Daily Coding Problem #263

Problem

This problem was asked by Nest.

Create a basic sentence checker that takes in a stream of characters and determines whether they form valid sentences. If a sentence is valid, the program should print it out.

We can consider a sentence valid if it conforms to the following rules:

1. The sentence must start with a capital letter, followed by a lowercase letter or a space.
2. All other characters must be lowercase letters, separators (`,`,`;`,`:`) or terminal marks (`.`,`?`,`!`,`‽`).
3. There must be a single space between each word.
4. The sentence must end with a terminal mark immediately following a word.

Solution

Since the input is a stream, we will only have access to one character at a time. As a result, we must come up with some rules to determine, based on the data we have seen earlier, whether adding the current character creates a valid sentence.

We know that a sentence must end in a terminal punctuation mark, such as a period or question mark. So maybe if the previous character was lowercase and the current character is one of these, we should print out the sentence. But we must also know if a capital letter started the sentence. So maybe we can set a variable to `True` if a capital letter has been seen. But maybe two spaces appeared at some point, and ... it turns out we need a system for these rules.

As it turns out, one system that suits this problem well is a finite state machine. After processing each element in our stream, we can update the state to be one of the following:

- 0: Begin
- 1: Uppercase
- 2: Lowercase
- 3: Space
- 4: Separator

Depending on the current character and state, different rules will apply. For example, if the current state is `0`, meaning the sentence has not started, only a capital letter can follow, moving us to state `1`. If any other character is seen, we remain in state `0`. If at any point when processing the stream we break one of the abovementioned rules, the state gets reset to `0`, and the sentence is reset to the empty string.

Finally, if we manage to reach a terminal punctuation mark, we `yield` the current sentence and reset both the state and sentence.

```
import string

UPPER = string.ascii_uppercase
LOWER = string.ascii_lowercase
SPACE = [' ']
SEP = [',', ';', ':']
TERM = ['.', '?', '!', '‽']

def check_sentences(stream):
    state = 0
    sentence = []
```

```
    while stream:
        char = stream.__next__()
        sentence.append(char)

        if char in UPPER and state == 0:
            state = 1

        elif char in LOWER and state in (1, 2, 3):
            state = 2

        elif char in SPACE and state in (1, 2, 4):
            state = 3

        elif char in SEP and state == 2:
            state = 4

        elif char in TERM and state == 2:
            yield ''.join(sentence)
            state = 0
            sentence = []

        else:
            sentence = []
```

Since appending to a list and updating our state are constant time operations, each element can be processed in `O(1)` time. The space required will be equal to the length of the longest sentence.

Press