```
import re
text1 = "DSA-0316 Natural Language Processing"
text2 = "DSA-0314 Natural Language"
word = "Natural Language Processing"
pattern = fr'\b{word}\b'
match1 = re.search(pattern, text1)
if match1:
    print("found the text")
else:
    print("not found in text")
match2 = re.search(pattern, text2)
if match2:
    print("found the text")
else:
    print("not found in text")
```

```
found the text
not found in text
```

```
INITIAL_STATE = 'q0'
ACCEPTING_STATE = 'q2'
TRANSITIONS = {
    'q0': {'a': 'q1', 'b': 'q0'},
    'q1': {'a': 'q1', 'b': 'q2'},
    'q2': {'a': 'q1', 'b': 'q0'}
}
def process_string(test_strings):
    current_state = INITIAL_STATE
    for char in test_strings:
        if char in TRANSITIONS.get(current_state, {}):
            current_state = TRANSITIONS[current_state][char]
        else:
            current_state = INITIAL_STATE
    return current_state == ACCEPTING_STATE
test_strings = ["ab", "aab", "bab", "bbaaab", "a", "b", "abc"]
for string in test_strings:
    if process_string(string):
        print(f"'{string}' is accepted by the FSA.")
    else:
        print(f"'{string}' is not accepted by the FSA.")
```

```
'ab' is accepted by the FSA.
'aab' is accepted by the FSA.
'bab' is accepted by the FSA.
'bbaaab' is accepted by the FSA.
'a' is not accepted by the FSA.
'b' is not accepted by the FSA.
'abc' is not accepted by the FSA.
```

```
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
text = "Hello, Students Welcome to SSE."
tokens = word_tokenize(text)
print("Word Tokens:")
print(tokens)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Word Tokens:
['Hello', ',', 'Students', 'Welcome', 'to', 'SSE', '.']
```

```
import nltk
from nltk import RegexpTokenizer

class PluralFiniteStateMachine:
    transitions = [
        (1, '([^aeiouy])$', r'\1\1'),        # Rule 1: Add 's' -> cats, dogs
        (2, '([aeiouy])$', r'\1s'),          # Rule 2: Add 'es' -> trees, boys
        (3, '([^aeiouy])y$', r'\1ies'),      # Rule 3: Change 'y' to 'ies' -> citie
        (4, '(o)$', r'\1es'),                # Rule 4: Add 'es' -> potatoes, tomatc
        (5, '([^sxz])$', r'\1s'),            # Rule 5: Add 's' -> books, pens
        (6, '(.*)', r'\1s')                  # Default: Add 's' -> nouns ending in
    ]

    tokenizer = RegexpTokenizer(r'\b\w+\b')

    def pluralize(noun):
        tokens = PluralFiniteStateMachine.tokenizer.tokenize(noun.lower())
        if tokens:
            singular = tokens[0]
            for state, pattern, replacement in PluralFiniteStateMachine.transitions
                if nltk.re.match(pattern, singular):
                    plural = nltk.re.sub(pattern, replacement, singular)
                    return plural
        return None
singular_nouns = ["cat", "dog", "tree", "boy", "city", "baby", "potato", "tomato",

for noun in singular_nouns:
    plural = PluralFiniteStateMachine.pluralize(noun)
    if plural:
        print(f"Singular: {noun}  =>  Plural: {plural}")
    else:
        print(f"No plural form generated for '{noun}'")
```

```
Singular: cat  =>  Plural: catss
Singular: dog  =>  Plural: dogss
Singular: tree  =>  Plural: treess
Singular: boy  =>  Plural: boyss
Singular: city  =>  Plural: cityss
Singular: baby  =>  Plural: babyss
Singular: potato  =>  Plural: potatoss
Singular: tomato  =>  Plural: tomatoss
```

```
Singular: book  =>  Plural: bookss
Singular: pen  =>  Plural: penss
Singular: bus  =>  Plural: busss
```

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = stopwords.words('english')
# Print some stopwords
print("Some English stopwords:")
print(stop_words[:20])
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print("\n tokens without stopwords:")
print(filtered_tokens)
from nltk.stem import PorterStemmer
porter = PorterStemmer()
stemmed_words = [porter.stem(word) for word in filtered_tokens]
print("\nStemmed Words:")
print(stemmed_words)
```

```
Some English stopwords:
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you'v

 tokens without stopwords:
['Hello', ',', 'Students', 'Welcome', 'SSE', '.']

Stemmed Words:
['hello', ',', 'student', 'welcom', 'sse', '.']
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```