Exercise - 1

Extract all named entities?

```
~~Ar~~ import nltk
from nltk.tree import Tree
from.nltk. tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
nltk.download('punkt')
nltk.download('averaged_perception_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

Sentence 1 = "Rajkumar said on Monday that WASHINGTON -- In the wake of a string of abuses by New York police officers in the 1990s, loretta E. Lynch, the top trust that African-Americans felt and said the responsibility for repairing generations of miscommunications and mistrust fell to law enforcement."

```
tokens = word_tokenize(sentence 1)
  tags = pos_tag(tokens)
ne_tree = ne_chunk(tags)
  print(ne_tree[:])

ne_tree = ne_chunk(pos_tag(word_tokenize
                            (sentence1)))
  for i in ne_tree:
      print(i)
```

Natural Language Processing Lab
Lab10. Named Entity Recognition

In this lab, you will extract named entities from the given text file using NLTK. You will also recognize entities based on the regular expression patterns.

EXERCISE-1

Extract all named entities from the following text:

> Sentence1 = "Rajkumar said on Monday that WASHINGTON -- In the wake of a string of abuses by New York police officers in the 1990s, Loretta E. Lynch, the top federal prosecutor in Brooklyn, spoke forcefully about the pain of a broken trust that African-Americans felt and said the responsibility for repairing generations of miscommunication and mistrust fell to law enforcement."

Source Code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk

tokens = word_tokenize(sentence1)
tags = pos_tag(tokens)
ne_tree = ne_chunk(tags)
print(ne_tree)
```

You can create a pipeline too:

```
ne_tree = ne_chunk(pos_tag(word_tokenize(sentence1)))
```

Question-1

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

Question-2

- Observe the results. Does named entity, "police officers" get recognized?.
- Write a regular expression patter to detect this. You will need nltk.RegexpParser class to define pattern and parse terms to detect patterns.

Question-3

- Does the named entity, "the top federal prosecutor" get recognized?.
- Write a regular expression pattern to detect this.

EXERCISE-2

Extract all named entities from the following text:

> sentence2 = "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"

Question-1

Observe the output. Does your code recognize the NE shown in BOLD?

Question. 1

```
import nltk
from collections import counter.
for chunk in ne_tree:
    if hasattr(chunk, 'label'):
        print ([Counter(label) for label in chunk])
```

Question : 2

```
(1) word = nltk.word_tokenize(sentence 1)
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)
```

```
(2) grammar = "NP: {<NN> <NNS>}".
   cp = nltk . Regexpparser (grammar)
   result = . cp. parse (ne_tree)
     NE = [" . " join (w for W, t . in else) .for ele
                    in result if ~~is~~instance(ele,
                                   nltk.Tree)]
     print (NE)
```

Write a regular expression that recognizes the entity, "**$5.1 billion**"
Detect and print this

Question-2

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

EXERCISE-3

In this exercise, you will extract all ingredients from the food recipes text file, food_recipes.txt".
For example, the following text shows one food recipe.

BEEF TENDERLOIN STEAKS WITH SMOKY BACON-BOURBON SAUCE
Serves: 4

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**

The ingredients are highlighted with BOLD in the above list.

Extract all Named Entities from the text file and display them.

Reference: https://sites.google.com/site/anu3bis/recipes-main.

Question: 3

```
out = cp.parse(tags)
print(out [5])
grammar = "NP: {<DT><JJ>*<NN>}"
    cp = nltk.Regexpparser(grammar)
result = .cp.parse(ne_tree)
NE = [" ".join(w for w,t in ele).for ele in
                                result.if isinstance(ele,
                                nltk.Tree)]
print(NE)
```

Sentence 2 = " 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".

```
tok = word_tokenize(sentence2)
tagged = nltk.pos_tag(tok)
ne_tree2 = nltk.ne_chunk(tagged, binary=False)
print(ne_tree2[:])
```

Write a regular expression that recognizes the entity.

```
word = nltk.word_tokenize(sentence2)
pos_tag = nltk.tag(word)
chunk = nltk.ne_chunk(pos_tag)
grammar = "NP: {<CD>|<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)
```

<u>Question 2.</u>

```
word = nltk.tokenize(sentence2)
pos_tag = nltk.pos_tag(word)
chunk = nltk.ne_chunk(postag)
grammar = "NP: {<DT><JJ>*<NN>}"
cp = nltk.Regexp Parser(grammar)
```

```
result = cp.parse(chunk)
NE = [" ".join(w for w, t
        in ele) for ele in
     result if isinstance
     (ele, nltk.tree)]
print(NE)
```