Documentation for numDecodings Method

Problem Statement

Given a string 's' containing only digits, determine the number of ways to decode it. Each letter from 'A' to 'Z' is encoded to a number from '1' to '26' as follows:

- 'A' -> "1"
- 'B' -> "2"
- ...
- 'Z' -> "26"

A valid decoding of the string is achieved by mapping groups of digits to their corresponding letters. For example:

- "11106" can be decoded as "AAJF" (grouped as 1 1 10 6) or "KJF" (grouped as 11 10 6).
- The grouping "1 11 06" is invalid because "06" cannot be mapped to "F".

Function Signature

```
class Solution:
    def numDecodings(self, s: str) -> int:
```

Input

- s: A string containing only digits. The length of s is between 1 and 100 inclusive.

Output

- An integer representing the number of ways to decode the input string s.

Constraints

- The string s contains only digits and may contain leading zeros.

Example 1

Input: s = "12"

Output: 2

Explanation: "12" can be decoded as "AB" (1 2) or "L" (12).

Example 2

Input: s = "226"

Output: 3

Explanation: "226" can be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).

Example 3

Input: s = "06"

Output: 0

Explanation: "06" cannot be mapped to "F" because of the leading zero ("6" is different from "06").

Method Description

The numDecodings method uses dynamic programming to compute the number of ways to decode the string s. The key steps are:

1. **Initialization:**

- If s is empty or starts with '0', return 0 because there are no valid decodings.
- Create an array dp where dp[i] represents the number of ways to decode the substring s[:i].
- Initialize dp[0] to 1 (one way to decode an empty string) and dp[1] based on whether the first character is '0'.

2. **Dynamic Programming Transition:**

- Iterate through the string starting from index 2 to n + 1.
- For each position i, check the last one and two digits:
- If the last single digit (i.e., s[i-1]) is between '1' and '9', add dp[i-1] to dp[i].
- If the last two digits (i.e., s[i-2:i]) form a number between 10 and 26, add dp[i-2] to dp[i].

3. **Result:**

- Return dp[n], which contains the total number of ways to decode the entire string s.

Explanation

Base Cases:

- dp[0] is set to 1 because there is one way to decode an empty string.
- dp[1] depends on the first character of s.

Iterative DP Update:

- For each position i from 2 to n:
- Check if the current character (s[i-1]) can be a valid single-digit decode.
- Check if the previous two characters (s[i-2:i]) can be a valid two-digit decode.

This approach ensures that all valid decodings are counted while invalid decodings (e.g., those involving '0') are excluded.