EXP:01 DATE: 12/01/24

# WORD ANALYSIS

## AIM:

To implement word analysis using Python and NLTK.

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

STEP10: String handling code has been generated and executed STEP11: Stop the program

## PROGRAM:

print(len("what it is what it isnt")) s=["what","it","is","what","it","isnt"] print(len(s))

x=sorted(s) print(s) print(x) d=x+s print(d)

## OUTPUT:

**RESULT:**

Word analysis using Python and NLTK is verified and executed.

EXP:02 DATE: 19/01/24

# WORD GENERATION

## AIM:

To implement word generation using Python and NLTK.

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

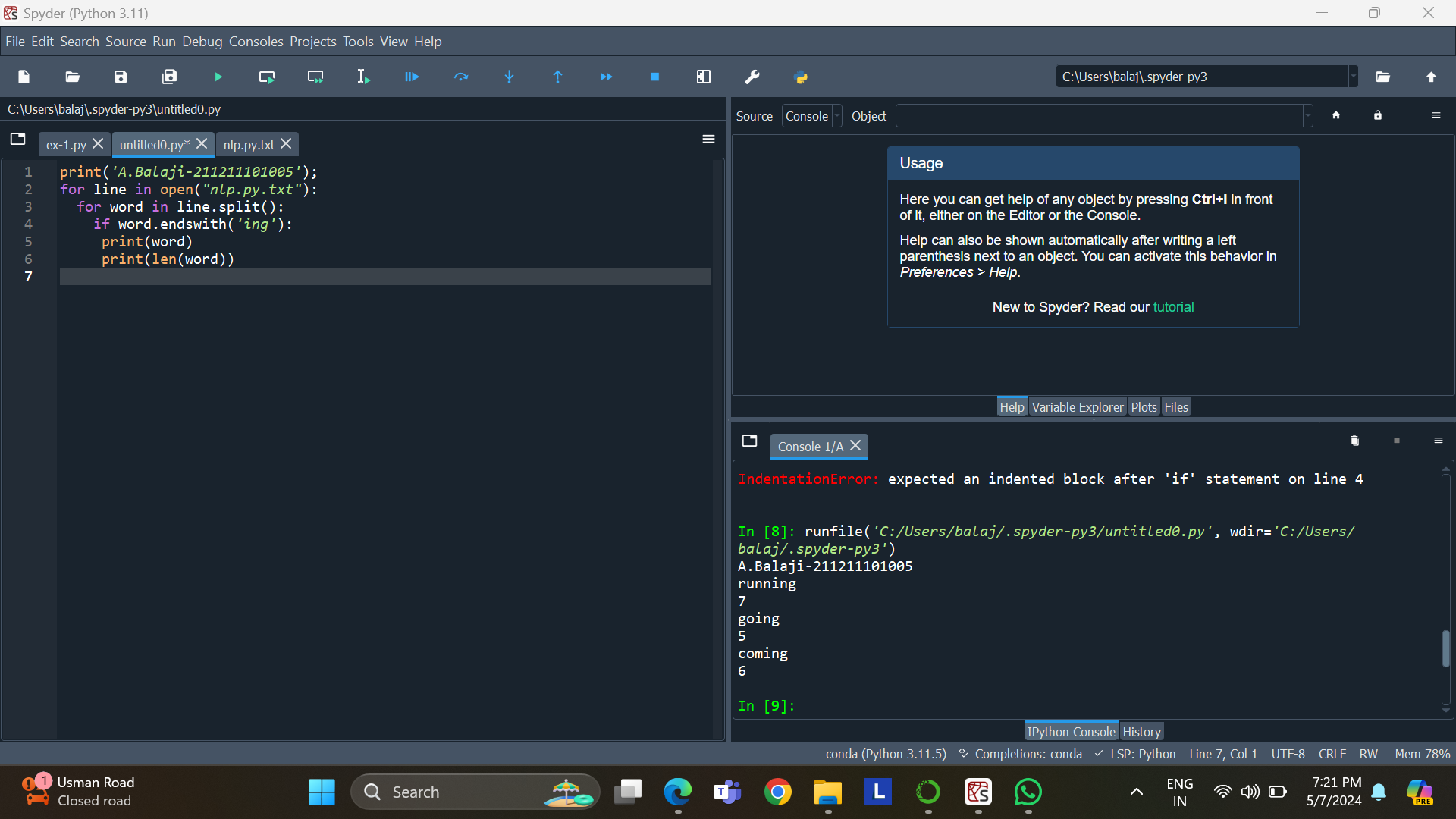
STEP10: File Handling is done using the process of tokenization and executed STEP11: Stop the program

## PROGRAM:

for line in open("nlp.py"): for word in line.split():

if word.endswith('ing'): print(word) print(len(word))

## OUTPUT:

****

**RESULT:**

Word generation using Python and NLTK is verified and executed.

EXP:03 DATE: 02/02/24

# MORPHOLOGY

## AIM:

To implement morphology using Python and NLTK.

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch STEP10: General Morphology Code and Stop Word Removal STEP11: Stop the program

## PROGRAM:

CODE:

import re input="The5biggestanimalsare1.Elephant,2Rhinoand3dinasaur" input=input.lower()

print(input) result=re.sub(r'\d+','',input) print(result)

STOP WORD REMOVAL:

def punctuations(raw\_review): text=raw\_review text=text.replace("n't",'not') text=text.replace("'s",'is') text=text.replace("'re",'are') text=text.replace("'ve",'have') text=text.replace("'m",'am') text=text.replace("'d",'would') text=text.replace("'ll",'will') text=text.replace("in",'ing') import re

letters\_only=re.sub("[^a-zA-Z]","",text) return(''.join(letters\_only))

t="Hows'smyteamdoin ,you'resupposedtobenotloosin" p=punctuations(t)

print(p)

SYNONYM:

import nltk nltk.download('omw-1.4') nltk.download('wordnet')

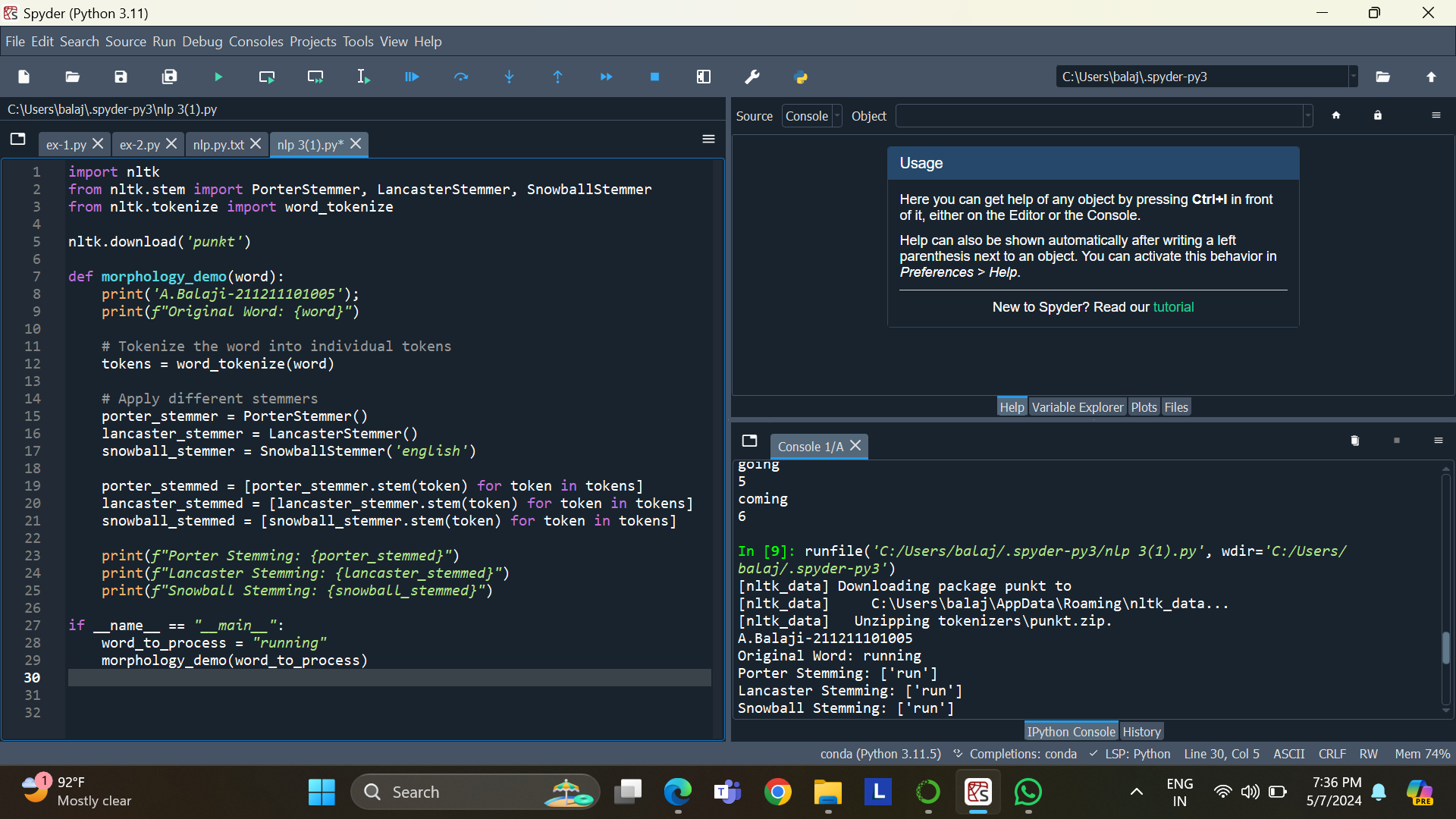
from nltk.corpus import wordnet synonyms = []

for syn in wordnet.synsets('Machine'): for lemma in syn.lemmas(): synonyms.append(lemma.name()) print(synonyms)

STEMMING:

from nltk.stem import PorterStemmer stemmer=PorterStemmer() print(stemmer.stem('eating')) print(stemmer.stem('ate')

## OUTPUT:

****

**RESULT:**

The Morphological Analysis Code of NLP is verified and executed

EX.NO:04 DATE: 09/02/24

# N-GRAMS

## AIM:

To implement N-Grams using Python and NLTK.

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch STEP10: N-Gram code has been generated and executed. STEP11: Stop the program

## PROGRAM:

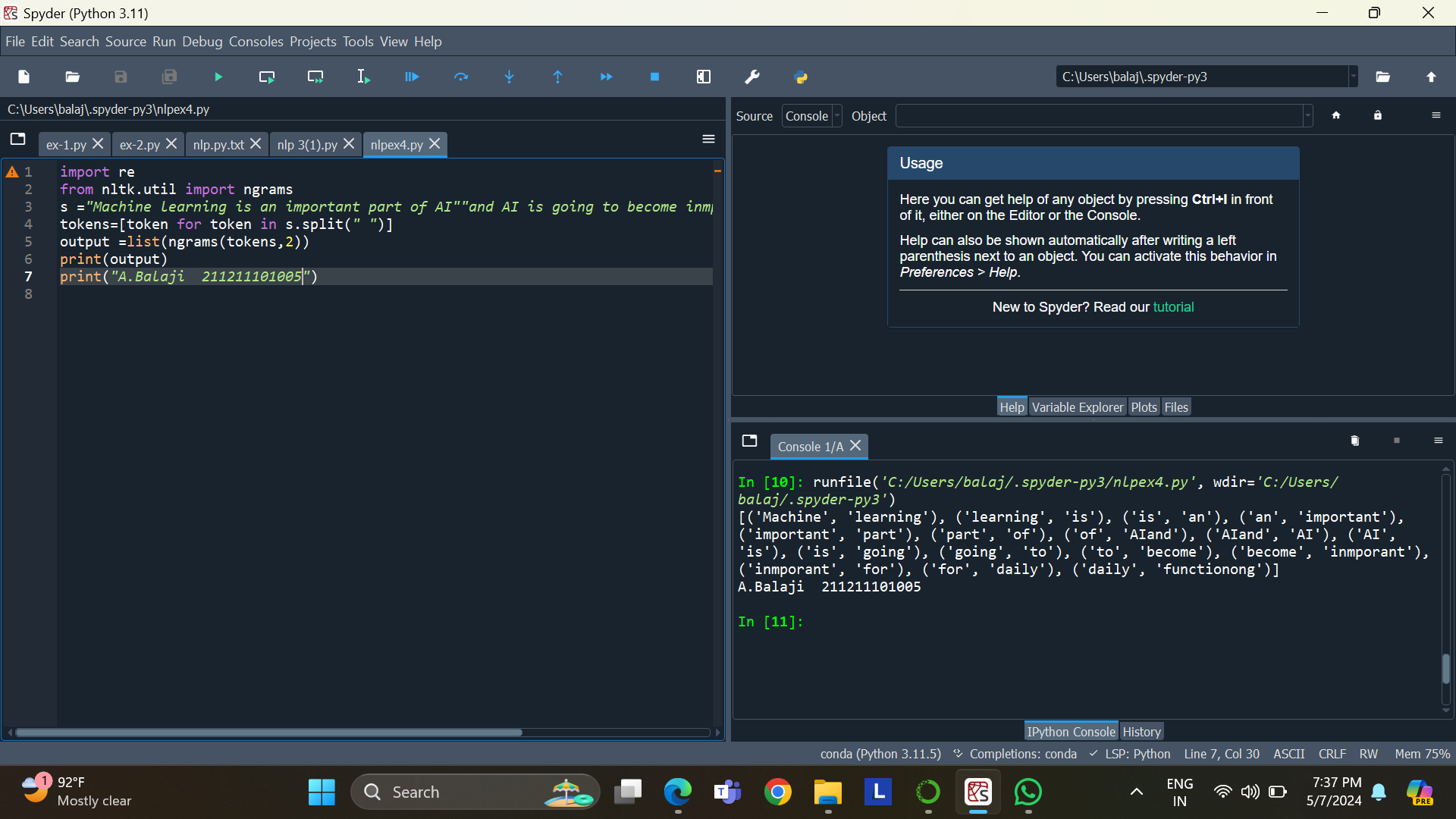
import re

from nltk.util import ngrams

s ="Machine learning is an important part of AI""and AI is going to become inmporant for daily functionong"

tokens=[token for token in s.split(" ")] output =list(ngrams(tokens,2)) print(output)

## OUTPUT:

****

**RESULT:**

The N Grams code has been executed and verified using Python and NLTK.

EX.NO:05 DATE: 15/02/24

# N-GRAMS SMOOTHING

## AIM:

To implement N-Grams Smoothing using Python and NLTK.

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

STEP10: N-Gram Smoothing code has been generated and executed. STEP11: Stop the program

## PROGRAM:

from collections import Counter import numpy as np

# Define corpus

corpus = "the quick brown fox jumps over the lazy dog" # Create unigrams

unigrams = Counter(corpus.split())

# Define function to compute n-grams def get\_ngrams(sentence, n):

return [tuple(sentence[i:i+n]) for i in range(len(sentence)-n+1)] # Create bigrams

bigrams = Counter(get\_ngrams(corpus.split(), 2)) # Define smoothing function

def add\_k\_smoothing(ngram\_counts, k, n\_1gram\_counts): # Calculate total number of n-grams

total\_ngrams = sum(ngram\_counts.values()) # Calculate vocabulary size

vocabulary\_size = len(n\_1gram\_counts)

# Calculate denominator for probability calculation denominator = total\_ngrams + k\*vocabulary\_size # Calculate smoothed probabilities

probabilities = {}

for ngram, count in ngram\_counts.items(): probabilities[ngram] = (count + k) / denominator

# Handle unseen n-grams

for ngram in set(n\_1gram\_counts.keys()) - set(ngram\_counts.keys()): probabilities[ngram] = k / denominator

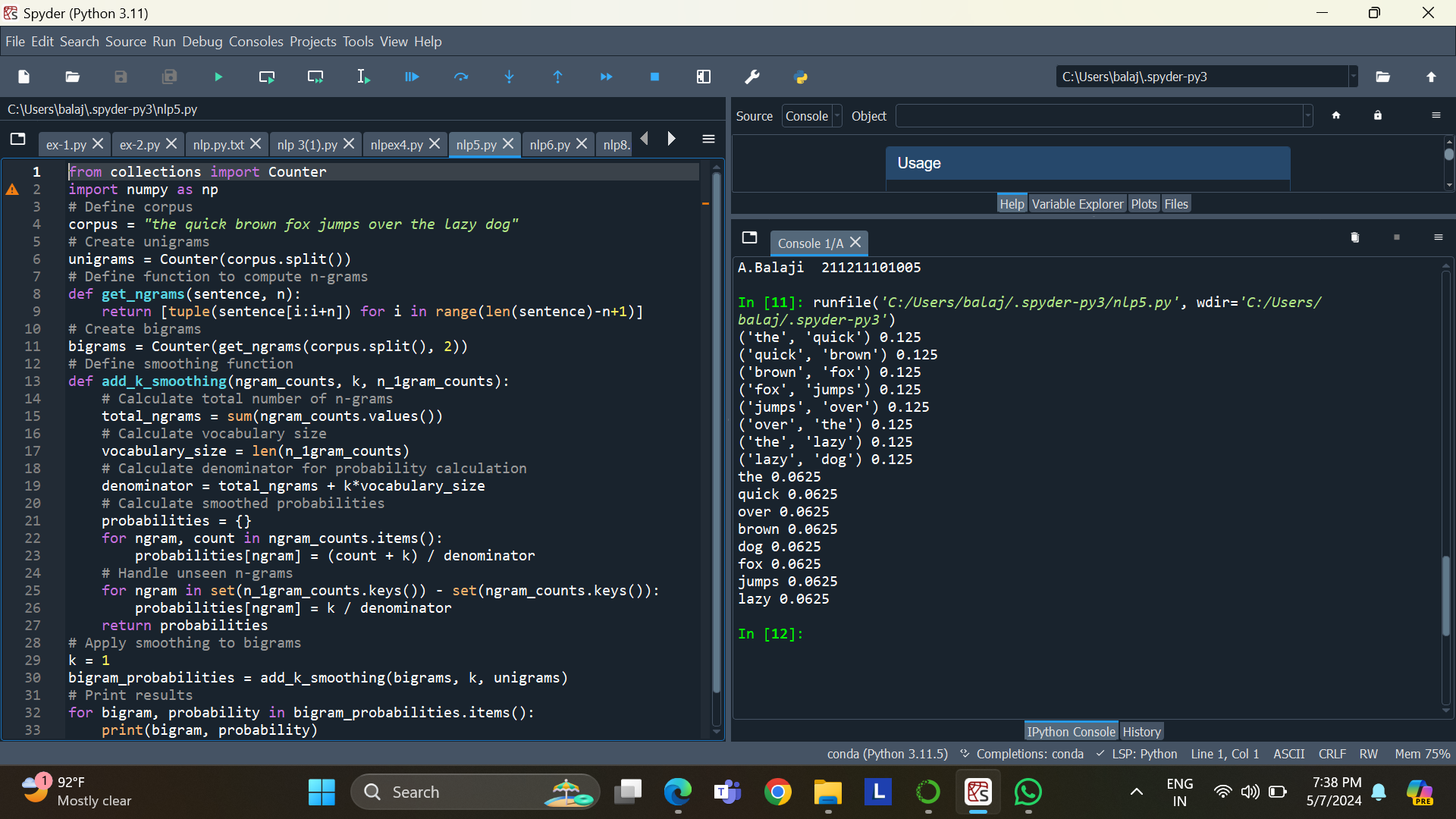
return probabilities

# Apply smoothing to bigrams k = 1

bigram\_probabilities = add\_k\_smoothing(bigrams, k, unigrams) # Print results

for bigram, probability in bigram\_probabilities.items(): print(bigram, probability)

## OUTPUT:



**RESULT:**

The N-Gram Smoothing code has been executed and verified using Python and NLTK.

EX.NO:06 DATE: 23/02/24

# POS – TAGGING: HIDDEN MARKOV MODEL

## AIM:

To implement POS-Tagging: Hidden Markov Model using Python and NLTK

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras, NLTK, Pandas, Numba and Random and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

STEP10:Using POS-Tagging, Hidden Markov Model code has been generated and executed. STEP11: Stop the program

## PROGRAM:

import nltk

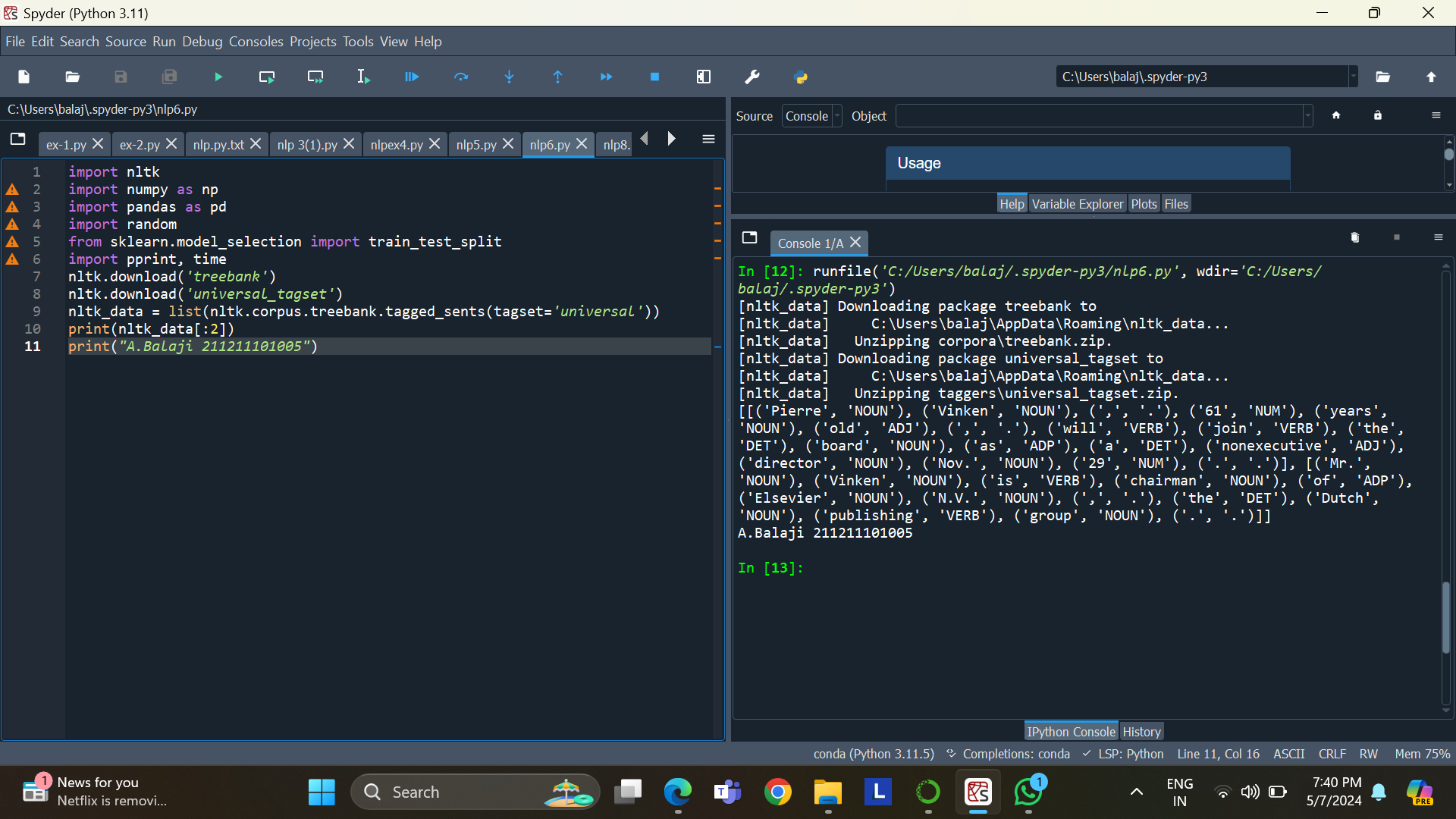
import numpy as np import pandas as pd import random

from sklearn.model\_selection import train\_test\_split import pprint, time

nltk.download('treebank') nltk.download('universal\_tagset')

nltk\_data = list(nltk.corpus.treebank.tagged\_sents(tagset='universal')) print(nltk\_data[:2]

## OUTPUT:

****

**RESULT:**

Using POS Tagging, Hidden Markov Model has been executed and verified using Python and NLTK.

## AIM:

EX.NO:07 DATE: 24/02/24

# POS – TAGGING: VITERBI DECODING

To implement POS-Tagging: Viterbi Decoding using Python and NLTK

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

STEP10:Using POS-Tagging, Viterbi Decoding has been generated and executed. STEP11: Stop the program.

## PROGRAM :

import nltk

from nltk.corpus import brown # Training data

sentences = brown.tagged\_sents()[:5000]

# Create tag frequency distribution and transition probability matrix

tag\_freq = nltk.FreqDist(tag for sentence in sentences for word, tag in sentence) transition\_prob = nltk.ConditionalFreqDist(

(tag1, tag2) for sentence in sentences for (\_, tag1), (\_, tag2) in nltk.bigrams(sentence)

)

# Define Viterbi function

def viterbi(sentence, tag\_freq, transition\_prob): # Initialize first word probabilities

v = [{}]

for tag in tag\_freq:

v[0][tag] = {"prob": tag\_freq[tag] / len(sentences), "prev": None} # Recursion step

for i in range(1, len(sentence)): v.append({})

for tag in tag\_freq: max\_prob = max(

v[i - 1][prev\_tag]["prob"] \* transition\_prob[prev\_tag][tag] \* tag\_freq[tag] / len(sentences)

for prev\_tag in tag\_freq

)

for prev\_tag in tag\_freq:

if v[i - 1][prev\_tag]["prob"] \* transition\_prob[prev\_tag][tag] \* tag\_freq[tag] / len(sentences) == max\_prob:

v[i][tag] = {"prob": max\_prob, "prev": prev\_tag} break

# Termination step

max\_prob = max(value["prob"] for value in v[-1].values()) current\_tag = None

for tag, data in v[-1].items():

if data["prob"] == max\_prob: current\_tag = tag

break

# Backtracking

tags = [current\_tag]

for i in range(len(v) - 1, 0, -1):

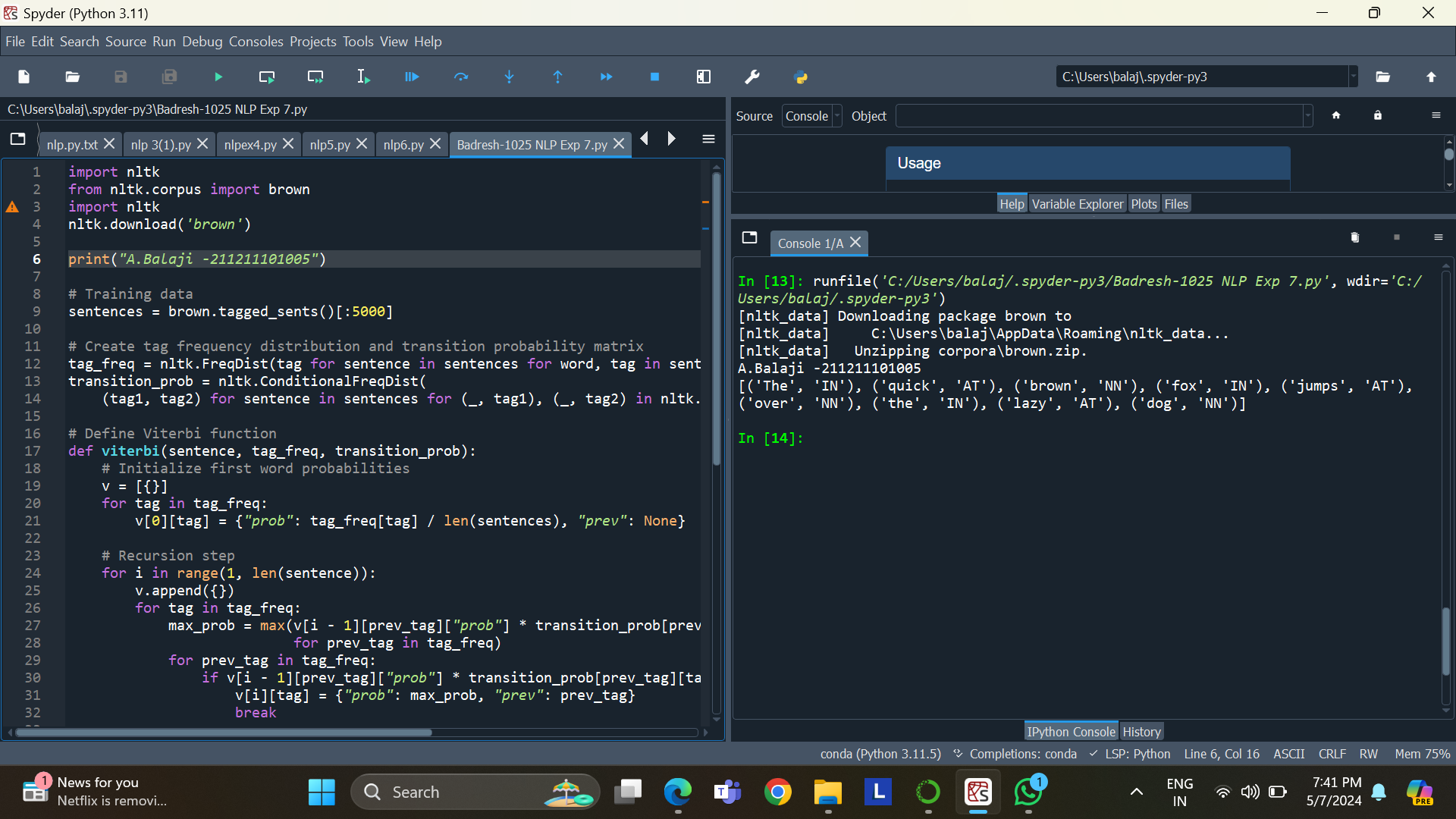
current\_tag = v[i][current\_tag]["prev"] tags.append(current\_tag)

tags.reverse()

return list(zip(sentence, tags)) # Example usage

sentence = "The quick brown fox jumps over the lazy dog".split() pos\_tags = viterbi(sentence, tag\_freq, transition\_prob) print(pos\_tags)

## OUTPUT :

****

**RESULT:**

Using POS Tagging, Viterbi Decoding has been executed and verified using Python and NLTK.

EX.NO:08 DATE: 01/03/24

# BUILDING POS TAGGER

## AIM:

To implement Building POS Tagger using Python and NLTK

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

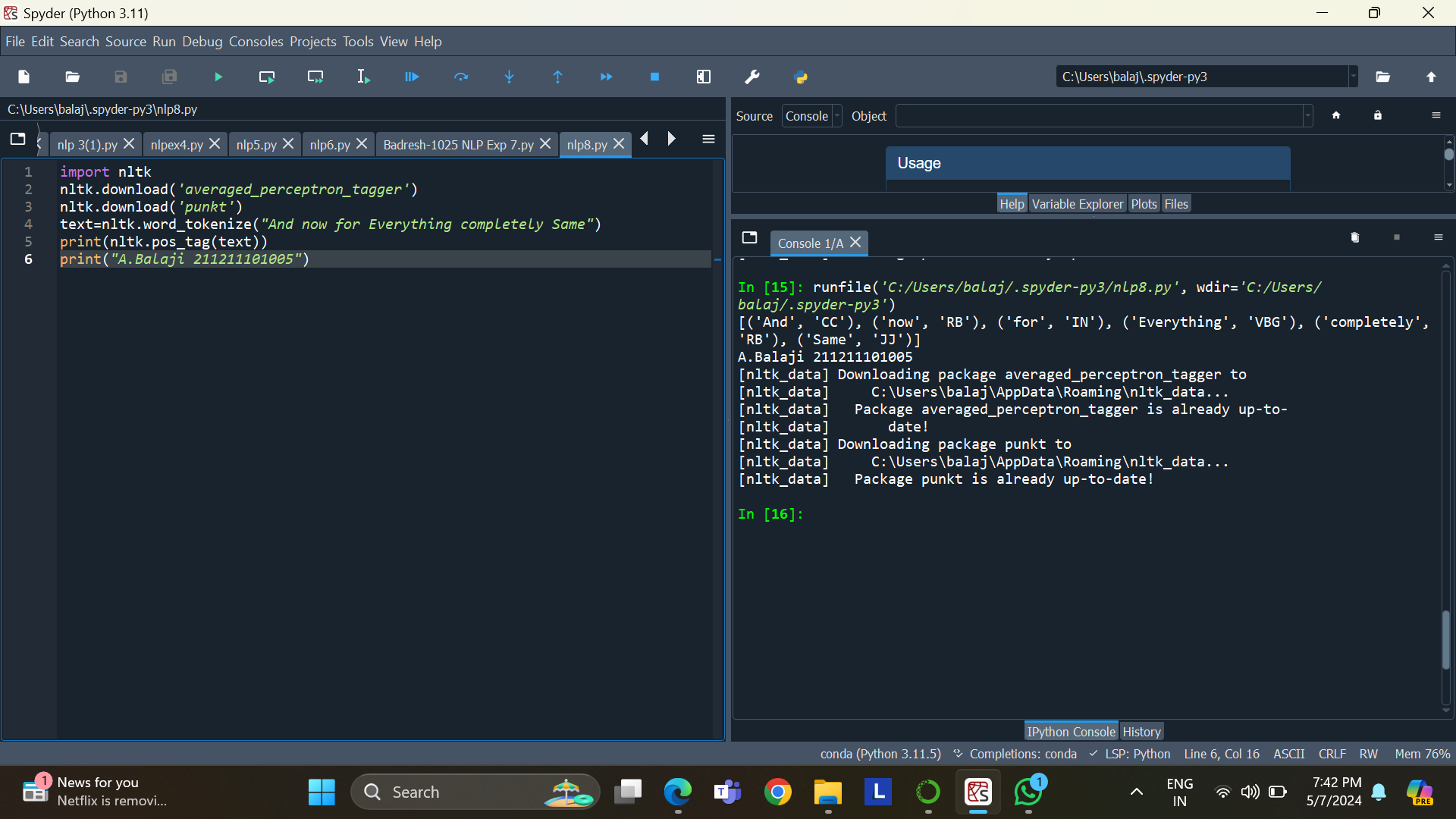
STEP10: Building POS Tagger has been generated and executed. STEP11: Stop the program

## PROGRAM:

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

text=nltk.word\_tokenize("And now for Everything completely Same") print(nltk.pos\_tag(text))

## OUTPUT :

****

**RESULT:**

Building POS Tagger code has been executed and verified using Python and NLTK.

EX.NO:09 DATE: 08/03/24

# CHUNKING

## AIM:

To implement Chunking code using Python and NLTK

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

STEP10: Chunking code is generated and verified, by printing the result. STEP11: Stop the program

## PROGRAM:

import nltk

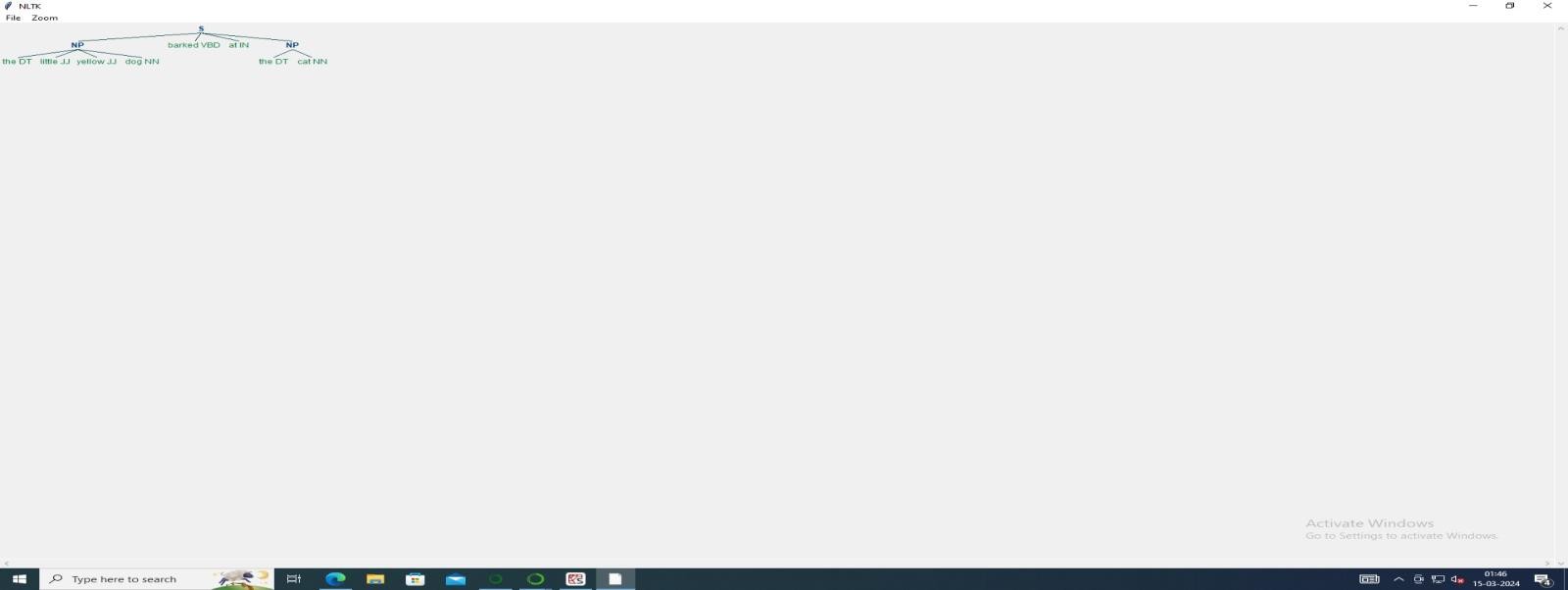
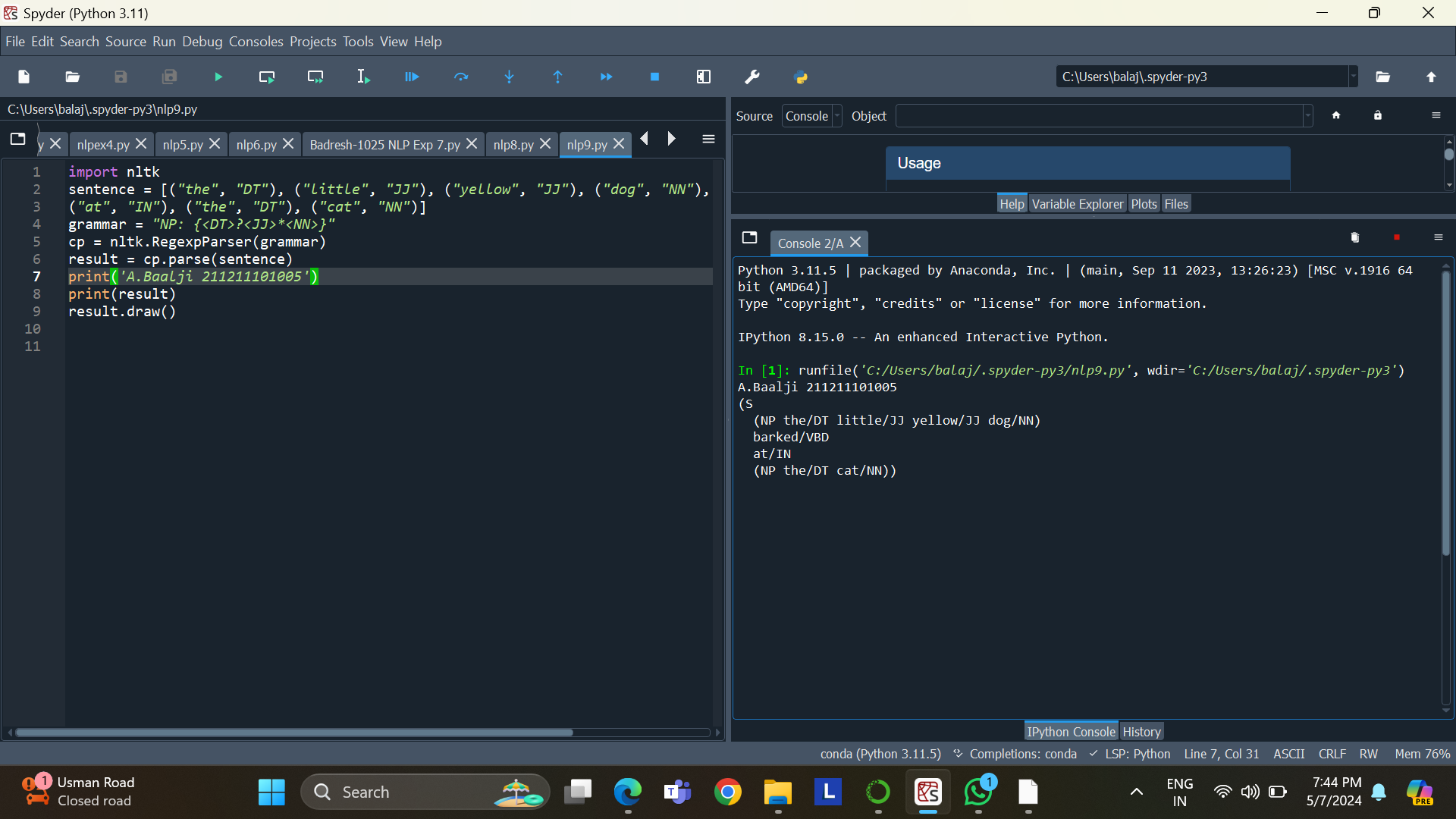
sentence = [("the", "DT"), ("little", "JJ"), ("yellow", "JJ"), ("dog", "NN"), ("barked", "VBD"), ("at", "IN"), ("the", "DT"), ("cat", "NN")]

grammar = "NP: {<DT>?<JJ>\*<NN>}"

cp = nltk.RegexpParser(grammar) result = cp.parse(sentence) print(result)

result.draw()

## OUTPUT :



**RESULT:**

The chunking code has been executed and verified using Python and NLTK

EX.NO:10 DATE: 22/03/24

# BUILDING CHUNKERS

## AIM:

To implement Building Chunkers code using Python and NLTK

## ALGORITHM:

STEP1: Start the program

STEP2: Download Anaconda version 3.9 STEP3: Click Environment

STEP4: Create new environment with name Tensorflow and click Create STEP5: Replace not installed option with installed

STEP6: Search tensorflow package and apply

STEP7: Repeat Step5, search Keras and NLTK and apply STEP8: Go to home, click Tensorflow

STEP9: Install Spyder, Jupyter Notebook and launch

STEP10: Building Chunker code is generated and verified, by printing the result. STEP11: Stop the program.

## PROGRAM:

from nltk import pos\_tag from nltk import RegexpParser

text ="learn php from kaala42 and make study easy".split() print("After Split:",text)

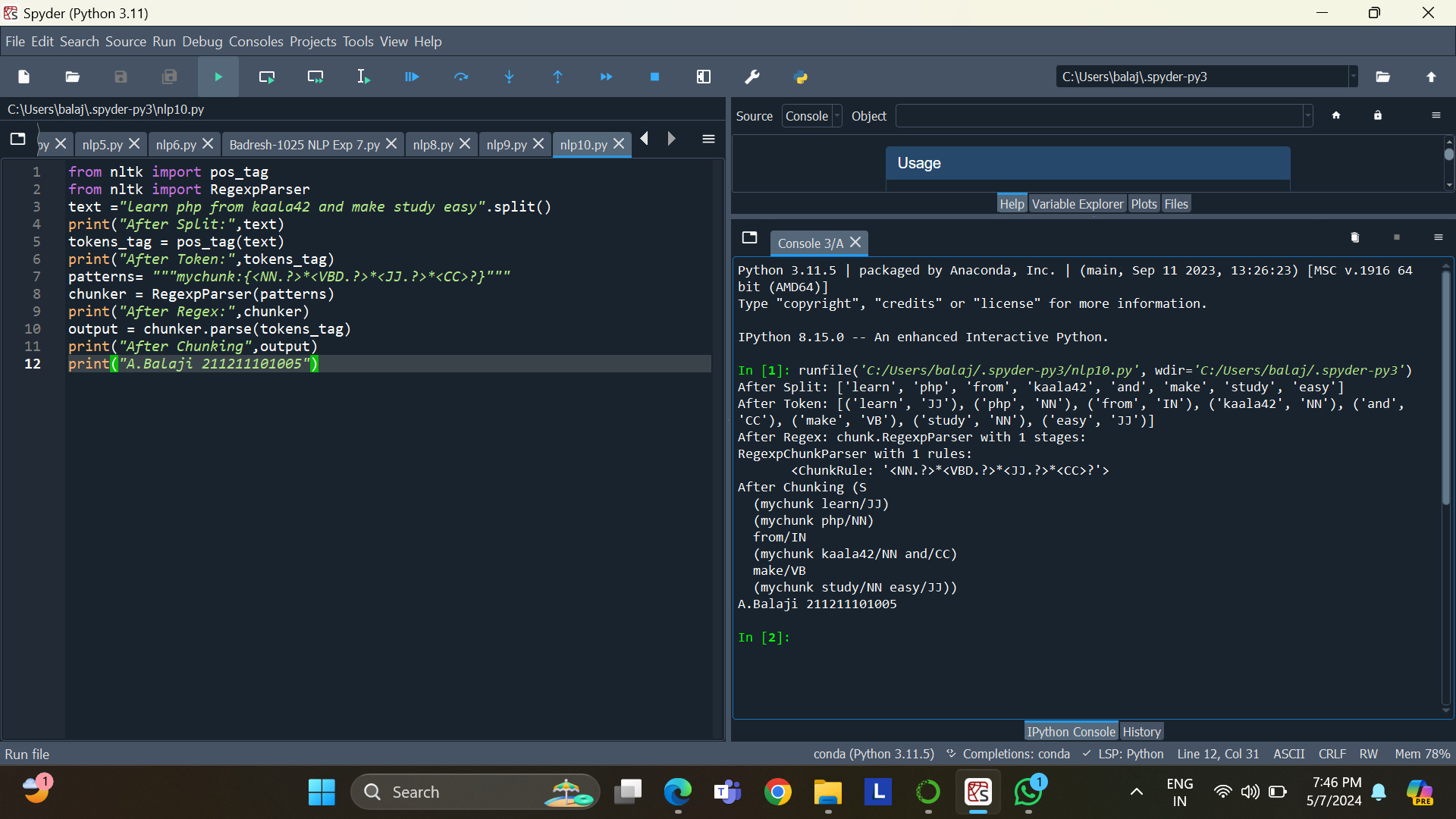
tokens\_tag = pos\_tag(text) print("After Token:",tokens\_tag)

patterns= """mychunk:{<NN.?>\*<VBD.?>\*<JJ.?>\*<CC>?}""" chunker

= RegexpParser(patterns)

print("After Regex:",chunker) output = chunker.parse(tokens\_tag) print("After Chunking",output)

## OUTPUT :

****

**RESULT:**

Thus, Building Chunkers code has been executed and verified using Python and NLTK.