```
Start coding or generate with AI.
```

Task

Perform text preprocessing by installing and importing NLTK and spaCy, loading a paragraph, counting its words, converting the text to lowercase, and then summarizing these preprocessing steps and their results.

Install and Import Libraries

Subtask:

Install and import the essential NLP libraries: NLTK and spaCy. This will prepare the environment for text processing.

Reasoning: To begin, I will install the necessary `nltk` and `spaCy` libraries using `pip` to set up the environment for natural language processing.

```
pip install nltk spacy
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages (3.8.11)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nl
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (frc
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (frc
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (frc
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.12/dist-packages (from spac
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/d
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.1
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from spacy)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from sp
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from reque
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from ji
```

```
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8
```

Reasoning: With `nltk` and `spaCy` installed, the next step is to download the necessary English language model for spaCy and then import both libraries to make them ready for use in text processing.

```
import spacy
import nltk

try:
    nlp = spacy.load("en_core_web_sm")
except OSError:
    print("Downloading spaCy model 'en_core_web_sm'...")
    !python -m spacy download en_core_web_sm
    nlp = spacy.load("en_core_web_sm")

print("NLTK and spaCy imported and spaCy model loaded successfully.")
```

```
NLTK and spaCy imported and spaCy model loaded successfully.
```

Load Text

Subtask:

Load a short paragraph of your choice into a string variable in the notebook.

Reasoning:

To continue with the text preprocessing, a sample paragraph is needed. This step involves defining a string variable `paragraph` and assigning a short, representative text to it. This will serve as the input for subsequent NLP operations such as word counting and text normalization.

Reasoning: Following the reasoning to load a sample paragraph, I will now define a Python string variable named `paragraph` and assign a suitable text to it. This variable will be used for subsequent NLP tasks.

```
paragraph = "Natural Language Processing (NLP) is a fascinating field at the intersection of comp
print("Paragraph loaded successfully.")
```

```
Paragraph loaded successfully.
```

Count Words

Subtask:

Count the total number of words in the loaded text using Python's string manipulation and splitting methods.

Reasoning: To count the words in the paragraph, I will first split the paragraph by whitespace to get a list of words, then count the number of elements in the list.

```
words = paragraph.split()
word_count = len(words)
print(f"Total word count in the paragraph: {word_count}")
```

```
Total word count in the paragraph: 63
```

Convert to Lowercase

Subtask:

Convert the entire loaded text to lowercase using Python's built-in string functions.

Reasoning: To normalize the text for further processing, the next step involves converting the entire `paragraph` to lowercase. This is achieved using the `.lower()` string method, which will ensure consistency in word matching and analysis by eliminating case distinctions. The result will be stored in `paragraph_lower`.

Reasoning: To convert the loaded text to lowercase, I will use Python's built-in `.lower()` string method on the `paragraph` variable, store the result in a new variable `paragraph_lower`, and then print a portion of it to verify the conversion.

```
paragraph_lower = paragraph.lower()
print(f"Converted paragraph (first 200 chars): {paragraph_lower[:200]}...")
```

```
Converted paragraph (first 200 chars): natural language processing (nlp) is a fascinating field at
```

Final Task

Subtask:

Summarize the preprocessing steps performed, including the word count and the lowercase conversion, and present the results clearly.

Summary:

Q&A

The preprocessing steps performed involved installing and importing NLTK and spaCy, loading a paragraph, counting its words, and converting the text to lowercase.

- **Word Count**: The paragraph contains 63 words.
- **Lowercase Conversion**: The entire paragraph was successfully converted to lowercase.

Data Analysis Key Findings

- NLTK and spaCy libraries were successfully installed, imported, and the `en_core_web_sm` spaCy model was loaded, preparing the environment for text processing.
- A paragraph detailing Natural Language Processing (NLP) was loaded into a string variable.

- The total word count for the loaded paragraph was determined to be 63 words using Python's string splitting method.
- The entire paragraph was successfully converted to lowercase, normalizing the text for consistent analysis.

Insights or Next Steps

- These preprocessing steps are fundamental for most NLP tasks, ensuring text consistency and preparing it for further analysis such as tokenization, stemming, or sentiment analysis.
- The preprocessed text is now ready for more advanced NLP operations like tokenization, removing stop words, or part-of-speech tagging to extract deeper linguistic insights.