```
import re
text = """Contact us at support@example.com or call +91-9876543210.
Visit our website https://www.mywebsite.org for details.
Follow us on Twitter @TechGuru and use the hashtag #AI2025.
Meeting scheduled on 28/07/2025. Beware of badword1 and badword2."""
emails = re.findall(r"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}",
text)
phones = re.findall(r"\+91-\d{10}", text)
urls = re.findall(r"https?://\S+", text)
hashtags = re.findall(r"#\w+", text)
mentions = re.findall(r"@\w+", text)
offensive_words = [word for word in ["badword1", "badword2",
"spamword"] if
word in text]
print("emails:", emails)
print("phones:", phones)
print("urls:", urls)
print("hashtags:", hashtags)
print("mentions:", mentions)
print("offensive words:", offensive_words)
```

```
emails: ['support@example.com']
phones: ['+91-9876543210']
urls: ['https://www.mywebsite.org']
hashtags: ['#AI2025']
mentions: ['@example', '@TechGuru']
offensive words: ['badword1', 'badword2', 'Follow']
```

```
import nltk, spacy
from transformers import BertTokenizer
nltk.download('punkt_tab')
nlp = spacy.load("en_core_web_sm")
text = "Artificial Intelligence is revolutionizing the world."
# Word-level tokenization
nltk_tokens = nltk.word_tokenize(text)
spacy_tokens = [token.text for token in nlp(text)]
# Character-level tokenization
char_tokens = list(text)
# Subword tokenization (WordPiece - BERT)
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
subword_tokens = tokenizer.tokenize(text)
print("NLTK Word Tokens:", nltk_tokens)
print("SpaCy Word Tokens:", spacy_tokens)

print("Character Tokens:", char_tokens[:20]) # first 20 chars
print("Subword Tokens (WordPiece):", subword_tokens)
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py
```

```
:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"648b6c439f4b450ca5f979d737093061","version_major":2,"vers
ion_minor":0}

{"model_id":"58b476c024024608a1d4fc9b5472aac5","version_major":2,"vers
ion_minor":0}

{"model_id":"1539fdfca94f4a72b98c670d3cea311c","version_major":2,"vers
ion_minor":0}

{"model_id":"8e953830877749cfa3e671398b312571","version_major":2,"vers
ion_minor":0}
```


```
NLTK Word Tokens: ['Artificial', 'Intelligence', 'is',
'revolutionizing', 'the', 'world', '.']
SpaCy Word Tokens: ['Artificial', 'Intelligence', 'is',
'revolutionizing', 'the', 'world', '.']
Character Tokens: ['A', 'r', 't', 'i', 'f', 'i', 'c', 'i', 'a', 'l', '
', 'I', 'n', 't', 'e', 'l', 'l', 'i', 'g', 'e']
Subword Tokens (WordPiece): ['artificial', 'intelligence', 'is',
'revolution', '##izing', 'the', 'world', '.']
```

```python
import nltk

nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')
# or, to fetch everything:
nltk.download('all')

# Install required packages if not already installed:
# pip install nltk spacy transformers

import nltk
import spacy
from transformers import BertTokenizer

# Download required resources
nltk.download('punkt')
nlp = spacy.load('en_core_web_sm')
```

```
# Given text
text = "Artificial Intelligence is revolutionizing the world."

# a) Word-level Tokenization using NLTK
nltk_tokens = nltk.word_tokenize(text)

# a) Word-level Tokenization using SpaCy
spacy_doc = nlp(text)
spacy_tokens = [token.text for token in spacy_doc]

# b) Character-level Tokenization
char_tokens = list(text)

# c) Subword Tokenization using WordPiece (BERT tokenizer from Hugging
Face)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
subword_tokens = tokenizer.tokenize(text)

# d) Compare the outputs
print("a) Word-level Tokenization:")
print("   NLTK:", nltk_tokens)
print("   SpaCy:", spacy_tokens)

print("\nb) Character-level Tokenization:")
print("   Characters:", char_tokens)

print("\nc) Subword Tokenization (WordPiece using BERT):")
print("   Subwords:", subword_tokens)

print("\nd) Comparison Summary:")
print("""
- NLTK and SpaCy both provide word-level tokens, but SpaCy also
includes punctuations as separate tokens.
- Character-level tokenization splits every character, including
spaces and punctuation.
- WordPiece (subword) tokenization breaks complex or unknown words
into smaller known units, e.g., "revolutionizing" → ["re",
"##volution", "##izing"].
""")

import nltk
nltk.download('punkt')
nltk.download('punkt_tab')   # explicitly include punkt_tab

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import spacy

# Download resources for NLTK
nltk.download('stopwords')
```

```
nltk.download('punkt')

# Define text
text = "This is an example sentence, showing the effect of stopword
removal."

# NLTK stopword removal
words = word_tokenize(text)
stop_words = set(stopwords.words('english'))
nltk_filtered = [w for w in words if w.lower() not in stop_words]

# SpaCy stopword removal
nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
spacy_filtered = [token.text for token in doc if not token.is_stop]

# Word counts comparison
print("Word counts before/after removal")
print(f"NLTK: before={len(words)}, after={len(nltk_filtered)}")
print(f"SpaCy: before={len([token.text for token in doc])},
after={len(spacy_filtered)}")

# Display results
print("\nNLTK filtered tokens:", nltk_filtered)
print("SpaCy filtered tokens:", spacy_filtered)

import nltk
from nltk.stem import PorterStemmer, LancasterStemmer,
WordNetLemmatizer
from nltk import pos_tag
from nltk.corpus import wordnet

# Make sure necessary corpora are available
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

words = ["running", "flies", "better", "studies", "wolves", "cities"]

porter = PorterStemmer()
lancaster = LancasterStemmer()
lemmatizer = WordNetLemmatizer()

# Helper to map NLTK POS tags to WordNet POS
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'): return wordnet.ADJ
    if treebank_tag.startswith('V'): return wordnet.VERB
    if treebank_tag.startswith('N'): return wordnet.NOUN
    if treebank_tag.startswith('R'): return wordnet.ADV
```

```
    return wordnet.NOUN

# a) Apply Porter and Lancaster stemmers
porter_output = [porter.stem(w) for w in words]
lancaster_output = [lancaster.stem(w) for w in words]

# b) Lemmatize using POS tagging
pos_tags = pos_tag(words)  # get POS for each
lemmatized = [
    lemmatizer.lemmatize(w, get_wordnet_pos(tag))
    for w, tag in pos_tags
]

# c) Compare vocabulary size reduction
orig_vocab = set(words)
porter_vocab = set(porter_output)
lanc_vocab = set(lancaster_output)
lemma_vocab = set(lemmatized)

print("Original:", words)
print("Porter stems:", porter_output)
print("Lancaster stems:", lancaster_output)
print("Lemmatized:", lemmatized)

print("\nVocabulary sizes:")
print(f"Original size = {len(orig_vocab)}")
print(f"Porter size = {len(porter_vocab)}")
print(f"Lancaster size = {len(lanc_vocab)}")
print(f"Lemmatized size = {len(lemma_vocab)}")

import re

# Input text
text = "RT @user123!!! The PRICE of Bitcoin hit $30,000 today!!!
#Crypto □"

# a) Convert text to lowercase
cleaned_text = text.lower()

# b) Remove punctuation, special symbols, hashtags, and mentions
cleaned_text = re.sub(r'[@#\$\w]+', '', cleaned_text)  # Remove
mentions, hashtags, and currency symbols
cleaned_text = re.sub(r'[^\w\s]', '', cleaned_text)  # Remove
punctuation

# c) Remove numbers
cleaned_text = re.sub(r'\d+', '', cleaned_text)  # Remove digits

# d) Display cleaned text
print("Cleaned Text:", cleaned_text)
```

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

# Sample text
text = "John loves eating pizza while Mary reads books in the
library."

# Tokenize the text
words = word_tokenize(text)

# Perform POS tagging
nltk_pos_tags = pos_tag(words)

# Display the results
print("NLTK POS Tagging:")
for word, tag in nltk_pos_tags:
    print(f"{word}: {tag}")

import spacy

# Load the SpaCy English model
nlp = spacy.load("en_core_web_sm")

# Process the text
doc = nlp(text)

# Display the results
print("\nSpaCy POS Tagging:")
for token in doc:
    print(f"{token.text}: {token.pos_}")

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

# Input sentence
sentence = "The dog chased the cat."

# Tokenize and POS tagging
tokens = word_tokenize(sentence)
nltk_tags = pos_tag(tokens)

print("NLTK POS Tags:", nltk_tags)

import spacy

# Load SpaCy's English model
nlp = spacy.load("en_core_web_sm")

# Process the sentence
```

```
doc = nlp(sentence)

# Extract tokens and their POS tags
spacy_tags = [(token.text, token.pos_) for token in doc]

print("SpaCy POS Tags:", spacy_tags)

import nltk
from nltk.tokenize import word_tokenize
from nltk import RegexpTagger

# Define regular expression patterns for POS tagging
patterns = [
    (r'\b(the|a|an)\b', 'DT'),  # Determiners
    (r'\b(?:dog|cat|cricket|TV)\b', 'NN'),  # Nouns
    (r'\b(?:plays|watches)\b', 'VB'),  # Verbs
    (r'\b(?:daily)\b', 'RB'),  # Adverb
    (r'\b(?:and)\b', 'CC'),  # Conjunction
    (r'\b(?:Ravi)\b', 'NNP'),  # Proper noun
    (r'\b\w+\b', 'NN')  # Default: Noun
]

# Create a RegexpTagger with the defined patterns
regexp_tagger = RegexpTagger(patterns)

# Input sentence
sentence = "Ravi plays cricket and watches TV daily."

# Tokenize and apply the tagger
tokens = word_tokenize(sentence)
tagged = regexp_tagger.tag(tokens)

print(tagged)

import nltk
from nltk.corpus import brown
from nltk.tokenize import word_tokenize

# Ensure necessary resources are downloaded
nltk.download('brown')
nltk.download('punkt')

# Prepare training and testing data
train_sents = brown.tagged_sents(categories='news')[:3000]
test_sents = brown.tagged_sents(categories='news')[3000:3100]

# Train the HMM tagger
trainer = nltk.tag.HiddenMarkovModelTrainer()
hmm_tagger = trainer.train(train_sents)

# Test sentence
```

```
test_sentence = "The quick brown fox jumps over the lazy dog."
tokens = word_tokenize(test_sentence)
tagged = hmm_tagger.tag(tokens)
print("Answer:")
print(tagged)

from transformers import pipeline

# Load the POS tagging pipeline
pos_tagger = pipeline("ner", model="dbmdz/bert-large-cased-finetuned-
conll03-english")

# Input sentence
sentence = "Elon Musk founded SpaceX in 2002."

# Apply POS tagging
tags = pos_tagger(sentence)

print(tags)
```