NAME : RETHINAGIRI G

ROLL NO : 225229130

COURSE TITLE : NATURAL LANGUAGE PRE-PROCESSING LAB

LAB_10. Named Entity Recognition

Exercise-I

In [35]:
```
import nltk
from nltk.tokenize import word_tokenize as wt
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
```

In [2]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker.zip.
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

Out[2]:
```
True
```

In [3]:
```
sentence1="Rajkumar said on monday that WASHINGTON-- In the wake of a string of abuses by Newyork police officers in the
```

In [4]:
```
sentence1
```

Out[4]:
```
'Rajkumar said on monday that WASHINGTON-- In the wake of a string of abuses by Newyork police officers in the 1990s,
Loretta E.Lynch, the federal prosecutor in Brooklyn,spoke forcefully about the pain of broken trust that African-Ameri
cans felt and said the responsibility for repairing generations of miscommunuication and mistrust fell to law enforcem
ent'
```

In [5]:
```
tokens=wt(sentence1)
tags=pos_tag(tokens)
ne_tree=ne_chunk(tags)
print(ne_tree)
```

```
(S
  (PERSON Rajkumar/NNP)
  said/VBD
  on/IN
  monday/NN
  that/IN
  (ORGANIZATION WASHINGTON/NNP)
  --/:
  In/IN
  the/DT
  wake/NN
  of/IN
  a/DT
  string/NN
  of/IN
  abuses/NNS
  by/IN
  (ORGANIZATION Newyork/NNP)
  police/NNS
  officers/NNS
  in/IN
  the/DT
  1990s/CD
  ,/,
  (PERSON Loretta/NNP E.Lynch/NNP)
  ,/,
  the/DT
  federal/JJ
  prosecutor/NN
  in/IN
  (GPE Brooklyn/NNP)
  ,/,
  spoke/VBD
  forcefully/RB
  about/IN
  the/DT
  pain/NN
  of/IN
  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
miscommunuication/NN
and/CC
mistrust/NN
fell/VBD
to/TO
law/NN
enforcement/NN)
```

In [6]:
```
ne_tree=ne_chunk(pos_tag(wt(sentence1)))
```

In [7]:
```
for i in  ne_tree:
    print(i)
```

```
(PERSON Rajkumar/NNP)
('said', 'VBD')
('on', 'IN')
('monday', 'NN')
('that', 'IN')
(ORGANIZATION WASHINGTON/NNP)
('--', ':')
('In', 'IN')
('the', 'DT')
('wake', 'NN')
('of', 'IN')
('a', 'DT')
('string', 'NN')
('of', 'IN')
('abuses', 'NNS')
('by', 'IN')
(ORGANIZATION Newyork/NNP)
('police', 'NNS')
('officers', 'NNS')
('in', 'IN')
('the', 'DT')
('1990s', 'CD')
(',', ',')
(PERSON Loretta/NNP E.Lynch/NNP)
(',', ',')
('the', 'DT')
('federal', 'JJ')
('prosecutor', 'NN')
('in', 'IN')
(GPE Brooklyn/NNP)
(',', ',')
('spoke', 'VBD')
('forcefully', 'RB')
('about', 'IN')
('the', 'DT')
('pain', 'NN')
('of', 'IN')
('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')
('miscommunuication', 'NN')
('and', 'CC')
('mistrust', 'NN')
('fell', 'VBD')
('to', 'TO')
('law', 'NN')
('enforcement', 'NN')
```

Questions

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

In [8]:
```
from collections import Counter
for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sentence1))):
    if hasattr(chunk,"label"):
        print([Counter(label)for label in chunk])
```

```
[Counter({'Rajkumar': 1, 'NNP': 1})]
[Counter({'WASHINGTON': 1, 'NNP': 1})]
[Counter({'Newyork': 1, 'NNP': 1})]
[Counter({'Loretta': 1, 'NNP': 1}), Counter({'E.Lynch': 1, 'NNP': 1})]
[Counter({'Brooklyn': 1, 'NNP': 1})]
```

2.Does named entity,"police officers" grt recognized?

In [9]:
```
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', 'Newyork', 'Loretta E.Lynch', 'Brooklyn']
```

'write a regular expression patter to detect this.you will need nltk.regexpparser class to define pattern and parse terms to detect patterns'

In [10]:
```
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', 'Newyork', 'Loretta E.Lynch', 'the federal prosecutor', 'Brooklyn',
'the pain', 'the responsibility']
```

3: Does the named entity ,'The Top Federal Prosecutor' get recognized?

In [11]:
```
parse = cp.parse(tags)
print(parse[:])
```

```
[('Rajkumar', 'NNP'), ('said', 'VBD'), ('on', 'IN'), ('monday', 'NN'), ('that', 'IN'), ('WASHINGTON', 'NNP'), ('--',
':'), ('In', 'IN'), Tree('NP', [('the', 'DT'), ('wake', 'NN')]), ('of', 'IN'), Tree('NP', [('a', 'DT'), ('string', 'N
N')]), ('of', 'IN'), ('abuses', 'NNS'), ('by', 'IN'), ('Newyork', 'NNP'), ('police', 'NNS'), ('officers', 'NNS'), ('i
n', 'IN'), ('the', 'DT'), ('1990s', 'CD'), (',', ','), ('Loretta', 'NNP'), ('E.Lynch', 'NNP'), (',', ','), Tree('NP',
[('the', 'DT'), ('federal', 'JJ'), ('prosecutor', 'NN')]), ('in', 'IN'), ('Brooklyn', 'NNP'), (',', ','), ('spoke', 'V
BD'), ('forcefully', 'RB'), ('about', 'IN'), Tree('NP', [('the', 'DT'), ('pain', 'NN')]), ('of', 'IN'), ('broken', 'J
J'), ('trust', 'NN'), ('that', 'IN'), ('African-Americans', 'NNP'), ('felt', 'VBD'), ('and', 'CC'), ('said', 'VBD'), T
ree('NP', [('the', 'DT'), ('responsibility', 'NN')]), ('for', 'IN'), ('repairing', 'VBG'), ('generations', 'NNS'), ('o
f', 'IN'), ('miscommunuication', 'NN'), ('and', 'CC'), ('mistrust', 'NN'), ('fell', 'VBD'), ('to', 'TO'), ('law', 'N
N'), ('enforcement', 'NN')]
```

Write a regular expression pattern to detect this

In [12]:
```
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', 'Newyork', 'Loretta E.Lynch', 'Brooklyn', 'the pain', 'the responsi
bility']
```

Exercise-II

Extract All Named Entities From The Following Text:

In [13]:
```
sentence2 = "European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in the mobile ph
```

In [14]:
```
sentence2
```

Out[14]:
```
'European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in the mobile phone market
and ordered the company to alter its practices'
```

1: Observe The Output. Does Your Code Recognize The NE Showns in Bold?

In [15]:
```
token=wt(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'), ('fo
r', 'IN'), ('abusing', 'VBG'), ('its', 'PRP$'), ('power', 'NN'), ('in', 'IN'), ('the', 'DT'), ('mobile', 'JJ'), ('phon
e', 'NN'), ('market', 'NN'), ('and', 'CC'), ('ordered', 'VBD'), ('the', 'DT'), ('company', 'NN'), ('to', 'TO'), ('alte
r', 'VB'), ('its', 'PRP$'), ('practices', 'NNS')]
```

'write a regular expression that recognizes the entity,$ 5.1 billion detect andprint this'

In [16]:
```
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 [17]:
```
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-III

BEEF TENDERLOIN STEAKS WITH SMOKY BACON-BORBON SAUCE

In [29]:
```
with open("recipe.txt", "w") as file:
    file.write("1 1/2 cups dry red wine\n")
    file.write("3 cloves garlic\n")
    file.write("1 3/4 cups beef broth\n")
    file.write("1 1/4 cups chicken broth\n")
    file.write("1 1/2 tablespoons tomato paste\n")
    file.write("1 bay leaf\n")
    file.write("1 sprig thyme\n")
    file.write("8 ounces bacon cut into 1/4 inch pieces\n")
    file.write("1 tablespoon flour\n")
    file.write("1 tablespoon butter\n")
    file.write("4 1 inch rib-eye steaks\n")
    file.write("1 tablespoon bourbon whiskey\n")
    file.close()
```

In [33]:
```
f=open("recipe.txt", "r")
text=f.read()
print(text)
```

```
1 1/2 cups dry red wine
3 cloves garlic
1 3/4 cups beef broth
1 1/4 cups chicken broth
1 1/2 tablespoons tomato paste
1 bay leaf
1 sprig thyme
8 ounces bacon cut into 1/4 inch pieces
1 tablespoon flour
1 tablespoon butter
4 1 inch rib-eye steaks
1 tablespoon bourbon whiskey
```

In [36]:
```
tokens=wt(text)
tags=pos_tag(tokens)
ne_tree=ne_chunk(tags)
print(ne_tree[:])
```

```
[('1', 'CD'), ('1/2', 'CD'), ('cups', 'NNS'), ('dry', 'JJ'), ('red', 'JJ'), ('wine', 'NN'), ('3', 'CD'), ('cloves', 'N
NS'), ('garlic', 'JJ'), ('1', 'CD'), ('3/4', 'CD'), ('cups', 'NNS'), ('beef', 'VBD'), ('broth', 'DT'), ('1', 'CD'),
('1/4', 'CD'), ('cups', 'NNS'), ('chicken', 'VBP'), ('broth', 'DT'), ('1', 'CD'), ('1/2', 'CD'), ('tablespoons', 'NN
S'), ('tomato', 'VBP'), ('paste', 'NN'), ('1', 'CD'), ('bay', 'NN'), ('leaf', 'NN'), ('1', 'CD'), ('sprig', 'NN'), ('t
hyme', 'NN'), ('8', 'CD'), ('ounces', 'NNS'), ('bacon', 'JJ'), ('cut', 'VBD'), ('into', 'IN'), ('1/4', 'CD'), ('inch',
'NN'), ('pieces', 'NNS'), ('1', 'CD'), ('tablespoon', 'RB'), ('flour', 'JJ'), ('1', 'CD'), ('tablespoon', 'NN'), ('but
ter', 'NN'), ('4', 'CD'), ('1', 'CD'), ('inch', 'JJ'), ('rib-eye', 'JJ'), ('steaks', 'NNS'), ('1', 'CD'), ('tablespoo
n', 'NN'), ('bourbon', 'NN'), ('whiskey', 'NN')]
```

In [37]:
```
ne_tree=ne_chunk(pos_tag(wt(text)))
```

In [38]:
```
for i in  ne_tree:
    print(i)
```

```
('1', 'CD')
('1/2', 'CD')
('cups', 'NNS')
('dry', 'JJ')
('red', 'JJ')
('wine', 'NN')
('3', 'CD')
('cloves', 'NNS')
('garlic', 'JJ')
('1', 'CD')
('3/4', 'CD')
('cups', 'NNS')
('beef', 'VBD')
('broth', 'DT')
('1', 'CD')
('1/4', 'CD')
('cups', 'NNS')
('chicken', 'VBP')
('broth', 'DT')
('1', 'CD')
('1/2', 'CD')
('tablespoons', 'NNS')
('tomato', 'VBP')
('paste', 'NN')
('1', 'CD')
('bay', 'NN')
('leaf', 'NN')
('1', 'CD')
('sprig', 'NN')
('thyme', 'NN')
('8', 'CD')
('ounces', 'NNS')
('bacon', 'JJ')
('cut', 'VBD')
('into', 'IN')
('1/4', 'CD')
('inch', 'NN')
('pieces', 'NNS')
('1', 'CD')
('tablespoon', 'RB')
('flour', 'JJ')
('1', 'CD')
('tablespoon', 'NN')
('butter', 'NN')
('4', 'CD')
('1', 'CD')
('inch', 'JJ')
```

```
('rib-eye', 'JJ')
('steaks', 'NNS')
('1', 'CD')
('tablespoon', 'NN')
('bourbon', 'NN')
('whiskey', 'NN')
```

In [39]:
```
from collections import Counter
for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(text))):
    if hasattr(chunk,"label"):
        print([Counter(label)for label in chunk])
```

In [40]:
```
word = nltk.word_tokenize(text)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = """mychunk:{<NN.?>*<VBD.?>*<JJ.?>*<CC>?}"""
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)
```

```
['cups dry red', 'wine', 'cloves garlic', 'cups beef', 'cups', 'tablespoons', 'paste', 'bay leaf', 'sprig thyme', 'oun
ces bacon', 'cut', 'inch pieces', 'flour', 'tablespoon butter', 'inch rib-eye', 'steaks', 'tablespoon bourbon whiske
y']
```

In [41]:
```
word = nltk.word_tokenize(text)
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)
```

```
['dry red wine', 'paste', 'bay', 'leaf', 'sprig', 'thyme', 'inch', 'tablespoon', 'butter', 'tablespoon', 'bourbon', 'w
hiskey']
```