Pratical no: 01

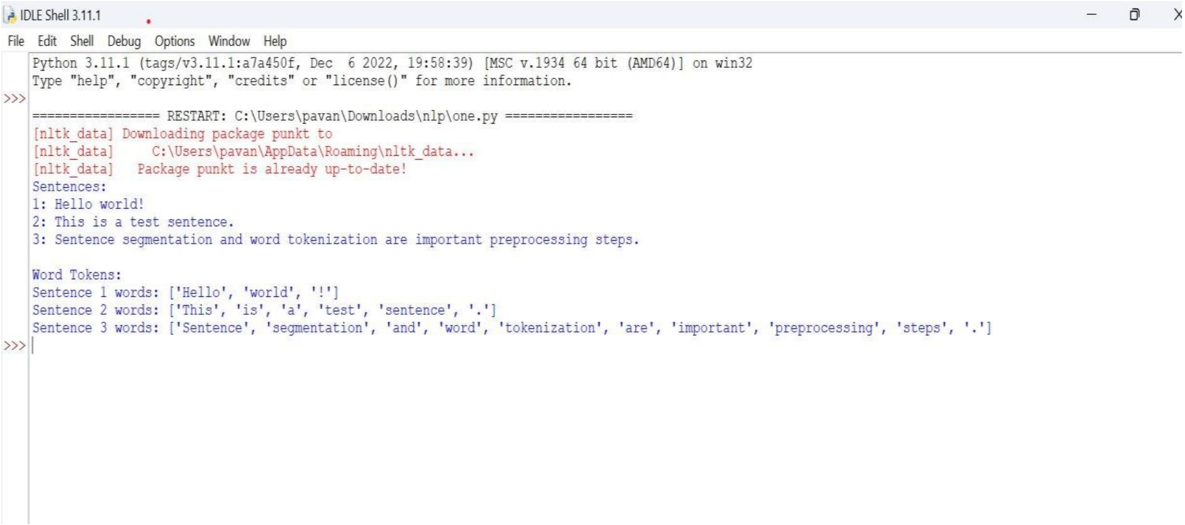
1. Write a program to implement sentence segmentation and word tokenization.

Input:

```
import nltk

from nltk.tokenize import sent_tokenize, word_tokenize

# Download the necessary resources nltk.download('punkt')

def segment_sentences(text):
    """
    Segment the input text into sentences.

    :param text: A string containing the text to be segmented.
    :return: A list of sentences.
    """
    sentences = sent_tokenize(text) return
    sentences

def tokenize_words(sentences): """
    Tokenize the input sentences into words.

    :param sentences: A list of sentences.
    :return: A list of lists, where each inner list contains the words of the corresponding
sentence. """
    word_tokens = [word_tokenize(sentence) for sentence in sentences] return
    word_tokens

if __name__ == "__main__": text = "Hello world! This is a test sentence. Sentence
    segmentation and word
tokenization are important preprocessing steps."

    # Segment the text into sentences

     sentences = segment_sentences(text)
    print("Sentences:") for i, sentence in
    enumerate(sentences):print(f"{i+1}:
    {sentence}")

    # Tokenize each sentence into words
    word_tokens = tokenize_words(sentences)
    print("\nWord Tokens:")
```

```
for i, words in enumerate(word_tokens):
 print(f"Sentence {i+1} words: {words}")
```

Output:



```
    Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    ================= RESTART: C:\Users\pavan\Downloads\nlp\one.py =================
    [nltk_data] Downloading package punkt to
    [nltk_data]     C:\Users\pavan\AppData\Roaming\nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
    Sentences:
    1: Hello world!
    2: This is a test sentence.
    3: Sentence segmentation and word tokenization are important preprocessing steps.

    Word Tokens:
    Sentence 1 words: ['Hello', 'world', '!']
    Sentence 2 words: ['This', 'is', 'a', 'test', 'sentence', '.']
    Sentence 3 words: ['Sentence', 'segmentation', 'and', 'word', 'tokenization', 'are', 'important', 'preprocessing', 'steps', '.']
>>>
```

Pratical no: 02

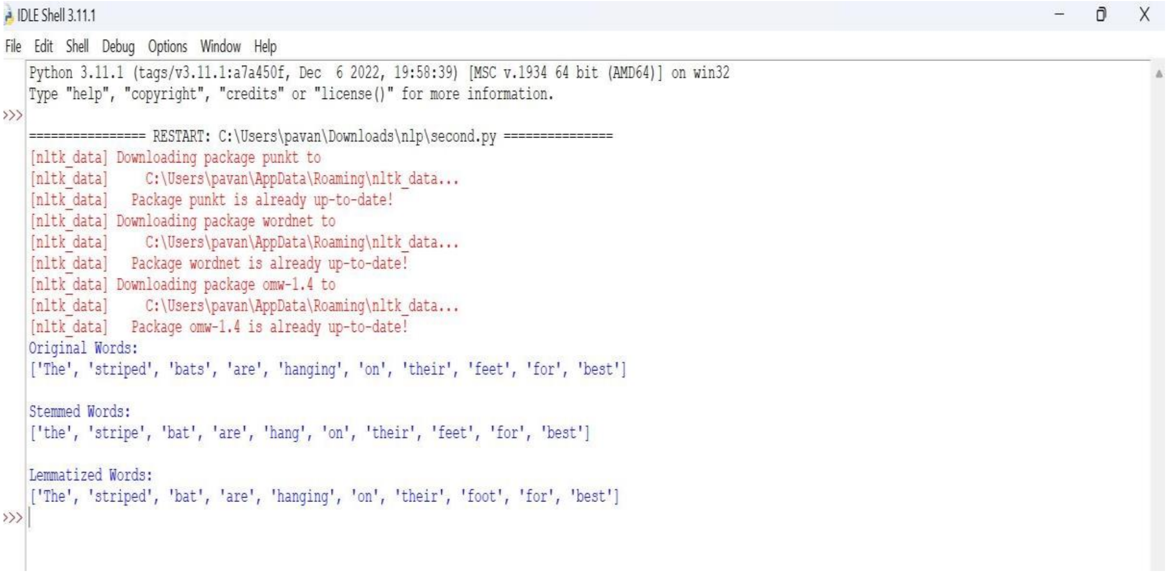
2. Write a program to implement stemming and lemmatization.

Input:

```
import nltk
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.tokenize import word_tokenize

# Download the necessary resources
nltk.download('punkt')
nltk.download('wordnet')
 nltk.download('omw-1.4')

def perform_stemming(words):
"""
    Perform stemming on the input words.

    :param words: A list of words to be stemmed.
    :return: A list of stemmed words.
    """
    stemmer = PorterStemmer()
    stemmed_words = [stemmer.stem(word) for word in words]

     return stemmed_words

def perform_lemmatization(words): """
    Perform lemmatization on the input words.

    :param words: A list of words to be lemmatized.
    :return: A list of lemmatized words.
    """
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]

     return lemmatized_words

if __name__ == "__main__": text = "The striped bats are hanging
    on their feet for best"

    # Tokenize the text into words words =
    word_tokenize(text)
    print("Original Words:")
    print(words)

    # Perform stemming
    stemmed_words = perform_stemming(words)
```

```
print("\nStemmed Words:") print(stemmed_words)

# Perform lemmatization

lemmatized_words = perform_lemmatization(words)
print("\nLemmatized Words:")
print(lemmatized_words)
```

Output:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\pavan\Downloads\nlp\second.py ===============
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\pavan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\pavan\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\pavan\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
Original Words:
['The', 'striped', 'bats', 'are', 'hanging', 'on', 'their', 'feet', 'for', 'best']

Stemmed Words:
['the', 'stripe', 'bat', 'are', 'hang', 'on', 'their', 'feet', 'for', 'best']

Lemmatized Words:
['The', 'striped', 'bat', 'are', 'hanging', 'on', 'their', 'foot', 'for', 'best']
>>>
```

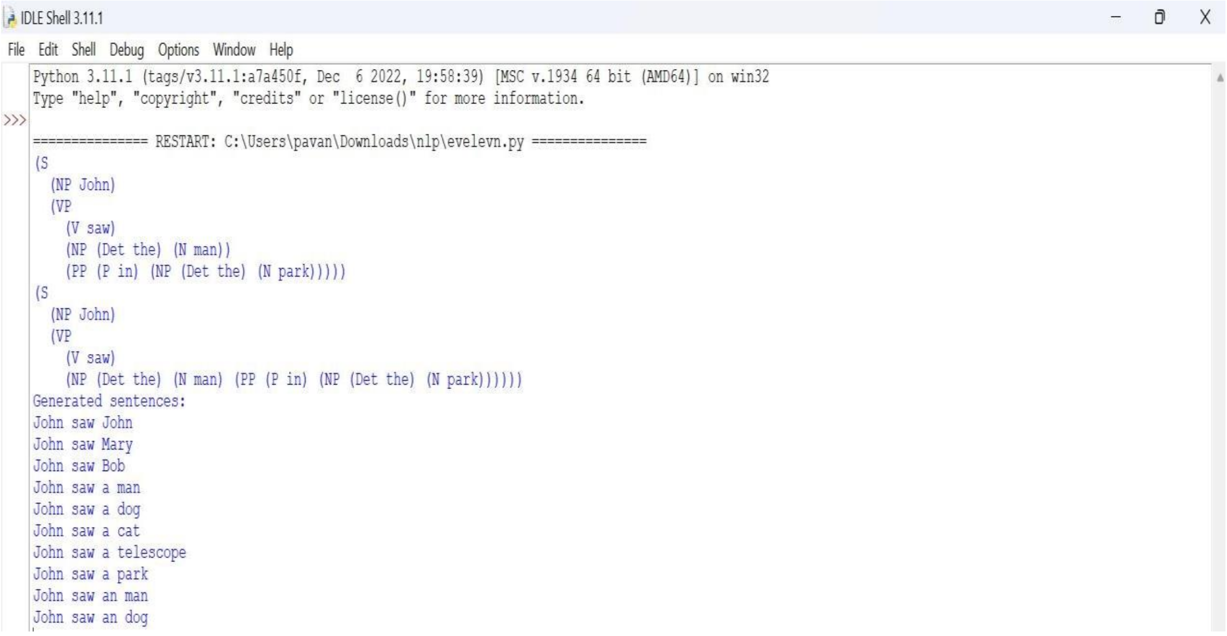
Pratical:03

3.Write a program to Implement syntactic parsing of a given text.

Input:

```
import nltk
from nltk import CFG
 from nltk.parse.generate import generate

# Define a simple grammar
grammar = CFG.fromstring("""
    S -> NP VP
    VP -> V NP | V NP PP
    PP -> P NP
    V -> "saw" | "ate" | "walked"
    NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
    Det -> "a" | "an" | "the" | "my"
    N -> "man" | "dog" | "cat" | "telescope" | "park"
   P -> "in" | "on" | "by" | "with"

     """)

# Create a parser

parser = nltk.ChartParser(grammar)

# Define a test sentence sentence = "John saw the man
in the park".split()

# Parse the sentence

 parses = list(parser.parse(sentence))

# Display the parse trees

for tree in parses:
 print(tree) tree.draw()

# If you want to generate all possible sentences according to the grammar
print("Generated sentences:") for sentence in generate(grammar, n=10): print('
'.join(sentence))
```

Output:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:\Users\pavan\Downloads\nlp\evelevn.py ===============
(S
  (NP John)
  (VP
    (V saw)
    (NP (Det the) (N man))
    (PP (P in) (NP (Det the) (N park)))))
(S
  (NP John)
  (VP
    (V saw)
    (NP (Det the) (N man) (PP (P in) (NP (Det the) (N park))))))
Generated sentences:
John saw John
John saw Mary
John saw Bob
John saw a man
John saw a dog
John saw a cat
John saw a telescope
John saw a park
John saw an man
John saw an dog
```

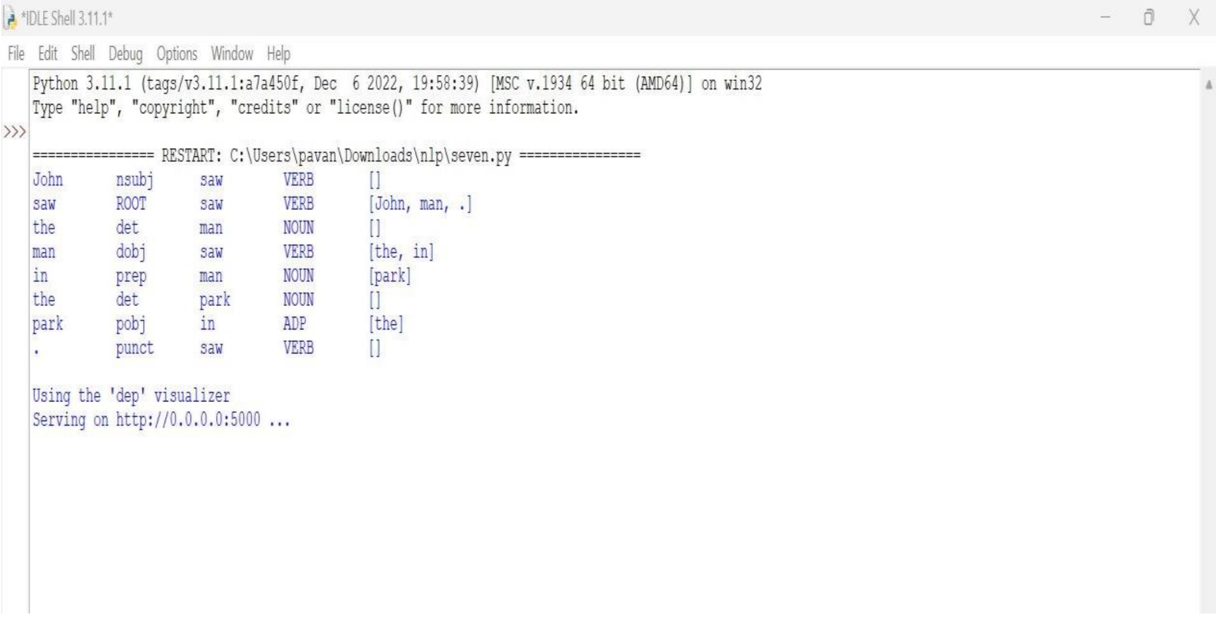
Pratical no:04

4.Write a program to Implement dependency parsing of a given text.

Input:

```
import spacy

# Load the pre-trained spaCy model
 nlp = spacy.load("en_core_web_sm")

# Define a test sentence
sentence = "John saw the man in the park."

# Parse the sentence
 doc = nlp(sentence)

# Display the syntactic structure
for token in doc:
      print(f"{token.text:10} {token.dep_:10} {token.head.text:10} {token.head.pos_:10}
{[child for child in token.children]}")

# Visualize the parse tree
 spacy.displacy.serve(doc, style="dep")
```

Output:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:\Users\pavan\Downloads\nlp\seven.py ===============
John       nsubj      saw        VERB       []
saw        ROOT       saw        VERB       [John, man, .]
the        det        man        NOUN       []
man        dobj       saw        VERB       [the, in]
in         prep       man        NOUN       [park]
the        det        park       NOUN       []
park       pobj       in         ADP        [the]
.          punct      saw        VERB       []

Using the 'dep' visualizer
Serving on http://0.0.0.0:5000 ...
```

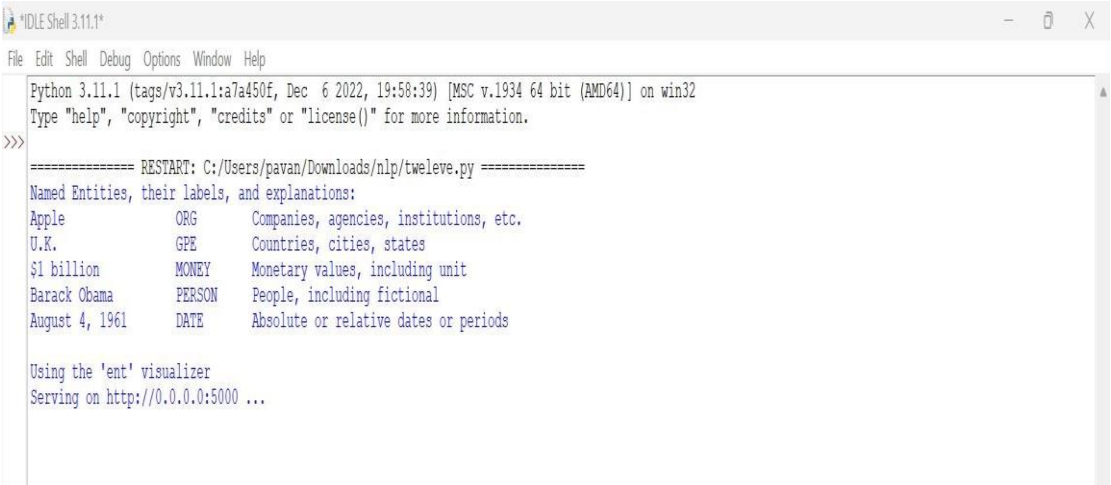
Practical:05

5.Write a program to Implement Named Entity Recognition (NER).

Input:

```
import spacy

# Load the pre-trained spaCy model

 nlp = spacy.load("en_core_web_sm")

# Define a test sentence
text = "Apple is looking at buying U.K. startup for $1 billion. Barack Obama was born on
August 4, 1961."

# Process the text
 doc = nlp(text)
# Display the named entities

 print("Named Entities, their labels, and explanations:")
for ent in doc.ents:
    print(f"{ent.text:20} {ent.label_:10} {spacy.explain(ent.label_)}")

# Visualize the named entities

 spacy.displacy.serve(doc, style="ent")
```

Output:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:/Users/pavan/Downloads/nlp/tweleve.py ===============
Named Entities, their labels, and explanations:
Apple                ORG        Companies, agencies, institutions, etc.
U.K.                 GPE        Countries, cities, states
$1 billion           MONEY      Monetary values, including unit
Barack Obama         PERSON     People, including fictional
August 4, 1961       DATE       Absolute or relative dates or periods

Using the 'ent' visualizer
Serving on http://0.0.0.0:5000 ...
```