```
import re

text = "The quick brown fox jumps over the lazy dog. Contact us at support@example.com or visit http://examp

words = re.findall(r"\b\w+\b", text)
print("Words in text:", words)

email_match = re.search(r"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}", text)
if email_match:
    print("Found email:", email_match.group())
else:
    print("No email found")

urls = re.findall(r"http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+", text)
print("URLs in text:", urls)

modified_text = re.sub(r"\bdog\b", "cat", text)
print("Text after replacement:", modified_text)

if re.match(r"The", text):
    print("Text starts with 'The'")
else:
    print("Text does not start with 'The'")

words_list = re.split(r"\s+", text)
print("Words split by whitespace:", words_list)
m
```

```
Words in text: ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', 'Contact', 'us',
Found email: support@example.com
URLs in text: ['http://example.com.']
Text after replacement: The quick brown fox jumps over the lazy cat. Contact us at support@example.com
Text starts with 'The'
Words split by whitespace: ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog.', 'Co
```

```
test_strings = ["cab", "aab", "abab", "abc", "bca", "ab", "a"]

for string in test_strings:
    state = 0
    for char in string:
        if state == 0:
            if char == 'a':
                state = 1
            else:
                state = 0
        elif state == 1:
            if char == 'b':
                state = 2
            elif char == 'a':
                state = 1
            else:
                state = 0
        elif state == 2:
            state = 0

    accepted = state == 2
```

```
    print(f"String '{string}' accepted? {accepted}")
```

```
String 'cab' accepted? True
String 'aab' accepted? True
String 'abab' accepted? False
String 'abc' accepted? False
String 'bca' accepted? False
String 'ab' accepted? True
String 'a' accepted? False
```

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import PorterStemmer, WordNetLemmatizer

stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

words = ["running", "jumps", "easily", "studies", "better"]

print("Morphological Analysis:")
print("=" * 30)

for word in words:
    stemmed = stemmer.stem(word)

    lemmatized = lemmatizer.lemmatize(word)

    print(f"Original: {word}")
    print(f"  Stemmed: {stemmed}")
    print(f"  Lemmatized: {lemmatized}")
    print("-" * 30)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
Morphological Analysis:
==============================
Original: running
  Stemmed: run
  Lemmatized: running
------------------------------
Original: jumps
  Stemmed: jump
  Lemmatized: jump
------------------------------
Original: easily
  Stemmed: easili
  Lemmatized: easily
------------------------------
Original: studies
  Stemmed: studi
  Lemmatized: study
------------------------------
Original: better
  Stemmed: better
  Lemmatized: better
------------------------------
```

```
nouns = ["cat", "dog", "bus", "box", "church", "baby", "lady", "toy"]

for noun in nouns:
    state = "initial"

    if noun.endswith(("s", "sh", "ch", "x", "z")):
        state = "add_es"
    elif noun.endswith("y") and not noun[-2] in "aeiou":
        state = "replace_y_with_ies"
    else:
        state = "add_s"

    if state == "add_es":
        plural = noun + "es"
    elif state == "replace_y_with_ies":
        plural = noun[:-1] + "ies"
    elif state == "add_s":
        plural = noun + "s"
    else:
        plural = noun
    print(f"Singular: {noun}, Plural: {plural}")
```

```
Singular: cat, Plural: cats
Singular: dog, Plural: dogs
Singular: bus, Plural: buses
Singular: box, Plural: boxes
Singular: church, Plural: churches
Singular: baby, Plural: babies
Singular: lady, Plural: ladies
Singular: toy, Plural: toys
```

```
import nltk
from nltk.stem import PorterStemmer

nltk.download('punkt')
words = ["running", "runner", "easily", "happiness", "cats", "running", "flies"]

stemmer = PorterStemmer()

stemmed_words = [stemmer.stem(word) for word in words]

for original, stemmed in zip(words, stemmed_words):
    print(f"Original: {original} -> Stemmed: {stemmed}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Original: running -> Stemmed: run
Original: runner -> Stemmed: runner
Original: easily -> Stemmed: easili
Original: happiness -> Stemmed: happi
Original: cats -> Stemmed: cat
Original: running -> Stemmed: run
Original: flies -> Stemmed: fli
```

```
import nltk
from nltk import bigrams
import random
from nltk.corpus import brown
```

```
nltk.download('brown')
nltk.download('punkt')

text = " ".join(brown.words(categories='news'))

words = nltk.word_tokenize(text)

bigrams_list = list(bigrams(words))

bigram_model = {}

for bigram in bigrams_list:
    if bigram[0] not in bigram_model:
        bigram_model[bigram[0]] = []
    bigram_model[bigram[0]].append(bigram[1])

def generate_text(bigram_model, length=20):
    word = random.choice(list(bigram_model.keys()))
    generated_words = [word]

    for _ in range(length - 1):
        next_word = random.choice(bigram_model[word])
        generated_words.append(next_word)
        word = next_word
    return ' '.join(generated_words)
```

```
generated_text = generate_text(bigram_model, length=20)
print("Generated Text:")
print(generated_text)
```

```
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Unzipping corpora/brown.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Generated Text:
disability pension plan alone . Their entry will be enforced now on the outgoing , separately and would
```