Name: Durva Kadam **Roll No.:23**

St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2022-2023
Class: BE-ITA/B, Semester: VII
Subject: Data Science Lab

Experiment – 4

1. **Aim:** To implement a Cognitive Computing Application

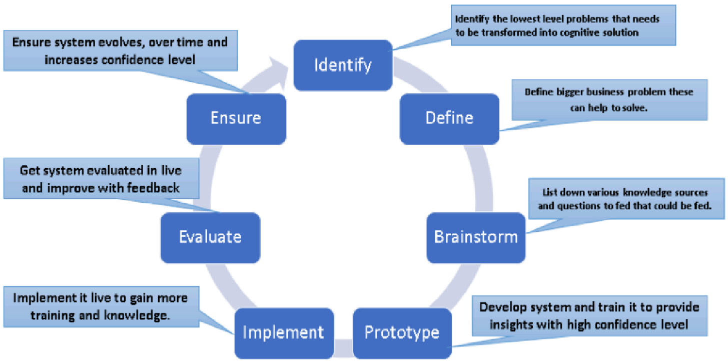
2. **Objectives:** Students should be able to design a solution for problem using Cognitive Computing.

3. **Prerequisite:** Python basics

4. **Requirements:** PC, Python 3.9, Windows 10/ MacOS/ Linux, IDLE IDE

5. **Pre-Experiment Exercise:**
Theory:
- The Cognitive is the mental action to learning and acquiring through thought, experience and the senses.
- Cognitive computing is computerized model that simulates human thought process in complex situations where the answer may be ambiguous and uncertain.
- Cognitive computing systems can recognize, understand, analyze, memorize and take out best possible result as or near about the human brain.
- The basic idea behind this type of computing is that to develop the computer system(include hardware and software) who interacts with human like humans.
- To accomplish this, cognitive computing makes use of AI and underlying technologies.
- If you look at cognitive computing as an analog to the human brain, you need to analyze in context all types of data, from structured data in databases to unstructured data in text, images, voice, sensors, and video.

Design Principles of Cognitive Computing:


Identify
Identify the lowest level problems that needs to be transformed into cognitive solution
Define
Define bigger business problem these can help to solve.
Brainstorm
List down various knowledge sources and questions to fed that could be fed.
Prototype
Develop system and train it to provide insights with high confidence level
Implement
Implement it live to gain more training and knowledge.
Evaluate
Get system evaluated in live and improve with feedback
Ensure
Ensure system evolves, over time and increases confidence level

Phases in NLP:

Phonological Analysis:
- It is applied if input is speech.

Morphological Analysis
- Deals with understanding distinct words according to their morphemes.
- Eg: Unhappiness: broken down into three morphemes (prefix, stem, suffix).
- Stem is considered as free morpheme and prefix and suffix are considered are

bound morphemes.

Lexical Analysis:

- Lexicon of a language means the collection of words and phrases in the language.
- Lexical analysis is dividing the whole chunk of text into paragraphs, sentences and words.
- Lexicon normalization is often needed in Lexical analysis.
- The most common lexicon normalization are:
    - Stemming: it is a rudimentary rule based process of stripping the suffixes. From word.
    - Lemmatization: organized procedure of obtaining the root form of the word by using dictionary and morphological analysis.

Syntactic Analysis:

- Deals with analyzing the words of a sentence so as to uncover the grammatical structure of the sentence.
- Eg: “Colorless green idea”
- Checked for dependency grammar and parts of speech tags .

Semantic Analysis:

- Determines possible meaning of the sentence by focusing on the interactions among word level meanings in the sentence.

Discourse Integration:

- Focuses on the properties of the text as a whole that convey meaning by making connections between component sentences.

Pragmatic Analysis:

- Explains how extra meaning is read into texts without actually being encoded in them.
- It helps user to discover intended effect by applying set of rules that characterize cooperative dialogues.

6. Laboratory Exercise

A. Procedure

i. Use google colab for programming.
ii. Import nltk package.
iii. Demonstrate all phases of NLP on a given text.
iv. Add relevant comments in your programs and execute the code. Test it for various cases.

7. Post-Experiments Exercise:

A. Extended Theory:

a. Explain design Principles of Cognitive Computing.

B. Post Lab Program:

a. Select a application of your choice in domain like health care, banking, finance and implement

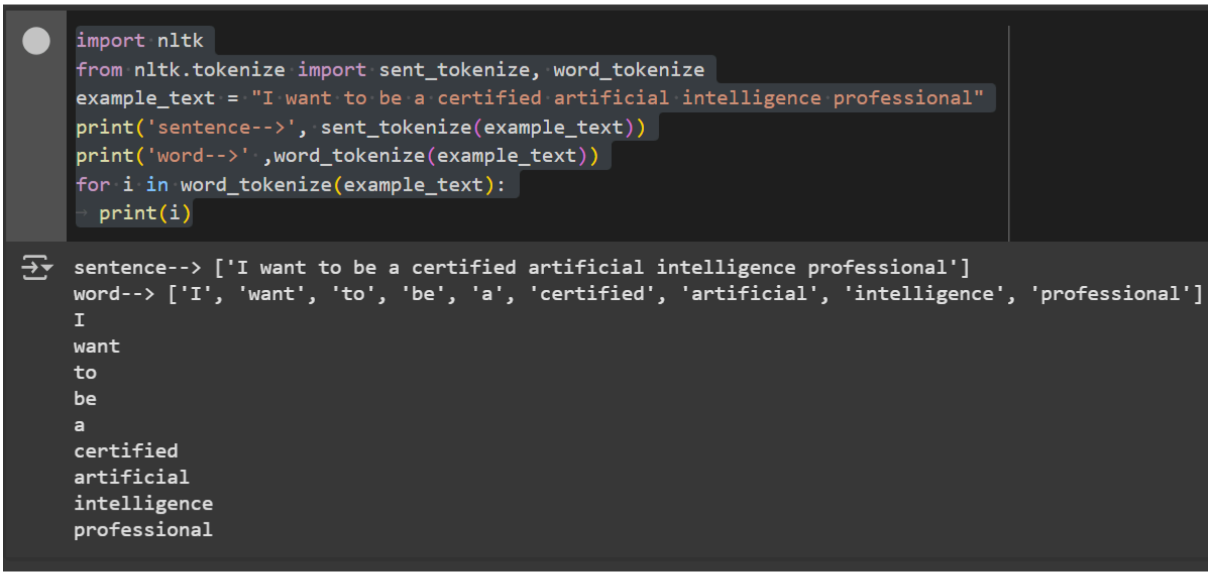
C. Conclusion:

1. Write what was performed in the program (s) .
2. What is the significance of program and what Objective is achieved?

D. References:

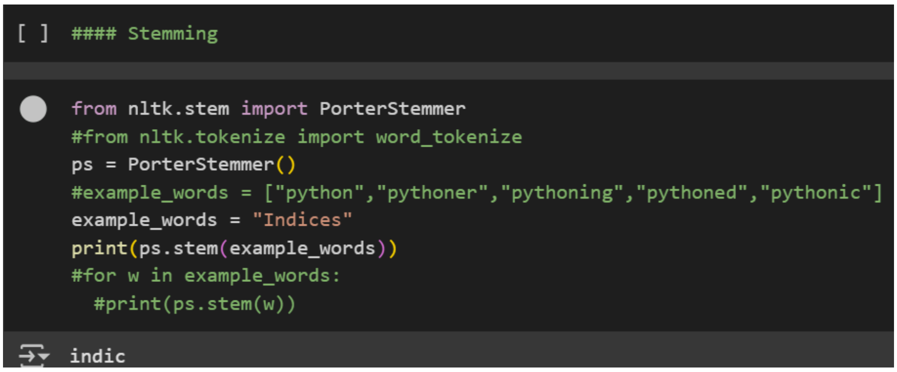
[1]Judith S. Hurwitz, Marcia Kaufman, Adrian Bowles, “Cognitive Computing and Big Data Analytics”, Wiley India, 2015

PRE EXPERIMENT EXPERIMENT:



```
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
example_text = "I want to be a certified artificial intelligence professional"
print('sentence-->', sent_tokenize(example_text))
print('word-->' ,word_tokenize(example_text))
for i in word_tokenize(example_text):
  print(i)
```

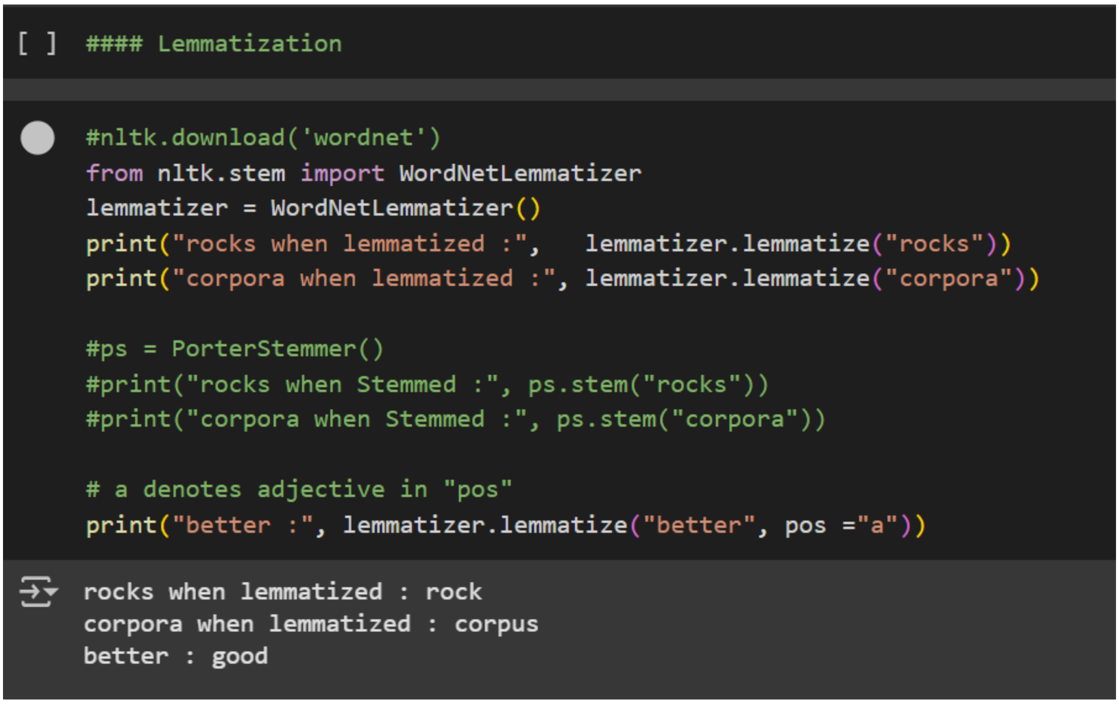
```
sentence--> ['I want to be a certified artificial intelligence professional']
word--> ['I', 'want', 'to', 'be', 'a', 'certified', 'artificial', 'intelligence', 'professional']
I
want
to
be
a
certified
artificial
intelligence
professional
```



```
[ ] #### Stemming
```

```
from nltk.stem import PorterStemmer
#from nltk.tokenize import word_tokenize
ps = PorterStemmer()
#example_words = ["python","pythoner","pythoning","pythoned","pythonic"]
example_words = "Indices"
print(ps.stem(example_words))
#for w in example_words:
  #print(ps.stem(w))
```

```
indic
```

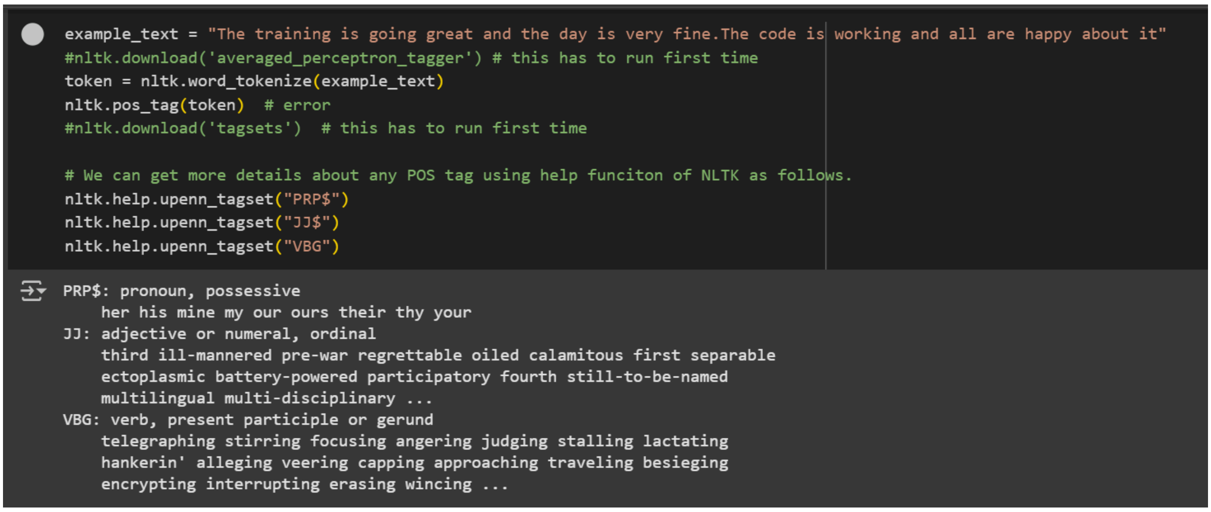


```
[ ] #### Lemmatization
```

```
#nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print("rocks when lemmatized :",   lemmatizer.lemmatize("rocks"))
print("corpora when lemmatized :", lemmatizer.lemmatize("corpora"))

#ps = PorterStemmer()
#print("rocks when Stemmed :", ps.stem("rocks"))
#print("corpora when Stemmed :", ps.stem("corpora"))

# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

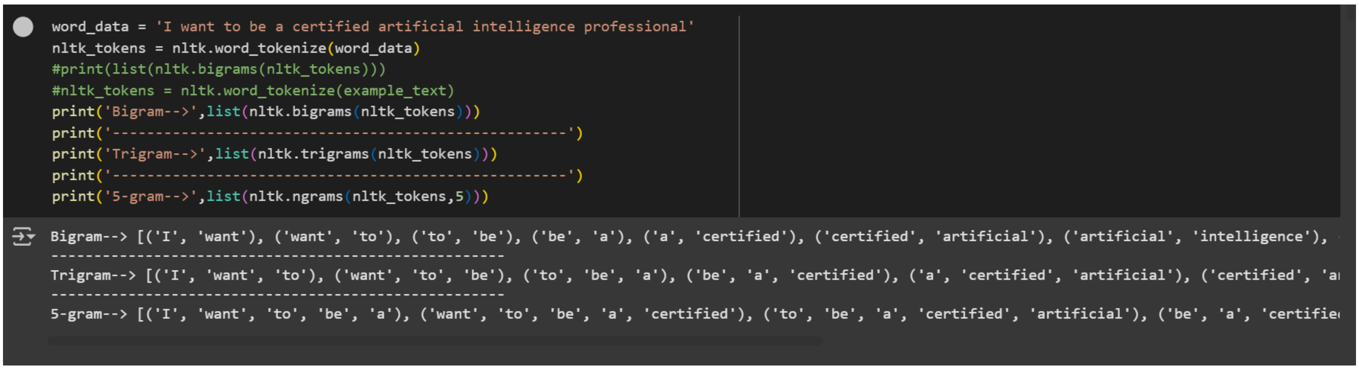
```
rocks when lemmatized : rock
corpora when lemmatized : corpus
better : good
```

Name: Shruti Kadam **Roll No.:24**



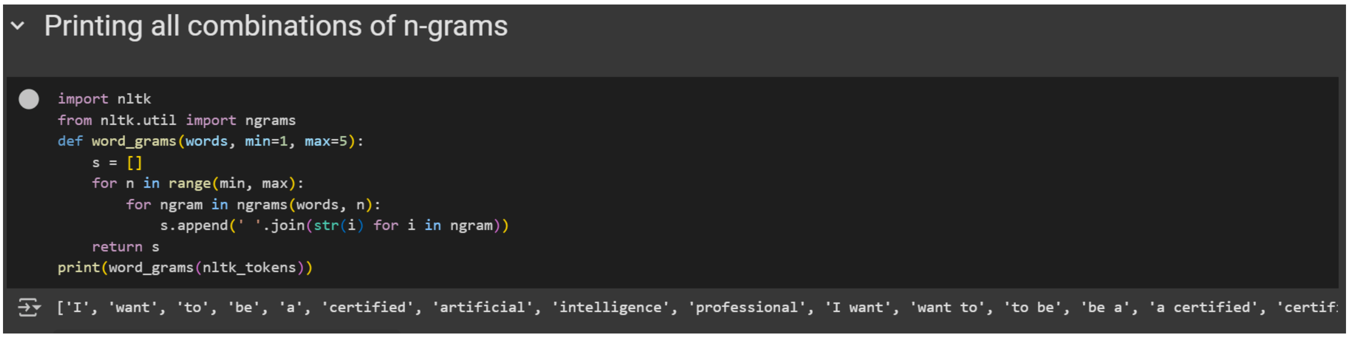
```
example_text = "The training is going great and the day is very fine.The code is working and all are happy about it"
#nltk.download('averaged_perceptron_tagger') # this has to run first time
token = nltk.word_tokenize(example_text)
nltk.pos_tag(token)  # error
#nltk.download('tagsets')  # this has to run first time

# We can get more details about any POS tag using help funciton of NLTK as follows.
nltk.help.upenn_tagset("PRP$")
nltk.help.upenn_tagset("JJ$")
nltk.help.upenn_tagset("VBG")
```

```
PRP$: pronoun, possessive
    her his mine my our ours their thy your
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first separable
    ectoplasmic battery-powered participatory fourth still-to-be-named
    multilingual multi-disciplinary ...
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lactating
    hankerin' alleging veering capping approaching traveling besieging
    encrypting interrupting erasing wincing ...
```



```
word_data = 'I want to be a certified artificial intelligence professional'
nltk_tokens = nltk.word_tokenize(word_data)
#print(list(nltk.bigrams(nltk_tokens)))
#nltk_tokens = nltk.word_tokenize(example_text)
print('Bigram-->',list(nltk.bigrams(nltk_tokens)))
print('-----------------------------------------------------')
print('Trigram-->',list(nltk.trigrams(nltk_tokens)))
print('-----------------------------------------------------')
print('5-gram-->',list(nltk.ngrams(nltk_tokens,5)))
```

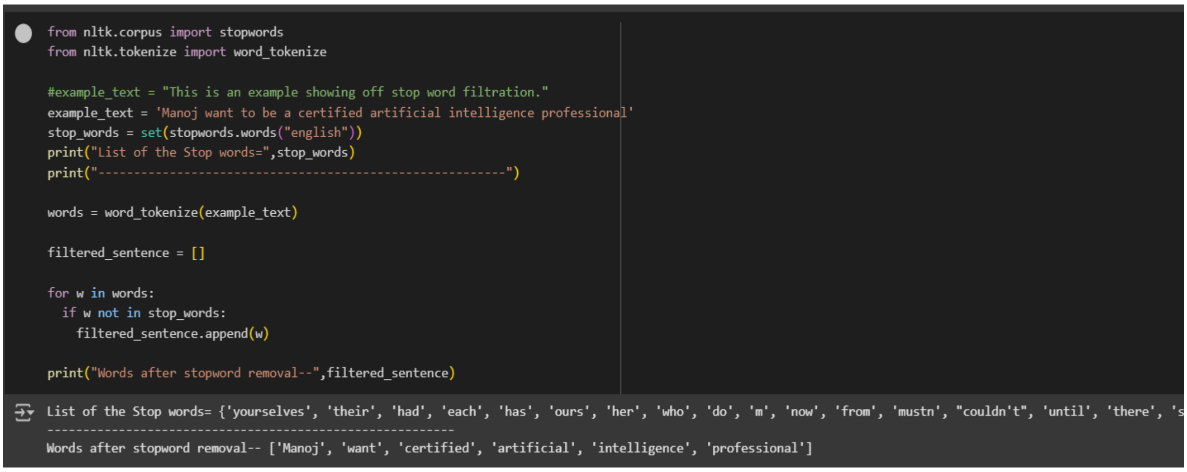
```
Bigram--> [('I', 'want'), ('want', 'to'), ('to', 'be'), ('be', 'a'), ('a', 'certified'), ('certified', 'artificial'), ('artificial', 'intelligence'),
-----------------------------------------------------
Trigram--> [('I', 'want', 'to'), ('want', 'to', 'be'), ('to', 'be', 'a'), ('be', 'a', 'certified'), ('a', 'certified', 'artificial'), ('certified', 'a
-----------------------------------------------------
5-gram--> [('I', 'want', 'to', 'be', 'a'), ('want', 'to', 'be', 'a', 'certified'), ('to', 'be', 'a', 'certified', 'artificial'), ('be', 'a', 'certifie
```



Printing all combinations of n-grams

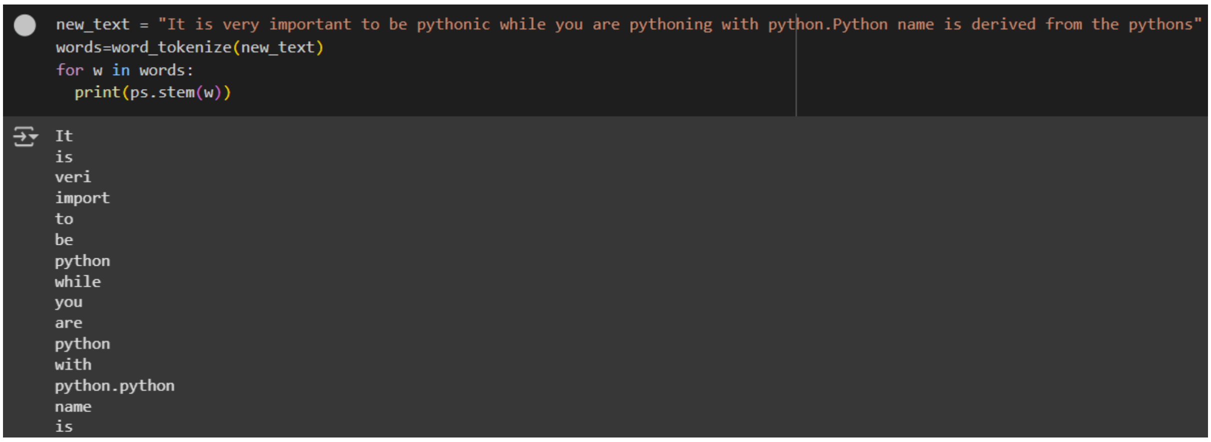
```
import nltk
from nltk.util import ngrams
def word_grams(words, min=1, max=5):
    s = []
    for n in range(min, max):
        for ngram in ngrams(words, n):
            s.append(' '.join(str(i) for i in ngram))
    return s
print(word_grams(nltk_tokens))
```

```
['I', 'want', 'to', 'be', 'a', 'certified', 'artificial', 'intelligence', 'professional', 'I want', 'want to', 'to be', 'be a', 'a certified', 'certif
```



```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

#example_text = "This is an example showing off stop word filtration."
example_text = 'Manoj want to be a certified artificial intelligence professional'
stop_words = set(stopwords.words("english"))
print("List of the Stop words=",stop_words)
print("----------------------------------------------------------")

words = word_tokenize(example_text)

filtered_sentence = []

for w in words:
  if w not in stop_words:
    filtered_sentence.append(w)

print("Words after stopword removal--",filtered_sentence)
```

```
List of the Stop words= {'yourselves', 'their', 'had', 'each', 'has', 'ours', 'her', 'who', 'do', 'm', 'now', 'from', 'mustn', "couldn't", 'until', 'there', 's
----------------------------------------------------------
Words after stopword removal-- ['Manoj', 'want', 'certified', 'artificial', 'intelligence', 'professional']
```



```
new_text = "It is very important to be pythonic while you are pythoning with python.Python name is derived from the pythons"
words=word_tokenize(new_text)
for w in words:
  print(ps.stem(w))
```

```
It
is
veri
import
to
be
python
while
you
are
python
with
python.python
name
is
```

POST LAB EXPERIMENT:
Health care:

```
import nltk
nltk.download('all')
# Cognitive Computing Application - Healthcare Symptom Checker
# Using NLP phases (tokenization, stemming, lemmatization, POS tagging, etc.)
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import stopwords, wordnet

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

# --- Sample healthcare knowledge base ---
symptom_db = {
   "fever": "Possible condition: Flu, Infection, or Covid-19",
   "cough": "Possible condition: Common Cold, Bronchitis, or Covid-19",
   "headache": "Possible condition: Migraine, Stress, or Dehydration",
   "stomach": "Possible condition: Food Poisoning, Gastritis, or Ulcer",
   "chest": "Possible condition: Asthma, Pneumonia, or Heart issue"
}

# --- Input Text ---
text = "I have a severe headache and mild fever since yesterday"
print("Input Text:", text)

# 1. Sentence Tokenization
sentences = sent_tokenize(text)
print("\nSentence Tokenization:", sentences)

# 2. Word Tokenization
words = word_tokenize(text)
print("\nWord Tokenization:", words)

# 3. Remove Stopwords
```

```
stop_words = set(stopwords.words('english'))
filtered_words = [w for w in words if w.lower() not in stop_words and w.isalpha()]
print("\nAfter Stopword Removal:", filtered_words)

# 4. Stemming
ps = PorterStemmer()
stemmed_words = [ps.stem(w) for w in filtered_words]
print("\nStemming:", stemmed_words)

# 5. Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_words = [lemmatizer.lemmatize(w.lower()) for w in filtered_words]
print("\nLemmatization:", lemmatized_words)

# 6. POS Tagging (Syntactic Analysis)
pos_tags = nltk.pos_tag(filtered_words)
print("\nPOS Tags:", pos_tags)

# --- Simple Cognitive Analysis ---
print("\n--- Cognitive Computing Symptom Analysis ---")
for word in lemmatized_words:
    for symptom in symptom_db:
        if symptom in word:
            print(f"Symptom detected: {word} → {symptom_db[symptom]}")
```

Output:

```
Input Text: I have a severe headache and mild fever since yesterday

Sentence Tokenization: ['I have a severe headache and mild fever since yesterday']

Word Tokenization: ['I', 'have', 'a', 'severe', 'headache', 'and', 'mild', 'fever', 'since', 'yesterday']

After Stopword Removal: ['severe', 'headache', 'mild', 'fever', 'since', 'yesterday']

Stemming: ['sever', 'headach', 'mild', 'fever', 'sinc', 'yesterday']

Lemmatization: ['severe', 'headache', 'mild', 'fever', 'since', 'yesterday']

POS Tags: [('severe', 'JJ'), ('headache', 'NN'), ('mild', 'NN'), ('fever', 'NN'), ('since', 'IN'), ('yesterday',
'NN')]

--- Cognitive Computing Symptom Analysis ---
Symptom detected: headache → Possible condition: Migraine, Stress, or Dehydration
Symptom detected: fever → Possible condition: Flu, Infection, or Covid-19
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```