```
install and import essential NLP libraries:NLTK and spaCY
```

```
  File "/tmp/ipython-input-2673306936.py", line 1
    install and import essential NLP libraries:NLTK and spaCY
            ^
SyntaxError: invalid syntax
```

Next steps: Explain error

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
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) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
```

```
import nltk
nltk.download('punkt') # Downloads essential data
nltk.download('averaged_perceptron_tagger') # Example of downloading a tagger
nltk.download('punkt_tab') # Download the missing resource

# Now you can use the library:
from nltk.tokenize import sent_tokenize, word_tokenize

text = "Hello world! This is a simple test."
print(word_tokenize(text))
```

```
[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!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
['Hello', 'world', '!', 'This', 'is', 'a', 'simple', 'test', '.']
```

```
!pip install nltk spacy
```

```
local/lib/python3.12/dist-packages (3.9.1)
/local/lib/python3.12/dist-packages (3.8.11)
/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
r/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
local/lib/python3.12/dist-packages (from nltk) (4.67.1)
3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.15)
=2.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.13)
,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
=8.3.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (8.3.10)
>=0.9.1 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.1.3)
=2.4.3 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.5.2)
.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.4.3)
0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.20.0)
 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.2)
0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.32.4)
,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.12.3)
r/local/lib/python3.12/dist-packages (from spacy) (3.1.6)
 /usr/local/lib/python3.12/dist-packages (from spacy) (75.2.0)
.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (25.0)
```

```
es>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.7.0)
==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (2.41.4)
ions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.15
tion>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.4.2
lizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4.4)
 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.11)
.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.5.0)
.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2025.11.12)
1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.3.3)
0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (0.1.5)
1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (0.23.0)
0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (7.5.0)
.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->spacy) (3.0.3)
/local/lib/python3.12/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.4.2->spacy) (2.0.1)
```

```
import nltk
import spacy
```

text = """ Artificial Intelligence is transforming industries by enabling machines to learn, analyze data, and make decisions. NLP helps computers understand human language. """

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

```
words = text.split()
word_count = len(words)

print("Number of words:", word_count)
```

```
Number of words: 7
```

```
lower_text = text.lower()

print("Text in lowercase:")
print(lower_text)
```

```
Text in lowercase:
hello world! this is a simple test.
```

```
print("Original Text:\n", text)
print("\nLowercase Text:\n", lower_text)
print("\nWord Count:", word_count)
```

```
Original Text:
 Hello world! This is a simple test.

Lowercase Text:
 hello world! this is a simple test.

Word Count: 7
```

Inline Commands and Docstrings Example

This cell demonstrates different types of 'inline commands' common in Colab notebooks and how to write docstrings for functions.

```
# Example of a shell command (an 'inline command' starting with '!')
# This command lists files in the current directory
!ls -la

# Example of a magic command (another type of 'inline command' starting with '%')
# This command measures the execution time of a single statement
%timeit [x**2 for x in range(1000)]


def calculate_square(number):
    """
    Calculates the square of a given number.

    This function takes a single numerical argument and returns its square.

    Args:
        number (int or float): The number to be squared.

    Returns:
        (int or float): The square of the input number.

    Examples:
        >>> calculate_square(5)
        25
        >>> calculate_square(2.5)
        6.25
    """
    return number * number

# Calling the function to demonstrate its use
result = calculate_square(7)
print(f"\nThe square of 7 is: {result}")

# Accessing the docstring
print("\nDocstring for calculate_square function:")
print(calculate_square.__doc__)
```

```
total 16
drwxr-xr-x 1 root root 4096 Dec 11 14:34 .
drwxr-xr-x 1 root root 4096 Jan  6 04:07 ..
drwxr-xr-x 4 root root 4096 Dec 11 14:34 .config
drwxr-xr-x 1 root root 4096 Dec 11 14:34 sample_data
80.3 µs ± 22.4 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)

The square of 7 is: 49

Docstring for calculate_square function:

    Calculates the square of a given number.

    This function takes a single numerical argument and returns its square.

    Args:
        number (int or float): The number to be squared.

    Returns:
        (int or float): The square of the input number.

    Examples:
        >>> calculate_square(5)
        25
        >>> calculate_square(2.5)
        6.25
```