```
import nltk
import re
```

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

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[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.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.tokenize import word_tokenize
```

+ Code \+ Text

```
# Initialize WordNet lemmatizer
lemmatizer = WordNetLemmatizer()

# Input text
input_text = "The quick brown foxes are jumping over the lazy dogs."

# Tokenize the input text
tokens = word_tokenize(input_text)

# Initialize lists to store morphological analysis and word generation results
morphological_analysis = []
word_generation = []
```

```
# Function to get WordNet POS tags from Penn Treebank POS tags
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN  # Default to noun
```

```
# Perform morphological analysis and word generation
for token in tokens:
    # Morphological Analysis
    pos_tags = nltk.pos_tag([token])
    lemma = lemmatizer.lemmatize(token, get_wordnet_pos(pos_tags[0][1]))
    morphological_analysis.append(f"Token: {token}, Lemma: {lemma}, POS: {pos_tags[0][1]}")

    # Word Generation Examples
    generated_words = []

    # Example 1: Verb conjugation (if the token is a verb)
    if pos_tags[0][1].startswith('V'):
        present_tense = token
        past_tense = lemma + 'ed'
        generated_words.append(f"Present tense: {present_tense}")
        generated_words.append(f"Past tense: {past_tense}")

    # Example 2: Noun pluralization (if the token is a noun)
    elif pos_tags[0][1].startswith('N'):
        plural = token + 's'
        generated_words.append(f"Plural form: {plural}")

    # Example 3: Adjective comparison (if the token is an adjective)
    elif pos_tags[0][1].startswith('J'):
        comparative = lemma + 'er'
        superlative = lemma + 'est'
        generated_words.append(f"Comparative form: {comparative}")
        generated_words.append(f"Superlative form: {superlative}")

    # Example 4: Adverb formation (if the token is an adjective)
    elif pos_tags[0][1].startswith('R'):
        adverb = lemma + 'ly'
        generated_words.append(f"Adverb form: {adverb}")

    # Example 5: Adjective to Adverb Conversion
    elif pos_tags[0][1].startswith('J'):
        comparative = lemma + 'er'
        superlative = lemma + 'est'
        adverb = lemma + 'ly'
        generated_words.append(f"Comparative form: {comparative}")
        generated_words.append(f"Superlative form: {superlative}")
        generated_words.append(f"Adverb form: {adverb}")

    # Example 6: Noun to Verb Conversion
    elif pos_tags[0][1].startswith('N'):
        verb_form = lemma + 'ize'
        generated_words.append(f"Verb form from noun: {verb_form}")

    # Example 7: Verb to Noun Conversion
    elif pos_tags[0][1].startswith('V'):
        noun_form = lemma + 'tion'
        generated_words.append(f"Noun form from verb: {noun_form}")

    # Example 8: Synonym Generation (using NLTK's WordNet)
    synonyms = set()
    for syn in wordnet.synsets(token):
        for lemma in syn.lemmas():
            synonyms.add(lemma.name())
    generated_words.append(f"Synonyms: {', '.join(synonyms)}")

    # Example 9: Antonym Generation (using NLTK's WordNet)
    antonyms = set()
    for syn in wordnet.synsets(token):
        for lemma in syn.lemmas():
            if lemma.antonyms():
                antonyms.add(lemma.antonyms()[0].name())
    generated_words.append(f"Antonyms: {', '.join(antonyms)}")

    word_generation.append(f"Token: {token}, Generated Words: {', '.join(generated_words)}")
```

```
# Print the results
print("Morphological Analysis:")
for analysis in morphological_analysis:
    print(analysis)

print("\nWord Generation:")
for generated_word in word_generation:
    print(generated_word)
```

```
Morphological Analysis:
Token: The, Lemma: The, POS: DT
Token: quick, Lemma: quick, POS: NN
Token: brown, Lemma: brown, POS: NN
Token: foxes, Lemma: fox, POS: NNS
Token: are, Lemma: be, POS: VBP
Token: jumping, Lemma: jumping, POS: NN
Token: over, Lemma: over, POS: IN
Token: the, Lemma: the, POS: DT
Token: lazy, Lemma: lazy, POS: NN
Token: dogs, Lemma: dog, POS: NNS
Token: ., Lemma: ., POS: .

Word Generation:
Token: The, Generated Words: Synonyms: , Antonyms: 
Token: quick, Generated Words: Plural form: quicks, Synonyms: straightaway, speedy, ready, spry, quickly, prompt, nimble, quick, agile, fast,
Token: brown, Generated Words: Plural form: browns, Synonyms: embrown, Robert_Brown, browned, chocolate-brown, brownish, brown, John_Brown, Br
Token: foxes, Generated Words: Plural form: foxess, Synonyms: dodger, befuddle, fuddle, confound, slyboots, Charles_James_Fox, throw, bedevil,
Token: are, Generated Words: Present tense: are, Past tense: beed, Synonyms: live, be, personify, exist, ar, make_up, represent, comprise, fol
Token: jumping, Generated Words: Plural form: jumpings, Synonyms: alternate, parachute, start, skip, pass_over, jump, stand_out, skip_over, ju
Token: over, Generated Words: Synonyms: complete, concluded, all_over, o'er, ended, terminated, over, Antonyms: 
Token: the, Generated Words: Synonyms: , Antonyms: 
Token: lazy, Generated Words: Plural form: lazys, Synonyms: slothful, indolent, lazy, faineant, work-shy, otiose, Antonyms: 
Token: dogs, Generated Words: Plural form: dogss, Synonyms: track, firedog, detent, frump, chase, give_chase, domestic_dog, hotdog, dog, go_af
Token: ., Generated Words: Synonyms: , Antonyms: 
```