Daily Coding Problem #114

Problem

This problem was asked by Facebook.

Given a string and a set of delimiters, reverse the words in the string while maintaining the relative order of the delimiters. For example, given "hello/world:here", return "here/world:hello"

Follow-up: Does your solution work for the following cases: "hello/world:here/", "hello//world:here"

Solution

The algorithm seems rather straightforward, but there are quite a few edge cases. We need to make sure that we handle when there are multiple delimiters in a row, as well as delimiters at the beginning and/or end of the string. One way we can handle these cases is by iterating through the string, and adding delimiters to the output when they appear. When we reach a word, we add the word from the end of the string instead of the current word. In order to do this, we pre-process the string into a list of words and reverse the list.

```
import re

def reverse(string, delimiters):
    # Parse the string into words between delimiters using regex
    # Keep adjacent delimiters together ("greedy match")
    words = re.split('[' + delimiters + ']+', string)
    if len(words) > 0 and words[-1] == '':
        words = words[:-1]
    # Reverse the list of words and convert to an iterator
    word_iter = reversed(words)

    output = []
    delimiter_found = True
```

```
    # Iterate through the original string
    for c in string:
        if c in delimiters:
            # When we reach a delimiter, then set the word length to 0
            delimiter_found = True
            output.append(c)
        else:
            # We've reached a non-delimiter character
            # If it's the first character of the word, add a word
            # from our reversed list of words (word_iter).
            if delimiter_found:
                try:
                    output.append(next(word_iter))
                except StopIteration:
                    pass
            delimiter_found = False


    return ''.join(output)
```

We can also simplify our code by using an additional regex to split the string into a list of words and a list of delimiters. Then, we reverse the list of words and merge the two lists together. We need to again be careful of cases where there are multiple consecutive delimiters or delimiters that appear at the beginning and/or end of the string.

```
import re

def reverse(string, delimiters):
    # Parse the string into words between delimiters using regex
    # Keep adjacent delimiters together ("greedy match")
    words = re.split('[' + delimiters + ']+', string)
    not_words = re.split('[^(' + delimiters + ')]+', string)
    if len(words) > 0 and words[-1] == '':
        words = words[:-1]
    # Reverse the list of words and convert to an iterator
    word_iter = reversed(words)

    output = []
    for d in not_words:
        output.append(d)
        try:
            output.append(next(word_iter))
        except StopIteration:
            pass

    return ''.join(output)
```

Both approaches take `O(N)` time and `O(N)` space, where `N` is the length of the input string.