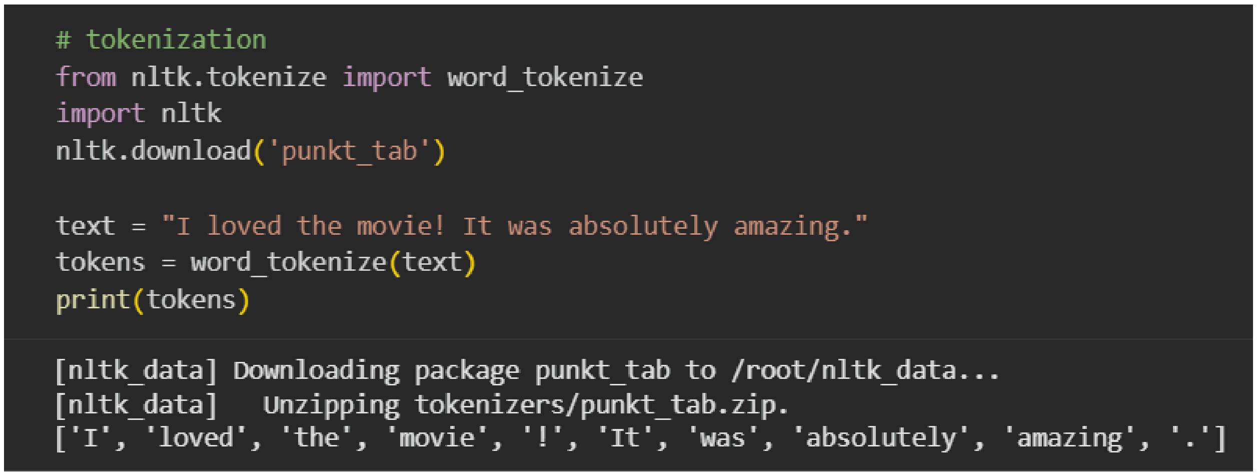
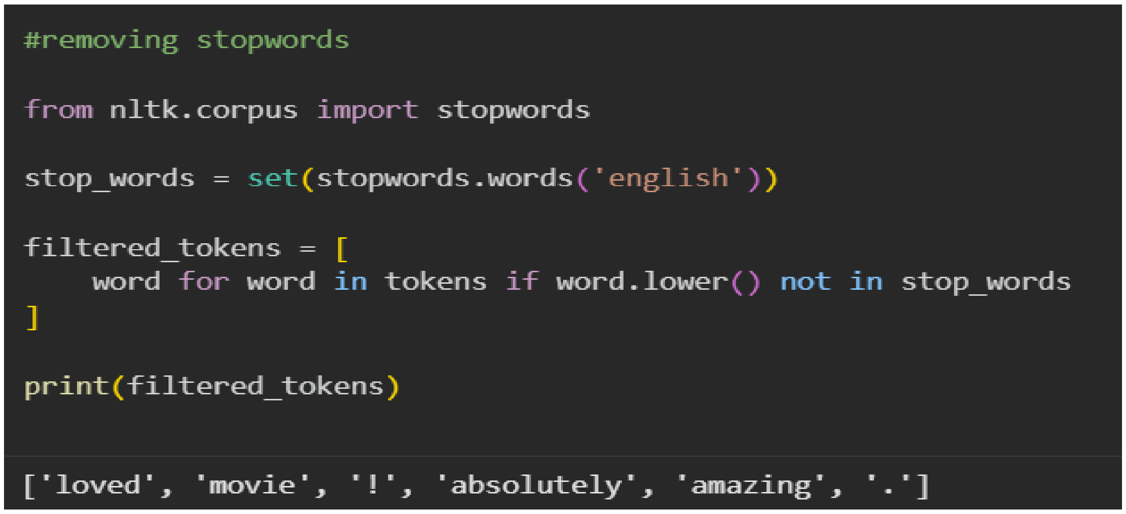
WEEK 1 REPORT



```
# tokenization
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt_tab')

text = "I loved the movie! It was absolutely amazing."
tokens = word_tokenize(text)
print(tokens)
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
['I', 'loved', 'the', 'movie', '!', 'It', 'was', 'absolutely', 'amazing', '.']
```

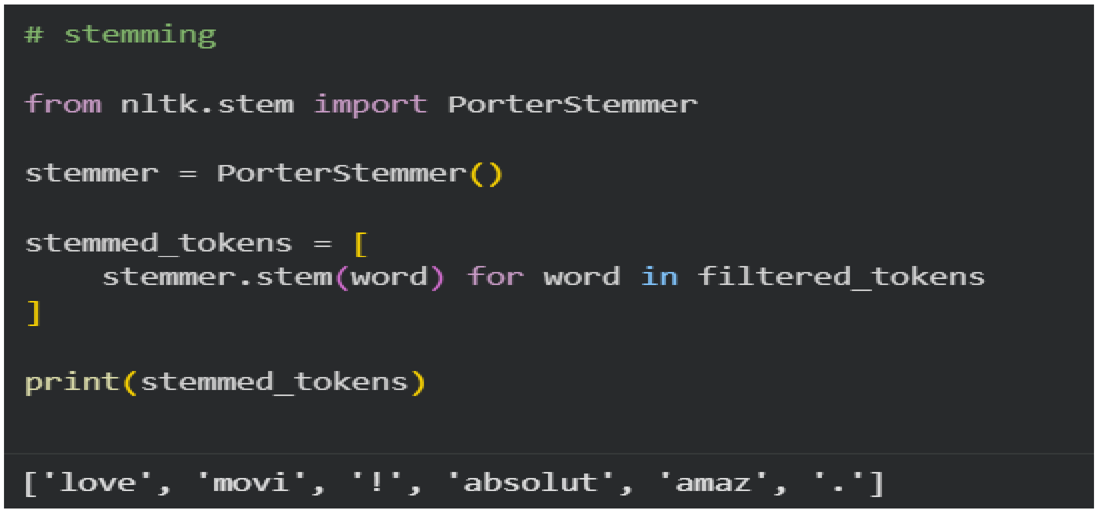
In this step, I did tokenization using NLTK library.
Tokenization is splitting of sentence into individual words and punctuation marks called tokens.



```
#removing stopwords

from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))

filtered_tokens = [
    word for word in tokens if word.lower() not in stop_words
]

print(filtered_tokens)
```

```
['loved', 'movie', '!', 'absolutely', 'amazing', '.']
```

Then I removed stopwords using the NLTK stopwords list. Stopwords are common words such as “the”, “is”, “was”, “and” that do not add much meaning to the text. These words were removed from the tokenized text to reduce noise and keep only important words that help in understanding the sentiment.



```
# stemming

from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

stemmed_tokens = [
    stemmer.stem(word) for word in filtered_tokens
]

print(stemmed_tokens)
```

```
['love', 'movi', '!', 'absolut', 'amaz', '.']
```

And then I applied stemming using the Porter Stemmer from the NLTK library. Stemming reduces words to their root form by removing suffixes. For example, words like “loved” are reduced to “love”. This helps in reducing the number of similar words and makes the text processing more efficient.

<u>Build a TF-IDF text classifier for sentiment using IMDb data</u>

- I performed sentiment analysis on movie reviews using basic Natural Language Processing (NLP) and Machine Learning techniques.
- I used the IMDb movie review dataset, which contains text reviews labeled as positive or negative.
- The text data was converted into numerical form using TF-IDF (Term Frequency–Inverse Document Frequency).
- I used Logistic Regression, a simple and effective machine learning classifier, to train the sentiment analysis model.
- The model was trained on TF-IDF features extracted from IMDb reviews to classify them as positive or negative. After training, the model was tested on new unseen reviews, and it successfully predicted whether a given review expressed a positive or negative sentiment.