Joining the List:

- `" ".join(...)`: This part takes the newly created list and joins its elements together, using a space `" "` as the separator.
    - The `join()` method is a string method that concatenates the elements of a sequence, in this case, the list of text values.

Jupyter Notebook shortcuts:

```
#### Heading
> Text that will be indented when the
Markdown is rendered.
_Italics_
__Bold__
<div class='alert-success'>alert warning,
alert success, alert danger,
alert-info</div>
```

Is not a stop word?

Whether "not" is considered a stop word can depend on the specific natural language processing (NLP) task and the chosen stop word list.

Traditionally, "not" is often included in stop word lists. This is because it's a very common word that doesn't typically carry much semantic meaning on its own. By removing stop words, we can focus on the more important words in a text.

—

>I learnt regex codes and python list comprehensions

> List comprehension: in is used to tell, that everything in that particular thing has to be covered.

—

Q. why this cleaned_text = ' '.join(lemmatized_words)

and not this cleaned_text=" ".join(word for word in lemmatized_words)

Both methods are valid for joining the words in `lemmatized_words` into a single string with spaces. However, the first method is often preferred due to its simplicity and clarity.

Here's a breakdown of both methods:

Method 1: Using `join()` directly on the list

Python
```
cleaned_text = ' '.join(lemmatized_words)
```

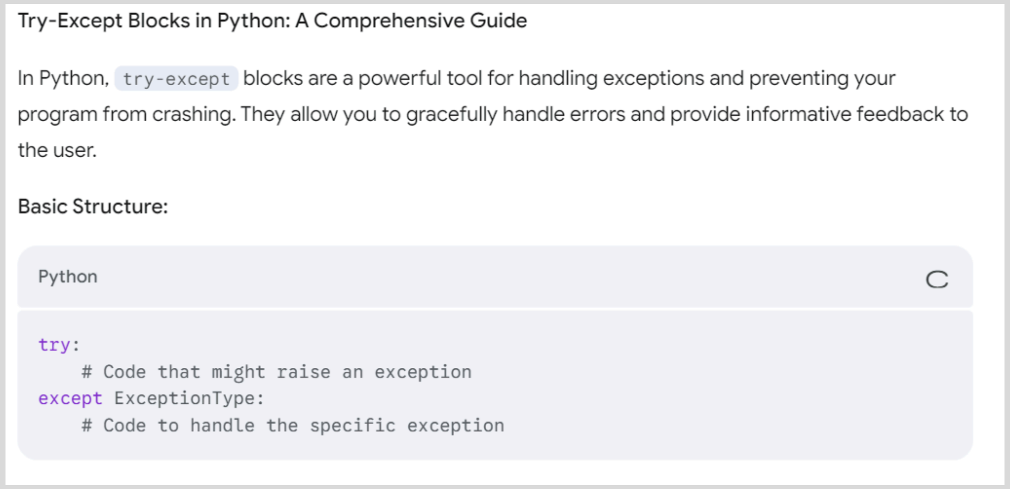
This method directly applies the `join()` method to the `lemmatized_words` list. The `' '` string is used as the separator between words. This is a concise and efficient way to achieve the desired result.

Method 2: Using a list comprehension with `join()`

Python
```
cleaned_text = " ".join(word for word in lemmatized_words)
```

While this method also works, it involves creating a new list using a list comprehension. This can be slightly less efficient, especially for large lists.

In most cases, the first method is sufficient and more readable. However, if you need to perform additional operations on the words during the joining process, the second method with list comprehension can be useful.



Try-Except Blocks in Python: A Comprehensive Guide

In Python, `try-except` blocks are a powerful tool for handling exceptions and preventing your program from crashing. They allow you to gracefully handle errors and provide informative feedback to the user.

Basic Structure:

Python
```
try:
    # Code that might raise an exception
except ExceptionType:
    # Code to handle the specific exception
```

—

In the code snippet, words are stored as keys in the dictionary `word_frequencies`, which is used to calculate the frequency of each word in the text. Dictionaries in Python allow us to associate a **key** (in this case, the word) with a **value** (its frequency). The dictionary's "array-like" behavior lets you access values using keys.

—

Extractive text summarization

As the name suggests, extractive text summarization ‘extracts’ notable information from the large dumps of text provided and groups them into clear and concise summaries.

The method is very straightforward as it extracts texts based on parameters such as the text to be summarized, the most important sentences (Top K), and the value of each of these sentences to the overall subject.

This, however, also means that the method is limited to predetermined parameters that can make extracted text biased under certain conditions.

Owing to its simplicity in most use cases, extractive text summarization is the most common method used by automatic text summarizers.

Abstractive text summarization

Abstractive text summarization generates legible sentences from the entirety of the text provided. It rewrites large amounts of text by creating acceptable representations, which is further processed and summarized by natural language processing.

What makes this method unique is its almost AI-like ability to use a machine’s semantic capability to process text and iron out the kinks using NLP.

Although it might not be as simple to use compared to the extractive method, in many situations, abstract summarization is far more useful. In a lot of ways, it is a precursor to full-fledged AI writing tools. However, this does not mean that there is no need for extractive summarization.