

## **Experiment No. 1**

**Title:Implementation of Tokenization with different  
python libraries**

Batch:B4

Roll No.: 16010420061

Experiment No.: 1

**Aim:** To implement Tokenization using different python libraries

---

**Activity:**

**Github Link:** [Sargundeeep/NLP \(github.com\)](https://github.com/Sargundeeep/NLP)

1. Apply tokenization using different python libraries with different text inputs such as tweets, news article and so on.

```
Punctuation Tokenisation

from nltk.tokenize import WordPunctTokenizer

tokenizer = WordPunctTokenizer()
tokenizer.tokenize("Let's see how it's working.")

['Let', "'", 's', 'see', 'how', 'it', "'", 's', 'working', '.']

+ Code + Text

PunktWord Tokenisation

[ ] import nltk
    from nltk.tokenize import sent_tokenize

text = "The picket fence had stood for years without any issue. That's all it was. A simple, white, picket fence. Why it had all of a sudden become a lightning rod
sent_tokenize(text)

['The picket fence had stood for years without any issue.',
 'That's all it was.',
 'A simple, white, picket fence.',
 'Why it had all of a sudden become a lightning rod within the community was still unbelievable to most.',
 'Yet a community that had once lived in harmony was now divided in bitter hatred and it had everything to do with the white picket fence.

import nltk.data

# Loading PunktSentenceTokenizer using English pickle file
tokenizer = nltk.data.load('tokenizers/punkt/PY3/english.pickle')

tokenizer.tokenize(text)

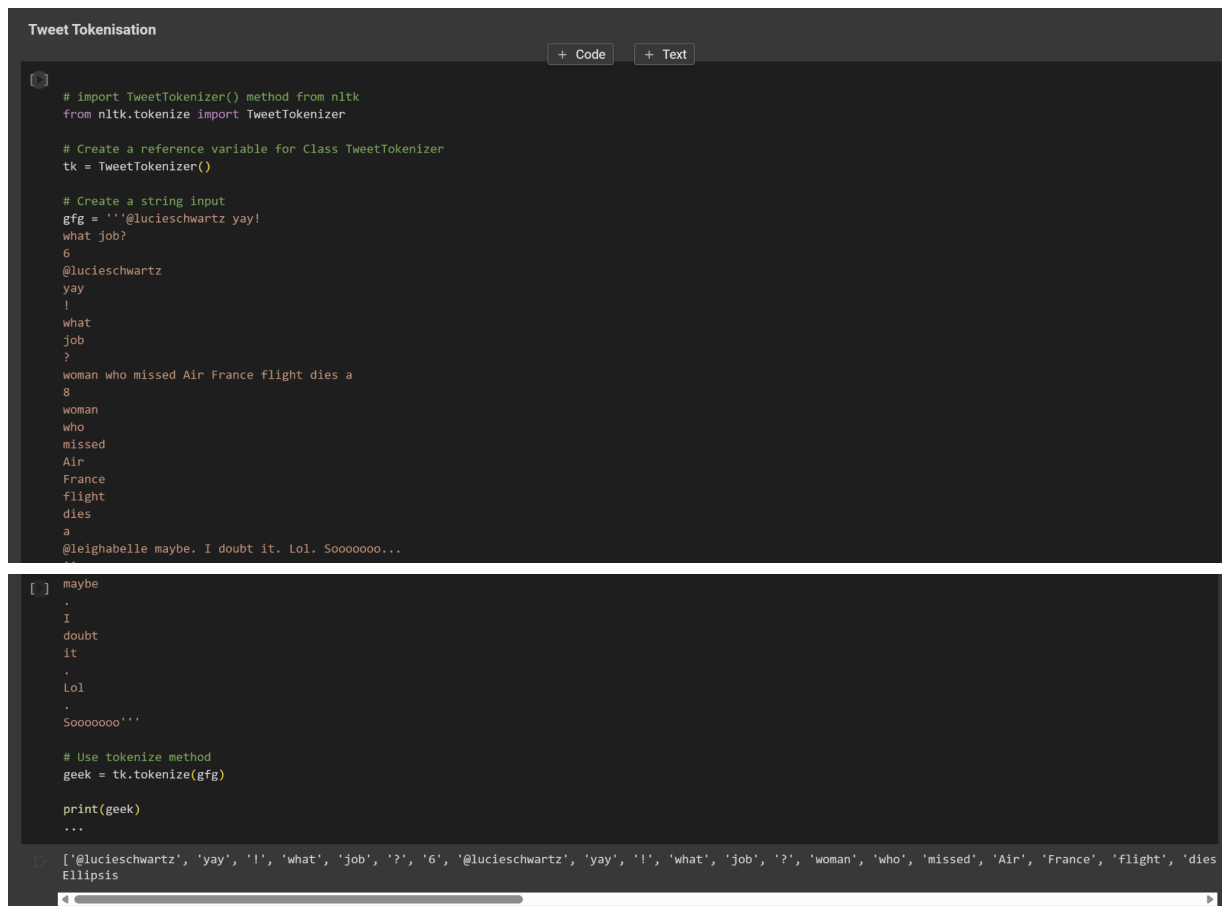
['The picket fence had stood for years without any issue.',
 'That's all it was.',
 'A simple, white, picket fence.',
 'Why it had all of a sudden become a lightning rod within the community was still unbelievable to most.',
 'Yet a community that had once lived in harmony was now divided in bitter hatred and it had everything to do with the white picket fence.

Default/TreeBank Tokenisation

[ ] from nltk.tokenize import word_tokenize

text = "Hello everyone. Welcome to GeeksforGeeks."
word_tokenize(text)

['Hello', 'everyone', '.', 'Welcome', 'to', 'GeeksforGeeks', '.']
```



```
# import TweetTokenizer() method from nltk
from nltk.tokenize import TweetTokenizer

# Create a reference variable for Class TweetTokenizer
tk = TweetTokenizer()

# Create a string input
gfg = '@lucieschwartz yay!
what job?
6
@lucieschwartz
yay
!
what
job
?
woman who missed Air France flight dies a
8
woman
who
missed
Air
France
flight
dies
a
@leighabelle maybe. I doubt it. lol. Soooooo...'

# Use tokenize method
geek = tk.tokenize(gfg)

print(geek)
...
```

```
['@lucieschwartz', 'yay', '!', 'what', 'job', '?', '6', '@lucieschwartz', 'yay', '!', 'what', 'job', '?', 'woman', 'who', 'missed', 'Air', 'France', 'flight', 'dies', 'a', '@leighabelle', 'maybe', 'I', 'doubt', 'it', 'lol', 'Sooooooo']
```

2. Write your observations regarding using different methods of tokenization with different input text data.

**WordPunctTokenizer for Punctuation Tokenization:** The WordPunctTokenizer from NLTK tokenizes text by considering words and punctuations as separate tokens. This method is useful when you want to separate words and punctuations while maintaining separate tokens for each.

**PunktSentenceTokenizer for Sentence Tokenization:** The PunktSentenceTokenizer from NLTK is specifically designed for sentence tokenization. It uses an unsupervised machine learning approach to identify sentence boundaries. This method is particularly useful when you need to split a text into sentences.

**Treebank Tokenization for Word Tokenization:** The word\_tokenize function from NLTK uses the Treebank tokenizer, which tokenizes text into words while considering common contractions and punctuation. It is useful when you want to split text into individual words.

**TweetTokenizer for Tokenizing Tweets:** The TweetTokenizer from NLTK is designed specifically for tokenizing tweets, which often contain hashtags, mentions, and emoticons. It's useful when you need to process social media text data.

**Questions:**

1. Explain the tokenization of regular expression using Regex tokenizer from NLTK with the help of an example

NLTK's RegexpTokenizer is a flexible tokenization method that allows you to tokenize text using regular expressions. This gives you more control over how the text is divided into tokens.

```
from nltk.tokenize import RegexpTokenizer

# Create a RegexpTokenizer using a regular expression to match words
tokenizer = RegexpTokenizer(r'\w+')

# Input text
text = "Hello, this is a simple example of tokenization using NLTK's RegexpTokenizer!"

# Tokenize the text using the RegexpTokenizer
tokens = tokenizer.tokenize(text)

print(tokens)
```

In this example, we are using the regular expression `r'\w+'` to match one or more word characters. Let's break down the regular expression:

`\w`: Matches any word character (alphanumeric character or underscore).

`+`: Matches one or more occurrences of the preceding character or group.

So, the regular expression `\w+` will match sequences of one or more word characters.

**Outcomes: CO 1: Understand fundamentals of NLP**

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

Successfully understood the NLP tokenization types and its implementation.