Exercise 1 Tokenization

```
doc1="""Natural language processing (NLP) is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
from nltk.tokenize import word_tokenize, sent_tokenize
```

```
doc_sentences = sent_tokenize(doc1)
print(doc_sentences)

doc_words = word_tokenize(doc1)
print(doc_words)

#unique_tokens = set(word_tokenize(doc1))
#print(unique_tokens)
```

```
['Natural language processing (NLP) is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence
['Natural', 'language', 'processing', '(', 'NLP', ')', 'is', 'an', 'interdisciplinary', 'subfield', 'of', 'linguistics', ',', 'comp
```

Exercise 2 Stopwords & Punctuations removal

```
doc1="""Natural language processing (NLP) is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence
```

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
from nltk.corpus import stopwords
```

```
stop_words = set(stopwords.words('english'))
tokenized_words = word_tokenize(doc1)
filtered_words = []
for w in tokenized_words:
    if w not in stop_words:
        filtered_words.append(w)
print(tokenized_words)
print(filtered_words)
```

```
['Natural', 'language', 'processing', '(', 'NLP', ')', 'is', 'an', 'interdisciplinary', 'subfield', 'of', 'linguistics', ',', 'comp
['Natural', 'language', 'processing', '(', 'NLP', ')', 'interdisciplinary', 'subfield', 'linguistics', ',', 'computer', 'science',
```

```
#Remove punctuations
import string as st
```

```
punctuations=list(st.punctuation)
filtered_tokens=[]

for i in filtered_words:
    if i not in punctuations:
        filtered_tokens.append(i)
print(filtered_tokens)
```

```
['Natural', 'language', 'processing', 'NLP', 'interdisciplinary', 'subfield', 'linguistics', 'computer', 'science', 'artificial', '
```

Exercise 3 Stemming & Lemmatization

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize


ps = PorterStemmer()
stemmed_words=[]

for w in filtered_tokens:
     stemmed_words.append(ps.stem(w))

print("Filtered Tokens After Removing Punctuations:",filtered_tokens)
print("Stemmed Tokens:",stemmed_words)
```

```
Filtered Tokens After Removing Punctuations: ['Natural', 'language', 'processing', 'NLP', 'interdisciplinary', 'subfield', 'linguis
Stemmed Tokens: ['natur', 'languag', 'process', 'nlp', 'interdisciplinari', 'subfield', 'linguist', 'comput', 'scienc', 'artifici',
```

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer


lem = WordNetLemmatizer()
lemm_words=[]
for w in filtered_tokens:
  lemm_words.append(lem.lemmatize(w,'v'))

print("Filtered Tokens After Removing Punctuations:",filtered_tokens)
print("Stemmed Tokens:",stemmed_words)
print("Lemmatized Word:",lemm_words)
```

```
Filtered Tokens After Removing Punctuations: ['Natural', 'language', 'processing', 'NLP', 'interdisciplinary', 'subfield', 'linguis
Stemmed Tokens: ['natur', 'languag', 'process', 'nlp', 'interdisciplinari', 'subfield', 'linguist', 'comput', 'scienc', 'artifici',
Lemmatized Word: ['Natural', 'language', 'process', 'NLP', 'interdisciplinary', 'subfield', 'linguistics', 'computer', 'science', '
```

Exercise 4 POS Tagging

```
nltk.download('averaged_perceptron_tagger')
```

```
[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!
True
```

```
from nltk.tokenize import word_tokenize
from nltk import pos_tag


print("Filtered Tokens After Removing Punctuations:",filtered_tokens)
```

```
Filtered Tokens After Removing Punctuations: ['Natural', 'language', 'processing', 'NLP', 'interdisciplinary', 'subfield', 'linguis
```

```
pos_tokens=pos_tag(filtered_tokens)
print("PoS tags:",pos_tokens)
```

```
PoS tags: [('Natural', 'JJ'), ('language', 'NN'), ('processing', 'NN'), ('NLP', 'NNP'), ('interdisciplinary', 'JJ'), ('subfield', '
```

Exercise 5 Named Entity Recognition

```
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
```

```
[nltk_data]   Package words is already up-to-date!
True
```

```
from nltk import ne_chunk
```

```
doc2="""In 1950, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the Tur
```

```
from nltk.tokenize import word_tokenize
from nltk import pos_tag
doc_words = word_tokenize(doc2)
pos_tokens=pos_tag(doc_words)
print("PoS tags:",pos_tokens)
"""
for chunk in ne_chunk(pos_tokens):
        if hasattr(chunk, 'label'):
          print(chunk.label(), ' '.join(c[0] for c in chunk))
"""
ck = ne_chunk(pos_tokens)
print(ck)
```

```
PoS tags: [('In', 'IN'), ('1950', 'CD'), (',', ','), ('Alan', 'NNP'), ('Turing', 'NNP'), ('published', 'VBD'), ('an', 'DT'), ('a
(S
  In/IN
  1950/CD
  ,/,
  (PERSON Alan/NNP Turing/NNP)
  published/VBD
  an/DT
  article/NN
  titled/VBN
  ``/``
  Computing/JJ
  (ORGANIZATION Machinery/NN)
  and/CC
  (ORGANIZATION Intelligence/NNP)
  ''/''
  which/WDT
  proposed/VBD
  what/WP
  is/VBZ
  now/RB
  called/VBN
  the/DT
  (GPE Turing/NNP)
  test/NN
  as/IN
  a/DT
  criterion/NN
  of/IN
  intelligence/NN
  ,/,
  though/RB
  at/IN
  the/DT
  time/NN
  that/WDT
  was/VBD
  not/RB
  articulated/VBN
  as/IN
  a/DT
  problem/NN
  separate/NN
  from/IN
  artificial/JJ
  intelligence/NN
  ./.
  The/DT
  proposed/JJ
  test/NN
  includes/VBZ
  a/DT
  task/NN
  that/WDT
  involves/VBZ
  the/DT
  automated/JJ
```

Exercise 6 Text Classification

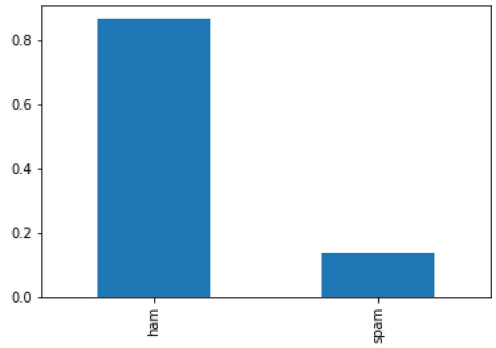
```
import pandas as pd
```

```
data = pd.read_csv('/content/spam.csv', encoding = "ISO-8859-1")
data.head()
```

	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
data['label'].value_counts(normalize = True).plot.bar()
```

```
<Axes: >
```




```
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer


lemmatizer = WordNetLemmatizer()
corp = []
txt = list(data['text'])

for i in range(len(txt)):
  r = re.sub('[’a-zA-Z]',' ', txt[i])
  r = r.lower()
  r = r.split()
  r = [word for word in r if word not in stopwords.words('english')]
  r = [lemmatizer.lemmatize(word) for word in r]
  r = ' ' .join(r)
  corp.append(r)

data['text']=corp
data.tail()
```

	label	text
5567	spam	2 2 . â£750 . 2 , 087187272008 1! 10 . - - .
5568	ham	ì_ ?
5569	ham	, * . ... ?
5570	ham	'
5571	ham	.

```
from sklearn.model_selection import train_test_split

X = data['text']
Y = data['label']

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2, random_state=123)

print('Training Data: ', X_train.shape)
print('Testing Data: ', X_test.shape)
```

```
Training Data:  (4457,)
Testing Data:  (1115,)
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X_train_cv = cv.fit_transform(X_train)

X_train_cv.shape
```

```
(4457, 618)
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train_cv, Y_train)
X_test_cv = cv.transform(X_test)
predictions = lr.predict(X_test_cv)

lr.score(X_test_cv,Y_test)
```

```
0.9524663677130045
```

```
from sklearn import metrics
df = pd.DataFrame(metrics.confusion_matrix(Y_test,predictions), index=['ham','spam'], columns=['ham','spam'])

print(df)
```

```
      ham  spam
ham   981     1
spam   52    81
```

Exercise 7 Sentiment Analysis

```
import numpy as np
import pandas as pd
import re
import nltk

data_source_url = "/content/Tweets.csv"
airline_tweets = pd.read_csv(data_source_url)
airline_tweets.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	

```
features = airline_tweets.iloc[:, 10].values
labels = airline_tweets.iloc[:, 1].values
processed_features = []
for sentence in range(0, len(features)):
  # Remove all the special characters
  processed_feature = re.sub(r'\W', ' ', str(features[sentence]))
  # Converting to Lowercase
  processed_feature = processed_feature.lower()
  processed_features.append(processed_feature)

print(processed_features)
```

```
[' virginamerica what  dhepburn said ', ' virginamerica plus you ve added commercials to the experience    tacky ', ' virginamerica
```

```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer (max_features=2500, min_df=7, max_df=0.8, stop_words=stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2, random_state=0)


from sklearn.ensemble import RandomForestClassifier
text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)

predictions = text_classifier.predict(X_test)



from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
[[1723  108   39]
 [ 326  248   40]
 [ 132   58  254]]
              precision    recall  f1-score   support

    negative       0.79      0.92      0.85      1870
     neutral       0.60      0.40      0.48       614
    positive       0.76      0.57      0.65       444

    accuracy                           0.76      2928
   macro avg       0.72      0.63      0.66      2928
weighted avg       0.75      0.76      0.74      2928

0.7599043715846995
```