|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Name:**    Natural Language processing Lab | | | | | | | | | **Course Code:** DLO8012 | |
| **Experiment No.:** 7 | | | | | | | | | | |
| **Experiment Title:** | | To perform Named Entity Recognition for sample input text. | | | | | | | | |
| **Name:** Hitesh Shikne | | | | | | **Roll No.:** 291 | | | | |
| **Year:** BE | | | | **Div:** B | | | **Batch:** B3 | | | |
| **Date of Performance:**   21 /03/2022 | | | | | | **Date of Submission:** 25/03/2022 | | | | |
| **Course /Lab Outcome:** | | |  | | | | | | | |
| **Assessment** | | | | | | | | | | |
| **Sr.**  **No.** | **Parameters for Assessment** | | | | **Marks** | **Rubrics** | | | |  |
| 1 | **Practical Performance / Active Participation (03 Marks)** | | | |  | Above Average  (03) | | Average  (02) | | Below Average  (01) |
| 2 | **Report Presentation**  **(02 Marks)** | | | |  | Above Average  (02) | | Average  (01) | | Below Average  (00) |
| 3 | **Understanding**  **(03 Marks)** | | | |  | Above Average  (03) | | Average  (02) | | Below Average  (01) |
| 4 | **Regularity in submission**  **(02 Marks)** | | | |  | Above Average  (02) | | Average  (01) | | Below Average  (00) |
| **Total Marks (10):** | | | | | | | | | |  |
| **Teacher’s Name & Signature:                                                                                   Date:** | | | | | | | | | |  |

**Aim:** To perform Named Entity Recognition for sample input text.

**Theory:**

[Named entity recognition](https://en.wikipedia.org/wiki/Named-entity_recognition) (NER)is probably the first step towards information extraction that seeks to locate and classify [named entities](https://en.wikipedia.org/wiki/Named_entity) in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. NER is used in many fields in [Natural Language Processing](https://en.wikipedia.org/wiki/Natural_language_processing) (NLP), and it can help answering many real-world questions, such as:

* Which companies were mentioned in the news article?
* Were specified products mentioned in complaints or reviews?
* Does the tweet contain the name of a person? Does the tweet contain this person’s location?

This article describes how to build named entity recognizer with [NLTK](https://www.nltk.org/book/ch07.html) and [SpaCy](https://spacy.io/usage/linguistic-features), to identify the names of things, such as persons, organizations, or locations in the raw text. Let’s get started!

**NLTK**

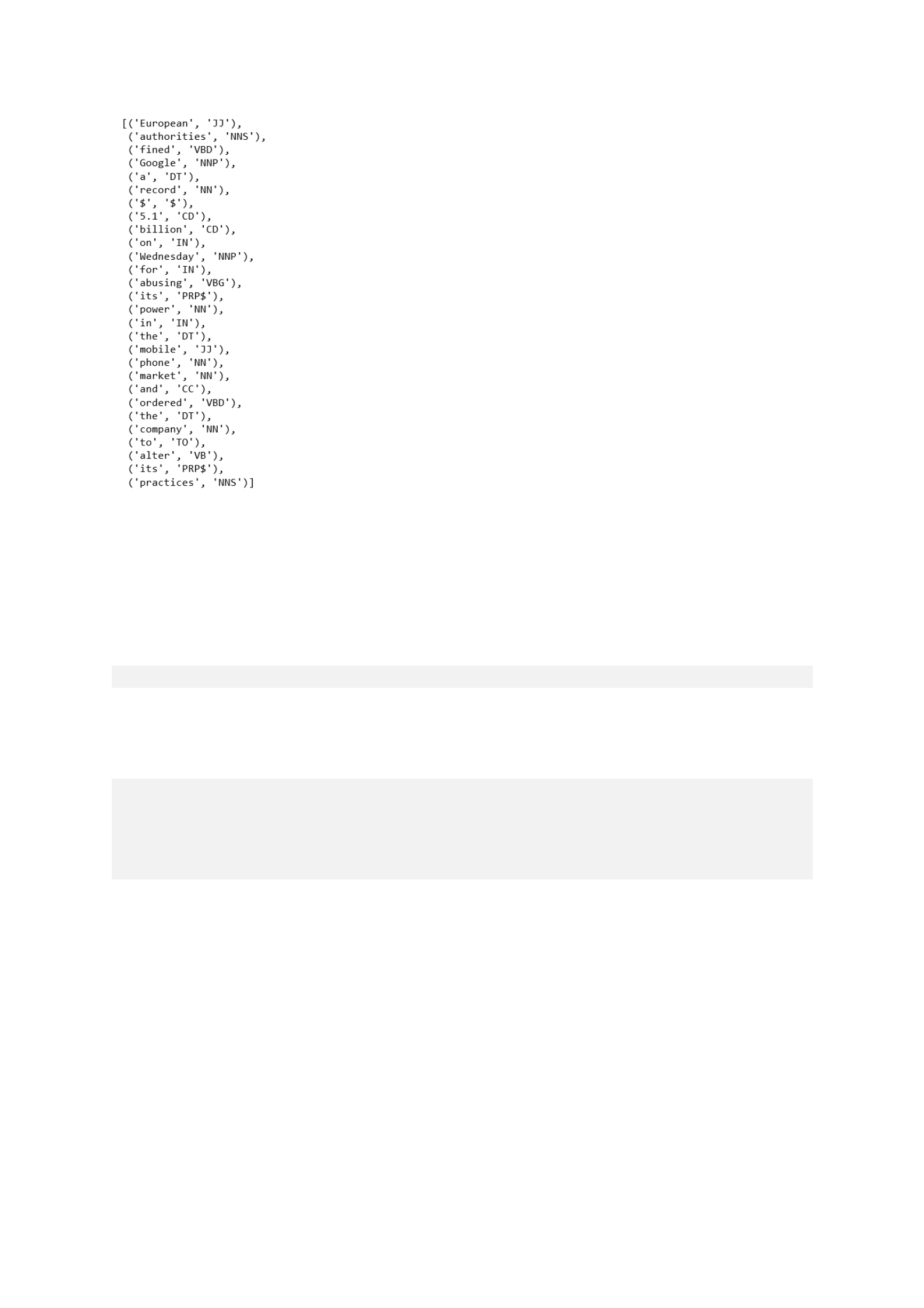
Import nltk

from nltk.tokenize import word\_tokenize

from nltk.tag import pos\_tag

* Information Extraction
* I took a sentence from [The New York Times](https://www.nytimes.com/2018/07/18/technology/google-eu-android-fine.html), “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.”
* ex = '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'
* Then we apply word tokenization and part-of-speech tagging to the sentence. defpreprocess(sent):

sent=nltk.word\_tokenize(sent) sent=nltk.pos\_tag(sent) return sent

****

* Let’s see what we get:

sent=preprocess(ex)

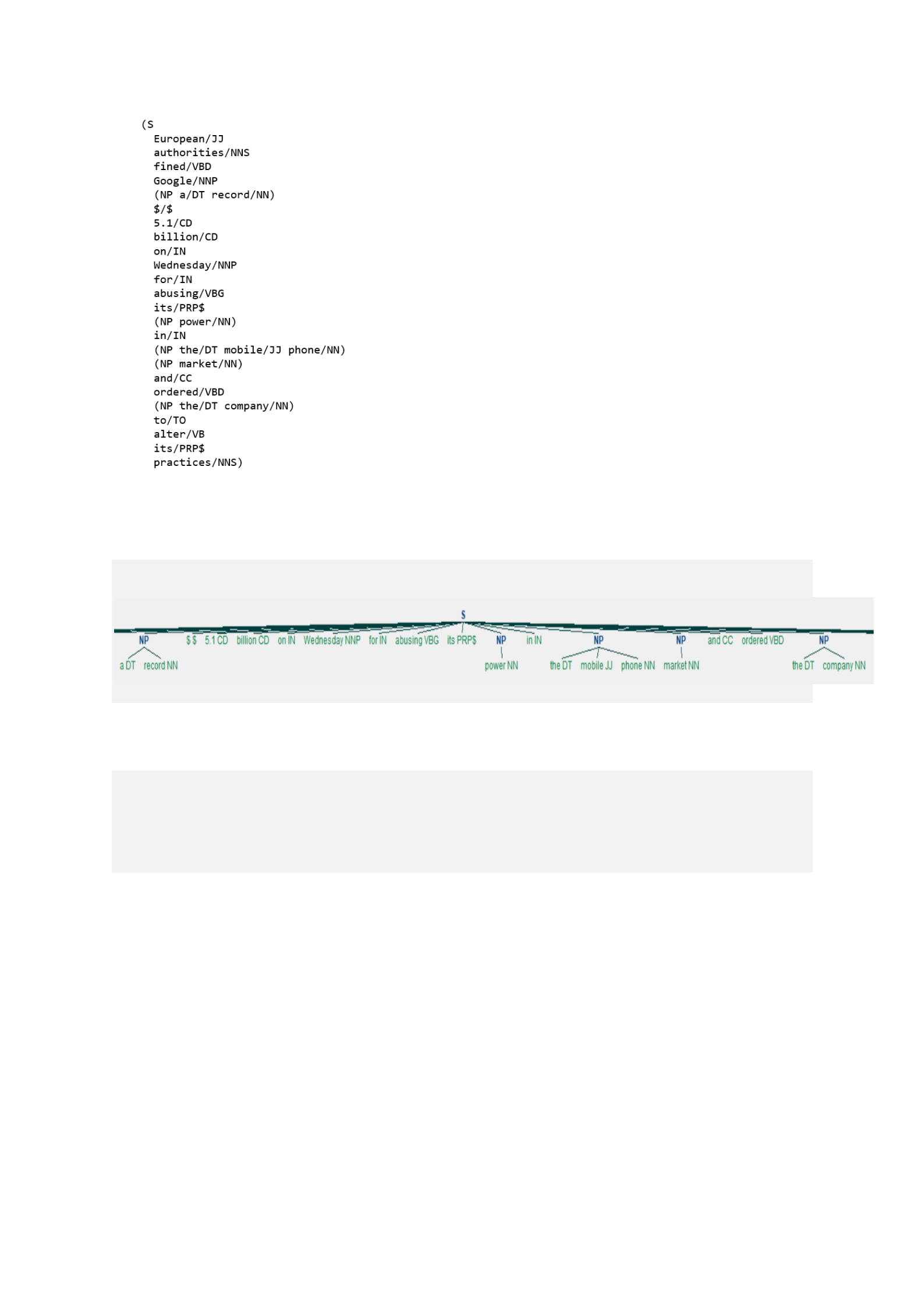
sent

We get a list of tuples containing the individual words in the sentence and their associated part-of-speech.Now we’ll implement noun phrase chunking to identify named entities using a regular expression consisting of rules that indicate how sentences should be chunked.Our chunk pattern consists of one rule, that a noun phrase, NP, should be formed whenever the chunker finds an optional determiner, DT, followed by any number of adjectives, JJ, and then a noun, NN.

pattern = 'NP: {<DT>?<JJ>\*<NN>}'

**Chunking**

Using this pattern, we create a chunk parser and test it on our sentence.

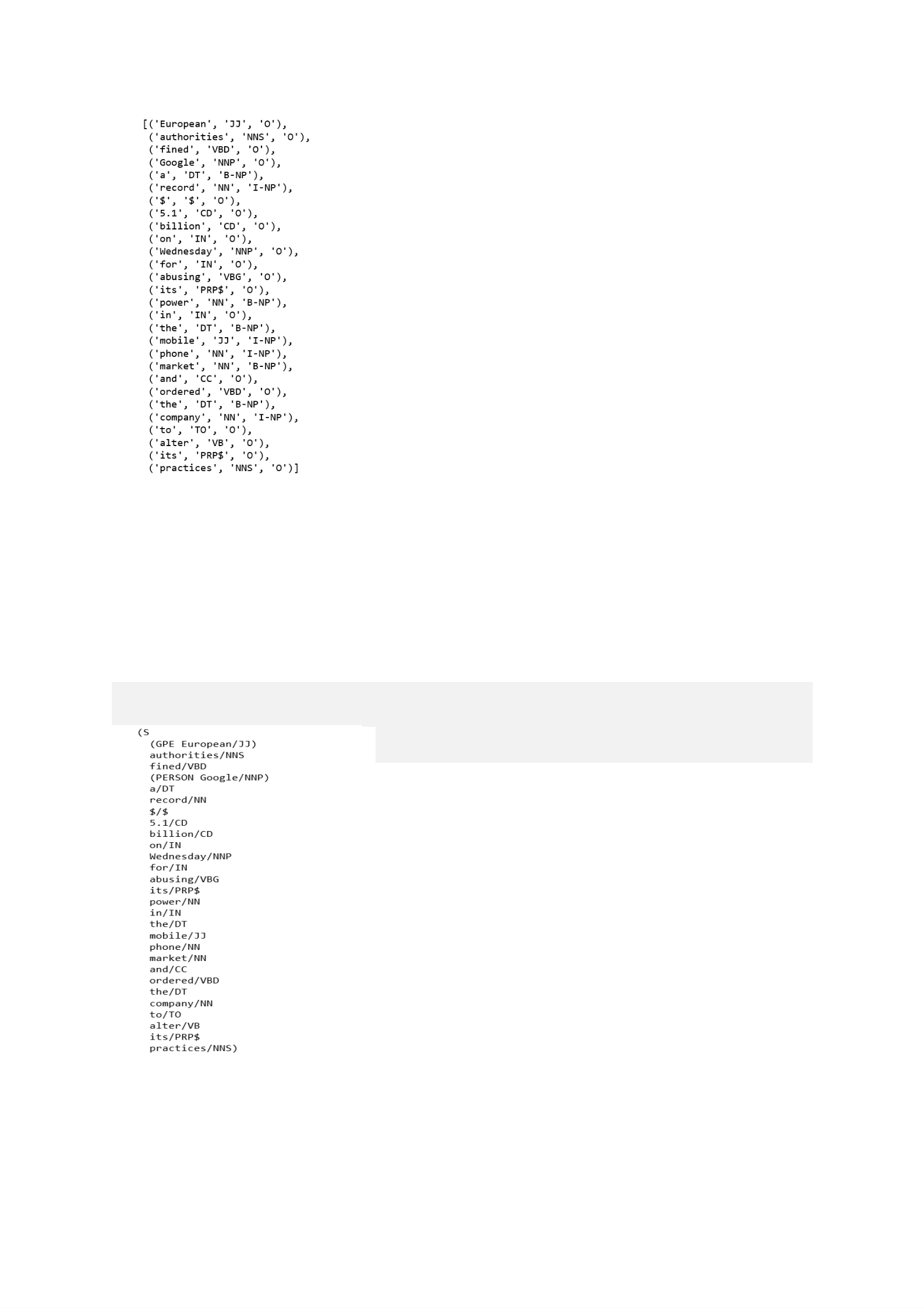
****cp=nltk.RegexpParser(pattern)

cs=cp.parse(sent)

print(cs)

The output can be read as a tree or a hierarchy with S as the first level, denoting sentence. we can also display it graphically.

IOB tags have become the standard way to represent chunk structures in files, and we will also be using this format.

****from nltk.chunk import conlltags2tree, tree2conlltags from pprint import pprintiob\_tagged = tree2conlltags(cs) pprint(iob\_tagged)

In this representation, there is one token per line, each with its part-of-speech tag and its named entity tag. Based on this training corpus, we can construct a tagger that can be used to label new sentences; and use the nltk.chunk.conlltags2tree() function to convert the tag sequences into a chunk tree.

With the function nltk.ne\_chunk(), we can recognize named entities using a classifier, the classifier adds category labels such as PERSON, ORGANIZATION, and GPE.

ne\_tree = ne\_chunk(pos\_tag(word\_tokenize(ex)))

print(ne\_tree)

**Program:**

import nltk

from nltk.tokenize import word\_tokenize

from nltk.tag import pos\_tag

ex = 'deepika, akshata ,abdul,aditya,bhushan and nikhil are my good friends.'

def preprocess(sent):

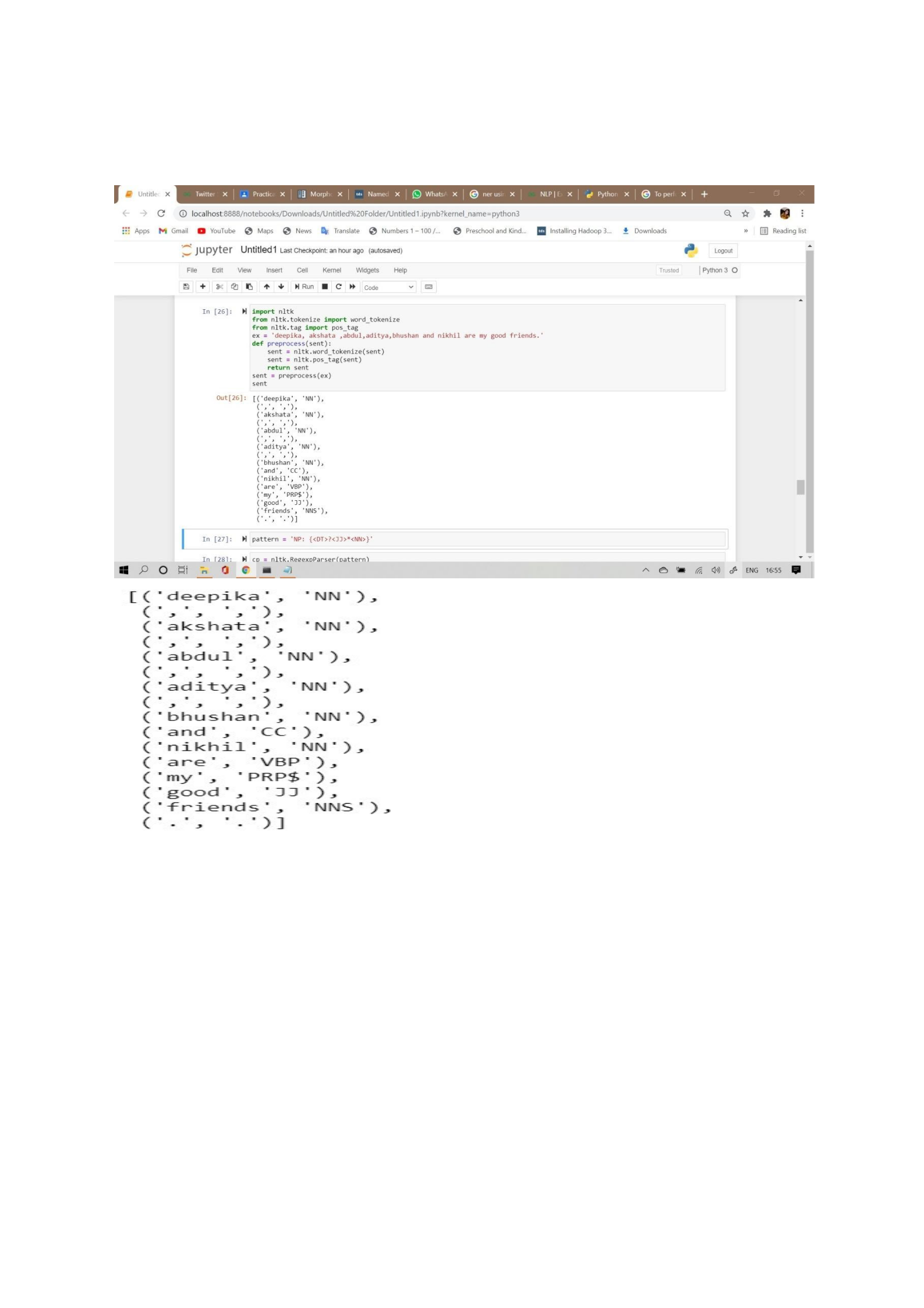
sent = nltk.word\_tokenize(sent)

sent = nltk.pos\_tag(sent)

return sent

sent = preprocess(ex)

sent

****

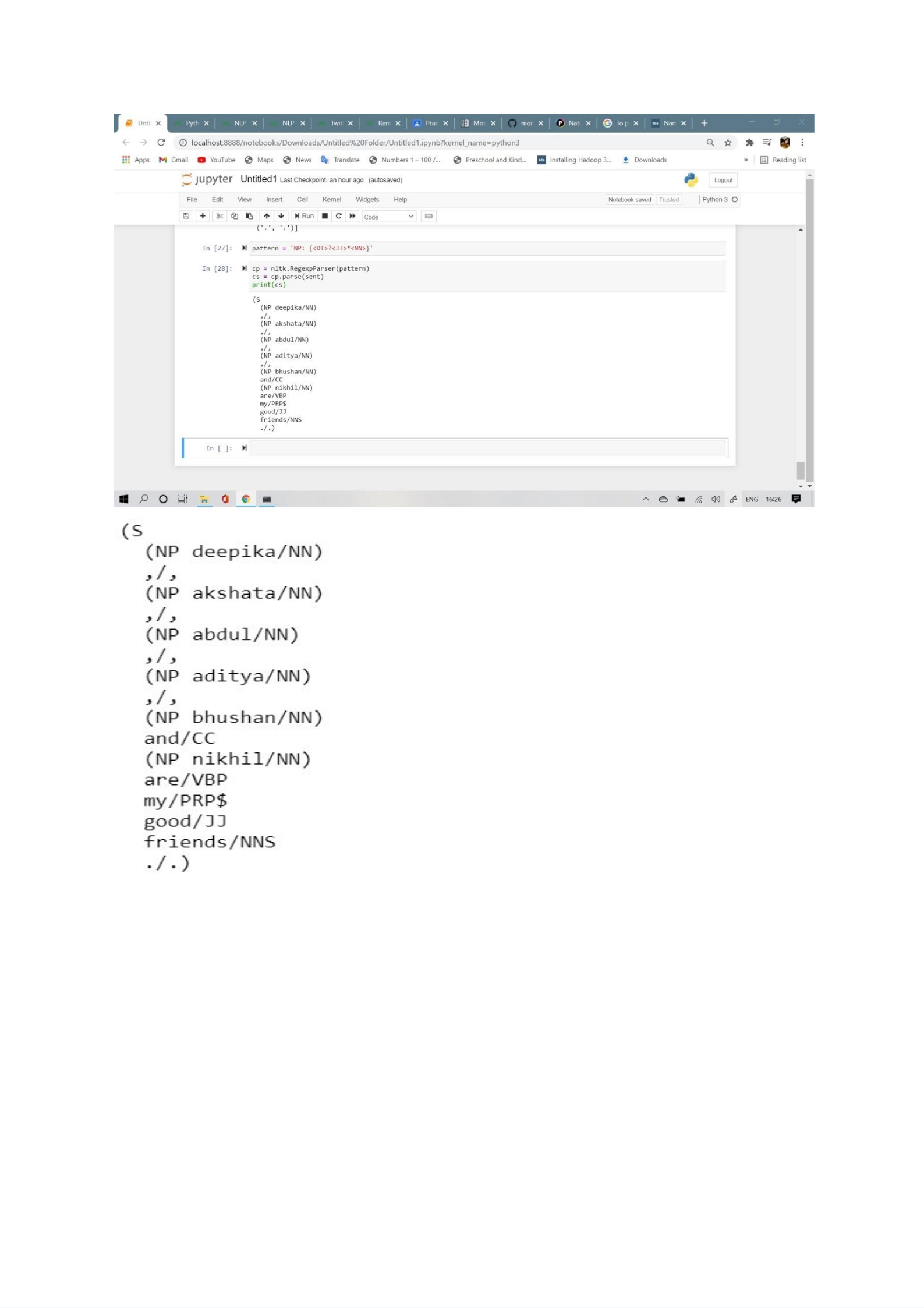
**Output:**

**Program:**

pattern = 'NP: {<DT>?<JJ>\*<NN>}'

cp = nltk.RegexpParser(pattern)

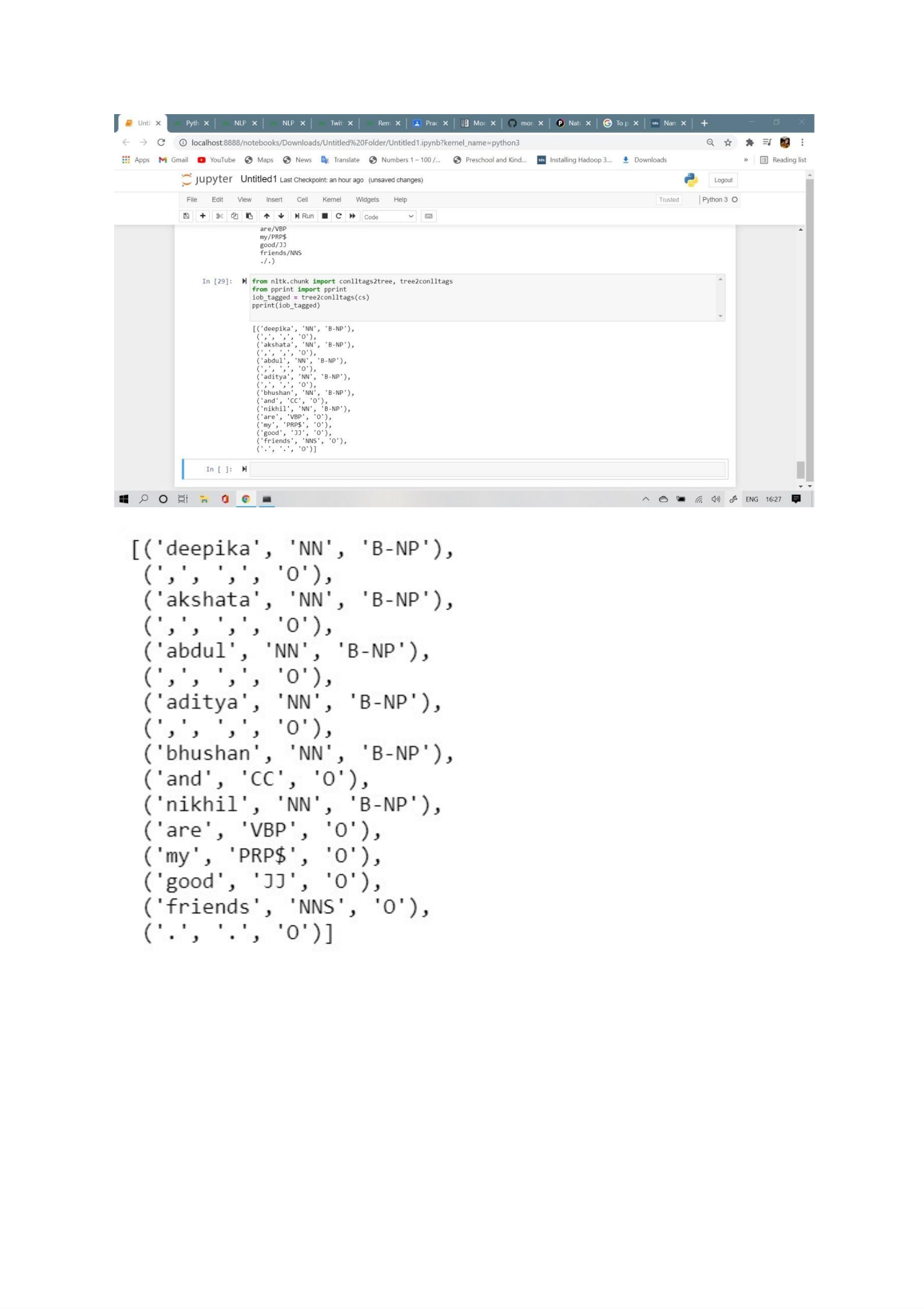
cs = cp.parse(sent)

****print(cs)

**Output:**

**Program:**

from nltk.chunk import conlltags2tree, tree2conlltags from print import print

****iob\_tagged = tree2conlltags(cs)

print(iob\_tagged)

**Output:**

**Conclusion**: We have successfully perform Named Entity Recognition for sample input text.