Experiment No :- 07

Aim- Perform Chunking for the given text input.

Theory:-

Chunking, one of the important processes in natural language processing, is used to identify parts of speech (POS) and short phrases. In other simple words, with chunking, we can get the structure of the sentence. It is also called *partial parsing*.
Chunk extraction or partial parsing is a process of meaningfully extracting short phrases from the sentence (tagged with Part-of-Speech).

Chunks are made up of words and the kinds of words are defined using the part-of-speech tags. One can even define a pattern or words that can't be a part of chuck and such words are known as chinks. A ChunkRule class specifies what words or patterns to include and exclude in a chunk.

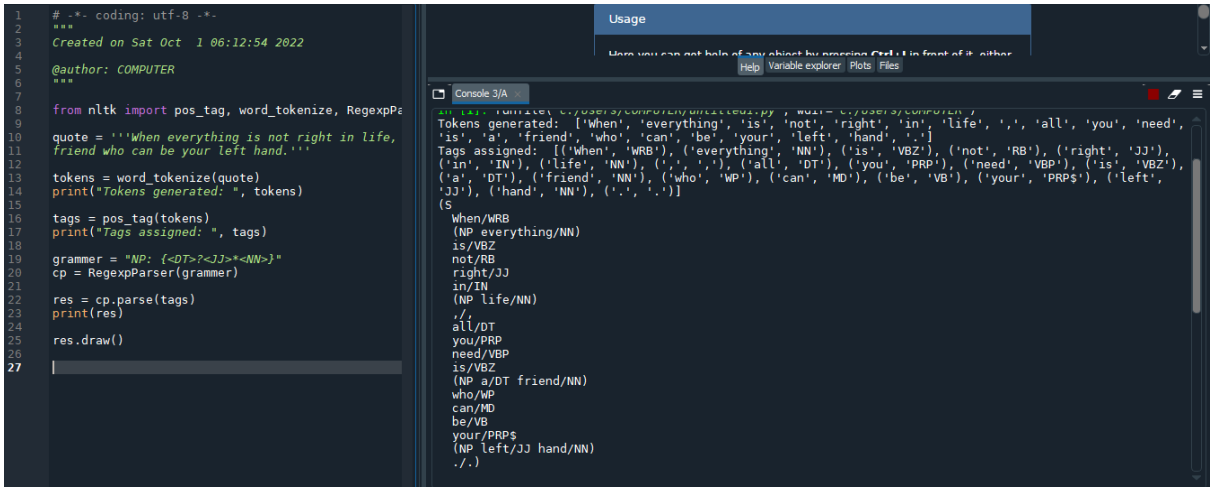
Defining Chunk patterns :

Chuck patterns are normal regular expressions which are modified and designed to match the part-of-speech tag designed to match sequences of part-of-speech tags. Angle brackets are used to specify an individual tag for example – *to match a noun tag*. One can define multiple tags in the same way.

Code

```
from nltk import pos_tag, word_tokenize, RegexpParser

quote = '''When everything is not right in life, all you need is a
friend who can be your left hand.'''

tokens = word_tokenize(quote)
print("Tokens generated: ", tokens)

tags = pos_tag(tokens)
print("Tags assigned: ", tags)

grammer = "NP: {<DT>?<JJ>*<NN>}"
cp = RegexpParser(grammer)

res = cp.parse(tags)
print(res)

res.draw()
```

Output



```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct  1 06:12:54 2022

@author: COMPUTER
"""

from nltk import pos_tag, word_tokenize, RegexpPa

quote = '''When everything is not right in life,
friend who can be your left hand.'''

tokens = word_tokenize(quote)
print("Tokens generated: ", tokens)

tags = pos_tag(tokens)
print("Tags assigned: ", tags)

grammer = "NP: {<DT>?<JJ>*<NN>}"
cp = RegexpParser(grammer)

res = cp.parse(tags)
print(res)

res.draw()
```

```
Usage

Help  Variable explorer  Plots  Files

Console 3/A

Tokens generated:  ['When', 'everything', 'is', 'not', 'right', 'in', 'life', ',', 'all', 'you', 'need',
'is', 'a', 'friend', 'who', 'can', 'be', 'your', 'left', 'hand', '.']
Tags assigned:  [('When', 'WRB'), ('everything', 'NN'), ('is', 'VBZ'), ('not', 'RB'), ('right', 'JJ'),
('in', 'IN'), ('life', 'NN'), (',', ','), ('all', 'DT'), ('you', 'PRP'), ('need', 'VBP'), ('is', 'VBZ'),
('a', 'DT'), ('friend', 'NN'), ('who', 'WP'), ('can', 'MD'), ('be', 'VB'), ('your', 'PRP$'), ('left',
'JJ'), ('hand', 'NN'), ('.', '.')]
(S
  When/WRB
  (NP everything/NN)
  is/VBZ
  not/RB
  right/JJ
  in/IN
  (NP life/NN)
  ,/,
  all/DT
  you/PRP
  need/VBP
  is/VBZ
  (NP a/DT friend/NN)
  who/WP
  can/MD
  be/VB
  your/PRP$
  (NP left/JJ hand/NN)
  ./.)
```

Conclusion:-

Thus ,we have successfully implemented Chunking for the given text input.