Deccan Education Society's
Kirti M. Doongursee College of Arts, Science and Commerce [AUTONOMOUS]

M.Sc. [Information Technology]

Practical Journal

Course Name: Natural Language Processing

Seat Number [ ]

(Academic Year 2022-2023)

Department of Computer Science and Information Technology

Department of Computer Science and Information Technology
Deccan Education Society's
Kirti M. Doongursee College of Arts, Science and Commerce
[AUTONOMOUS]

C E R T I F I C A T E

This is to certify that Miss. Mrunal Gajanan Aher of M.Sc. (I.T.) with Seat No. has complete ___8___Practical of Paper-(Course Name- **Natural Language Processing**) under mysupervision in this College during the Fourth Semester of academic year 2022-2023.

Prof. Jaymala Deshpande Dr. Apurva Yadav H.O.D.
Lecturer-In-Charge Department of
Computer Science & IT

Date: / /2023 Date:

Examined by: Remarks: Date:

INDEX

Sr. No.	Practical	Sign
1	A. Convert the given text to speech.	
	B. Convert audio file Speech to Text.	
2	A. Create and use your own corpora (plaintext, categorical)	
	B. Study of tagged corpora with methods like tagged_sents, tagged_words.	
	C. Map Words to Properties Using Python Dictionaries.	
3	A. Study DefaultTagger,	
	B. Study UnigramTagger	
4	A. Study of Wordnet Dictionary with methods as synsets,	

	definitions, examples, antonyms	
	B. Write a program using python to find synonym and antonym of word "active" using Wordnet.	
	C. Compare two nouns	
5	A. Tokenization using Python’s split() function	
	B. Tokenization using Regular Expressions (RegEx)	
	C. Tokenization using Keras	
6	A. Named Entity recognition using user defined text.	
	B. Named Entity recognition with diagram using NLTK corpus – treebank.	
7	A. Define grammar using nltk. Analyze a sentence using the same.	
	B. Implementation of Deductive Chart Parsing using context free grammar and a given sentence.	
8	Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer Study WordNetLemmatizer	

Practical 1(A)

Aim: Convert the given text to speech.

Program:

```
from playsound import playsound

from gtts import gTTS

mytext="happy birthday to you"

language="en"

myobj=gTTS(text=mytext,lang=language,slow=False)

myobj.save("myfile.mp3")

playsound("myfile.mp3")
```

Output:

welcomeNLP.mp3 audio file is getting created and it plays the file with playsound() method, while running the program

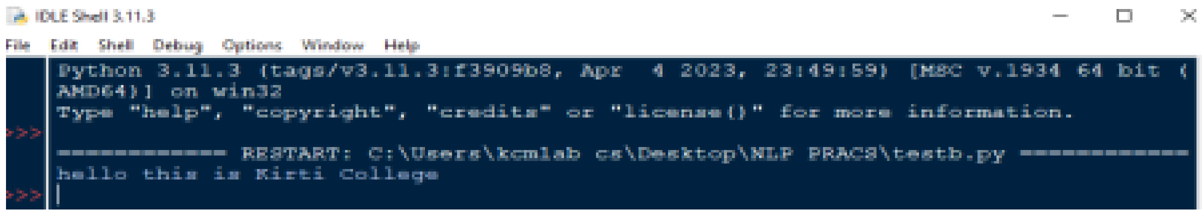
Practical 1(B)

Aim: Convert audio file Speech to Text.

Program:

```
import speech_recognition as sr

filename="C:/Users/kcmlab cs/Desktop/NLP PRACS/kirti.wav"

r=sr.Recognizer()

with sr.AudioFile(filename)as source:

 audio_data=r.record(source)

 text=r.recognize_google(audio_data)

 print(text)
```

Output:



```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:\Users\kcmlab cs\Desktop\NLP PRACS\testb.py ============
hello this is Kirti College
>>> 
```

Practical 2(A)

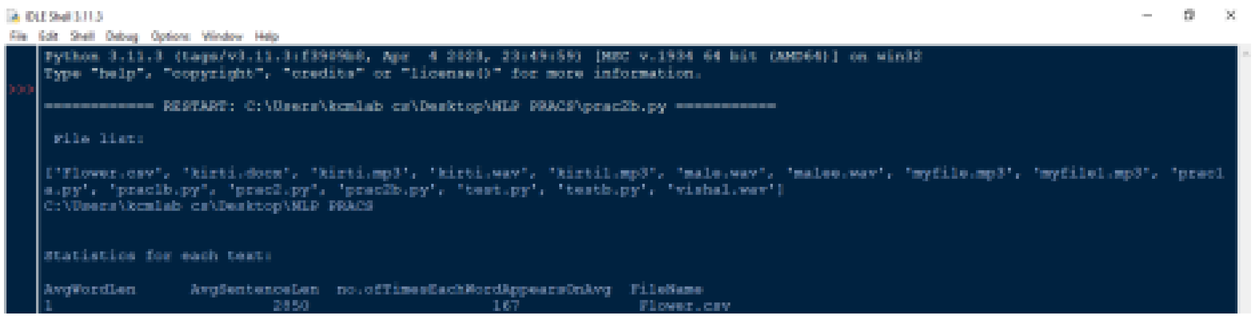
Aim: Create and use your own corpora (plaintext, categorical)

Program:

```
import nltk

from nltk.corpus import PlaintextCorpusReader

corpus_root = 'C:/Users/kcmlab cs/Desktop/NLP PRACS'

filelist = PlaintextCorpusReader(corpus_root, '.*')

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

print (filelist.fileids())

print (filelist.root)


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

```
corresponding to the filelist file identifiers listed earlier and then computing statistics

for each text.'''

print ('\n\nStatistics 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', fileid
```

Output:



```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

============ RESTART: C:\Users\kcmlab cs\Desktop\NLP PRACS\prac2b.py ============

 File list:

['Flower.csv', 'kirti.docx', 'kirti.mp3', 'kirti.wav', 'kirti1.mp3', 'male.wav', 'malee.wav', 'myfile.mp3', 'myfile1.mp3', 'prac1
a.py', 'prac1b.py', 'prac2.py', 'prac2b.py', 'test.py', 'testb.py', 'vishal.wav']
C:\Users\kcmlab cs\Desktop\NLP PRACS


Statistics for each text:

AvgWordLen      AvgSentenceLen  no.ofTimesEachWordAppearsOnAvg  FileName
1                      2850                    167                     Flower.csv
```

Practical 2(B)

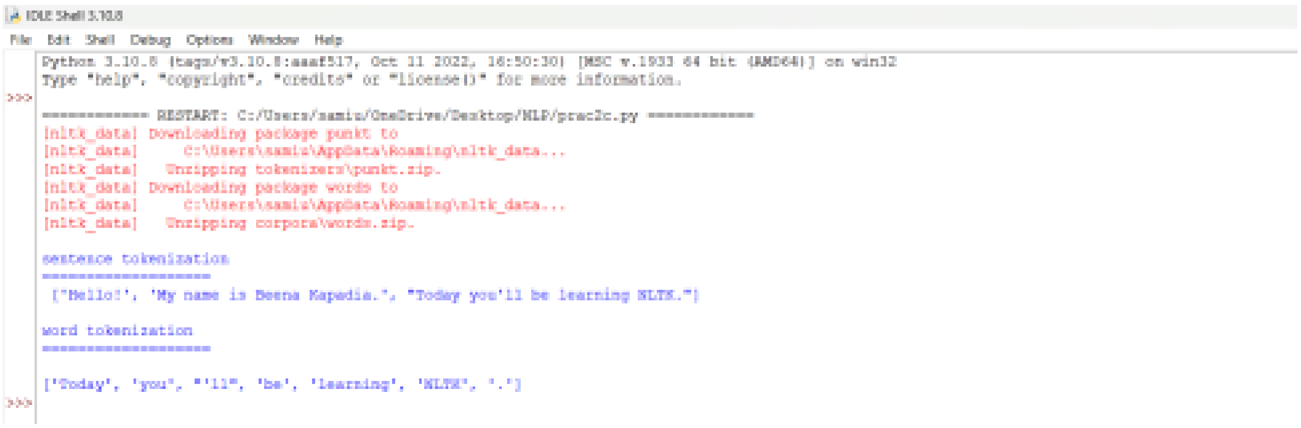
Aim: Study of tagged corpora with methods like tagged_sents,

tagged_words. **Program:**

```
import nltk

from nltk import tokenize

nltk.download('punkt')

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)
```

Output:



```
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/samiu/OneDrive/Desktop/NLP/prac2c.py ============
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\samiu\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\samiu\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.

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

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

['Today', 'you', "'ll", 'be', 'learning', 'NLTK', '.']
>>>
```

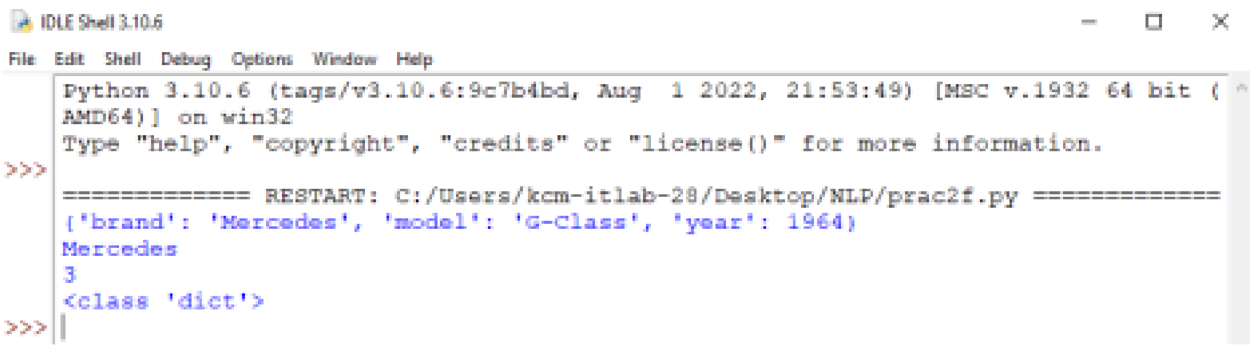
Practical 2(C)

Aim: Map Words to Properties Using Python Dictionaries

Program:

```
thisdict= {
 "brand":"Mercedes",
 "model": "G-Class",
 "year":1964
 }
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```

Output:



```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:/Users/kcm-itlab-28/Desktop/NLP/prac2f.py =============
{'brand': 'Mercedes', 'model': 'G-Class', 'year': 1964}
Mercedes
3
<class 'dict'>
>>>
```

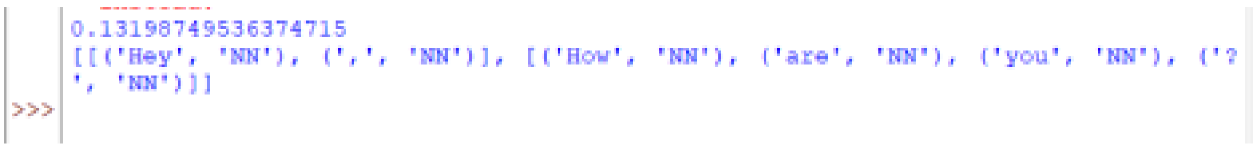
Practical 3(A)

Aim: Study Default Tagger

Program:

```
import nltk

from nltk.tag import DefaultTagger

exptagger=DefaultTagger('NN')

from nltk.corpus import treebank

testsentences=treebank.tagged_sents()[1000:]

print(exptagger.evaluate(testsentences))


import nltk

from nltk.tag import DefaultTagger

exptagger=DefaultTagger

exptagger=DefaultTagger('NN')

print(exptagger.tag_sents([['Hey',','],['How','are','you','?']]))
```

Output:



```
0.13198749536374715
[[('Hey', 'NN'), (',', 'NN')], [('How', 'NN'), ('are', 'NN'), ('you', 'NN'), ('?
', 'NN')]]
>>>
```

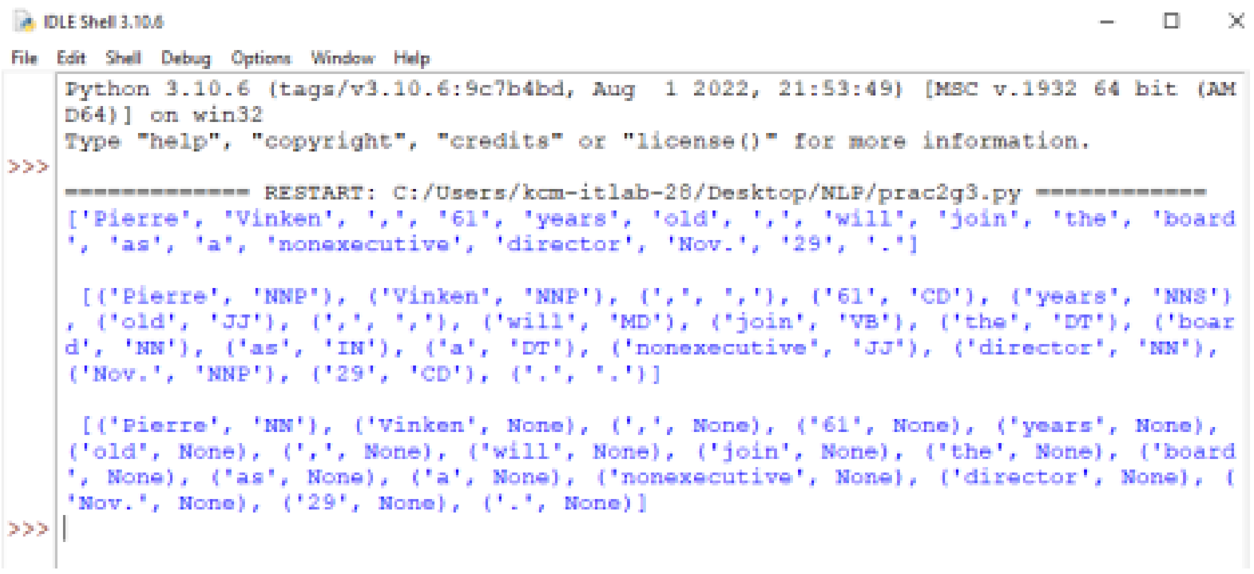
Practical 3(B)

Aim: Study Unigram Tagger

Program:

```
# 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:



```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:/Users/kcm-itlab-28/Desktop/NLP/prac2g3.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'), ('boar
d', 'NN'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive', 'JJ'), ('director', 'NN'),
('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')]

 [('Pierre', 'NN'), ('Vinken', None), (',', None), ('61', None), ('years', None),
('old', None), (',', None), ('will', None), ('join', None), ('the', None), ('board
', None), ('as', None), ('a', None), ('nonexecutive', None), ('director', None), (
'Nov.', None), ('29', None), ('.', None)]
>>> |
```

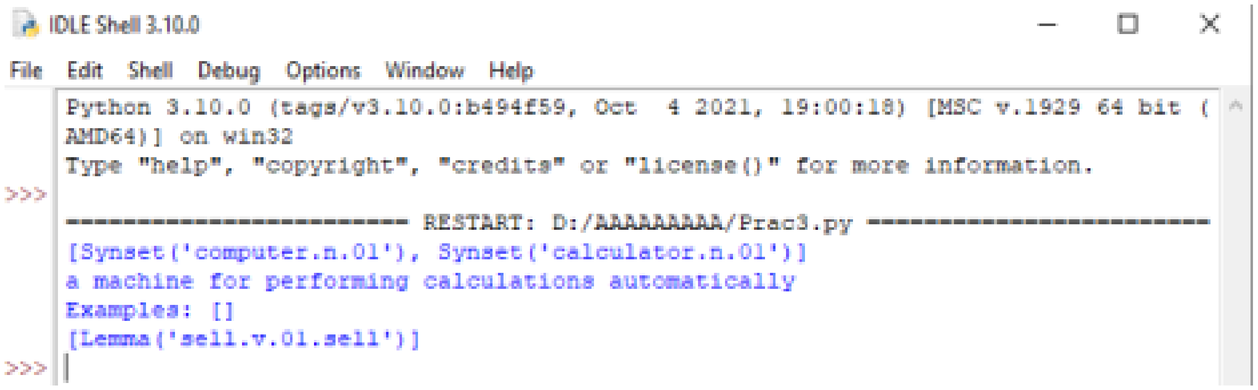
Practical 4(A)

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

Program:

```
'''WordNet provides synsets which is the collection of synonym words also called

“lemmas”'''

import nltk

from nltk.corpus import wordnet

print(wordnet.synsets("computer"))

# definition and example of the word ‘computer’

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

#examples

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

#get Antonyms

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

Output:



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================== RESTART: D:/AAAAAAAAA/Prac3.py ==========================
[Synset('computer.n.01'), Synset('calculator.n.01')]
a machine for performing calculations automatically
Examples: []
[Lemma('sell.v.01.sell')]
>>>
```

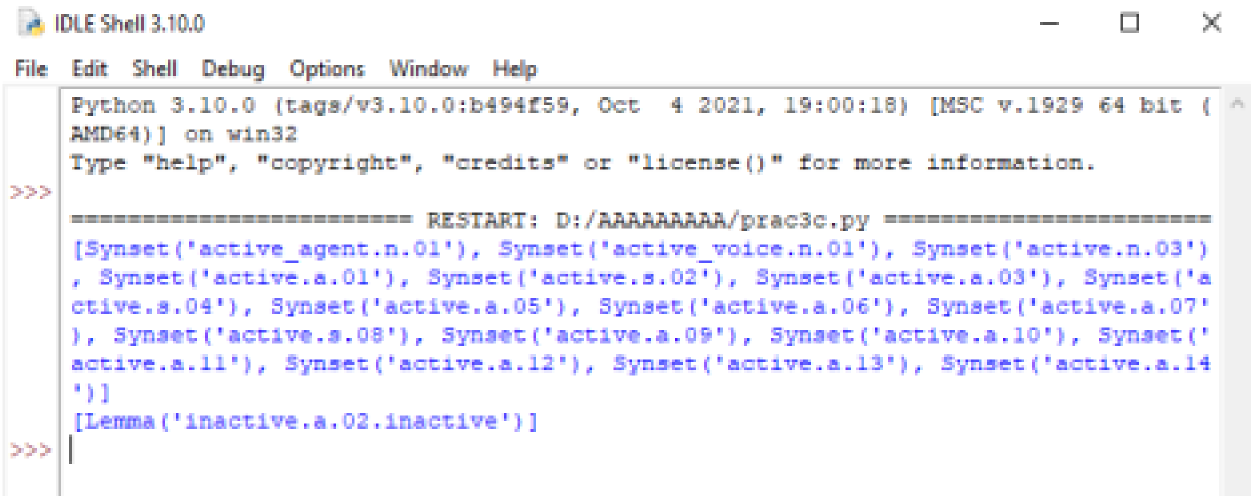
Practical 4(B)

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

Program:

```
from nltk.corpus import wordnet

print( wordnet.synsets("active"))

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

Output:



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: D:/AAAAAAAAA/prac3c.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')]
>>>
```

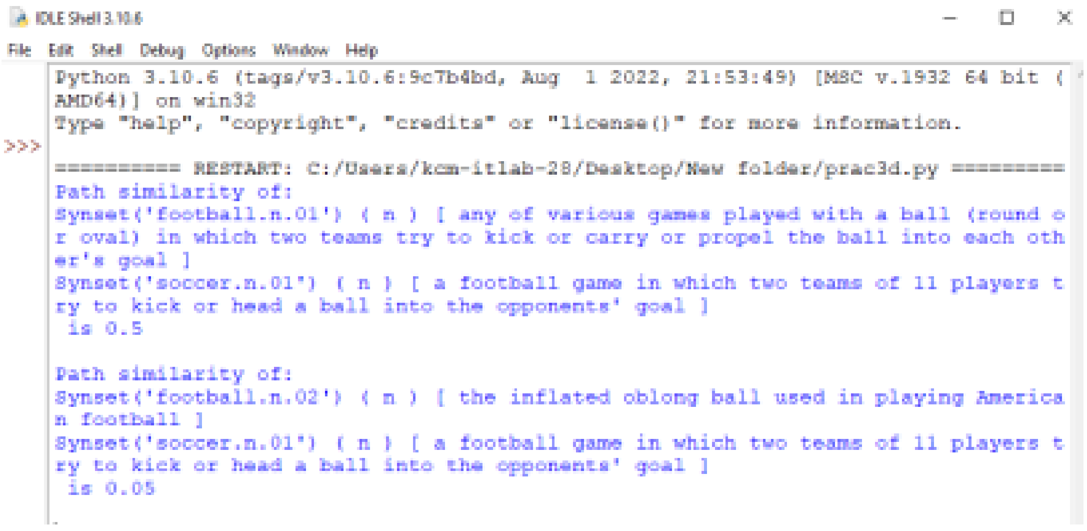
Practical 4(C)

Aim: Compare two nouns

Program:

```
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:



```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/kcm-itlab-28/Desktop/New folder/prac3d.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
```

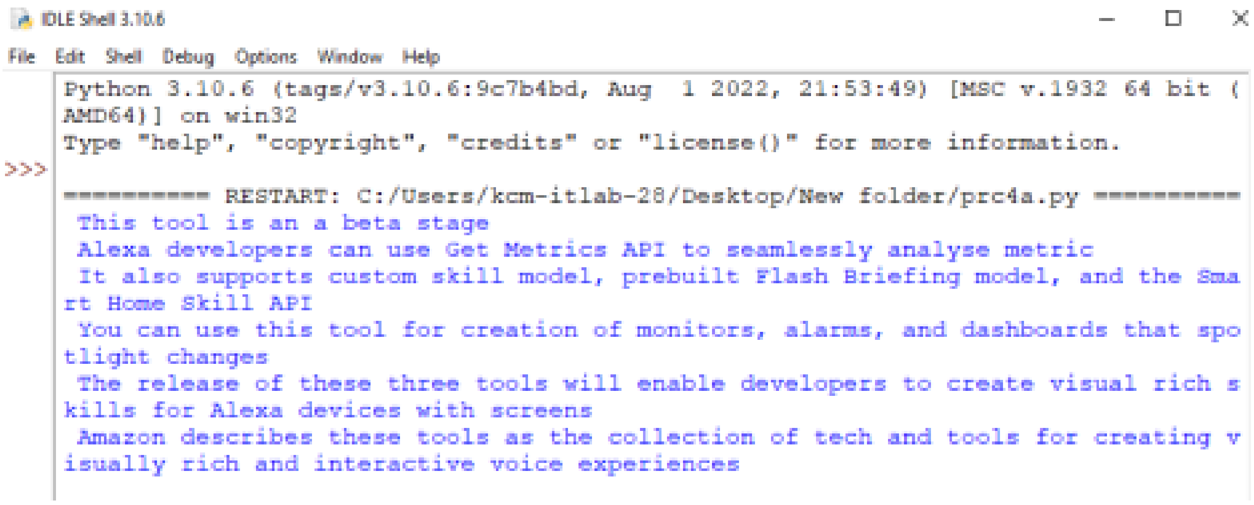
Practical 5(A)

Aim: Tokenization using Python's split() function

Program:

```
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:



```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/kcm-itlab-28/Desktop/New folder/prc4a.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 Sma
rt Home Skill API
 You can use this tool for creation of monitors, alarms, and dashboards that spo
tlight changes
 The release of these three tools will enable developers to create visual rich s
kills for Alexa devices with screens
 Amazon describes these tools as the collection of tech and tools for creating v
isually rich and interactive voice experiences
```

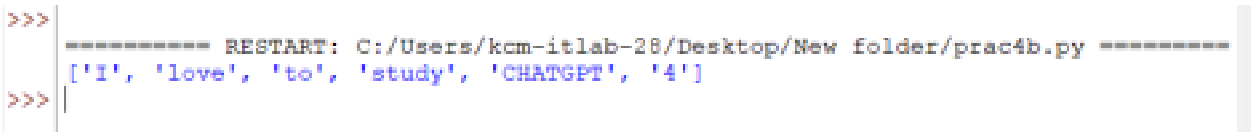
Practical 5(B)

Aim: Tokenization using Regular Expressions (RegEx)

Program:

```
import nltk

# import RegexpTokenizer() method from nltk

from nltk.tokenize import RegexpTokenizer

# Create a reference variable for Class RegexpTokenizer

tk = RegexpTokenizer('\s+', gaps = True)

# Create a string input

str = "I love to study CHATGPT 4"

# Use tokenize method

tokens = tk.tokenize(str)

print(tokens)
```

Output:



```
>>>
========== RESTART: C:/Users/kcm-itlab-28/Desktop/New folder/prac4b.py =========
['I', 'love', 'to', 'study', 'CHATGPT', '4']
>>>
```

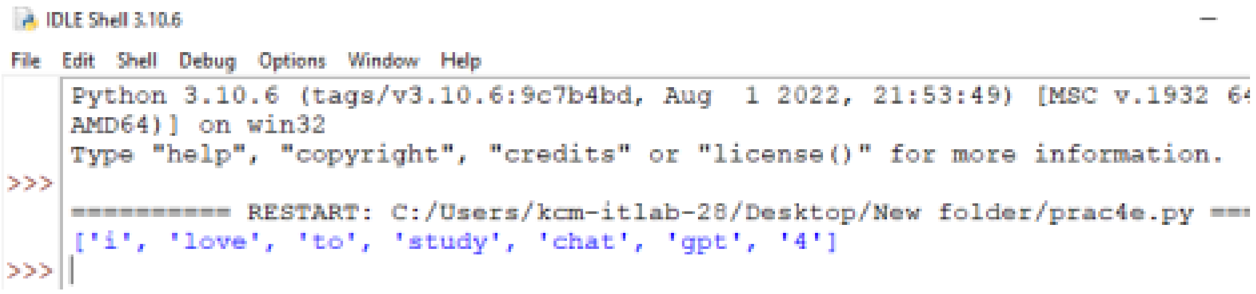
Practical 5(C)

Aim: Tokenization using Keras

Program:

```
import keras

from keras.preprocessing.text import text_to_word_sequence

# Create a string input

str = "I love to study Chat GPT 4"

# tokenizing the text

tokens = text_to_word_sequence(str)

print(tokens)
```

Output:



```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 6
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/kcm-itlab-28/Desktop/New folder/prac4e.py ===
['i', 'love', 'to', 'study', 'chat', 'gpt', '4']
>>>
```

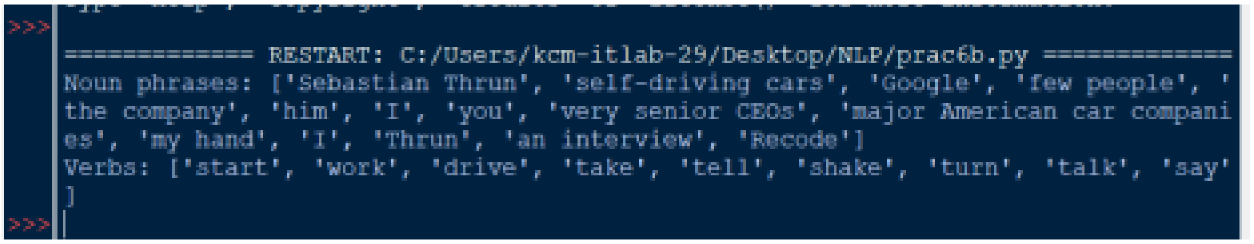
Practical 6(A)

Aim: Named Entity recognition using user defined text.

Program:

```
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/kcm-itlab-29/Desktop/NLP/prac6b.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'
]
>>>
```

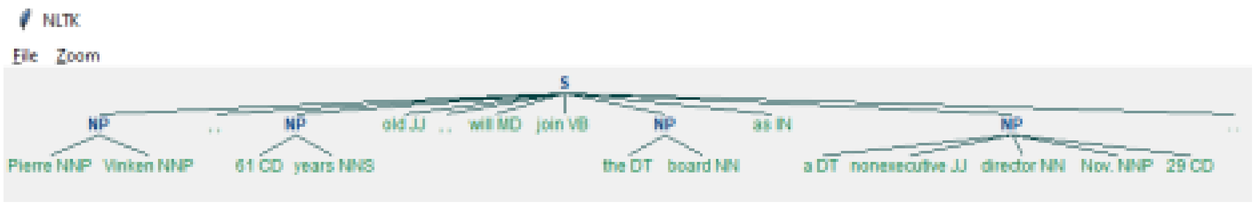
Practical 6(B)

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

Program:

```
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 7(A)

Aim: Define grammar using nltk. Analyze a sentence using the

same **Program:**

```
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:

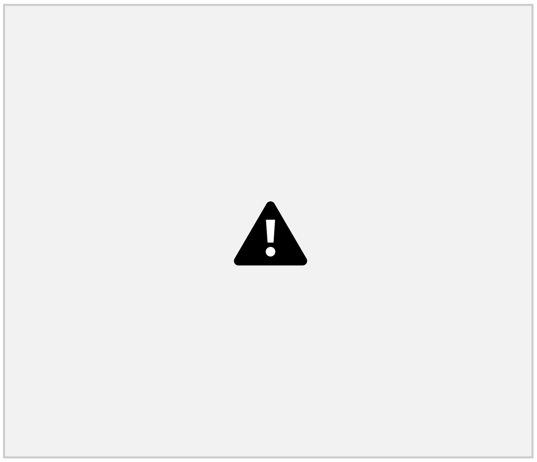
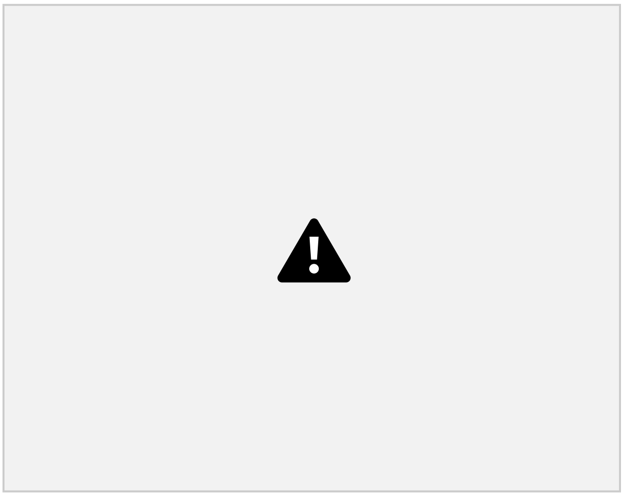
Practical 7(B)

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

Program:

```
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:

Practical 8

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

Program:

```
print('PorterStemmer')

import nltk

from nltk.stem import PorterStemmer

word_stemmer = PorterStemmer()

print(word_stemmer.stem('writing'))


print('LancasterStemmer')
```

```
import nltk

from nltk.stem import LancasterStemmer

Lanc_stemmer = LancasterStemmer()

print(Lanc_stemmer.stem('writing'))


print('RegexpStemmer')

import nltk

from nltk.stem import RegexpStemmer

Reg_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4)

print(Reg_stemmer.stem('writing'))


print('SnowballStemmer')

import nltk

from nltk.stem import SnowballStemmer
english_stemmer = SnowballStemmer('english')

print(english_stemmer.stem ('writing'))


print('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: