```
pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\2020s\anaconda3\lib\site-packages (3.7)Note: you may need to restart the kernel to use
updated packages.

Requirement already satisfied: tqdm in c:\users\2020s\anaconda3\lib\site-packages (from nltk) (4.64.1)
Requirement already satisfied: regex>=2021.8.3 in c:\users\2020s\anaconda3\lib\site-packages (from nltk) (2022.7.9)
Requirement already satisfied: joblib in c:\users\2020s\anaconda3\lib\site-packages (from nltk) (1.1.0)
Requirement already satisfied: click in c:\users\2020s\anaconda3\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: colorama in c:\users\2020s\anaconda3\lib\site-packages (from click->nltk) (0.4.5)
```

Word tokenization

1)Using NLTK

Input :

```
import nltk
from nltk.tokenize import word_tokenize

#nltk.download('punkt')  # word_tokenize used punkt for tokenization. punkt is already trained ML model for tokenization

text = '''I saw my life branching out before me like the green fig tree in the story.
From the tip of every branch, like a fat purple fig, a wonderful future beckoned and winked.
One fig was a husband and a happy home and children, and another fig was a famous poet and another fig was a brilliant professor,
and another fig was Ee Gee, the amazing editor, and another fig was Europe and Africa and South America,
and another fig was Constantin and Socrates and Attila and a pack of other lovers with queer names and offbeat professions,
and another fig was an Olympic lady crew champion, and beyond and above these figs were many more figs I couldn't quite make out.
I saw myself sitting in the crotch of this fig tree, starving to death, just because I couldn't make up my mind which of the figs
I would choose. I wanted each and every one of them, but choosing one meant losing all the rest, and, as I sat there, unable to decide,
the figs began to wrinkle and go black, and, one by one, they plopped to the ground at my feet.'''

tokens = word_tokenize(text)
print(tokens)
```

Output :

```
['I', 'saw', 'my', 'life', 'branching', 'out', 'before', 'me', 'like', 'the', 'green', 'fig', 'tree', 'in', 'the', 'story', '.', 'Fro
m', 'the', 'tip', 'of', 'every', 'branch', ',', 'like', 'a', 'fat', 'purple', 'fig', ',', 'a', 'wonderful', 'future', 'beckoned', 'an
d', 'winked', '.', 'One', 'fig', 'was', 'a', 'husband', 'and', 'a', 'happy', 'home', 'and', 'children', ',', 'and', 'another', 'fig',
'was', 'a', 'famous', 'poet', 'and', 'another', 'fig', 'was', 'a', 'brilliant', 'professor', ',', 'and', 'another', 'fig', 'was', 'Ee',
'Gee', ',', 'the', 'amazing', 'editor', ',', 'and', 'another', 'fig', 'was', 'Europe', 'and', 'Africa', 'and', 'South', 'America', ',',
'and', 'another', 'fig', 'was', 'Constantin', 'and', 'Socrates', 'and', 'Attila', 'and', 'a', 'pack', 'of', 'other', 'lovers', 'with',
'queer', 'names', 'and', 'offbeat', 'professions', ',', 'and', 'another', 'fig', 'was', 'an', 'Olympic', 'lady', 'crew', 'champion',
',', 'and', 'beyond', 'and', 'above', 'these', 'figs', 'were', 'many', 'more', 'figs', 'I', 'could', "n't", 'quite', 'make', 'out',
'.', 'I', 'saw', 'myself', 'sitting', 'in', 'the', 'crotch', 'of', 'this', 'fig', 'tree', ',', 'starving', 'to', 'death', ',', 'just',
'because', 'I', 'could', "n't", 'make', 'up', 'my', 'mind', 'which', 'of', 'the', 'figs', 'I', 'would', 'choose', '.', 'I', 'wanted',
'each', 'and', 'every', 'one', 'of', 'them', ',', 'but', 'choosing', 'one', 'meant', 'losing', 'all', 'the', 'rest', ',', 'and', ',',
'as', 'I', 'sat', 'there', ',', 'unable', 'to', 'decide', ',', 'the', 'figs', 'began', 'to', 'wrinkle', 'and', 'go', 'black', ',', 'an
d', ',', 'one', 'by', 'one', ',', 'they', 'plopped', 'to', 'the', 'ground', 'at', 'my', 'feet', '.']
```

2) Without Using NLTK

Input :

```
#Without NLTK
tokens_without_nltk = text.split()

print(tokens_without_nltk)
```

Output :

```
['I', 'saw', 'my', 'life', 'branching', 'out', 'before', 'me', 'like', 'the', 'green', 'fig', 'tree', 'in', 'the', 'story.', 'F
rom', 'the', 'tip', 'of', 'every', 'branch,', 'like', 'a', 'fat', 'purple', 'fig,', 'a', 'wonderful', 'future', 'beckoned', 'an
d', 'winked.', 'One', 'fig', 'was', 'a', 'husband', 'and', 'a', 'happy', 'home', 'and', 'children,', 'and', 'another', 'fig',
'was', 'a', 'famous', 'poet', 'and', 'another', 'fig', 'was', 'a', 'brilliant', 'professor,', 'and', 'another', 'fig', 'was',
'Ee', 'Gee,', 'the', 'amazing', 'editor,', 'and', 'another', 'fig', 'was', 'Europe', 'and', 'Africa', 'and', 'South', 'Americ
a,', 'and', 'another', 'fig', 'was', 'Constantin', 'and', 'Socrates', 'and', 'Attila', 'and', 'a', 'pack', 'of', 'other', 'love
rs', 'with', 'queer', 'names', 'and', 'offbeat', 'professions,', 'and', 'another', 'fig', 'was', 'an', 'Olympic', 'lady', 'cre
w', 'champion,', 'and', 'beyond', 'and', 'above', 'these', 'figs', 'were', 'many', 'more', 'figs', 'I', "couldn't", 'quite', 'm
ake', 'out.', 'I', 'saw', 'myself', 'sitting', 'in', 'the', 'crotch', 'of', 'this', 'fig', 'tree,', 'starving', 'to', 'death,',
'just', 'because', 'I', "couldn't", 'make', 'up', 'my', 'mind', 'which', 'of', 'the', 'figs', 'I', 'would', 'choose.', 'I', 'wa
nted', 'each', 'and', 'every', 'one', 'of', 'them,', 'but', 'choosing', 'one', 'meant', 'losing', 'all', 'the', 'rest,', 'an
d,', 'as', 'I', 'sat', 'there,', 'unable', 'to', 'decide,', 'the', 'figs', 'began', 'to', 'wrinkle', 'and', 'go', 'black,', 'an
d,', 'one', 'by', 'one,', 'they', 'plopped', 'to', 'the', 'ground', 'at', 'my', 'feet.']
```

Regional language filtration

Input :

```
#Regional language filtration
def is_hindi_char(char):
    return '\u0900' <= char <= '\u097F'

sentence = '''This is a verse from Rashmirathi : ‘उदयाचल मेरा दीप्त भाल,
भूमंडल वक्षस्थल विशाल,
भुज परिधि-बन्ध को घेरे हैं,
मैनाक-मेरु पग मेरे हैं।
दिपते जो ग्रह नक्षत्र निकर,
सब हैं मेरे मुख के अन्दर । '''

hindi_words = []

current_word = ""

for char in sentence:
    if is_hindi_char(char):
        current_word += char
    else:
        if current_word:  #if there is some content inside current_word string then it will enter the if block
            hindi_words.append(current_word)
            current_word = ""

#if the last word is hindi then to append that word in the hindi_words list this line of code is useful
if current_word:
    hindi_words.append(current_word)

print("Detected Hindi words:", hindi_words)
```

Output :

```
Detected Hindi words: ['उदयाचल', 'मेरा', 'दीप्त', 'भाल', 'भूमंडल', 'वक्षस्थल', 'विशाल', 'भुज', 'परिधि', 'बन्ध', 'को', 'घेरे', 'हैं', 'मैनाक',
'मेरु', 'पग', 'मेरे', 'हैं।', 'दिपते', 'जो', 'ग्रह', 'नक्षत्र', 'निकर', 'सब', 'हैं', 'मेरे', 'मुख', 'के', 'अन्दर', '।']
```

Stop word filtration

Input :

```
#Stop word filtration
#Filtration - Filtration involves removing unwanted elements from the text, such as punctuation and stopwords.

from nltk.corpus import stopwords
import string

#nltk.download('stopwords')

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

filtered_tokens = [word for word in tokens if word.lower() not in stop_words and word.lower() not in punctuation]
print(filtered_tokens)
```

Output :

```
['saw', 'life', 'branching', 'like', 'green', 'fig', 'tree', 'story', 'tip', 'every', 'branch', 'like', 'fat', 'purple', 'fig', 'wonder
ful', 'future', 'beckoned', 'winked', 'One', 'fig', 'husband', 'happy', 'home', 'children', 'another', 'fig', 'famous', 'poet', 'anothe
r', 'fig', 'brilliant', 'professor', 'another', 'fig', 'Ee', 'Gee', 'amazing', 'editor', 'another', 'fig', 'Europe', 'Africa', 'South',
'America', 'another', 'fig', 'Constantin', 'Socrates', 'Attila', 'pack', 'lovers', 'queer', 'names', 'offbeat', 'professions', 'anothe
r', 'fig', 'Olympic', 'lady', 'crew', 'champion', 'beyond', 'figs', 'many', 'figs', 'could', "n't", 'quite', 'make', 'saw', 'sitting',
'crotch', 'fig', 'tree', 'starving', 'death', 'could', "n't", 'make', 'mind', 'figs', 'would', 'choose', 'wanted', 'every', 'one', 'cho
osing', 'one', 'meant', 'losing', 'rest', 'sat', 'unable', 'decide', 'figs', 'began', 'wrinkle', 'go', 'black', 'one', 'one', 'ploppe
d', 'ground', 'feet']
```

Punctuation Filtration

Input :

```
#Punctuation Filtration
import string

punct = string.punctuation

sentence = '''there's a bluebird in my heart that wants to get out but I'm too tough for him, I say,
stay in there, I'm not going to let anybody see you.'''

filtered_sent = ""

for str in sentence:
    if str not in punct:
        filtered_sent += str

print("Filtered Senetence (without any puncutation) : " , filtered_sent)
```

Output :

```
Filtered Senetence (without any puncutation) :  theres a bluebird in my heart that wants to get out but Im too tough for him I say
stay in there Im not going to let anybody see you
```

Email Validation

Input :

```python
#Email Validation
import re

def is_valid_email(email):
    exp = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

    match = re.match(exp, email)

    return bool(match)

email = input("Your email : ")

if is_valid_email(email):
    print(f"{email} is a valid email.")
else:
    print(f"{email} is not a valid email.")
```

Output :

```
Your email : shreya@ves.ac.in
shreya@ves.ac.in is a valid email.
```

Phone number validation

Input :

```python
#phone number validation
import re

def is_valid_phone_number(phone_number):
    pattern = r'^\+\d{10,15}$'

    match = re.match(pattern, phone_number)
    return bool(match)

phone_number = input("Enter a phone number in international format (+1234567890): ")

if is_valid_phone_number(phone_number):
    print("Valid phone number")
else:
    print("Invalid phone number")
```

Output :

```
Enter a phone number in international format (+1234567890): +9119451557
Valid phone number
```

Name Validation

Input :

```
#Name Validation
import re

def is_valid_name(name):
    pattern = r'^[a-zA-Z\-\_\'\s]+$'
    match = re.match(pattern, name)
    return bool(match)

name = input("Enter a name: ")

if is_valid_name(name):
    print("Valid name")
else:
    print("Invalid name")
```

Output :

```
Enter a name: sus_shreyaww
Valid name
```

Filtration

Input :

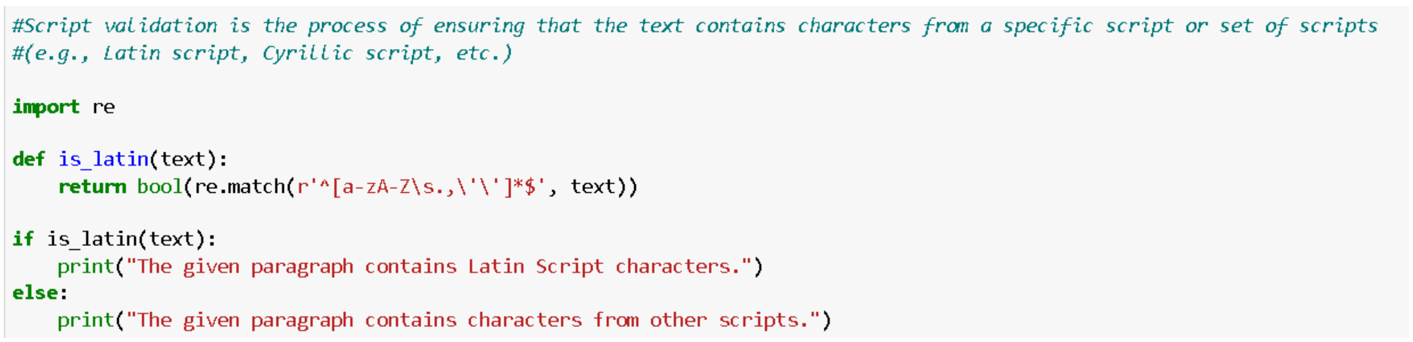
```
#Filtration - Filtration involves removing unwanted elements from the text, such as punctuation and stopwords.

from nltk.corpus import stopwords
import string

#nltk.download('stopwords')

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

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

Output :

```
['saw', 'life', 'branching', 'like', 'green', 'fig', 'tree', 'story', 'tip', 'every', 'branch', 'like', 'fat', 'purple', 'fig', 'wonder
ful', 'future', 'beckoned', 'winked', 'One', 'fig', 'husband', 'happy', 'home', 'children', 'another', 'fig', 'famous', 'poet', 'anothe
r', 'fig', 'brilliant', 'professor', 'another', 'fig', 'Ee', 'Gee', 'amazing', 'editor', 'another', 'fig', 'Europe', 'Africa', 'South',
'America', 'another', 'fig', 'Constantin', 'Socrates', 'Attila', 'pack', 'lovers', 'queer', 'names', 'offbeat', 'professions', 'anothe
r', 'fig', 'Olympic', 'lady', 'crew', 'champion', 'beyond', 'figs', 'many', 'figs', 'could', "n't", 'quite', 'make', 'saw', 'sitting',
'crotch', 'fig', 'tree', 'starving', 'death', 'could', "n't", 'make', 'mind', 'figs', 'would', 'choose', 'wanted', 'every', 'one', 'cho
osing', 'one', 'meant', 'losing', 'rest', 'sat', 'unable', 'decide', 'figs', 'began', 'wrinkle', 'go', 'black', 'one', 'one', 'ploppe
d', 'ground', 'feet']
```

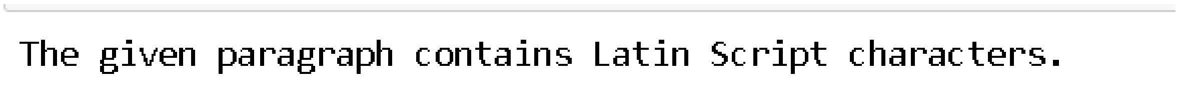
Script Validation

Input :



```python
#Script validation is the process of ensuring that the text contains characters from a specific script or set of scripts
#(e.g., Latin script, Cyrillic script, etc.)

import re

def is_latin(text):
    return bool(re.match(r'^[a-zA-Z\s.,\'\']*$', text))

if is_latin(text):
    print("The given paragraph contains Latin Script characters.")
else:
    print("The given paragraph contains characters from other scripts.")
```

Output :



```
The given paragraph contains Latin Script characters.
```

Github Repo

https://github.com/Shreyaww/Sem7_NLP/blob/main/Experiment2.ipynb