

**Experiment No. 2**

**Title: Implementation of removal of punctuations, stop words, extra white spaces, URLs and HTML code from Text**

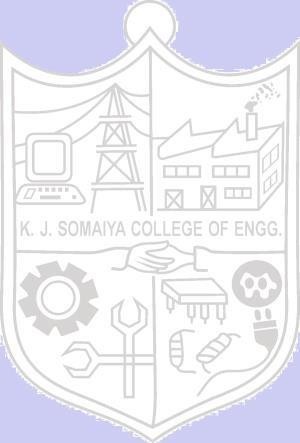
**Batch: 1 Roll No.: 16010420008 Experiment No.:2**

## **Aim**: To implement removal of punctuations, stop words, extra white spaces, URLs and HTML code from Text.

**Resources needed: Text Editor, Python Interpreter Theory:**

After Tokenization, the next step is NLP pipeline is stop words removal. Depending on the requirement, along with stop words, punctuations, extra white spaces, URLs and HTML code are also removed sometimes.

**Remove stop words**

Stop words are common words that do not add significant meaning to the text, such as “a,” “an,” and “the” and so on.

To remove common stop words from a list of tokens using NLTK, the nltk.corpus.stopwords.words() function can be used to get a list of stopwords in a specific language and filter the tokens using this list.

Example:

*import nltk import word\_tokenize* from nltk.corpus import stopwords

*# input text*

*text = "Natural language processing is a field of artificial intelligence that* deals with the interaction between computers and human (natural) language."

*# tokenize the text*

*tokens = word\_tokenize(text)*

*# get list of stopwords in English* stopwords = stopwords.words("english")

*# remove stopwords*

*filtered\_tokens = [token for token in tokens if token.lower() not in* stopwords]

*print("Tokens without stopwords:", filtered\_tokens)* Output generated:

*Tokens without stopwords: ['Natural', 'language', 'processing', 'field',* 'artificial', 'intelligence', 'deals', 'interaction', 'computers', 'human', '(', 'natural', ')', 'language', '.']

## **Note:** When you run this code the first time, it is possible that you will get a Python error, including the following message at the end: Resource stopwords not found. Please use the NLTK Downloader to obtain the resource:

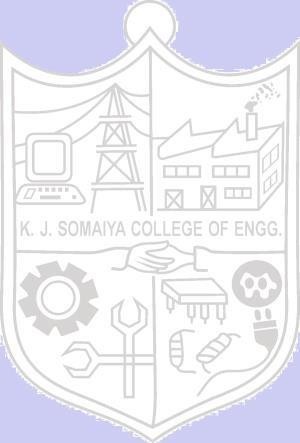
import nltk nltk.download(’stopwords’) **Remove punctuations**

## Removing punctuation marks simplifies the text and make it easier to process.To remove punctuation from a list of tokens using NLTK, string module can be used to check if each token is a punctuation character.

Example:

*import nltk* import string

*# input text*

*text = "Natural language processing is a field of artificial intelligence that* deals with the interaction between computers and human (natural) language."

*# tokenize the text*

*tokens = nltk.word\_tokenize(text)*

*# remove punctuation*

*filtered\_tokens = [token for token in tokens if token not in* string.punctuation]

*print("Tokens without punctuation:", filtered\_tokens)* Output generated:

*Tokens without punctuation: ['Natural', 'language', 'processing', 'is', 'a',* 'field', 'of', 'artificial', 'intelligence', 'that', 'deals', 'with', 'the', 'interaction', 'between', 'computers', 'and', 'human', 'natural', 'language']

# Remove extra whitespaces

## To remove extra white space from a string of text using NLTK the string.strip() function can be used to remove leading and trailing white space, and the string.replace() function can be used to replace multiple consecutive white space characters with a single space.

*import nltk* import string

*# input text with extra white space*

*text = " Natural language processing is a field of artificial* intelligence that deals with the interaction between computers and human (natural) language. "

*# remove leading and trailing white space*

*text = text.strip()*

*# replace multiple consecutive white space characters with a single space* text = " ".join(text.split())

*print("Cleaned text:", text)*

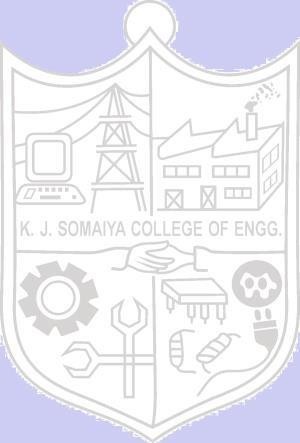
*Output generated:*

*Cleaned text: Natural language processing is a field of artificial* intelligence that deals with the interaction between computers and human (natural) language.

# Remove URLs

## To remove URLs from a string of text using NLTK, a regular expression pattern can be used to identify URLs and can be replaced with an empty string.

Example:

*import nltk* import re

*# input text with URLs*

*text = "Natural language processing is a field of artificial intelligence that* deals with the interaction between computers and human (natural) language.

*Check out this article for more information:* https://en.wikipedia.org/wiki/Natural\_language\_processing"

*# define a regular expression pattern to match URLs*

*pattern = r"(http|ftp|https)://([\w\_-]+(?:(?:\.[\w\_-]+)+))([\w.,@?^=%&:/~+#-*

*]\*[\w@?^=%&/~+#-])?"*

*# replace URLs with an empty string* cleaned\_text = re.sub(pattern, "", text)

*print("Text without URLs:", cleaned\_text)*

*Output generated:*

*Text without URLs: Natural language processing is a field of artificial* intelligence that deals with the interaction between computers and human (natural) language. Check out this article for more information:

# Remove HTML Code

## To remove HTML code from a string of text using NLTK, regular expression pattern can be used to identify HTML tags and then can be replaced with an empty string.

Example:

*import nltk* import re

*# input text with HTML code*

*text = "Natural language processing is a field of artificial intelligence that* deals with the interaction between computers and human (natural) language.

*<b>This is an example of bold text.</b>"*

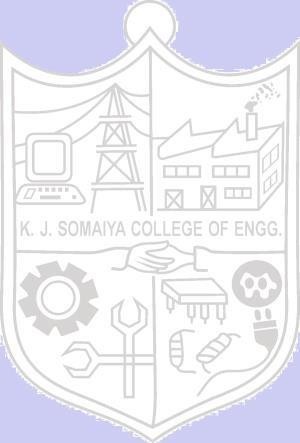
*# define a regular expression pattern to match HTML tags* pattern = r"<[^>]+>"

*# replace HTML tags with an empty string* cleaned\_text = re.sub(pattern, "", text)

*print("Text without HTML code:", cleaned\_text)*

*Output generated:*

*Text without HTML code: Natural language processing is a field of artificial intelligence that deals with the interaction between computers and human (natural) language. This is an example of bold text.*



**Note** that sometimes depending on the need, for example, while working with text data derived from speech, punctuations might be required to be added to the extracted text. Similarly, rather than removing URLs and HTML code from the text, this data might be of interest to us while working with some NLP applications.

**Activity:**

1. Add custom list of stop words to English language stop words and use this list of stop words to remove stop words from text
2. Apply stop word removal, punctuation removal, space removal, URL and HTML code removal to a dataset of technical discussion forum such as dataset of stack overflow.

**Results: (Program with snapshot of output) Code:**

import pandas as pd import nltk

from nltk.corpus import stopwords import string

import re nltk.download('punkt') nltk.download('stopwords') from nltk import word\_tokenize

def stopwords\_removal(stopwords, token\_words): stopwords = stopwords.words("english") stopwords.append("ies") stopwords.append("ed")

stopwords\_removal\_tokens = [token for token in token\_words if token.lower() not in stopwords]

return stopwords\_removal\_tokens

def punctuations\_removal(token\_words):

punctuations\_removal\_tokens = [token for token in token\_words if token not in string.punctuation]

return punctuations\_removal\_tokens

def whitespace\_removal(text): text = text.strip()

whitespace\_removal\_text = " ".join(text.split()) return text

def url\_removal(text):

pattern = r"(http|ftp|https)://([\w\_-]+(?:(?:\.[\w\_-]+)+))([\w.,@?^=%&:/~+#-

]\*[\w@?^=%&/~+#-])?"

url\_removal\_text = re.sub(pattern, "", text) return url\_removal\_text

def html\_removal(text): pattern = r"<[^>]+>"

html\_removal\_text = re.sub(pattern, "", text) return html\_removal\_text

csv\_file = pd.read\_csv('/content/Answers.csv') csv\_file\_column = csv\_file[csv\_file['Body'].notna()]

answer\_column = [] for i in range(0, 10):

answer\_column.append(csv\_file\_column['Body'][i])

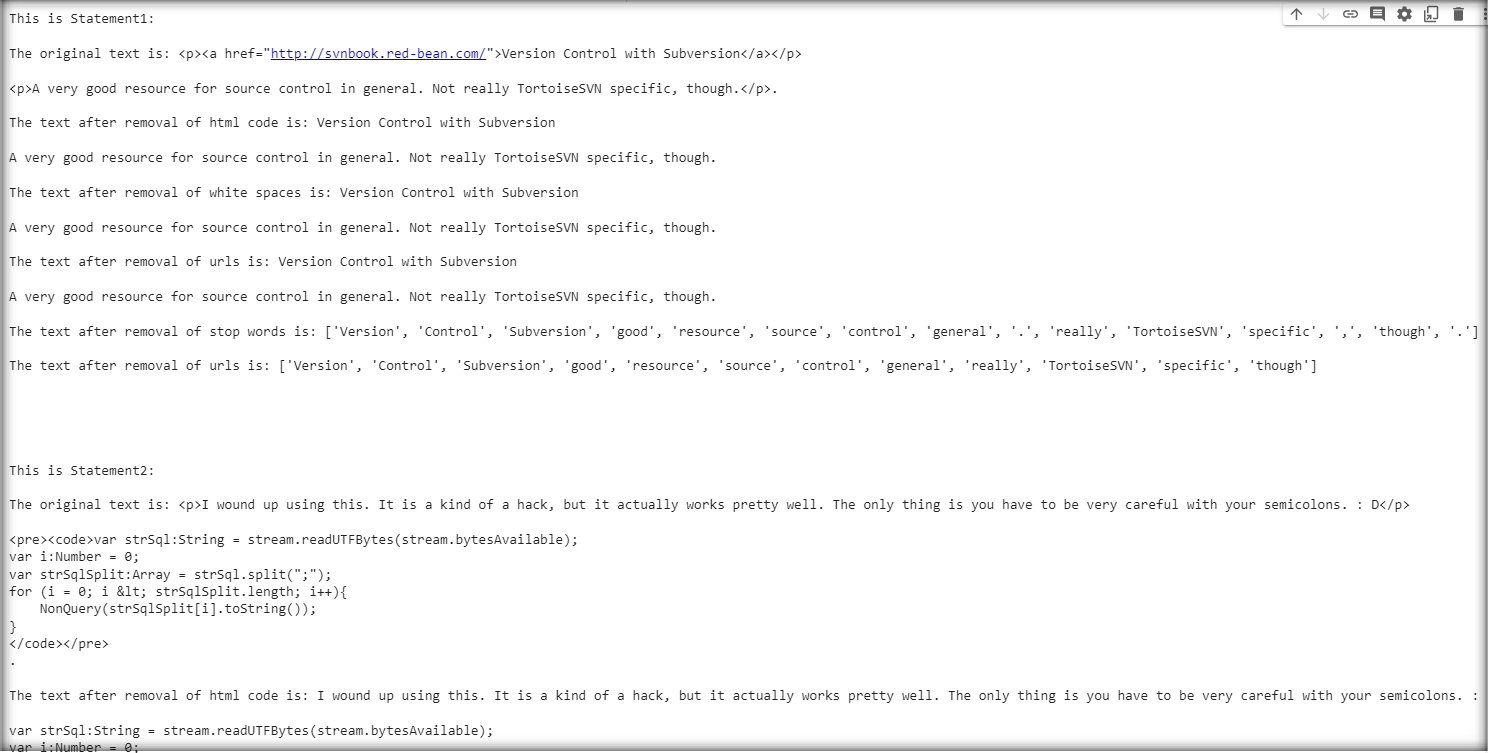
for i in answer\_column:

print(f"This is Statement{answer\_column.index(i) + 1}: \n") print(f"The original text is: {i}.\n")

html\_removal\_text = html\_removal(i)

print(f"The text after removal of html code is: {html\_removal\_text}\n") whitespace\_removal\_text = whitespace\_removal(html\_removal\_text)

print(f"The text after removal of white spaces is:



{whitespace\_removal\_text}\n")

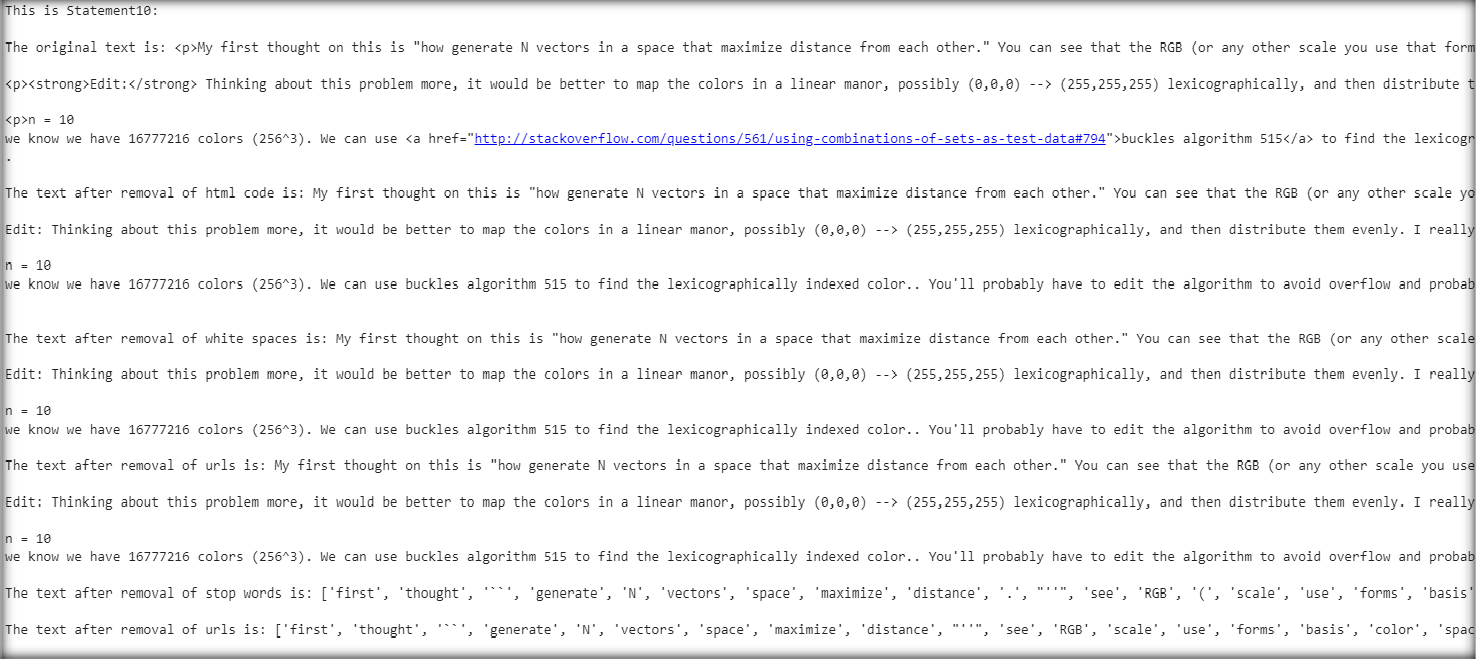
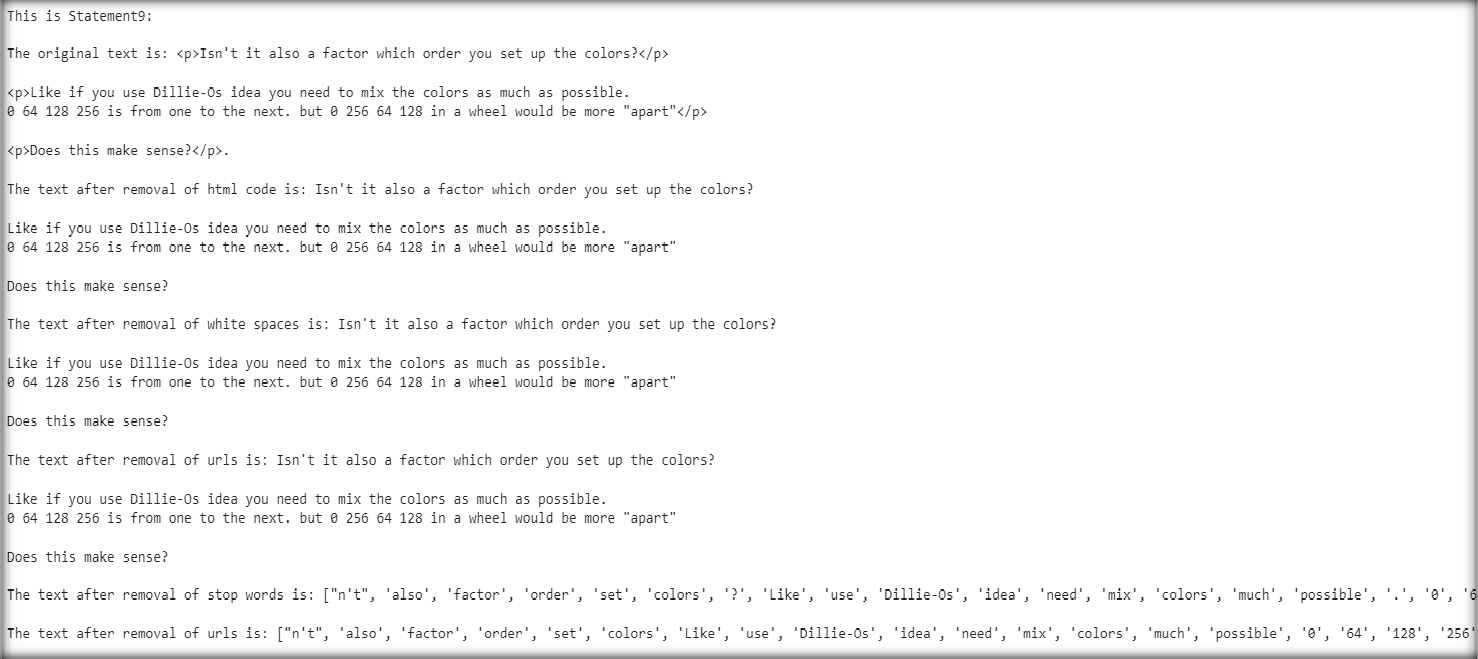
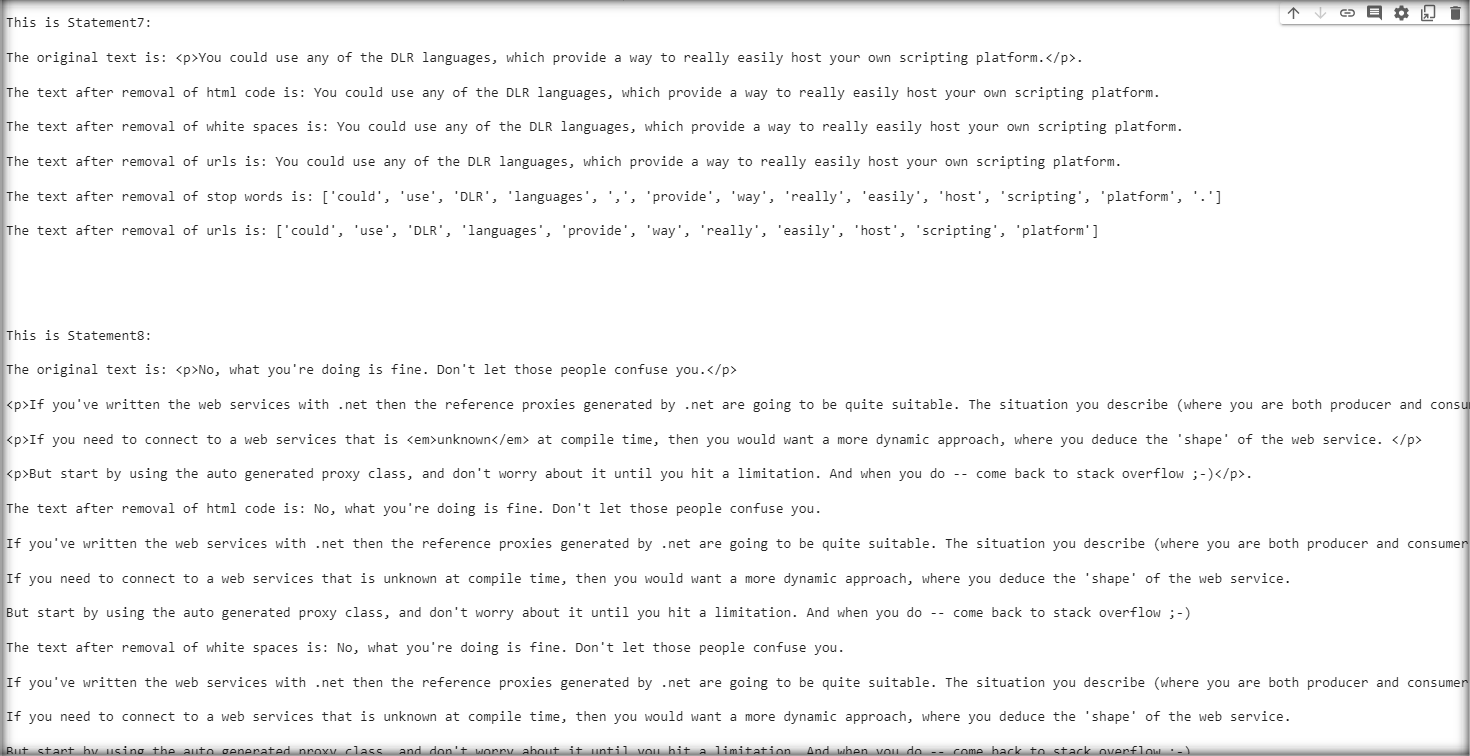
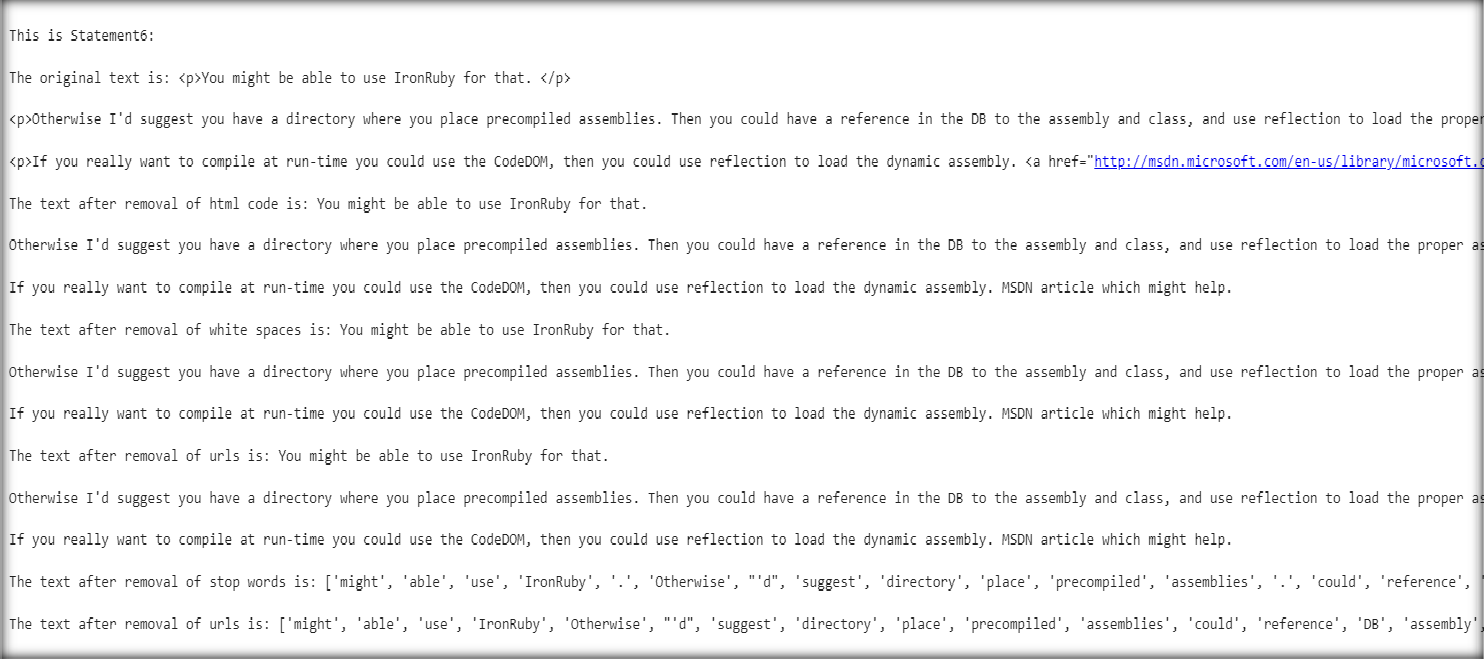
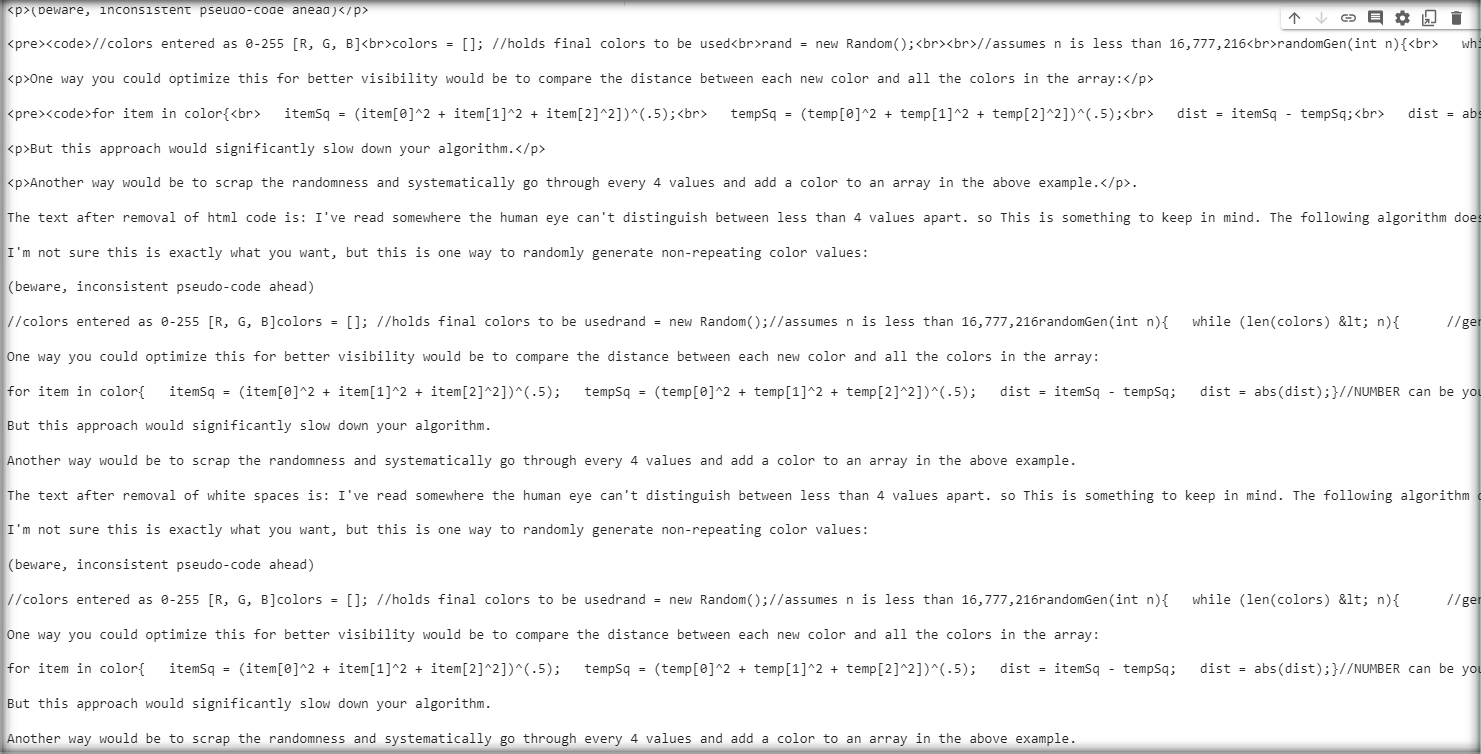
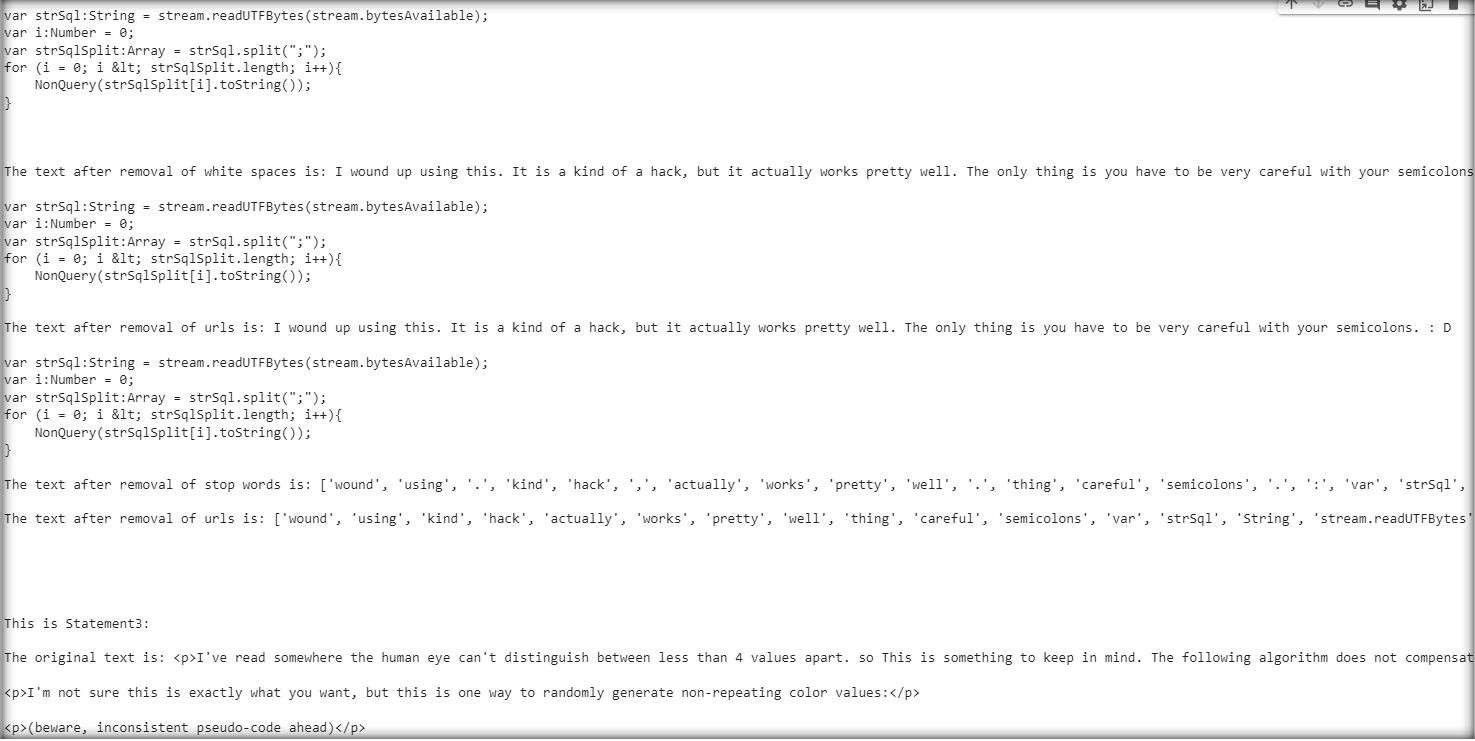
url\_removal\_text = url\_removal(whitespace\_removal\_text) print(f"The text after removal of urls is: {url\_removal\_text}\n") tokens = word\_tokenize(url\_removal\_text) stopwords\_removal\_tokens = stopwords\_removal(stopwords, tokens) print(f"The text after removal of stop words is:

{stopwords\_removal\_tokens}\n") punctuations\_removal\_tokens =

punctuations\_removal(stopwords\_removal\_tokens) print(f"The text after removal of urls is:

{punctuations\_removal\_tokens}\n\n\n\n\n")

**Output:**



**Questions:**

1. As discussed there might be need to add punctuations or retain URLs in the given Text.
   1. Write the sample python code for extracting text from audio and add appropriate punctuations to it
   2. Write a sample python code to identify the URLs from the text data and extract URLs from text data

A)

a)

**Code:**

import speech\_recognition as sr import string

recognizer = sr.Recognizer() audio\_file = "sample\_audio.wav"

with sr.AudioFile(audio\_file) as source: audio = recognizer.record(source)

try:

text = recognizer.recognize\_google(audio) print("Extracted Text:", text)

text\_with\_punctuations = text + " " + string.punctuation print("Text with Punctuations:", text\_with\_punctuations)

except sr.UnknownValueError:

print("Speech Recognition could not understand audio") except sr.RequestError as e:

print(f"Could not request results from Google Speech Recognition service; {e}")

b)

**Code:**

import re

text = "Check out this website: https://travelog.surge.sh."

url\_pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\\(\\),]|(?:%[0-9a- fA-F][0-9a-fA-F]))+'

urls = re.findall(url\_pattern, text) for url in urls:

print("Extracted URL:", url)

**Outcomes:**

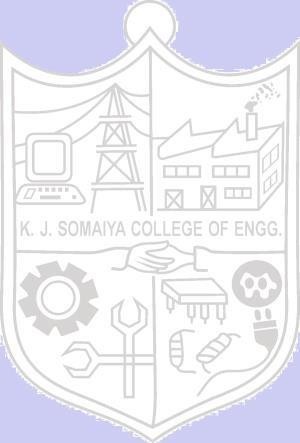
**CO2: Comprehend Words and Word Forms in NLP**

**Conclusion: (Conclusion to be based on the outcomes achieved)**

**We understood the concept of need of removal of punctuations, stop words, extra white spaces, URLs and HTML code from Text. We also implemented the same on a field of text from our dataset and observed the results.**

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date



**References:**

**Books/ Journals/ Websites:**

1. Allen.James, Natural Language Understanding, Benjamin Cumming, Second Edition, 1995
2. Jurafsky, Dan and Martin, James, Speech and Language Processing, Prentice Hall, 2008
3. Palash Goyal, Karan Jain, Sumit Pandey,Deep Learning for Natural Language Processing: Creating Neural Networks with Python, Apress, 2018