## INDEX

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No.** | **Practical Name** | **Page No.** |
| 1 | 1. Install NLTK 2. Convert the given text to speech. 3. Convert audio file Speech to Text. | 1 |
| 2 | 1. Study of various Corpus–Brown, Inaugural, Reuters, udhr with various method like fields, raw, words, sents, categories. 2. Study Condition a frequency distributions 3. Studio f tagged corpora with methods like tagged sents, tagged\_words. 4. Map Words to Properties Using Python Dictionaries. | 6 |
| 3 | 1. Study of Word net Dictionary with methods assents,definitions, examples, antonyms 2. Study lemmas, hyponyms, hypernyms. 3. Write a program using python to find synonym andantonym of word "active" using Word net. | 11 |
| 4 | 1. Tokenization using Python’s split() function 2. Tokenization using Regular Expressions (Reg Ex) 3. Tokenization using NLTK 4. Tokenization using the spa cylibrary | 13 |
| 5 | Import NLPL ibraries for Indian Language sand perform word tokenizationin Hindi | 15 |
| 6 | Illustrate part of speech tagging.   1. Part of speech Tagging and chunking of user deined text. 2. Named Entity recognition of user defined text. 3. Named Entity recognition with diagram using NLTK corpus– Treebank | 16 |
| 7 | Finitestate automata   1. Define grammar using nltk. Analyze a sentence using the same. 2. Accept the input string with Regular expression of Finite Automaton: 101+. | 18 |

|  |  |  |
| --- | --- | --- |
|  | C.Accept the input string with Regular expression of FA: (a+b)\*bba. |  |
| 8 | Study Porter Stemmer, Lancaster Stemmer, Regexp Stemmer,Snowball Stemmer  Study Word Net Lemmatizer | 21 |
| 9 | Implement Naive Bayes classifier | 22 |
| 10 | 1. Speech tagging using nktl 2. B.Usage ofGiveandGaveinthePennTreebanksample 3. Probabilistic parser 4. Malt parsing: Parse a sentence and draw a tree using malt parsing. | 24 |
| 11 | 1. Multi word Expressionsin NLP 2. Normalized Web Distance and Word Similarity 3. Word Sense Disambiguation | 29 |

**1A]Install NLTK**

**PRACTICAL: 1**

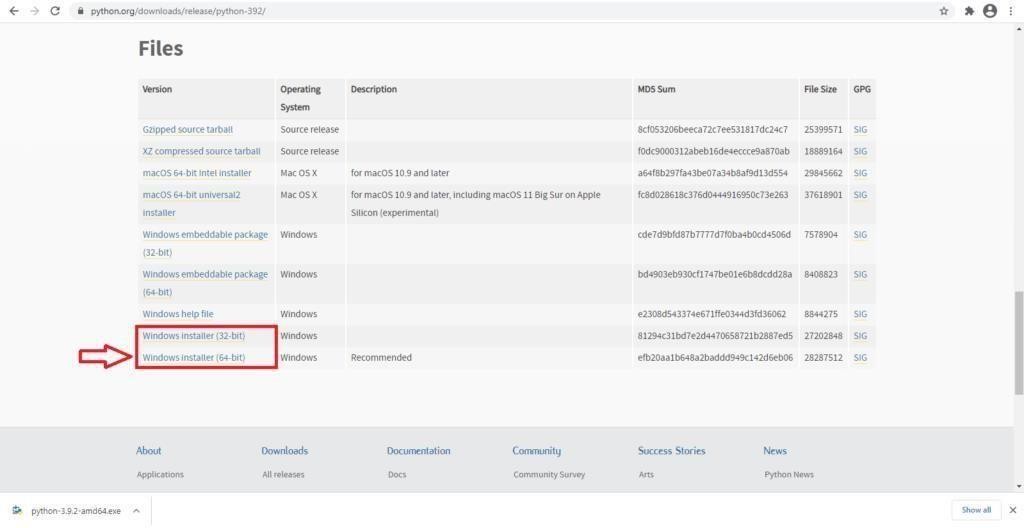
###### Python3.9.2InstallationonWindows InstallNLTK

**Python3.9.2InstallationonWindows**

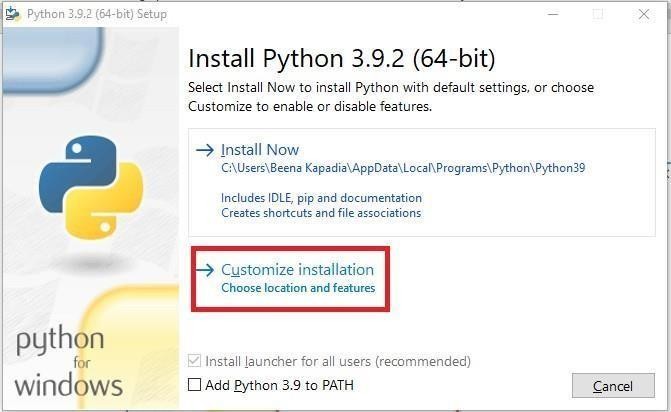
Step1)**Gotolink**https:/[/www.p](http://www.python.org/downloads/)y[thon.org/downloads/,](http://www.python.org/downloads/)**andselectthe latestversionforwindows.**



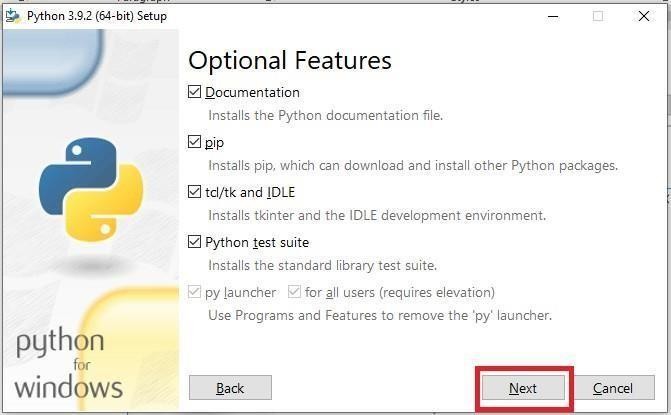
**Note**:Ifyoudon'twanttodownloadthelatestversion,youcanvisit thedownloadtab and seeallreleases.



**Step2)**ClickontheWindowsinstaller(64bit) **Step3)**SelectCustomizeInstallation

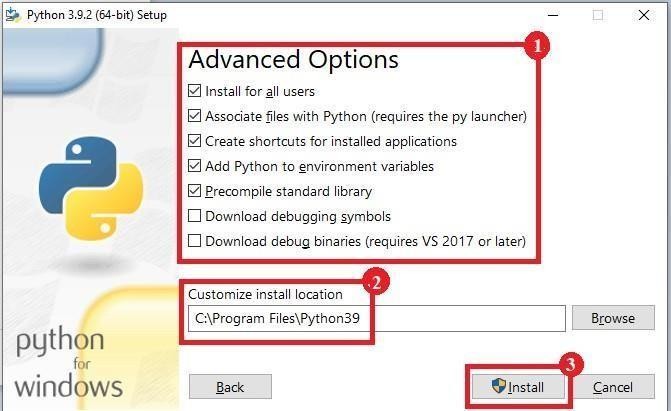


**Step4)**ClickNEXT



**Step5)** Innextscreen

1. Selecttheadvancedoptions
2. GiveaCustominstalllocation.Keepthedefaultfolderas c:\Programfiles\Python39
3. ClickInstall

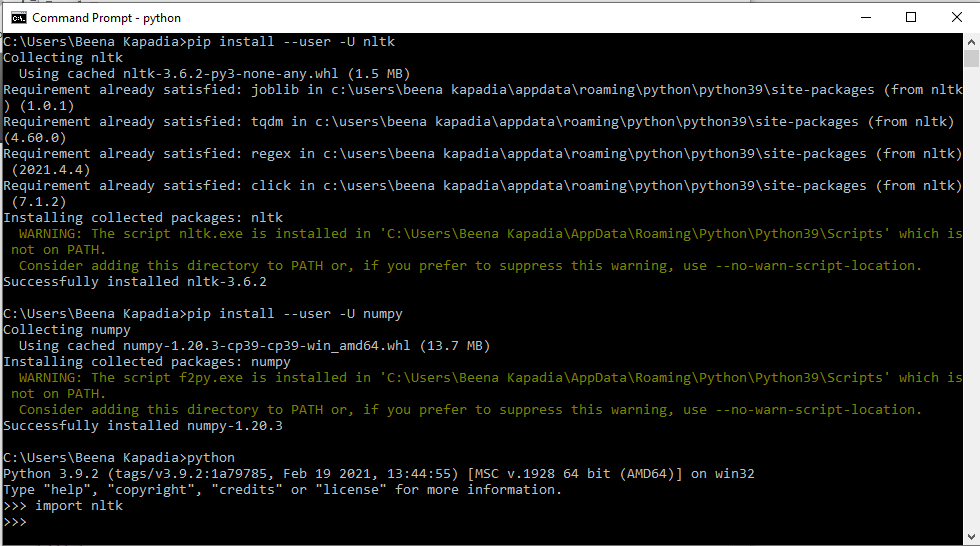


**Step6)**ClickClosebuttononceinstallisdone.

**Step7)open**commandpromptwindowandrunthefollowing commands:C:\Users\BeenaKapadia>pip install--upgradepip C:\Users\Beena Kapadia> pip install --user -U nltkC:\Users\BeenaKapadia>>pipinstall --user-U numpyC:\Users\BeenaKapadia>python

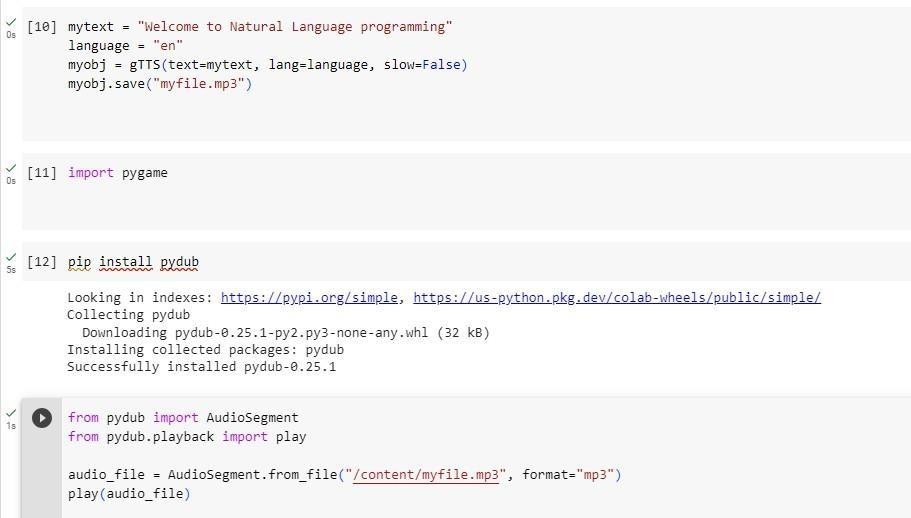
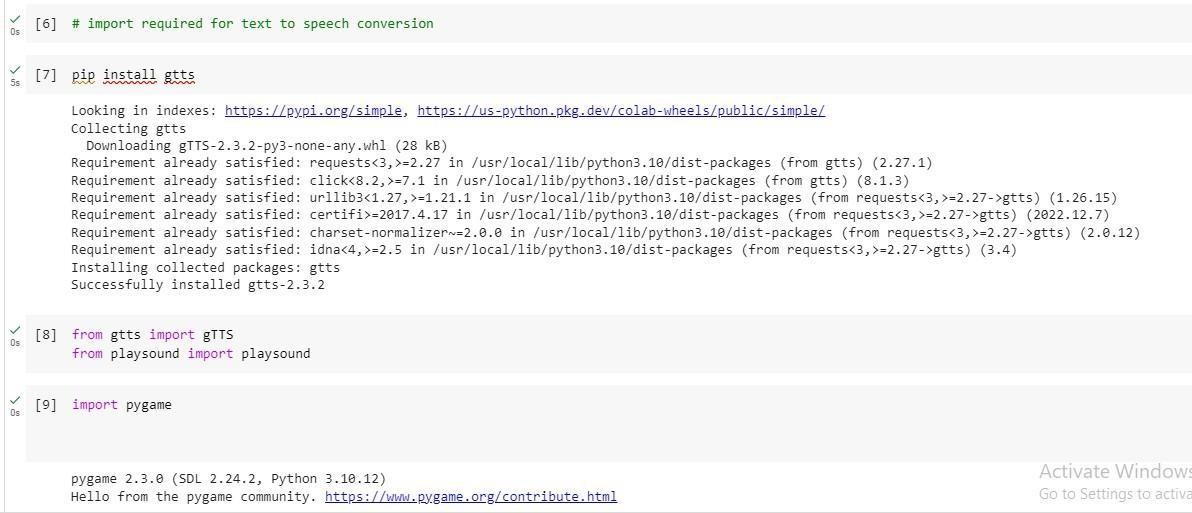
>>>importnltk

>>>



###### PillaiHOCCollegeof Arts,Science &Commerce Page |3

**1B]Convertthegiventextto speech.**



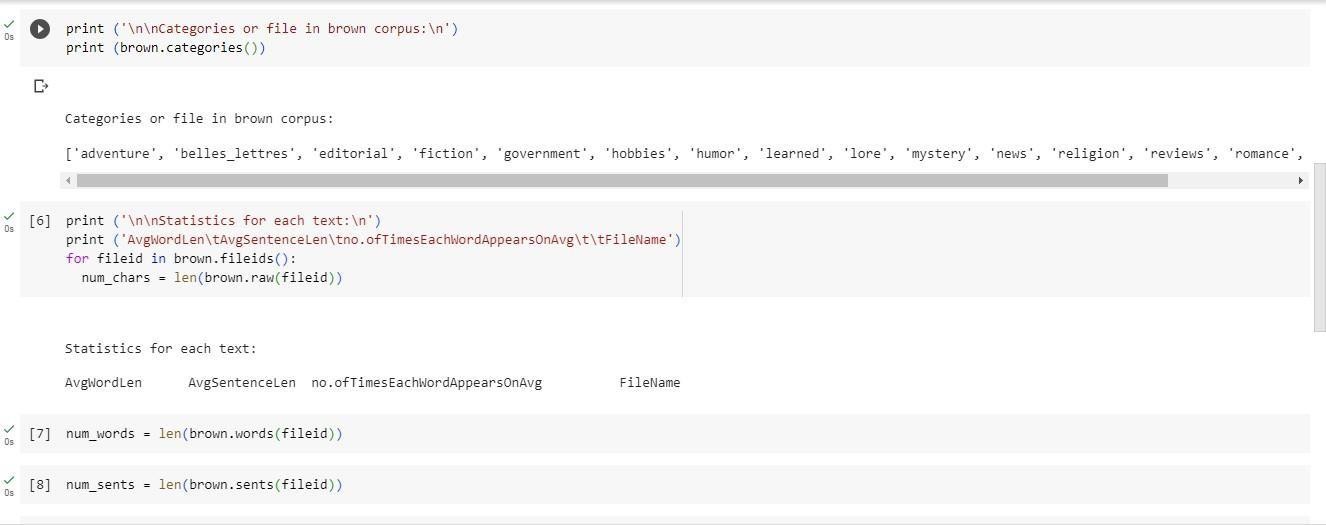
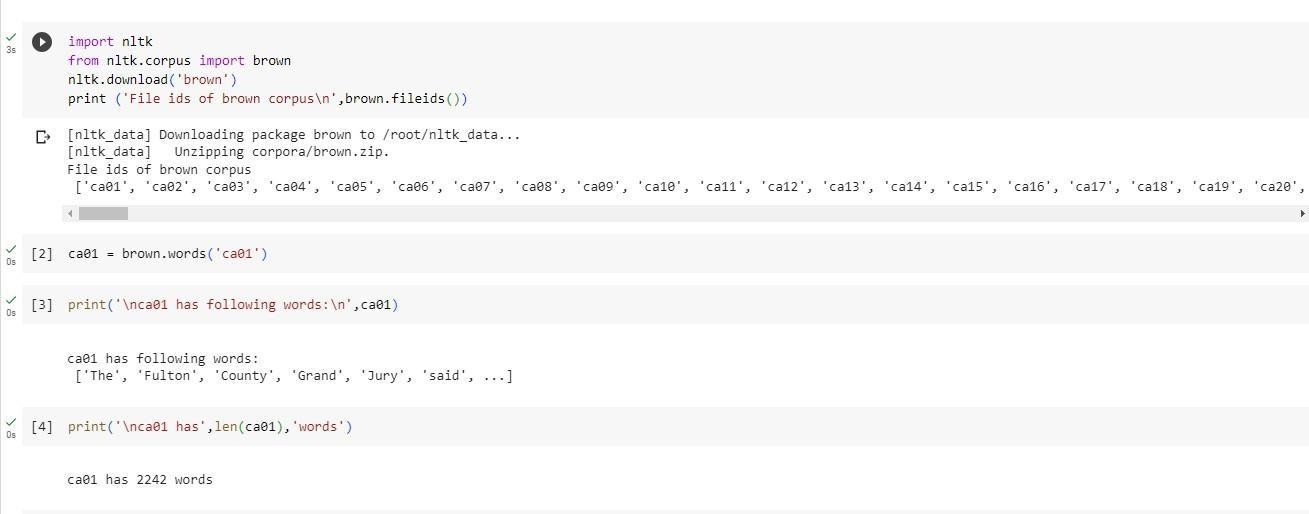
**1C]ConvertaudiofileSpeechto Text.**





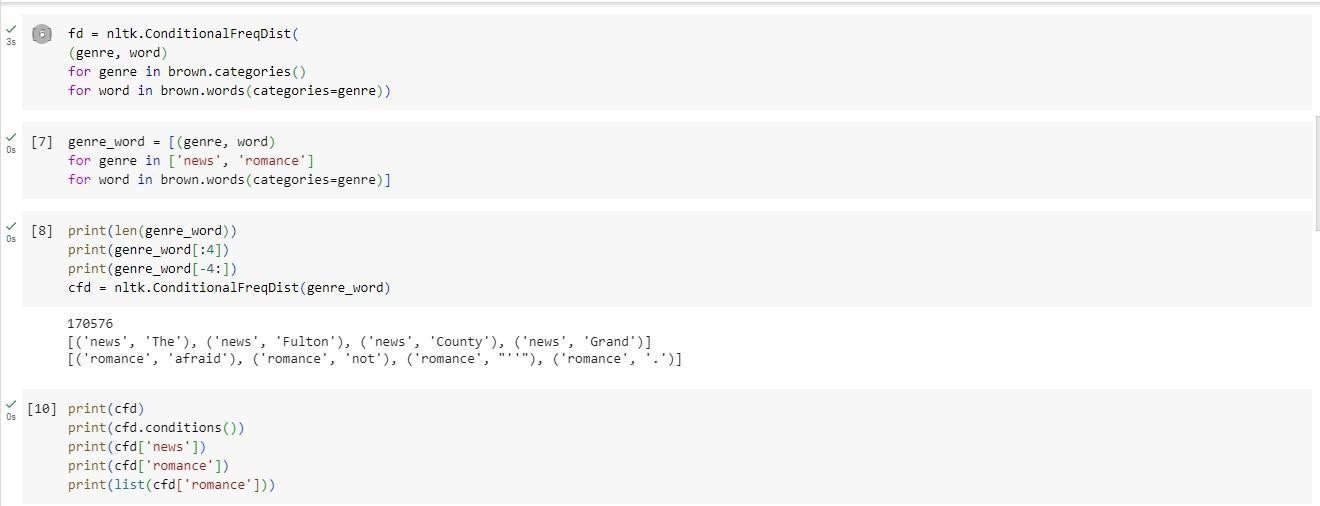
### PRACTICAL:2

###### 2A]StudyofvariousCorpus–Brown,Inaugural,Reuters,udhrwithvariousmethod likefilelds, raw,words, sents, categories.



**2C]StudyConditionalfrequency distributions**

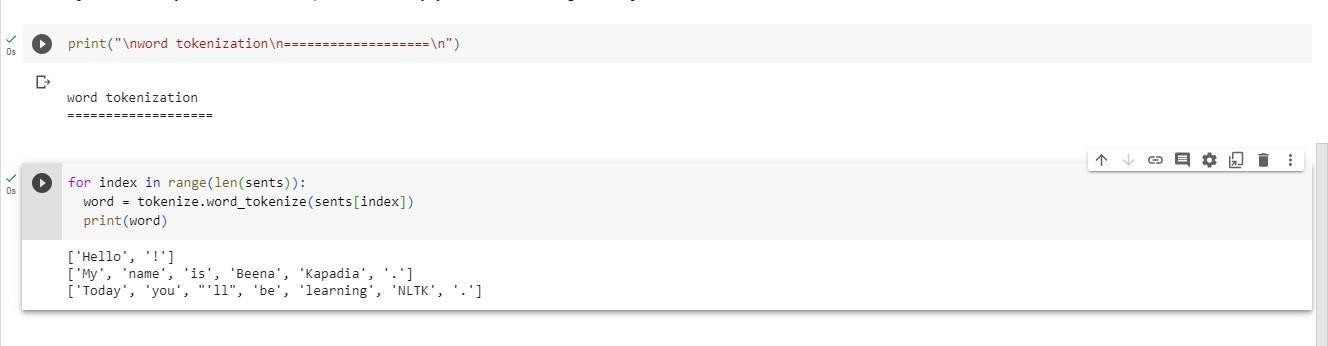




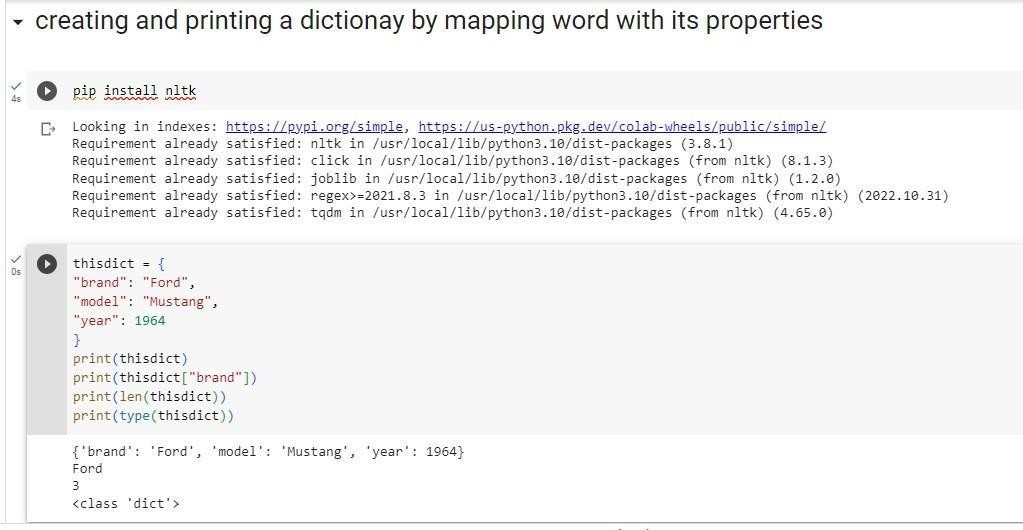




###### 2D]Studyof taggedcorporawithmethodsliketagged\_sents,tagged\_words.



**2F]MapWordsto PropertiesUsingPython Dictionaries.**



### PRACTICAL:3

###### 3A]Studyof WordnetDictionarywith methodsassynsets,definitions,examples,antonyms



**3B]Studylemmas,hyponyms,hypernyms.**



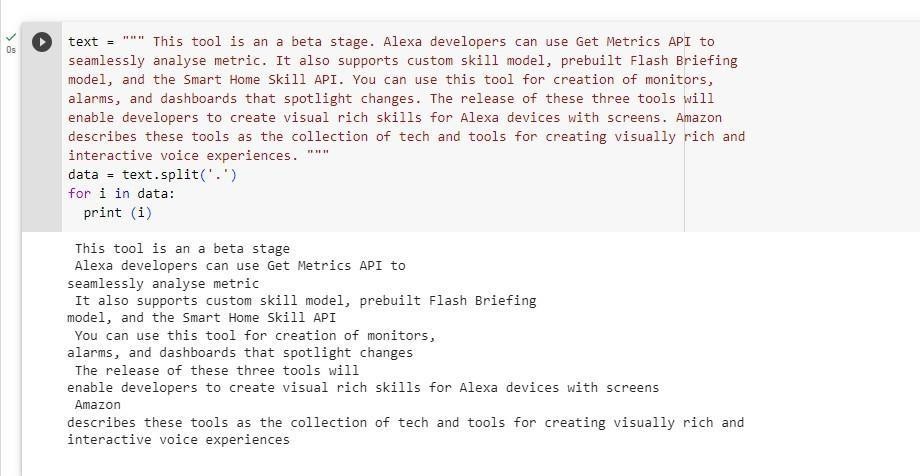


**3C]Writeaprogramusingpythontofindsynonymandantonymofword "active"usingWordnet.**



### PRACTICAL:4

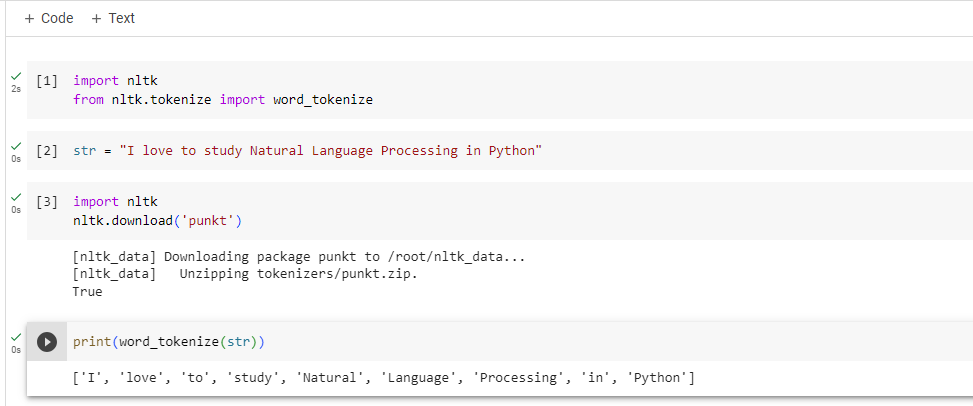
###### 4A]TokenizationusingPython’ssplit() function



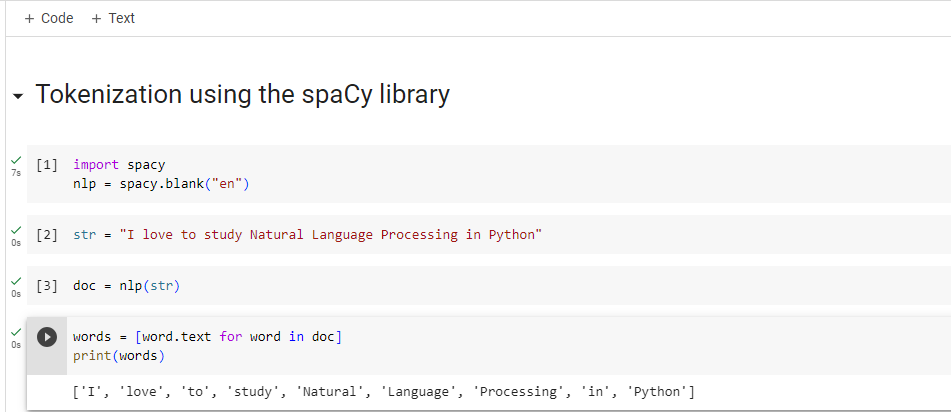
**4B]TokenizationusingRegularExpressions(RegEx)**



###### 4C]Tokenization usingNLTK

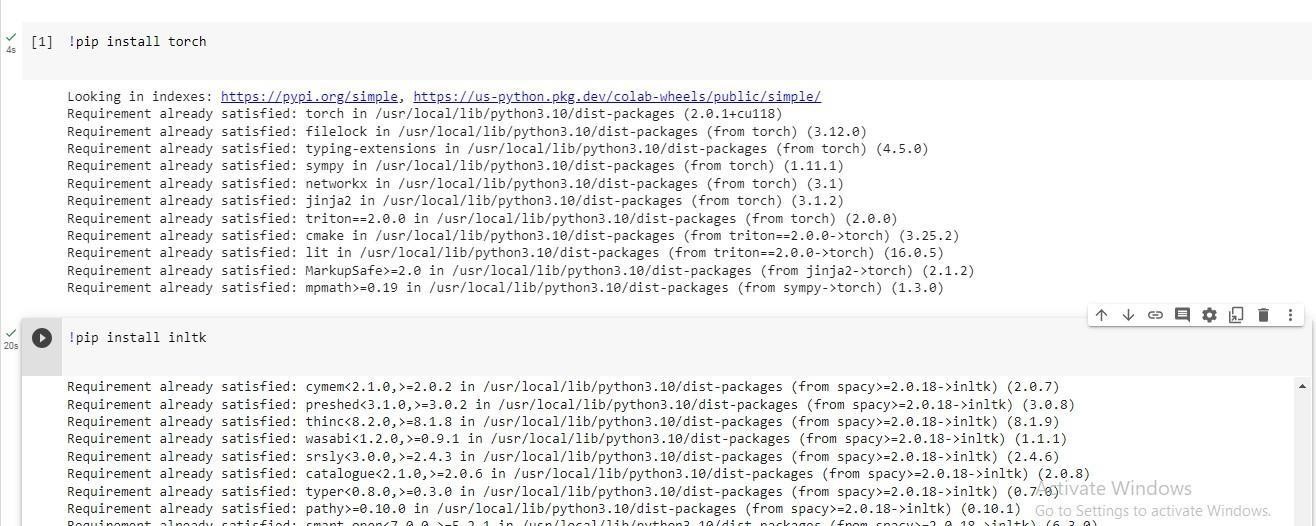


**4D]Tokenizationusingthe spaCylibrary**



### PRACTICAL:5

###### ImportNLPLibrariesforIndianLanguagesandperform 5A] word tokenization in Hindi

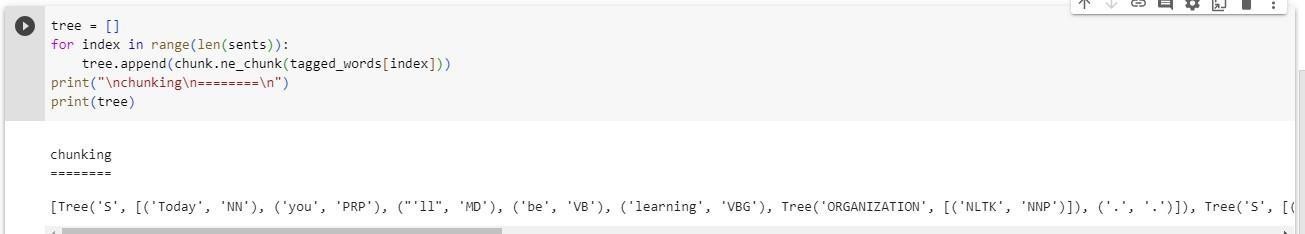
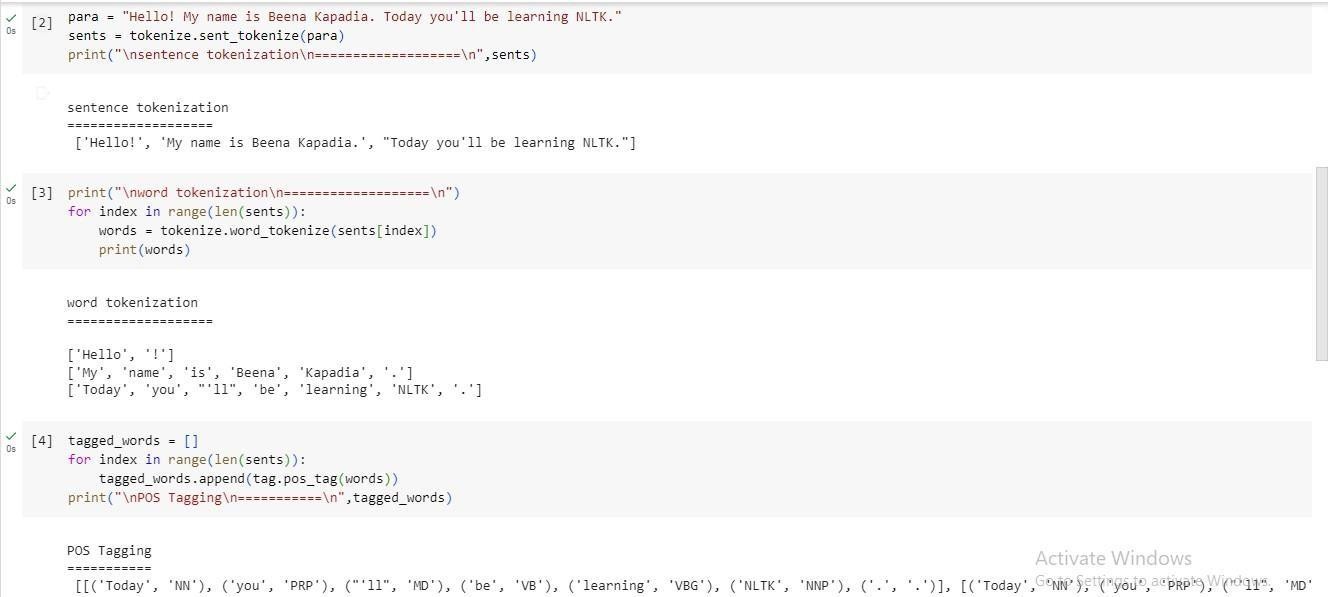


**Illustratepartofspeechtagging.**

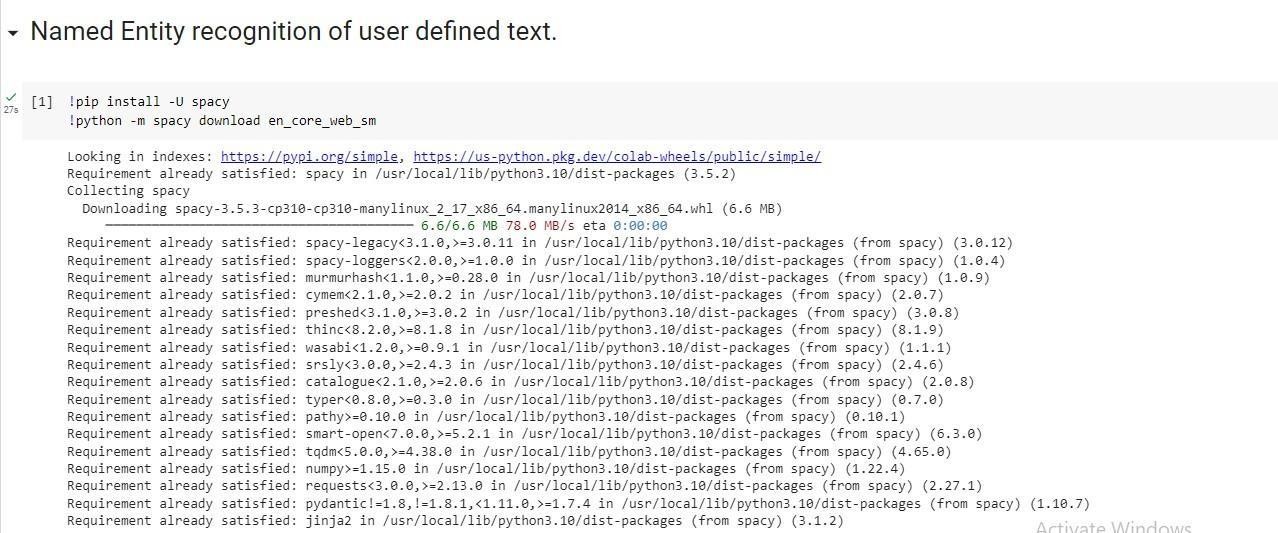
### PRACTICAL:6

###### 6A]PartofspeechTaggingandchunkingof userdefinedtext.





**6B]NamedEntityrecognitionofuserdefinedtext.**

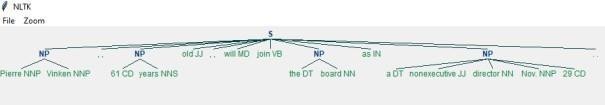


###### 6C]Named Entity recognitionwith diagramusingNLTKcorpus– Treebank

importnltknltk.download('treebank'

)

fromnltk.corpus import treebank\_chunktreebank\_chunk.tagged\_sents()[0] treebank\_chunk.chunked\_sents()[0] treebank\_chunk.chunked\_sents()[0].draw()



**Finitestate automata**

### PRACTICAL:7

###### 7A]Definegrammarusingnltk.Analyzeasentenceusingthesame.

importnltk fromnltkimporttokenize

grammar1=nltk.CFG.fromstring(""" S

-> VP

VP->VPNP NP->DetNP

Det->'that'

NP->singularNoun NP

-> 'flight'

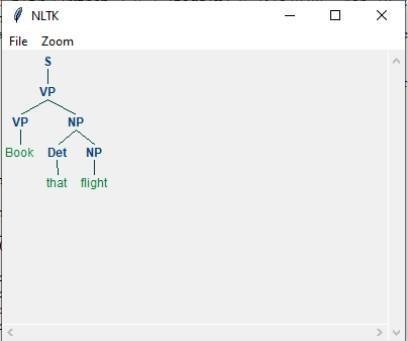
VP->'Book' """)

sentence="Bookthatflight" forindexin 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()



###### 7B]AccepttheinputstringwithRegularexpressionof Finite Automaton:101+.

defFA(s):

#ifthelengthislessthan3,itcan'tbeaccepted,soendtheprocess iflen(s) <

3:

return"Rejected"

#firstthreecharactersarefixed,checkingthemusingindex if s[0] == '1':

ifs[1]=='0':

ifs[2]=='1':

#Afterindex2,only"1"canappear,so break theprocess if anyothercharacter is

detected

fori inrange(3, len(s)):

ifs[i] !='1':

return"Rejected" return"Accepted"#ifallnestedifsaretrue return "Rejected"# else of 3rd if return"Rejected"#elseof2ndif

return "Rejected"# else of 1st if

inputs=['1','10101','101','10111','01010','100','','10111101','1011111']

fori in inputs:

print(FA(i))



###### 7C]AccepttheinputstringwithRegularexpressionof FA:(a+b)\*bba.

defFA(s):

size= 0

#scancompletestringandmakesurethatitcontainsonly'a'&'b' for i in s: ifi=='a'ori=='b': size

+= 1

else: return"Rejected"

#Aftercheckingthatitcontainsonly'a'&'b' # check its length; it should be at least 3

if size>=3:

#checkthelast3elements if s[size-3] == 'b':

ifs[size-2]=='b':

ifs[size-1]=='a': return"Accepted"#ifall3ifsaretrue return "Rejected"# else of 3rd if return "Rejected"# else of 2nd if return "Rejected"# else of 1st ifreturn "Rejected"

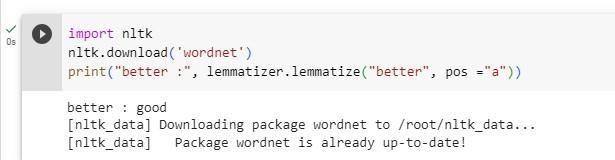
inputs=['bba','ababbba','abba','abb','baba','bbb',''] for i in inputs:

print(FA(i))



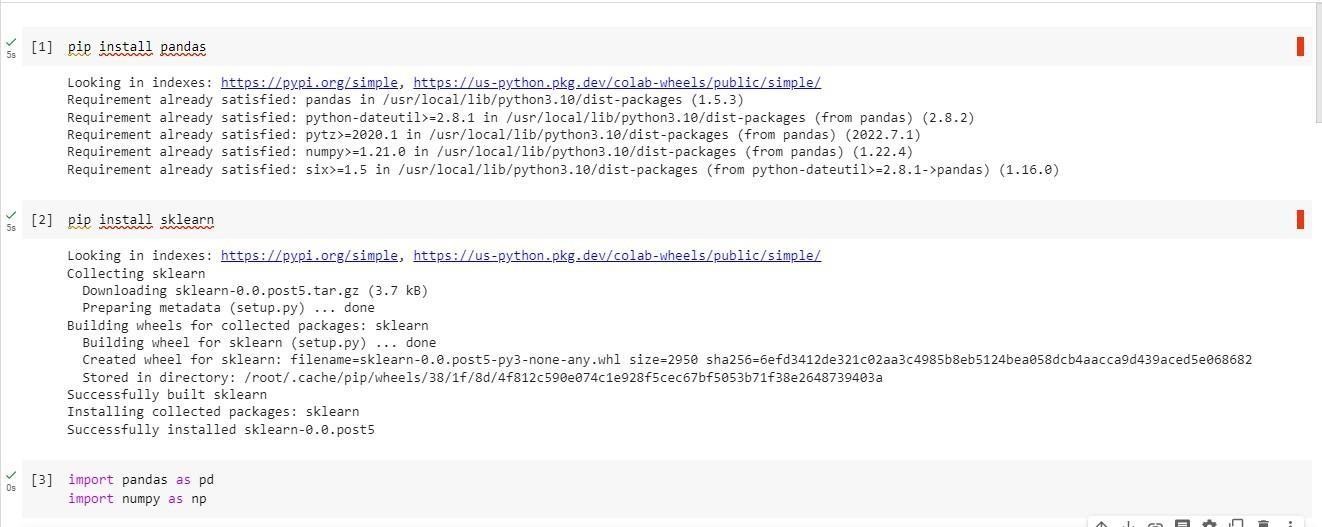
### PRACTICAL:8

###### StudyPorterStemmer,LancasterStemmer,RegexpStemmer,SnowballStemmer Study WordNetLemmatizer



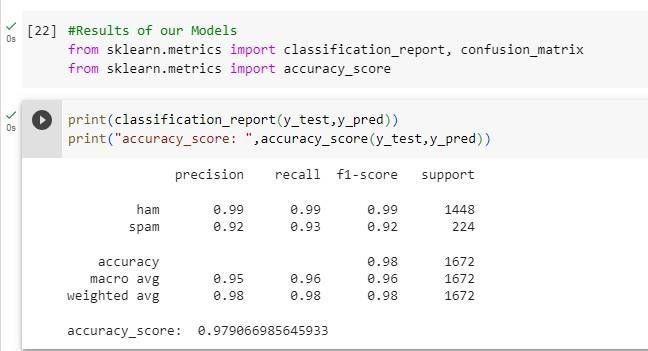
**ImplementNaiveBayes classifier**

### PRACTICAL:9



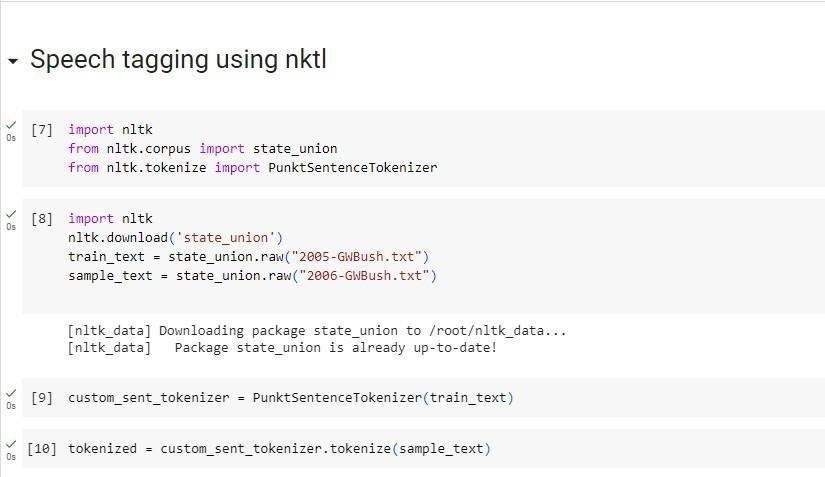


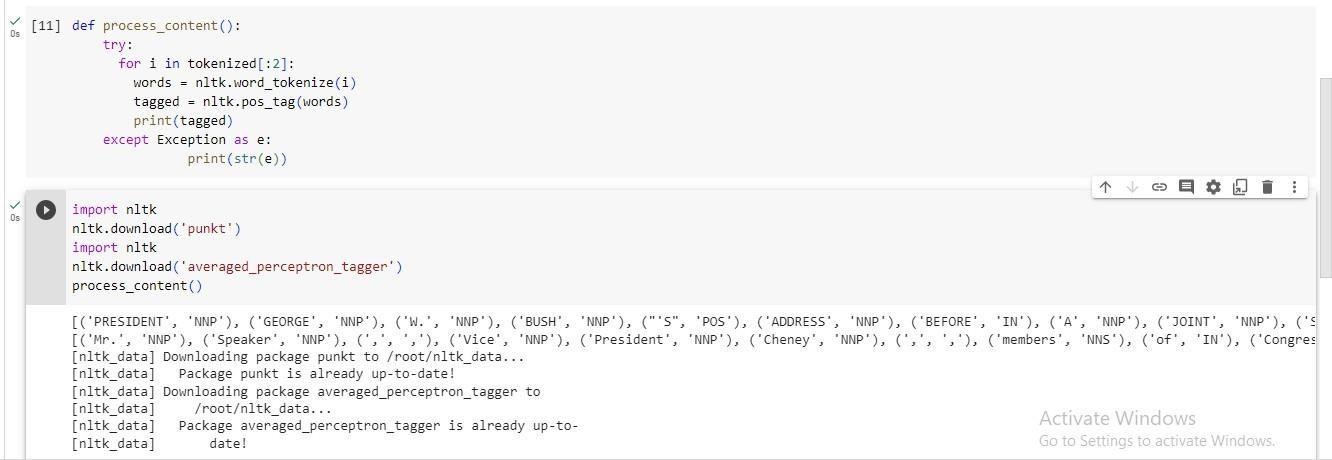




**10AII]Speech taggingusing nktl**

### PRACTICAL:10

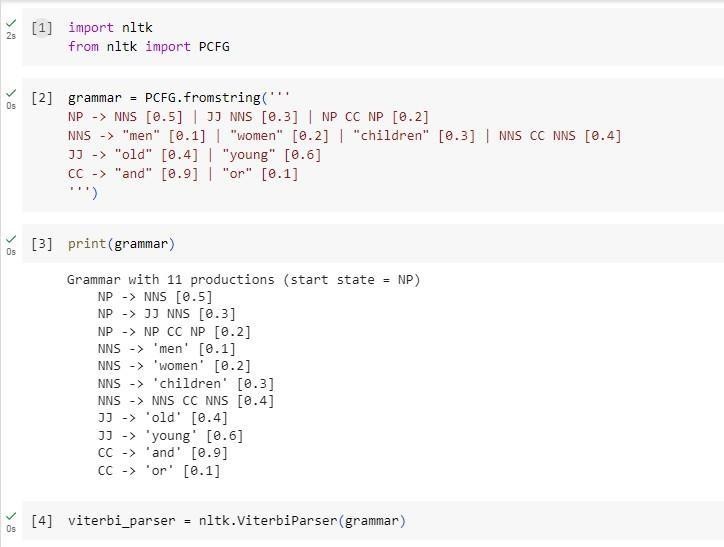




###### 10BI]Usageof Giveand Gavein the PennTreebanksample



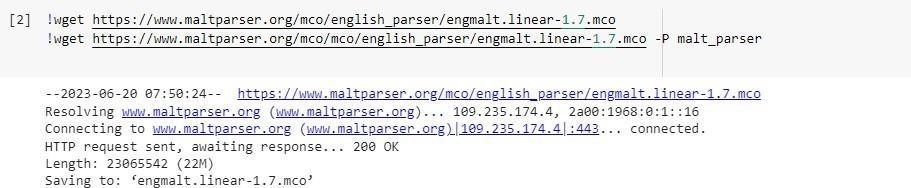
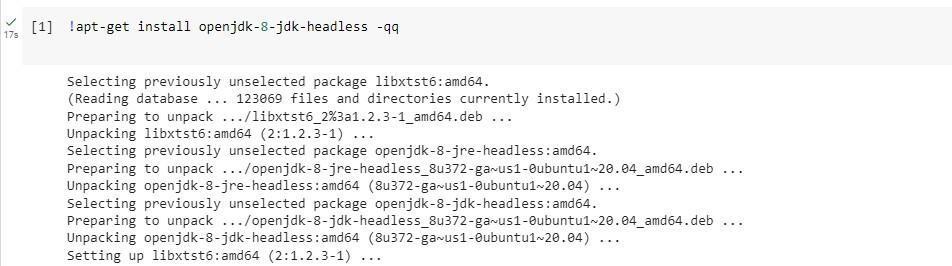
**10BII]probabilisticparser**





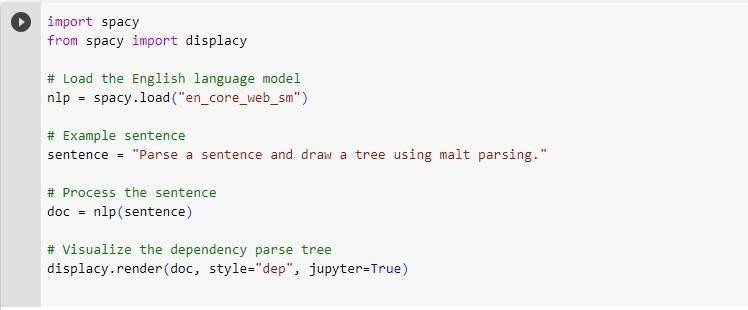
###### 10C]Malt parsing:

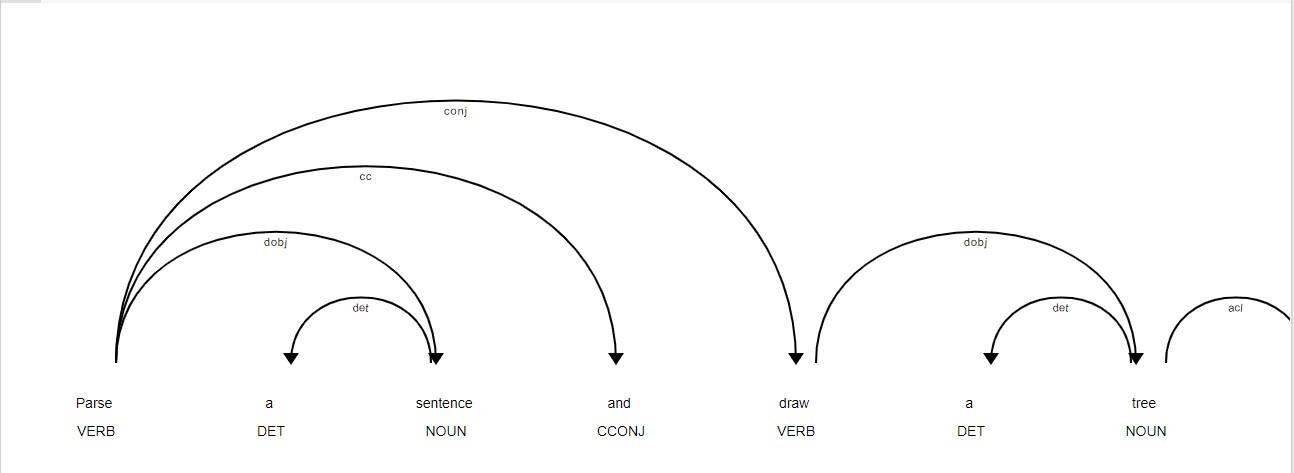
**Parseasentenceanddraw atreeusingmaltparsing.**





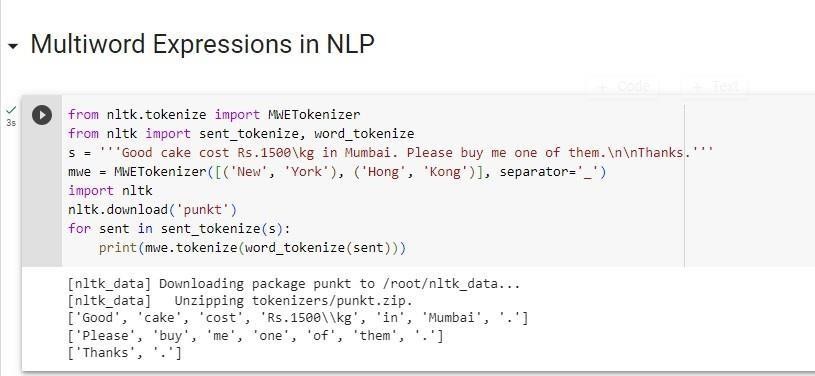






**11A]MultiwordExpressionsinNLP**

### PRACTICAL:11



###### 11B]NormalizedWeb DistanceandWordSimilarity

importnumpyasnp import re importtextdistanceimp ortsklearn

fromsklearn.clusterimportAgglomerativeClustering

texts =[

'Reliancesupermarket','Reliancehypermarket','Reliance','Reliance','Reliance', 'downtown', 'Reliance market', 'Mumbai','MumbaiHyper','Mumbaidxb','mumbai airport', 'k.mtrading','KMTrading','KMtrade', 'K.M.Trading', 'KM.Trading'

]

defnormalize(text):

"""Keeponlylower-casedtextandnumbers""" returnre.sub('[^a-z0-9]+', ' ', text.lower())

defgroup\_texts(texts, threshold=0.4): """Replaceeachtextwiththerepresentativeofitscluster"""

normalized\_texts = np.array([normalize(text) for text in texts]) distances = 1 - np.array([

[textdistance.jaro\_winkler(one,another)foroneinnormalized\_texts]

foranotherin normalized\_texts

])

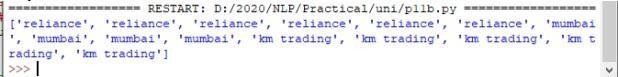
clustering=AgglomerativeClustering( distance\_threshold=threshold, affinity="precomputed",linkage="complete",n\_clusters=None

).fit(distances) centers=dict()

forcluster\_idin set(clustering.labels\_): index=clustering.labels\_==cluster\_id

centrality = distances[:, index][index].sum(axis=1) centers[cluster\_id]=normalized\_texts[index][centrality.argmin()] return [centers[i] for i in clustering.labels\_]

print(group\_texts(texts))



**11C]WordSenseDisambiguation**

