**Practical No : 1**

**A ] Convert the given text to speech.**

**Code :**

# text to speech

# pip install gtts

# pip install playsound

from playsound import playsound

# import required for text to speech conversion

from gtts import gTTS

mytext = "Welcome to Natural Language programming"

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.

**B ] Convert audio file Speech to Text.**

**Code :**

import speech\_recognition as sr

filename = "male.wav"

# initialize the recognizer

r = sr.Recognizer()

# open the file

with sr.AudioFile(filename) as source:

# listen for the data (load audio to memory)

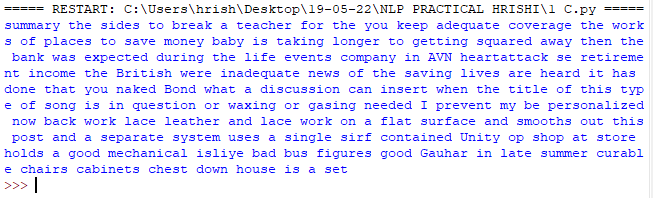
audio\_data = r.record(source)

# recognize (convert from speech to text)

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

print(text)

**Output :**

****

**Practical No : 2**

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

**Code :**

import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

from nltk.tokenize import word\_tokenize

text = "john 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

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

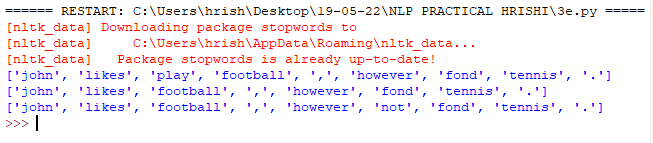
all\_stopwords.remove('not')

text\_tokens = word\_tokenize(text)

tokens\_without\_sw = [word for word in text\_tokens if not word in all\_stopwords]

print(tokens\_without\_sw)

**Output :**

****

**Practical No : 3**

**A ] Tokenization using Regular Expressions (RegEx)**

**Code :**

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

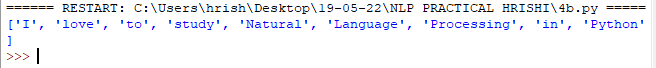
str = "I love to study Natural Language Processing in Python"

# Use tokenize method

tokens = tk.tokenize(str)

print(tokens)

**Output :**

****

**B ] Tokenization using NLTK**

**Code :**

import nltk

from nltk.tokenize import word\_tokenize

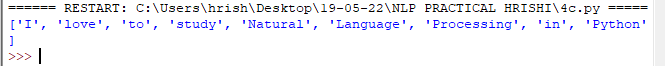
# Create a string input

str = "I love to study Natural Language Processing in Python"

# Use tokenize method

print(word\_tokenize(str))

**Output :**

****

**C ] Tokenization using Keras**

**Code :**

#pip install keras

#pip install tensorflow

import keras

from keras.preprocessing.text import text\_to\_word\_sequence

# Create a string input

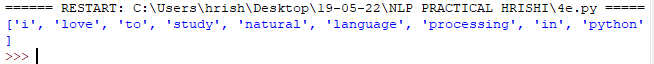
str = "I love to study Natural Language Processing in Python"

# tokenizing the text

tokens = text\_to\_word\_sequence(str)

print(tokens)

**Output :**

****

**Practical No : 4**

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

**Code :**

**#PorterStemmer**

import nltk

from nltk.stem import PorterStemmer

word\_stemmer = PorterStemmer()

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

**Output :**

****

**Code :**

**#LancasterStemmer**

import nltk

from nltk.stem import LancasterStemmer

Lanc\_stemmer = LancasterStemmer()

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

**Output :**

****

**Code :**

**# RegexpStemmer**

import nltk

from nltk.stem import RegexpStemmer

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

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

**Output :**

****

**Code :**

**# SnowballStemmer**

import nltk

from nltk.stem import SnowballStemmer

english\_stemmer = SnowballStemmer('english')

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

**Output :**

****

**Code :**

**#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 :**

