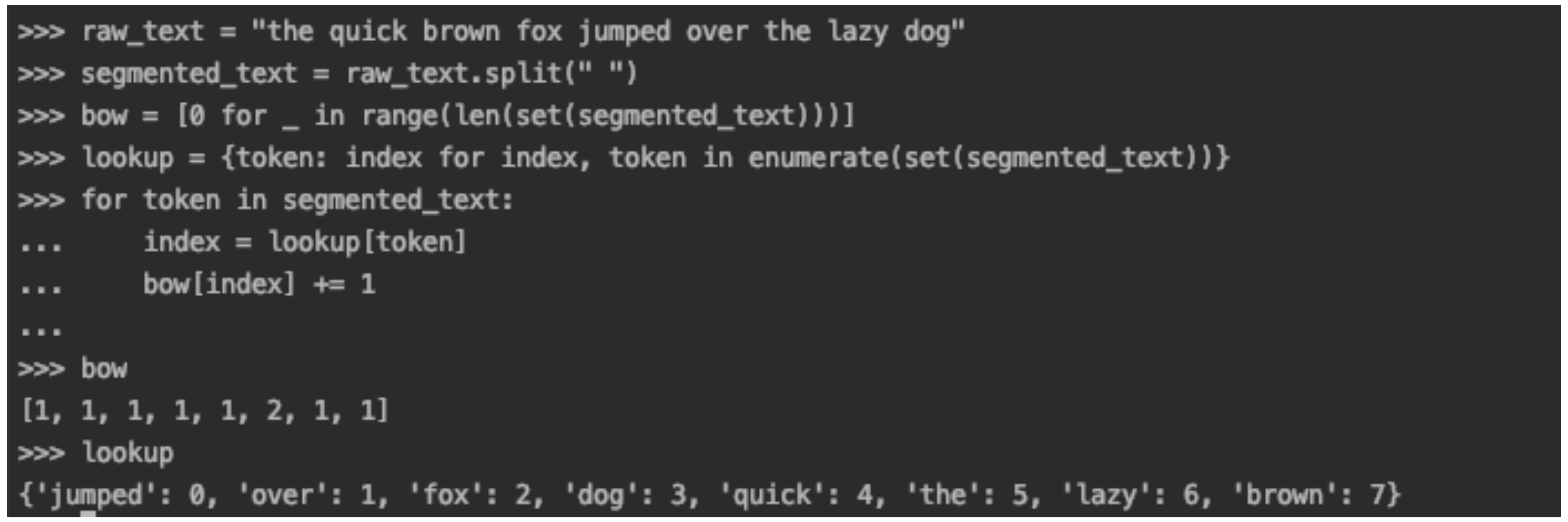
Tokenization

- Very common approach: segment on the space character

raw text: "The quick brown fox jumped over the lazy dog"

segmented text: ["The", "quick", "brown", "fox", "jumped", "over", "the", "lazy", "dog"]

- Once a tokenization scheme is decided on, we typically construct a lookup table mapping each token (or key) to an index that references the feature representation for that token. Here is an example of the BOW representation of the above text:



```
>>> raw_text = "the quick brown fox jumped over the lazy dog"
>>> segmented_text = raw_text.split(" ")
>>> bow = [0 for _ in range(len(set(segmented_text)))]
>>> lookup = {token: index for index, token in enumerate(set(segmented_text))}
>>> for token in segmented_text:
...     index = lookup[token]
...     bow[index] += 1
...
>>> bow
[1, 1, 1, 1, 1, 2, 1, 1]
>>> lookup
{'jumped': 0, 'over': 1, 'fox': 2, 'dog': 3, 'quick': 4, 'the': 5, 'lazy': 6, 'brown': 7}
```

- There are data-driven methods to do this that don't rely on naively splitting on whitespace, and other feature representations too!

Tokenization

- Whitespace tokenization
  - Segment on whitespace, compute vocabulary from top-k ranked words, add extra token for OOV words.
- Character tokenization
  - Segments on characters, very simple, but often limits performance on downstream tasks
- Subword tokenization methods
  - Byte-Pair Encoding (BPE)
    - Recursive algorithm to compute the common unicode character sequences in a dataset. Recursion stops based on frequency hyperparameter.
  - WordPiece
    - Similar to BPE, but instead of adding a sequence based on frequency alone, it normalizes frequency by the frequency of its constituent unicode character(s) / pairs.
  - Unigram Language Model
    - Starts with a complete vocabulary, and progressively shrinks it by removing tokens that result in the bottom percentile of log likelihood loss when removed.
  - SentencePiece
    - Completely agnostic to whitespace by including “\s” in the set of characters it recognizes, and is thus the only language agnostic tokenizer. It uses BPE+Unigram tokenization for subword regularization.