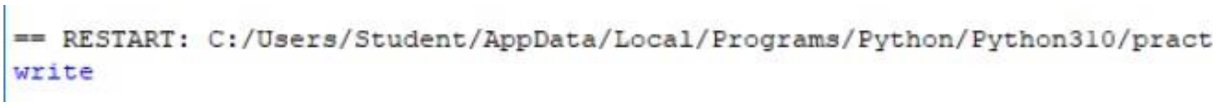
**Output:**



```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract
writ
```

```
#SnowballStemmer
import nltk
from nltk.stem import SnowballStemmer
english_stemmer = SnowballStemmer('english')
print(english_stemmer.stem ('writing'))
```

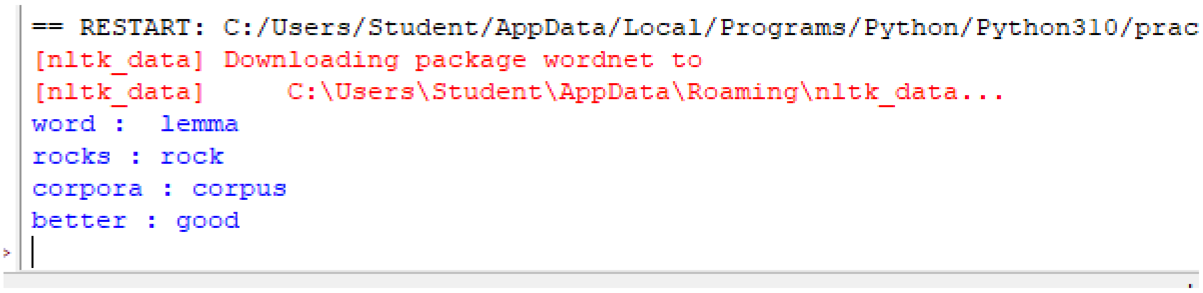
**Output:**



```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/pract
write
```

**#WordNetLemmatizer**
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
    print("word :\tlemma")
    print("rocks :",
    lemmatizer.lemmatize("rocks"
    ))
print("corpora :", lemmatizer.lemmatize("corpora"))
print("better :", lemmatizer.lemmatize("better", pos ="a"))

**Output:**



```
== RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python310/prac
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Student\AppData\Roaming\nltk_data...
word :  lemma
rocks : rock
corpora : corpus
better : good
>
```

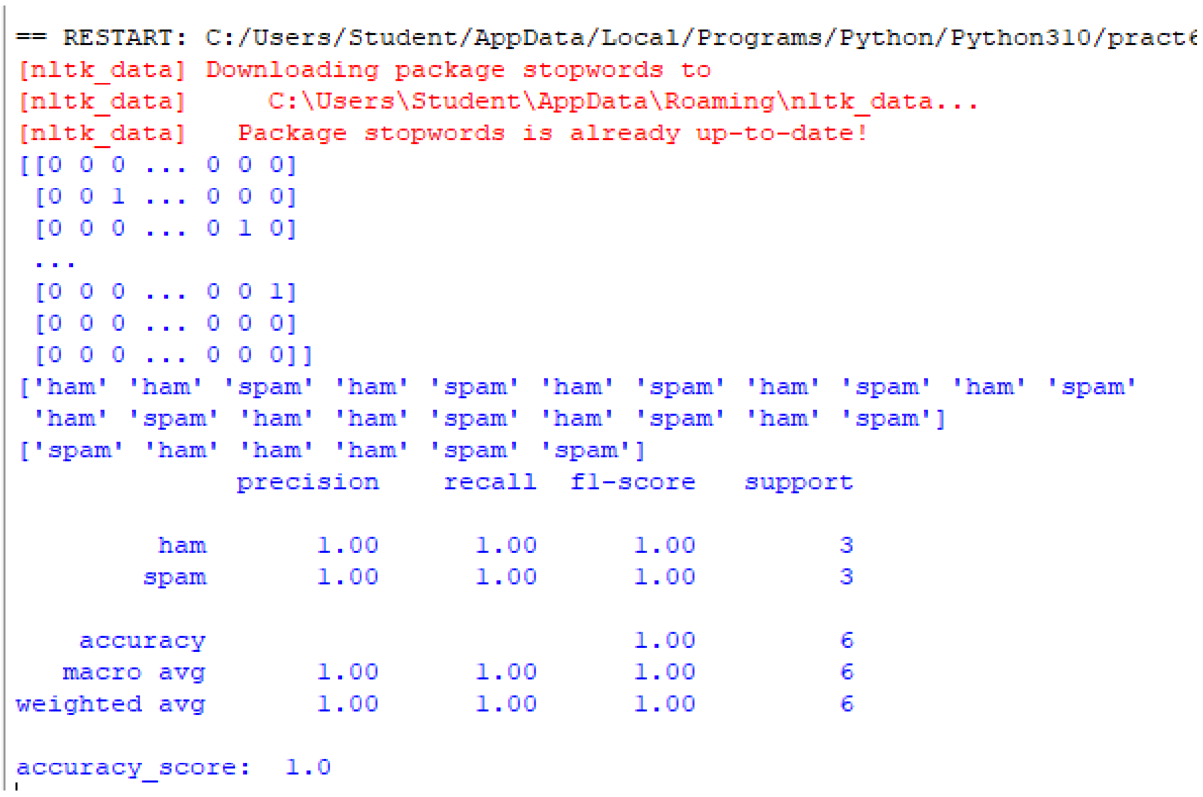
# Practical No. 8

## Implement Naive Bayes classifier

**Source Code:**

```
import pandas as pd
import numpy as np
sms_data = pd.read_csv("D:\spam.csv", encoding='latin-1')
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
nltk.download('stopwords')
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/Student/AppData/Local/Programs/Python/Python310/pract
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[[0 0 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 [0 0 0 ... 0 1 0]
 ...
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
['ham' 'ham' 'spam' 'ham' 'spam' 'ham' 'spam' 'ham' 'spam' 'ham' 'spam'
 'ham' 'spam' 'ham' 'ham' 'spam' 'ham' 'spam' 'ham' 'spam']
['spam' 'ham' 'ham' 'ham' 'spam' 'spam']
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00         3
        spam       1.00      1.00      1.00         3

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6

accuracy_score:  1.0
```