**Practical No: 1**

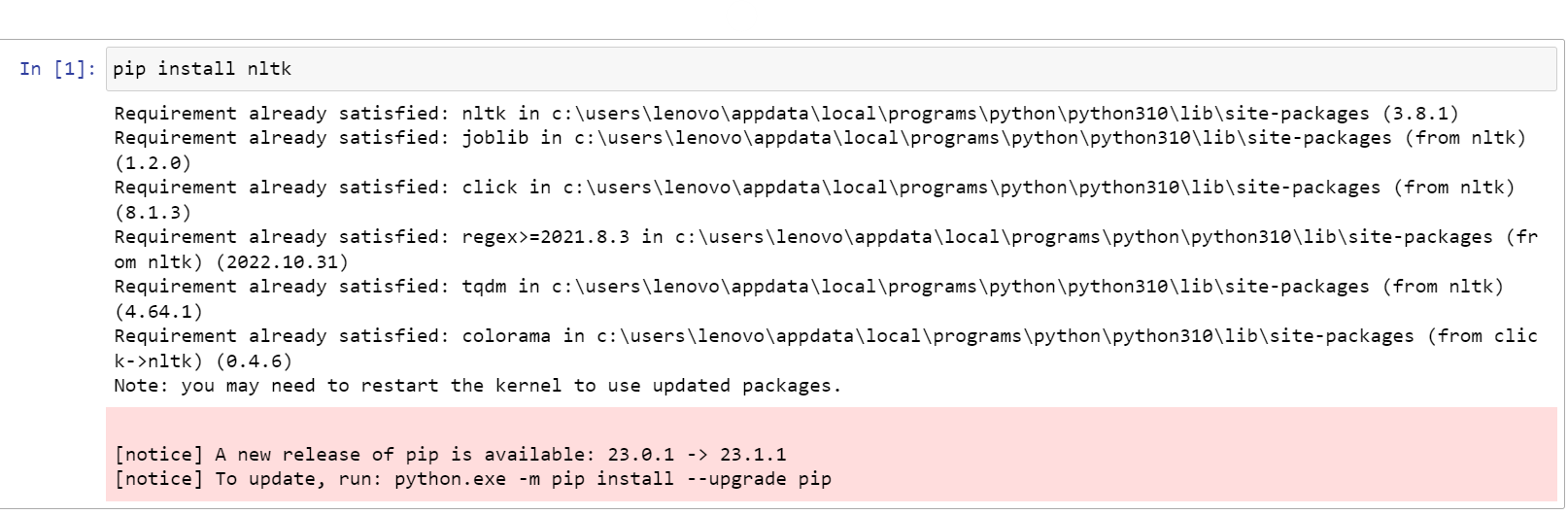
**Practical 1 a)**  **Date: 28/02/2023**

**Aim:** Install NLTK

**Code:**

pip install nltk

**Output:**



**Practical 1 b)** **Date: 28/02/2023**

**Aim:** Convert the given text to speech.

**Code:**

pip install gTTS

# Import the gTTS module for text

# to speech conversion

from gtts import gTTS

pip install playsound

# This module is imported so that we can

# play the converted audio

from playsound import playsound

# It is a text value that we want to convert to audio

text\_val = 'All the best for your exam.'

# Here are converting in English Language

language = 'en'

# Passing the text and language to the engine,

# here we have assign slow=False. Which denotes

# the module that the transformed audio should

# have a high speed

obj = gTTS(text=text\_val, lang=language, slow=False)

#Here we are saving the transformed audio in a mp3 file named

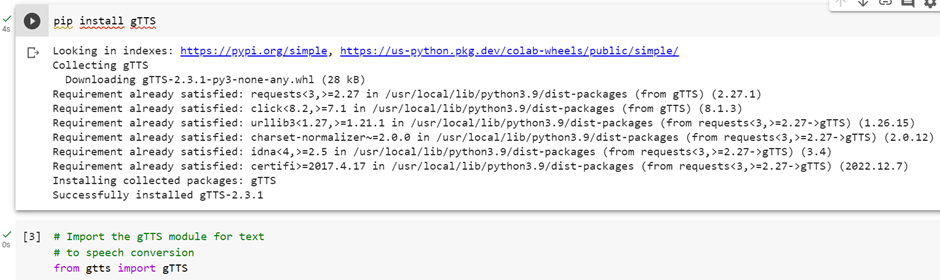
# exam.mp3

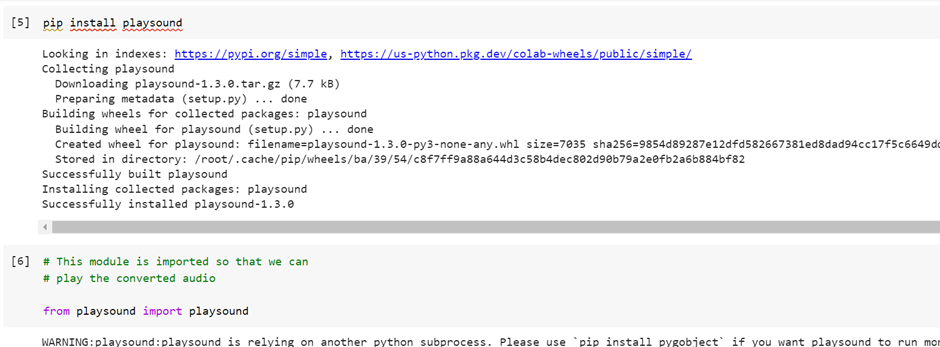
obj.save("exam.mp3")

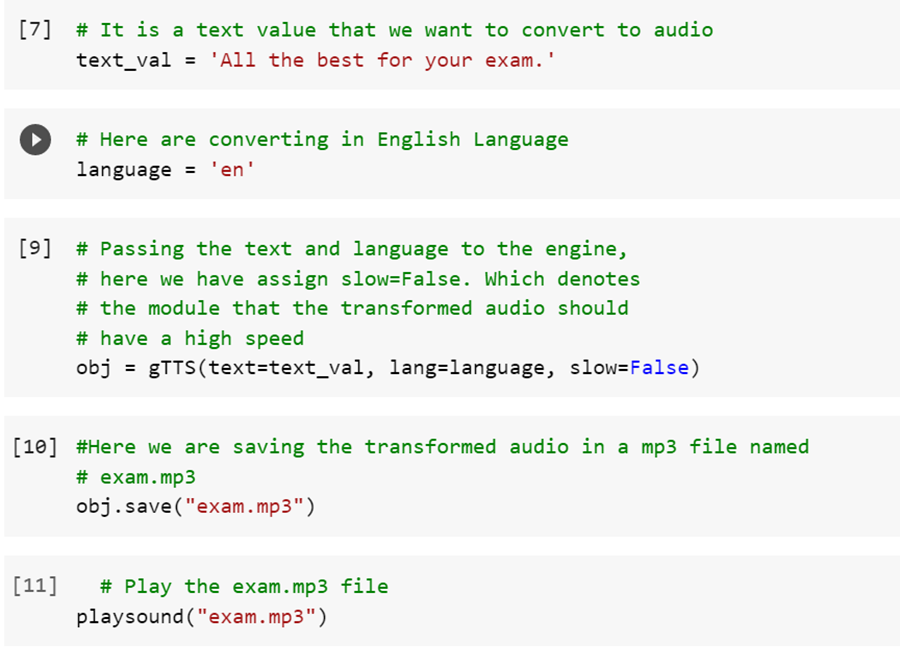
# Play the exam.mp3 file

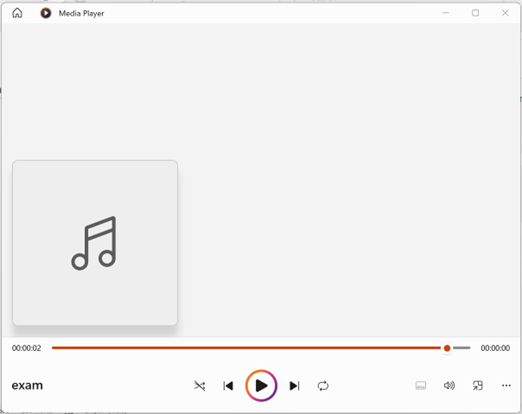
playsound("exam.mp3")

**Output:**









**Practical 1 c)** **Date: 28/02/2023**

**Aim: Convert the Speech of .wav audio file to Text.**

**Code:**

!pip install SpeechRecognition pydub

#import library

import speech\_recognition as sr

r= sr.Recognizer()

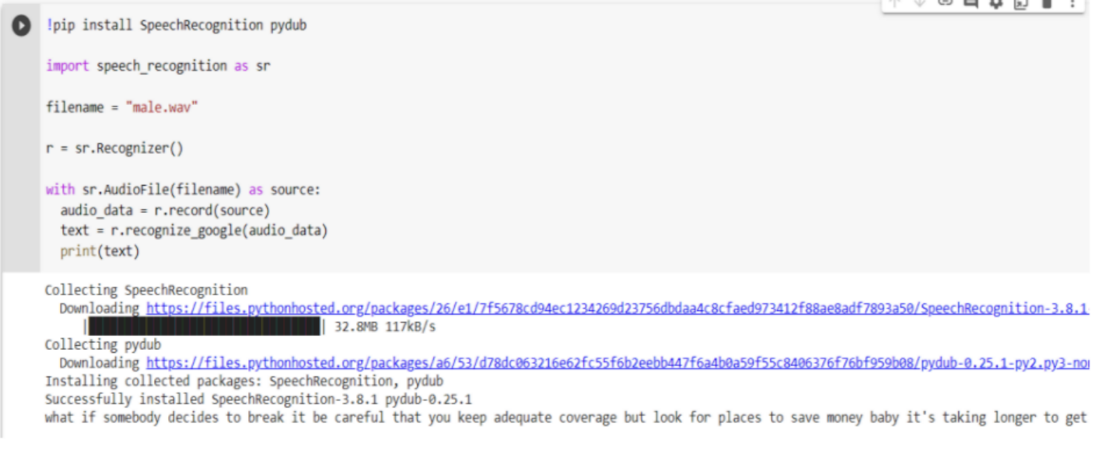
with sr.AudioFile("male.wav")as source:

audio\_data = r.record(source)

text = r.recognize\_google(audio\_data)

print(text)

**Output:**



**Practical No: 2**

**Practical 2 a) Date: 14-03-23**

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

**Code:**

import nltk

from nltk.corpus import brown

nltk.download('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('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFileName')

for fileid in brown.fileids():

num\_chars = len(brown.raw(fileid))

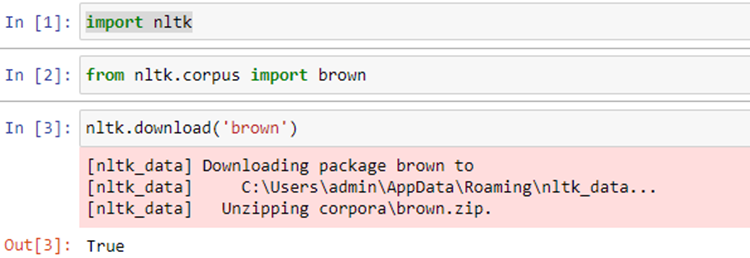
num\_words = len(brown.words(fileid))

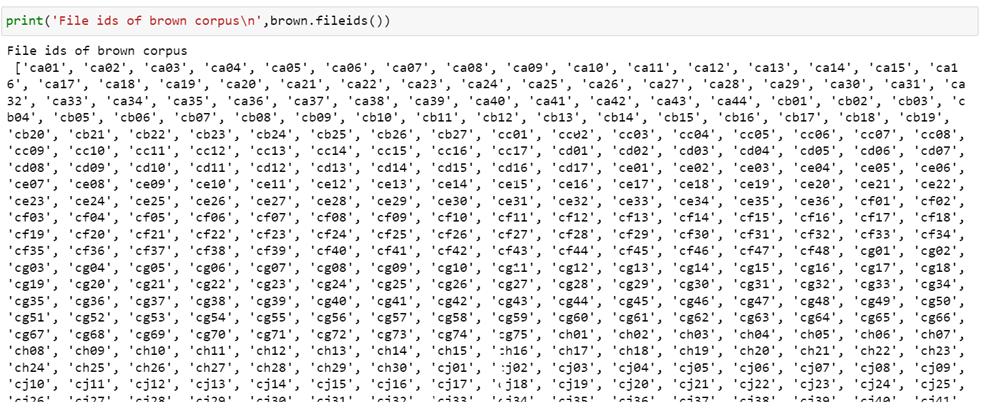
num\_sents = len(brown.sents(fileid))

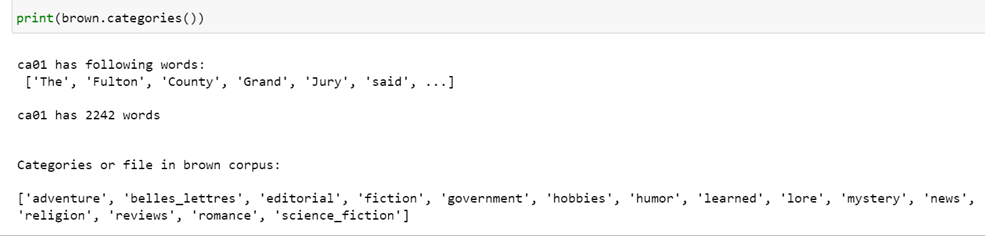
num\_vocab = len(set([w.lower() for w in brown.words(fileid)]))

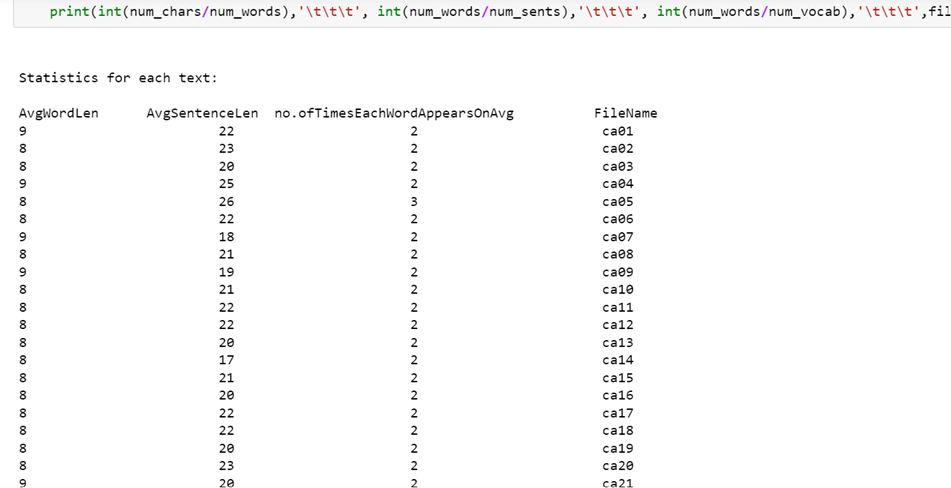
print(int(num\_chars/num\_words),'\t\t\t', int(num\_words/num\_sents),'\t\t\t', int(num\_words/num\_vocab),'\t\t\t',fileid)

**Output:**









**Practical 2 b) Date: 14-03-23**

**Aim: Create and use your own corpora (plaintext, categorical)**

**Code:**

import nltk

from nltk.corpus import PlaintextCorpusReader

corpus\_root="C:/Users/admin/Desktop/corpus"

filelist=PlaintextCorpusReader(corpus\_root,'.\*')

print('\n File list:\n')

print(filelist.fileids())

print(filelist.root)

import nltk

nltk.download('punkt')

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

print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\tFileName')

for fileid in filelist.fileids():

num\_chars=len(filelist.raw(fileid))

num\_words=len(filelist.words(fileid))

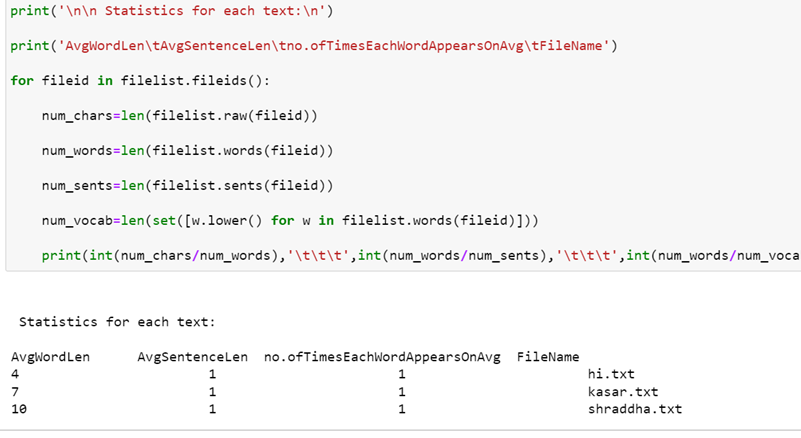
num\_sents=len(filelist.sents(fileid))

num\_vocab=len(set([w.lower() for w in filelist.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:**





**Practical 2 c) Date: 14-03-23**

**Aim: 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 Shraddha Kasar . Today I'll be learning NLTK"

sents=tokenize.sent\_tokenize(para)

print("\nsentence tokenization\n======================\n",sents)

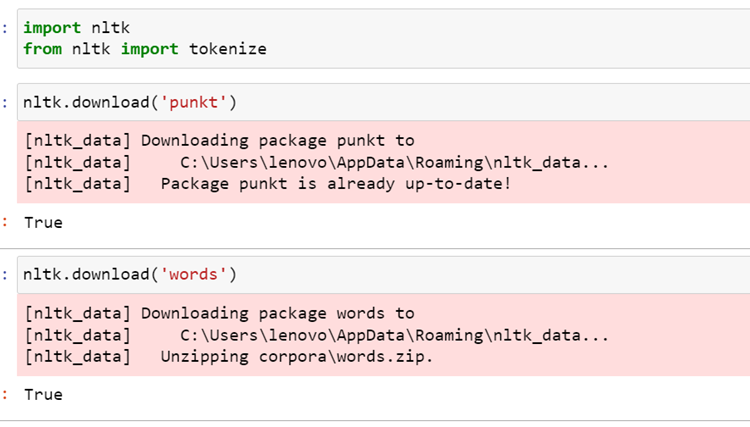
print("\nword tokenization\n==================\n")

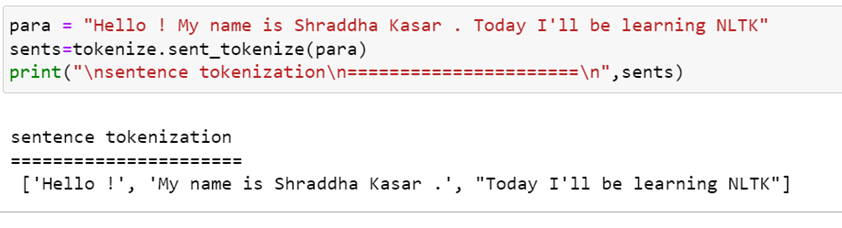
for index in range (len(sents)):

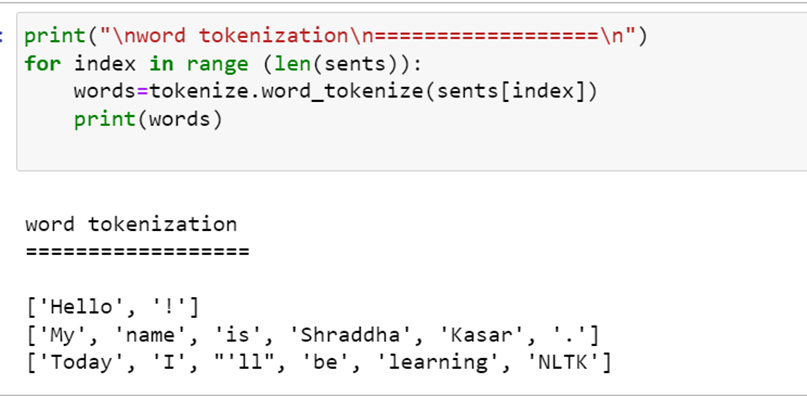
words=tokenize.word\_tokenize(sents[index])

print(words)

**Output:**







**Practical 2 d)** **Date: 14-03-23**

**Aim: Write a program to find the most frequent noun tags.**

**Code:**

import nltk

from collections import defaultdict

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

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:

temp[sub] += 1

res = max(temp, key = temp.get)

print("Word with maximum frequency : " + str(res))

**Output:**

--



**Practical No: 3**

**Practical 3 a) Date: 21/03/2023**

**Aim:** Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms

**Code:**

import nltk

from nltk.corpus import wordnet

nltk.download('wordnet')

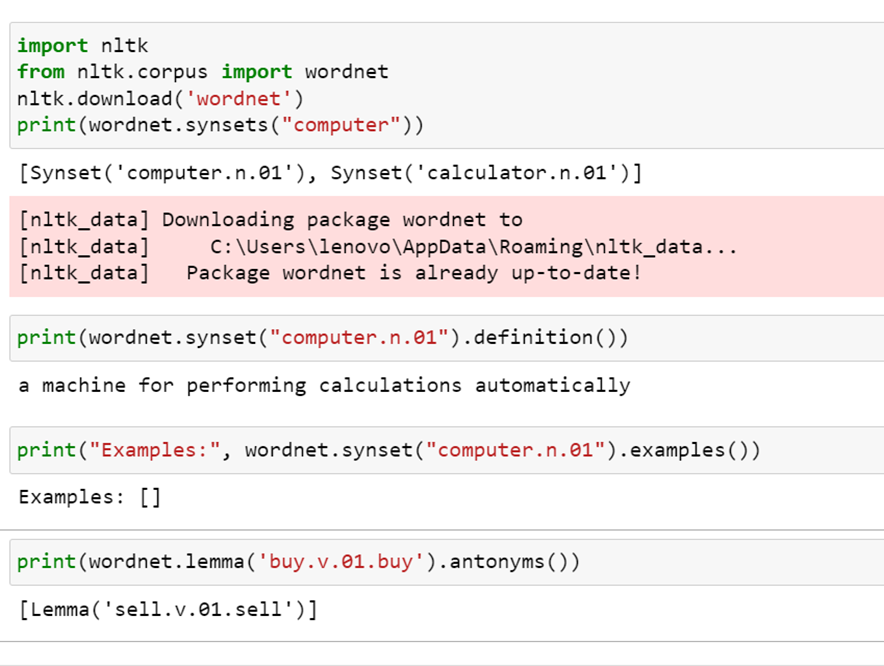
print(wordnet.synsets("computer"))

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

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

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

**Output:**



**Practical 3 b) Date: 21/03/2023**

**Aim: S**tudy lemmas, hyponyms, hypernyms.

**Code:**

import nltk

from nltk.corpus import wordnet

nltk.download('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)

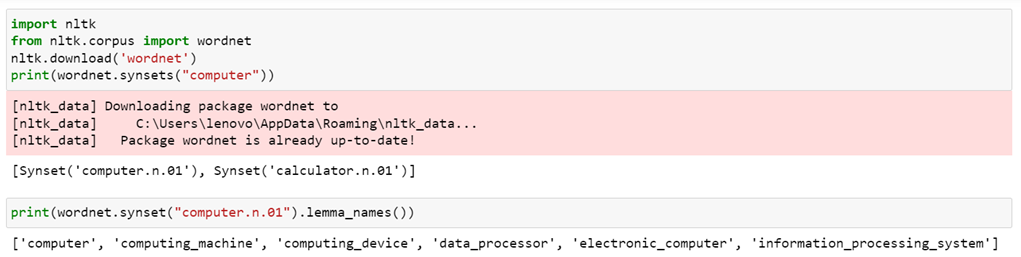
print([lemma.name() for synset in syn.hyponyms() for lemma in synset.lemmas()])

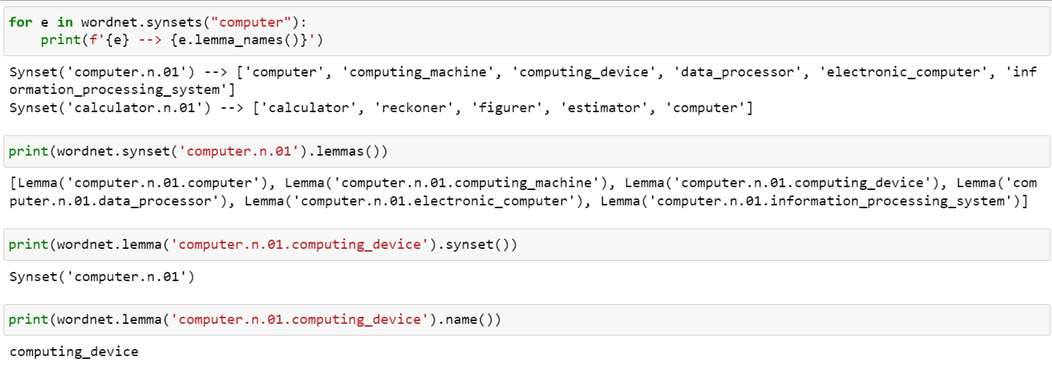
vehicle = wordnet.synset('vehicle.n.01')

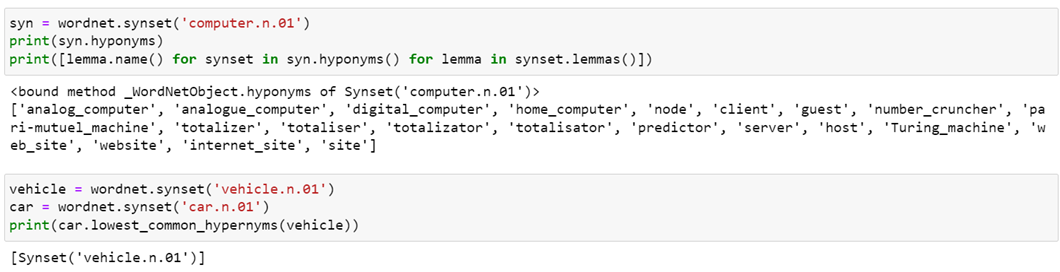
car = wordnet.synset('car.n.01')

print(car.lowest\_common\_hypernyms(vehicle))

**Output:**







**Practical 3 c)** **Date: 21/03/2023**

**Aim:** sentence tokenization, word tokenization, 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 Shraddha kasar. 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)):para = "Hello! My name is Shraddha kasar. Today you'll be learning NLTK."

sents = tokenize.sent\_tokenize(para)

print("\nsentence tokenization\n=================\n",sents)

words = tokenize.word\_tokenize(sents[index])

print(words)

tagged\_words = []

for index in range(len(sents)):

tagged\_words.append(tag.pos\_tag(words))

print("\nPOS Tagging\n=======\n")

tree = []

for index in range(len(sents)):

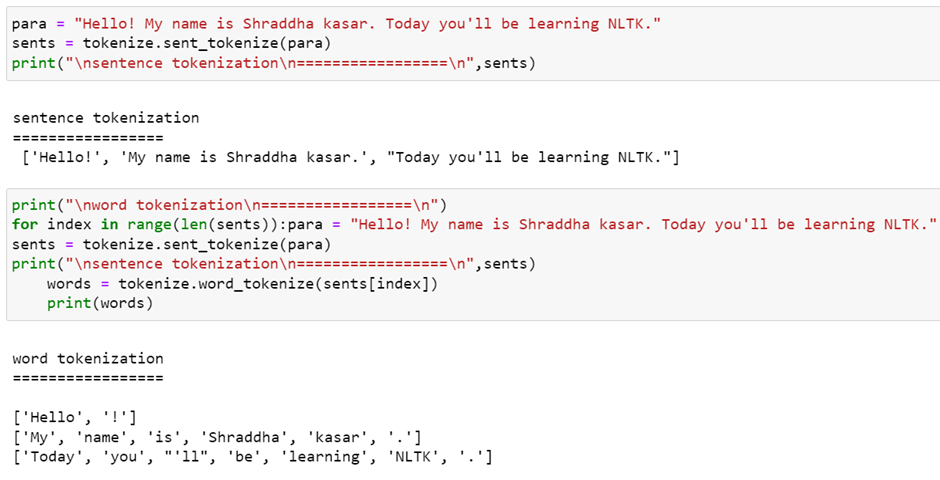
tree.append(chunk.ne\_chunk(tagged\_words[index]))

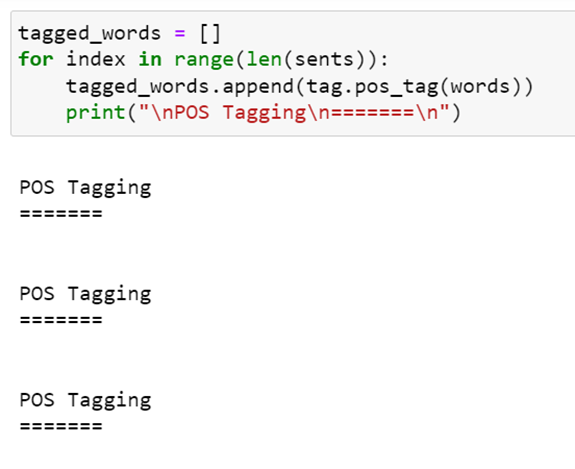
print("\nchunking\n=======\n")

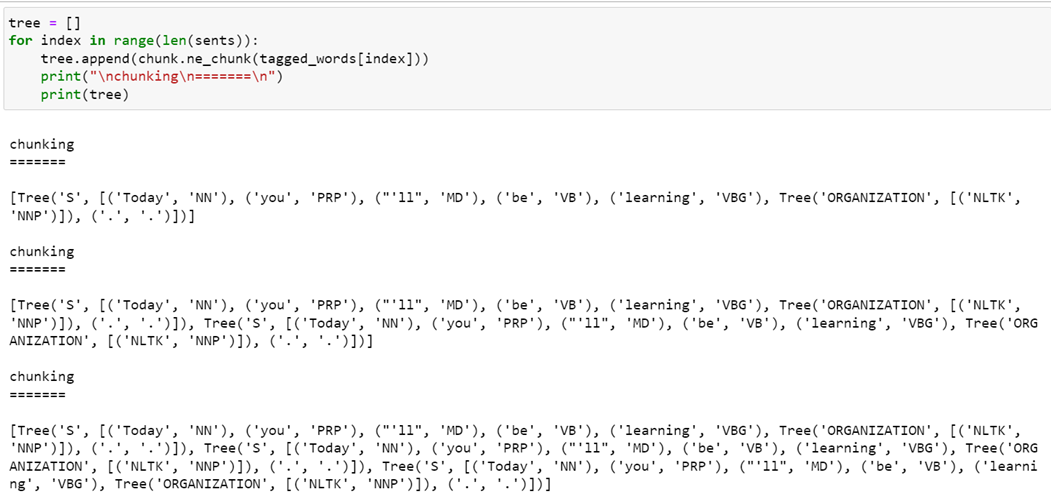
print(tree)

**Output:**









**Practical No: 4**

**Practical 4 a)** **Date: 05/04/2023**

**Aim: Named Entity recognition using user defined text.**

**Code:**

!pip install -U spacy

!python -m spacy download en\_core\_web\_sm

import spacy

nlp = spacy.load("en\_core\_web\_sm")

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 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 interviwe with Recode earlier"

"this week.''')

doc = nlp(text)

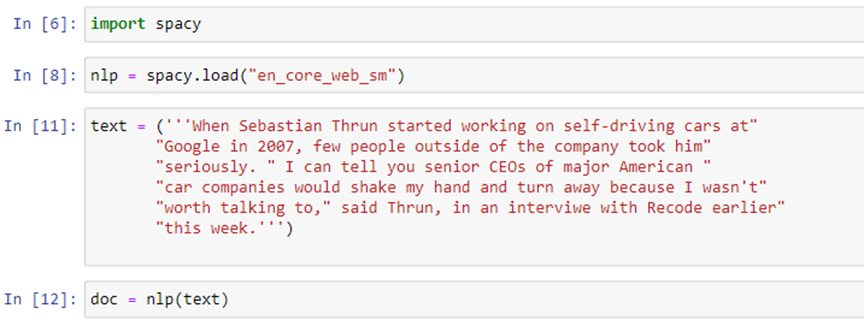
print("Noun phrases:", [chunk.text for chunk in doc.noun\_chunks])

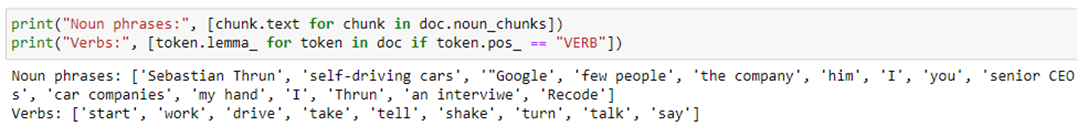
print("Verbs:", [token.lemma\_ for token in doc if token.pos\_ == "VERB"])

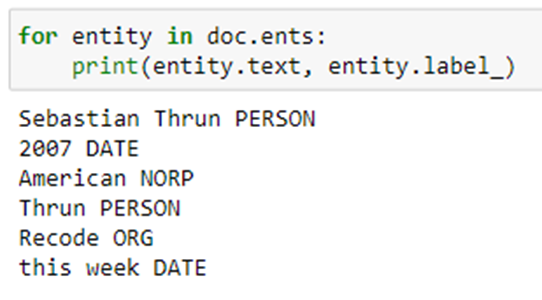
for entity in doc.ents:

print(entity.text, entity.label\_)

**Output:**







**Practical 4 b)** **Date: 05/04/2023**

**Aim: Named Entity recognition with diagram using NLTK corpus – treebank.**

**Code:**

import nltk

nltk.download('treebank')

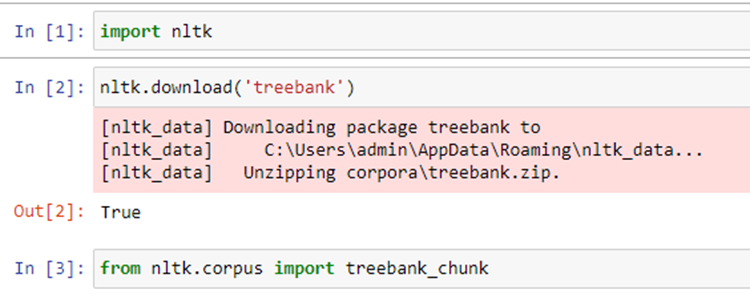
from nltk.corpus import treebank\_chunk

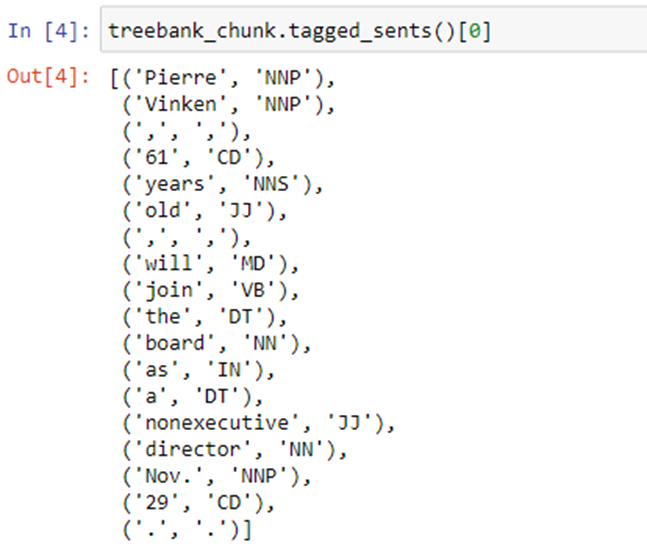
treebank\_chunk.tagged\_sents()[0]

treebank\_chunk.chunked\_sents()[0]

treebank\_chunk.chunked\_sents()[0].draw()

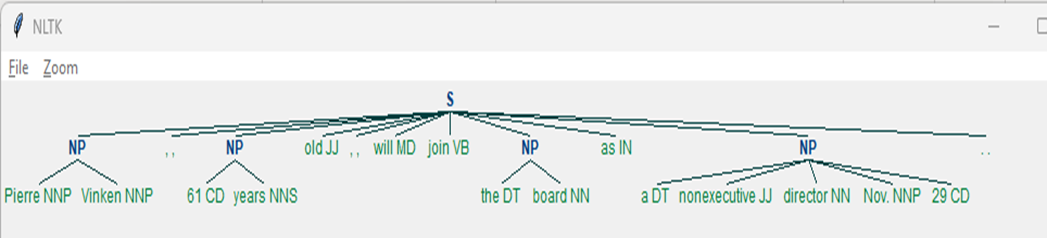
**Output:**











**Practical No: 5**

**Practical 5 a)** **Date: 05/04/2023**

**Aim: Chart parsing using the string "Book that flight".**

**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)

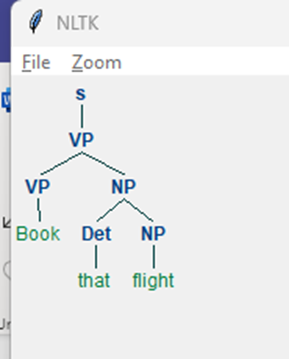
for tree in parser.parse(all\_tokens):

print(tree)

tree.draw()

**Output:**





**Practical 5 b)** **Date: 05/04/2023**

**Aim: Chart parsing using the string "I saw a bird in my balcony".**

**Code:**

import nltk

from nltk import tokenize

grammar1 = nltk.CFG.fromstring("""

s -> NP VP

NP -> 'I'

VP -> VP PP

VP -> V NP

V -> 'saw'

NP -> Det N

Det -> 'a'

N -> singular Noun

N -> 'bird'

PP -> P NP

P -> 'in'

NP -> Det N

Det -> 'my'

N -> 'balcony'

""")

sentence = "I saw a bird in my balcony"

for index in range(len(sentence)):

all\_tokens = tokenize.word\_tokenize(sentence)

print(all\_tokens)

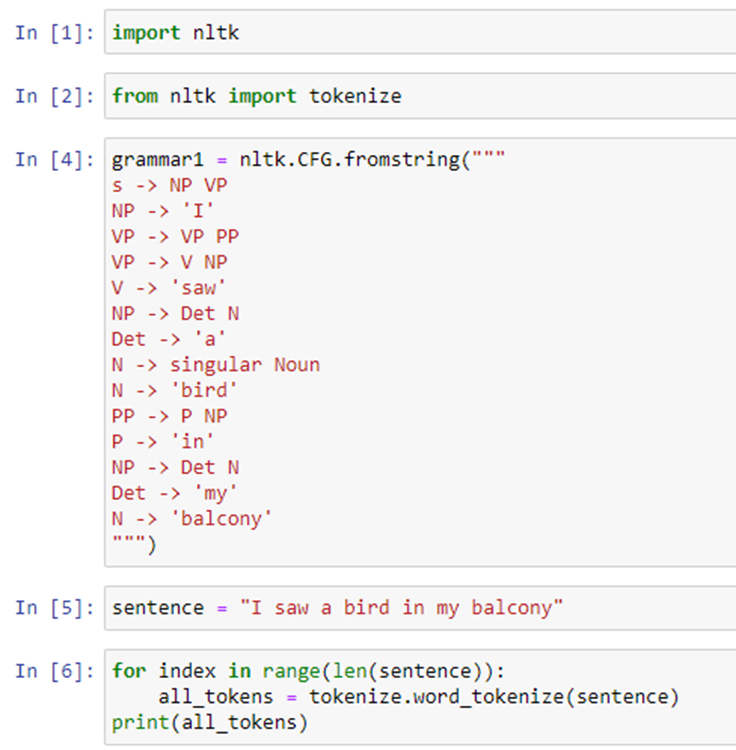
parser = nltk.ChartParser(grammar1)

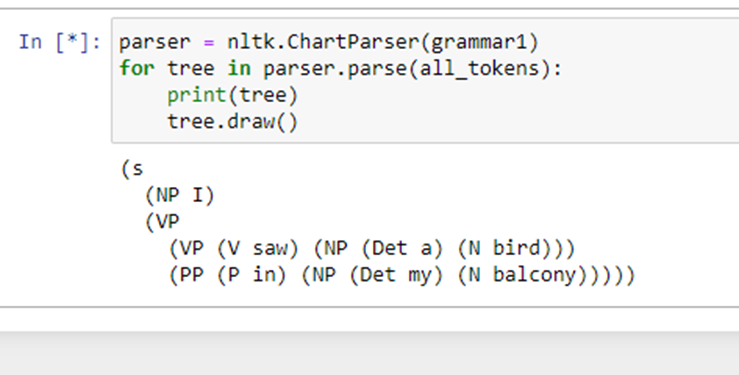
for tree in parser.parse(all\_tokens):

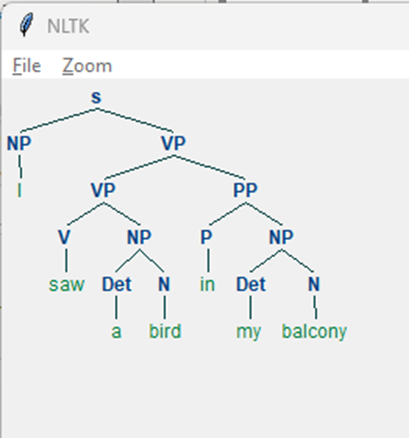
print(tree)

tree.draw()

**Output:**







**Practical No: 6**

**Practical 6 a)**  **Date: 12/04/2023**

**Aim:** Analyzing the meaning of sentence by querying a database. Find the cities of India by applying a query - 'What cities are located in India' and the context free grammar from the file 'sqlIndia.fcfg'.

Hint: sqlIndia.fcfg file should be in the same folder.

**Data file: sqlIndia.fcfg**

## Natural Language Toolkit: sqlIndia.fcfg

##

## Deliberately naive string-based grammar for

## deriving SQL queries from English

##

% start S

S[SEM=(?np + WHERE + ?vp)] -> NP[SEM=?np] VP[SEM=?vp]

VP[SEM=(?v + ?pp)] -> IV[SEM=?v] PP[SEM=?pp]

VP[SEM=(?v + ?ap)] -> IV[SEM=?v] AP[SEM=?ap]

NP[SEM=(?det + ?n)] -> Det[SEM=?det] N[SEM=?n]

PP[SEM=(?p + ?np)] -> P[SEM=?p] NP[SEM=?np]

AP[SEM=?pp] -> A[SEM=?a] PP[SEM=?pp]

NP[SEM='Country="india"'] -> 'India'

Det[SEM='SELECT'] -> 'Which' | 'What'

N[SEM='City FROM city\_table'] -> 'cities'

IV[SEM=''] -> 'are'

A[SEM=''] -> 'located'

P[SEM=''] -> 'in'

**Code:**

import nltk

from nltk import load\_parser

nltk.download('book\_grammars')

nltk.data.show\_cfg('grammars/book\_grammars/sql0.fcfg')

from nltk.parse import load\_parser

cp = load\_parser('grammars/book\_grammars/sql0.fcfg')

query = 'What cities are located in China'

trees = list(cp.parse(query.split()))

print (trees)

answer = trees[0].label()['SEM']

print (answer)

answer = [s for s in answer if s]

q = ' '.join(answer)

print(q)

from nltk.sem import chat80

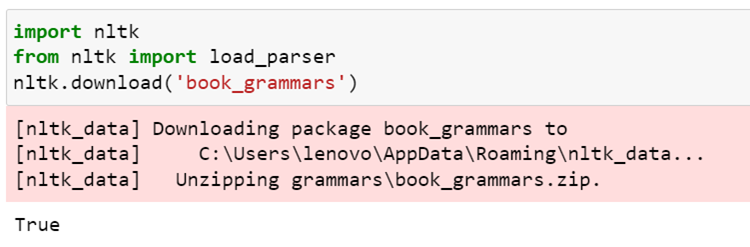
nltk.download('city\_database')

rows = chat80.sql\_query('corpora/city\_database/city.db', q)

for r in rows:

print(r)

**Output:**









**Practical 6 b)** **Date: 12/04/2023**

**Aim:**

b. Building a Discourse Representation Theory (DRT) by parsing a string representation - Angus owns a dog.

**Code:**

#pip install nltk

get\_ipython().system('pip install nltk')

import nltk

read\_the\_expr = nltk.sem.DrtExpression.fromstring

drs1 = read\_the\_expr('([x, y], [Angus(x), dog(y), own(x, y)])')

print(drs1)

drs1.draw()

print(drs1.fol())

from nltk import load\_parser

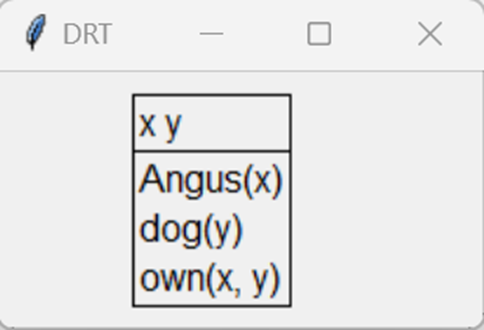
parser = load\_parser('grammars/book\_grammars/drt.fcfg', logic\_parser=nltk.sem.drt.DrtParser())

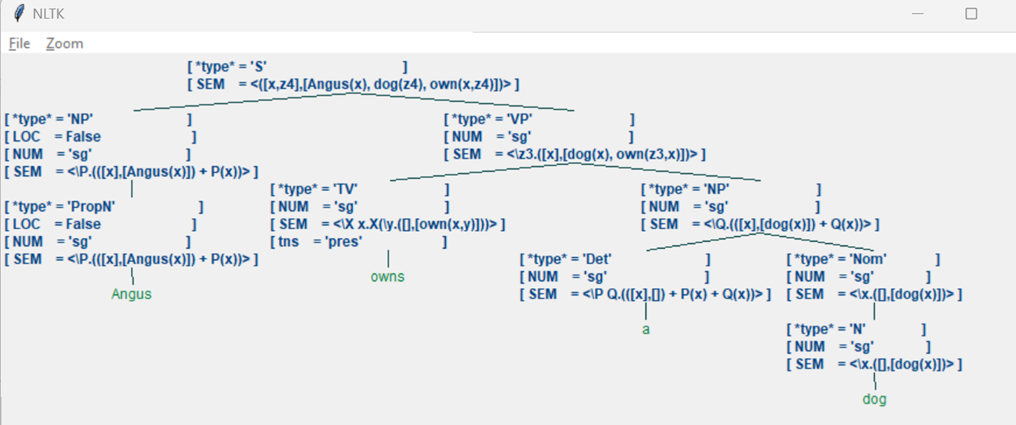
trees = list(parser.parse('Angus owns a dog'.split()))

print(trees[0].label()['SEM'].simplify())

trees[0].draw()

**Output:**





**Practical No: 7** **Date: 14/04/2023**

**Aim:** Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer and WordNetLemmatizer

**Code:**

# **PorterStemmer**

import nltk

from nltk.stem import PorterStemmer

word\_stemmer = PorterStemmer()

print(word\_stemmer.stem('writing'))

**#LancasterStemmer**

import nltk

from nltk.stem import LancasterStemmer

Lanc\_stemmer = LancasterStemmer()

print(Lanc\_stemmer.stem('writing'))

**#RegexpStemmer**

import nltk

from nltk.stem import RegexpStemmer

Reg\_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4)

print(Reg\_stemmer.stem('writing'))

**#SnowballStemmer**

import nltk

from nltk.stem import SnowballStemmer

english\_stemmer = SnowballStemmer('english')

print(english\_stemmer.stem ('writing'))

**#WordNetLemmatizer**

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("word :\tlemma")

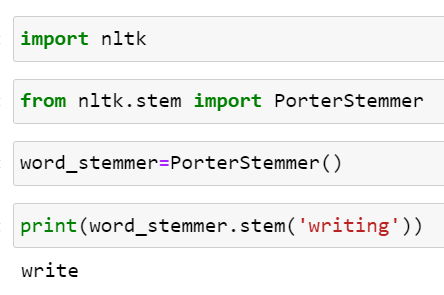
print("rocks :", lemmatizer.lemmatize("rocks"))

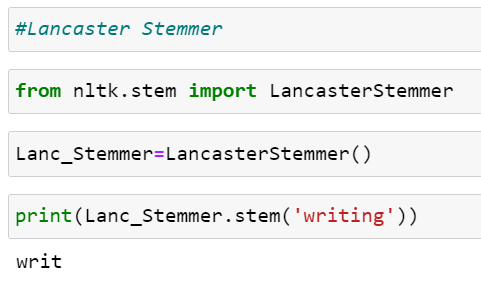
print("corpora :", lemmatizer.lemmatize("corpora"))

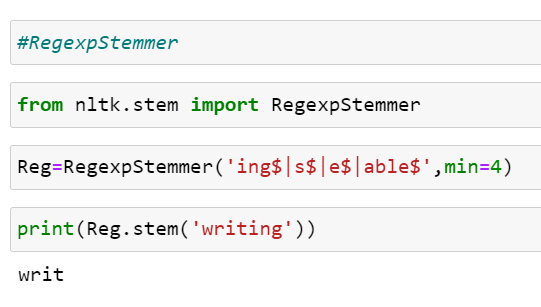
# a denotes adjective in "pos"

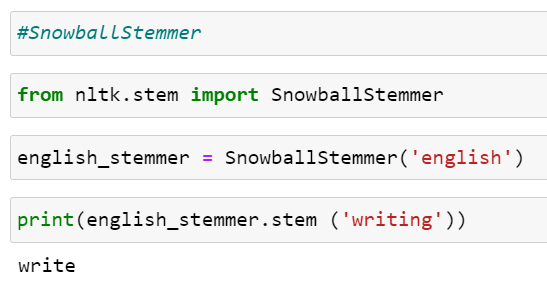
print("better :", lemmatizer.lemmatize("better", pos ="a"))

**Output:**











**Practical No: 8**

**Practical No: 8 a** **Date: 14/04/2023**

**Aim:** a) Parse a sentence - "old men and women" and draw a tree using 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)

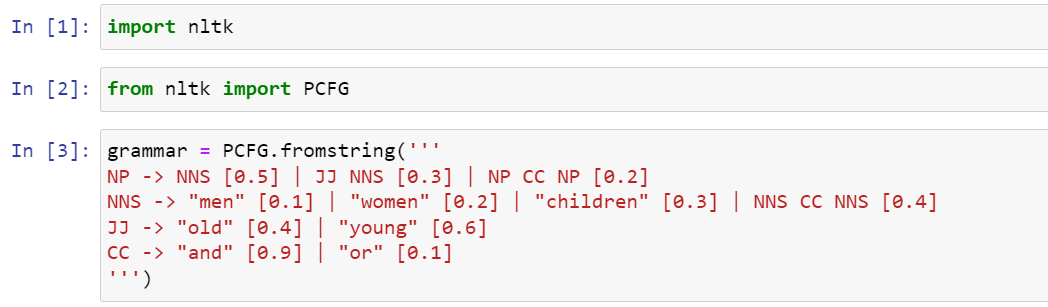
print("Output: ")

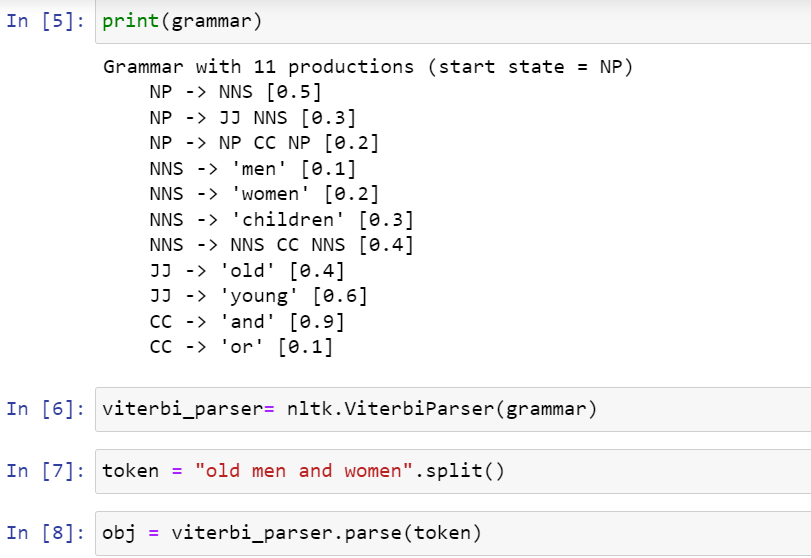
for x in obj:

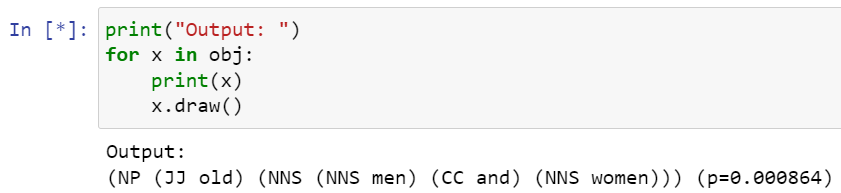
print(x)

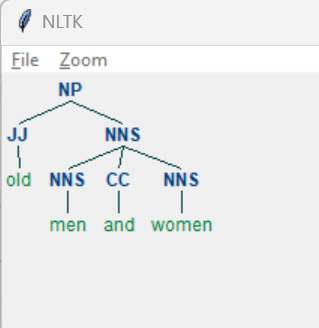
x.draw()

**Output:**









**Practical No: 8 b** **Date: 14/04/2023**

**Aim:** Parse a sentence - 'I saw a bird from my window.' and draw a tree using malt parsing.

**HINT:**

Set the environment variable -> System Variable -> New ->

Variable Name:(MALT-PARSER) -> Variable Value:(C:\Users\lenovo\AppData\Local\Programs\Python\Python310\maltparser-1.7.2)

Variable Name:(MALT-MODEL) -> Variable Value:(C:\Users\lenovo\AppData\Local\Programs\Python\Python310\ engmalt.linear-1.7.mco)

**Code:**

from nltk.parse import malt

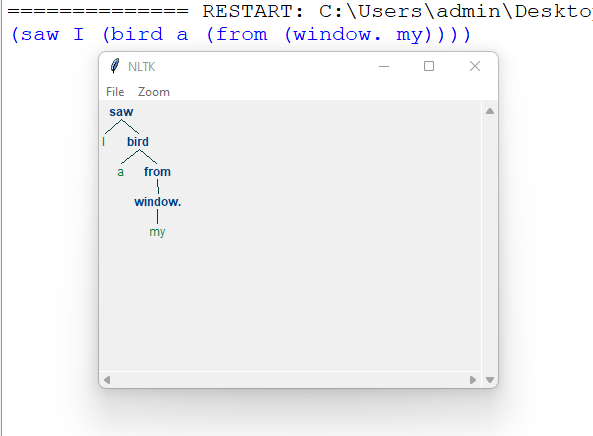
mp = malt.MaltParser('maltparser-1.7.2','engmalt.linear-1.7.mco')

t = mp.parse\_one('I saw a bird from my window.'.split()).tree()

print(t)

t.draw()

**Output:**



**Practical No: 9**

**Practical No: 9 a** **Date: 14/04/2023**

**Aim:** Multiword Expressions in NLP for multiword – ‘New Delhi’ in '''Good cake cost Rs.1500\kg in New Delhi. Please buy me one of them.\n\nThanks.'''

**Code:**

from nltk.tokenize import MWETokenizer

from nltk import sent\_tokenize, word\_tokenize

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

mwe = MWETokenizer([('New','Delhi'), ('New','Bombay')], separator='\_')

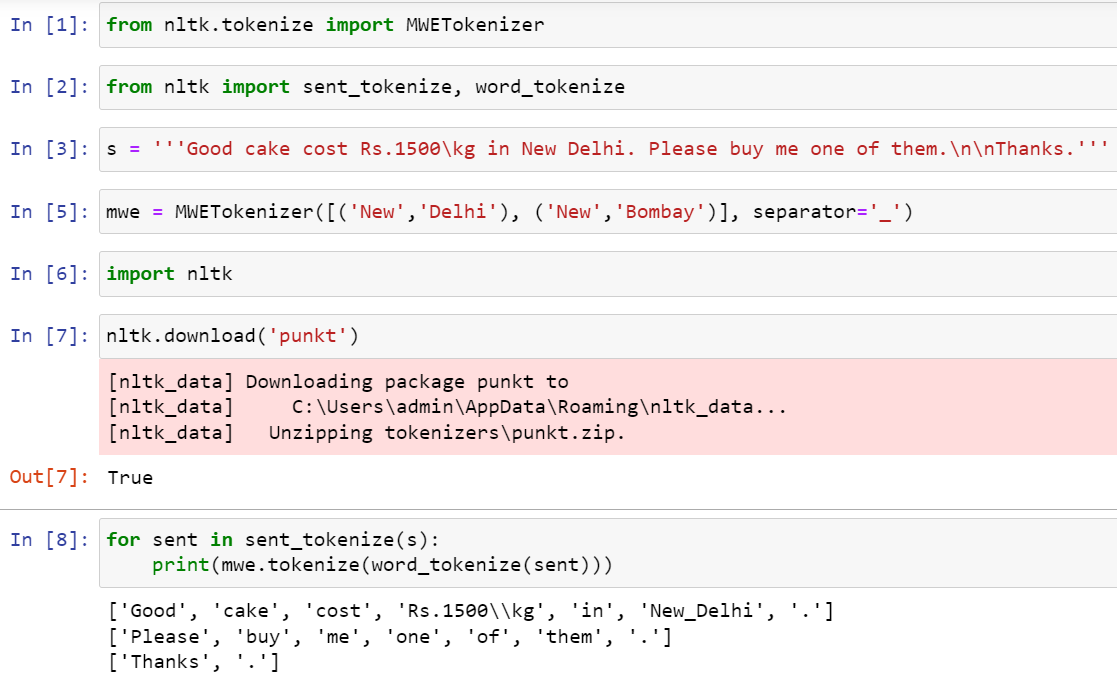
import nltk

nltk.download('punkt')

for sent in sent\_tokenize(s):

print(mwe.tokenize(word\_tokenize(sent)))

**Output:**



**Practical No: 9 b** **Date: 14/04/2023**

**Aim:** Word Sense Disambiguation for the keyword ‘jam’ in the sentences - 'This device is used to jam the signal' and 'I am stuck in a traffic jam'. Also, for the keyword ‘book’ in the sentences - 'I love reading books on coding.' and 'The table was already booked by someone else.'

**Code:**

from nltk.wsd import lesk

from nltk.tokenize import word\_tokenize

import nltk

nltk.download('omw-1.4')

nltk.download('punkt')

nltk.download('wordnet')

a1= lesk(word\_tokenize('This device is used to jam the signal'),'jam')

print(a1,a1.definition())

a2 = lesk(word\_tokenize('I am stuck in a traffic jam'),'jam')

print(a2,a2.definition())

b1= lesk(word\_tokenize('I love reading books on coding.'),'book')

print(b1,b1.definition())

b2 = lesk(word\_tokenize('The table was already booked by someone else.'),'book')

print(b2,b2.definition())

**Output:**

