225229124

Exercise-1

```
In [1]: Sentence1="Rajkumar said on Monday that WASHINGTON -- In the wake of a string of ab
```

```
In [2]: from nltk.tokenize import word_tokenize
        from nltk.tag import pos_tag
        from nltk.chunk import ne_chunk
```

```
In [7]: tokens=word_tokenize(Sentence1)
        tags=pos_tag(tokens)
        ne_tree=ne_chunk(tags)
        print(ne_tree)
```

```
(S
  (PERSON Rajkumar/NNP)
  said/VBD
  on/IN
  Monday/NNP
  that/IN
  (ORGANIZATION WASHINGTON/NNP)
  --/:
  In/IN
  the/DT
  wake/NN
  of/IN
  a/DT
  string/NN
  of/IN
  abuses/NNS
  by/IN
  (GPE New/NNP York/NNP)
  police/NN
  officers/NNS
  in/IN
  the/DT
  1990s/CD
  ,/,
  (PERSON Loretta/NNP E./NNP Lynch/NNP)
  ,/,
  the/DT
  top/JJ
  federal/JJ
  prosecutor/NN
  in/IN
  (GPE Brooklyn/NNP)
  ,/,
  spoke/VBD
  forcefully/RB
  about/IN
  the/DT
  pain/NN
  of/IN
  a/DT
  broken/JJ
  trust/NN
  that/IN
  African-Americans/NNP
  felt/VBD
  and/CC
  said/VBD
  the/DT
  responsibility/NN
  for/IN
  repairing/VBG
  generations/NNS
  of/IN
  miscommunication/NN
  and/CC
  mistrust/NN
  tell/NN
  to/TO
  law/NN
  enforcement/NN
  ./.)
```

In [8]:
```python
import nltk
```

```
nltk.download('words')
```

```
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\8mpra\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Out[8]:
```
True
```

In [9]:
```
ne_tree=ne_chunk(pos_tag(word_tokenize(Sentence1)))
```

In [10]:
```
print(ne_tree)
```

```
(S
  (PERSON Rajkumar/NNP)
  said/VBD
  on/IN
  Monday/NNP
  that/IN
  (ORGANIZATION WASHINGTON/NNP)
  --/:
  In/IN
  the/DT
  wake/NN
  of/IN
  a/DT
  string/NN
  of/IN
  abuses/NNS
  by/IN
  (GPE New/NNP York/NNP)
  police/NN
  officers/NNS
  in/IN
  the/DT
  1990s/CD
  ,/,
  (PERSON Loretta/NNP E./NNP Lynch/NNP)
  ,/,
  the/DT
  top/JJ
  federal/JJ
  prosecutor/NN
  in/IN
  (GPE Brooklyn/NNP)
  ,/,
  spoke/VBD
  forcefully/RB
  about/IN
  the/DT
  pain/NN
  of/IN
  a/DT
  broken/JJ
  trust/NN
  that/IN
  African-Americans/NNP
  felt/VBD
  and/CC
  said/VBD
  the/DT
  responsibility/NN
  for/IN
  repairing/VBG
  generations/NNS
  of/IN
  miscommunication/NN
  and/CC
  mistrust/NN
  tell/NN
  to/TO
  law/NN
  enforcement/NN
  ./.)
```

Question-1

Count and print the number of PERSON, LOCATION AND ORGANIZATION in the given sentence

In [11]:
```
person_entities = []
location_entities = []
organization_entities = []

for node in ne_tree:
    if isinstance(node, nltk.tree.Tree):
        if node.label() == "PERSON":
            person_entities.append(' '.join([child[0] for child in node]))
        elif node.label() == "LOCATION":
            location_entities.append(' '.join([child[0] for child in node]))
        elif node.label() == "ORGANIZATION":
            organization_entities.append(' '.join([child[0] for child in node]))

print("Number of PERSON entities:", len(person_entities))
print("PERSON entities:",person_entities)
print("\n")
print("Number of LOCATION entities:",len(location_entities))
print("LOCATION entities:",location_entities)
print("\n")
print("Number of ORGANIZATION entities:",len(organization_entities))
print("ORGANIZATION entities:",organization_entities)
```

```
Number of PERSON entities: 2
PERSON entities: ['Rajkumar', 'Loretta E. Lynch']


Number of LOCATION entities: 0
LOCATION entities: []


Number of ORGANIZATION entities: 1
ORGANIZATION entities: ['WASHINGTON']
```

Question-2

Observe the results. Does named enity,"police officers" get recognized ?

In [12]:
```
word = nltk.word_tokenize(Sentence1)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<NN><NNS>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.Tree)]
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'New York', 'police officers', 'Loretta E. Lynch', 'Bro
oklyn']
```

Write a regular expression patter to detect this. You will need nltk.RegexParser class to define pattern and parse terms to detect patterns.

In [13]:
```
grammar = "NP: {<DT><JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.Tree)]
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'the wake', 'a string', 'New York', 'Loretta E. Lynch',
'the top federal prosecutor', 'Brooklyn', 'the pain', 'a broken trust', 'the respo
nsibility']
```

Question-3

Does the named entity,"the top federal prosecutor" get recognized ?

In [14]:
```
parse = cp.parse(tags)
print(parse[:])
```

```
[('Rajkumar', 'NNP'), ('said', 'VBD'), ('on', 'IN'), ('Monday', 'NNP'), ('that',
'IN'), ('WASHINGTON', 'NNP'), ('--', ':'), ('In', 'IN'), Tree('NP', [('the', 'D
T'), ('wake', 'NN')]), ('of', 'IN'), Tree('NP', [('a', 'DT'), ('string', 'NN')]),
('of', 'IN'), ('abuses', 'NNS'), ('by', 'IN'), ('New', 'NNP'), ('York', 'NNP'),
('police', 'NN'), ('officers', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('1990s', 'C
D'), (',', ','), ('Loretta', 'NNP'), ('E.', 'NNP'), ('Lynch', 'NNP'), (',', ','),
Tree('NP', [('the', 'DT'), ('top', 'JJ'), ('federal', 'JJ'), ('prosecutor', 'N
N')]), ('in', 'IN'), ('Brooklyn', 'NNP'), (',', ','), ('spoke', 'VBD'), ('forceful
ly', 'RB'), ('about', 'IN'), Tree('NP', [('the', 'DT'), ('pain', 'NN')]), ('of',
'IN'), Tree('NP', [('a', 'DT'), ('broken', 'JJ'), ('trust', 'NN')]), ('that', 'I
N'), ('African-Americans', 'NNP'), ('felt', 'VBD'), ('and', 'CC'), ('said', 'VB
D'), Tree('NP', [('the', 'DT'), ('responsibility', 'NN')]), ('for', 'IN'), ('repai
ring', 'VBG'), ('generations', 'NNS'), ('of', 'IN'), ('miscommunication', 'NN'),
('and', 'CC'), ('mistrust', 'NN'), ('tell', 'NN'), ('to', 'TO'), ('law', 'NN'),
('enforcement', 'NN'), ('.', '.')]
```

Write a regular expression pattern to detect this.

In [15]:
```
grammar = "NP: {<DT><JACJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.Tree)
print (NE)
```

```
['Rajkumar', 'WASHINGTON', 'the wake', 'a string', 'New York', 'Loretta E. Lynch',
'Brooklyn', 'the pain', 'the responsibility']
```

Exercise-2

In [16]:
```
Sentence2="European authorities fined Google a record $5.1 billion on Wednesday for
```

Question-1

Observe the Output. Does your code recognize the NE shown in BOLD

In [17]:
```
token=word_tokenize(Sentence2)
tag=nltk.pos_tag(token)
ne_tree=ne_chunk(tag)
print(ne_tree[:])
```

```
[Tree('GPE', [('European', 'JJ')]), ('authorities', 'NNS'), ('fined', 'VBD'), Tree
('PERSON', [('Google', 'NNP')]), ('a', 'DT'), ('record', 'NN'), ('$', '$'), ('5.
1', 'CD'), ('billion', 'CD'), ('on', 'IN'), ('Wednesday', 'NNP'), ('for', 'IN'),
('abusing', 'VBG'), ('its', 'PRP$'), ('power', 'NN'), ('in', 'IN'), ('the', 'DT'),
('mobile', 'JJ'), ('phone', 'NN'), ('market', 'NN'), ('and', 'CC'), ('ordered', 'V
BD'), ('the', 'DT'), ('company', 'NN'), ('to', 'TO'), ('alter', 'VB'), ('its', 'PR
P$'), ('practices', 'NNS')]
```

Write a regular expression that recognizes the entity,"$5.1 billion"

In [18]:
```
word = nltk.word_tokenize(Sentence2)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<CD>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.Tree)
print (NE)
```

```
['European', 'Google', '5.1', 'billion']
```

Question-2

Write a regular expression that recognizes the entity,"the mobile phone" and similar to this entity such as "the company"

In [19]:
```
word = nltk.word_tokenize(Sentence2)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<DT><JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(chunk)
NE = [ " ".join(w for w, t in ele) for ele in result if isinstance(ele, nltk.Tree)
print (NE)
```

```
['European', 'Google', 'a record', 'the mobile phone', 'the company']
```

Exercise-3

In this exercise, you will extract all ingredients from the food recipes text files, "food_recipes.txt".

In [24]:
```
f=open("food_recipes.txt")
s=f.readlines()
```

In [25]:
```
s
```

Out[25]:
```
['\t\n',
 'BEEF TENDERLOIN STEAKS WITH SMOKY BACON-BOURBON SAUCE\n',
 'Serves: 4\n',
 '\n',
 '1 1/2 cups dry red wine\n',
 '3 cloves garlic\n',
 '1 3/4 cups beef broth\n',
 '1 1/4 cups chicken broth\n',
 '1 1/2 tablespoons tomato paste\n',
 '1 bay leaf\n',
 '1 sprig thyme\n',
 '8 ounces bacon cut into 1/4 inch pieces\n',
 '1 tablespoon flour\n',
 '1 tablespoon butter\n',
 '4 1 inch rib-eye steaks\n',
 '1 tablespoon bourbon whiskey']
```

In [27]:
```
import re
with open("food_recipes.txt") as f:
    s = f.read()
bold_pattern = r'\*([^*]+)\*'
bold_words = re.findall(bold_pattern, s)
for word in bold_words:
    print(word)
```