1. Tokenization & Stop Word Removal

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
nltk.download('punkt_tab')
nltk.download('stopwords')
text = "Natural Language Processing is a fascinating field of AI."
# Tokenization
tokens = word_tokenize(text)
print("Tokens:", tokens)
# Stopword Removal
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print("Filtered Tokens (No Stopwords):", filtered_tokens)
```

```
Tokens: ['Natural', 'Language', 'Processing', 'is', 'a', 'fascinating', 'field', 'of', 'AI', '.']
Filtered Tokens (No Stopwords): ['Natural', 'Language', 'Processing', 'fascinating', 'field', 'AI', '.']
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

2. Stemming & Lemmatization

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

nltk.download('wordnet')
nltk.download('omw-1.4')

text = "The leaves on the trees were falling gracefully"

# Tokenize the text
tokens = word_tokenize(text)

# Stemming
stemmer = PorterStemmer()
stemmed = [stemmer.stem(word) for word in tokens]
print("Stemmed Words:", stemmed)

# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized = [lemmatizer.lemmatize(word) for word in tokens]
print("Lemmatized Words:", lemmatized)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
Stemmed Words: ['the', 'leav', 'on', 'the', 'tree', 'were', 'fall', 'grace']
Lemmatized Words: ['The', 'leaf', 'on', 'the', 'tree', 'were', 'falling', 'gracefully']
```

3. POS Tagging

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

nltk.download('averaged_perceptron_tagger_eng')

text = "The quick brown fox jumps over the lazy dog"
tokens = word_tokenize(text)

# POS Tagging
pos_tags = pos_tag(tokens)
print("POS Tags:", pos_tags)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
POS Tags: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy',
```

4. Top Down Parsing (Recursive Descent Parsing) & Bottom Up Parsing (Shift Reduce Parser)

```
import nltk
from nltk import CFG
from nltk.parse import RecursiveDescentParser, ShiftReduceParser

# Define a simple grammar
grammar = CFG.fromstring("""
S -> NP VP
NP -> Det N
VP -> V NP
Det -> 'the' | 'a'
N -> 'cat' | 'dog'
V -> 'chased' | 'saw'
""")

# Tokenized sentence
sentence = ['the', 'cat', 'chased', 'a', 'dog']

# Top Down Parsing (Recursive Descent)
print("Top Down Parsing (Recursive Descent):")
rd_parser = RecursiveDescentParser(grammar)
for tree in rd_parser.parse(sentence):
    print(tree)

# Bottom Up Parsing (Shift-Reduce)
print("\nBottom Up Parsing (Shift-Reduce):")
sr_parser = ShiftReduceParser(grammar)
for tree in sr_parser.parse(sentence):
    print(tree)
```

```
Top Down Parsing (Recursive Descent):
(S (NP (Det the) (N cat)) (VP (V chased) (NP (Det a) (N dog))))

Bottom Up Parsing (Shift-Reduce):
(S (NP (Det the) (N cat)) (VP (V chased) (NP (Det a) (N dog))))
```

Start coding or generate with AI.