# 1119. Remove Vowels from a String

Easy String

## Problem Description

In this problem, we are given a string `s` that consists of lowercase letters. Our task is to remove all the vowels from this string. The vowels that we need to remove are 'a', 'e', 'i', 'o', and 'u'. After removing all these vowels from the string `s`, we should return the resulting string.

The main challenge is to process the string and efficiently eliminate the characters that are considered vowels.

## Intuition

The intuitive approach to solving this problem is to create a new string by iterating over each character in the original string and including only the characters that are not vowels. We can accomplish this by checking every character in the string `s` and appending it to the new string if it is not 'a', 'e', 'i', 'o', or 'u'.

To make our code more concise and Pythonic, we use a generator expression inside the `join` method. A generator expression is an elegant way to transform each item from a sequence according to a given condition or rule. In our solution:

1. We iterate over each character `c` in the string `s`.
2. We check if `c` is not in the string `"aeiou"` which contains all the vowels we want to exclude.
3. If `c` is not a vowel, it is included in the generator expression.
4. The `"".join()` function is used to concatenate all the characters accepted by our condition into a new string without vowels.

## Solution Approach

The implementation for removing vowels from a string in the given solution can be outlined as follows:

1. **Data Structure**: We use a simple string data structure. Strings in Python are immutable, so we're actually creating a new string as we iterate through the original one.
2. **Algorithm**: The core of the algorithm is a traversal of the input string `s`. This is the classic iteration pattern.
3. **Patterns Used**: We make use of a generator expression, which is an efficient way to handle sequences in Python. It provides a concise way to handle elements one by one, applying a filter or function to each one.

The steps of the algorithm are as follows:

- We begin by defining a `class Solution`, which will contain our method `removeVowels`.
- Inside `removeVowels`, we perform a generator expression `"".join(c for c in s if c not in "aeiou")`.
  - `c for c in s`: This part creates a generator that goes through each character `c` in the string `s`.
  - `if c not in "aeiou"`: This condition serves as a filter. It checks whether each character `c` is not one of the vowels 'a', 'e', 'i', 'o', or 'u'.
  - `"".join(...)`: This function takes all characters that pass the filter (i.e., all non-vowel characters) and joins them together into a new string. The `""` indicates that we are not using any delimiter between characters, so they are simply concatenated together.

The efficiency of this approach lies in the fact that we avoid constructing intermediary lists or arrays; we are directly constructing the new string without vowels in a single pass through the input string.

### Example Walkthrough

Let's walk through an example to illustrate the solution approach using the string `s = "leetcode"`.

1. We start by creating an instance of the `Solution` class.
2. We invoke the `removeVowels` method and pass our example string `s` to it.
3. The `removeVowels` method begins executing a generator expression within a `join` function.
4. The generator iterates over each character in `s`. Here's how it processes our example:
   - `'l'`: Since `'l'` is not in `"aeiou"`, it passes the filter. The generator yields `'l'`.
   - `'e'`: `'e'` is a vowel; it does not pass the filter and is not yielded by the generator.
   - `'e'`: `'e'` is a vowel; it does not pass the filter and is not yielded by the generator.
   - `'t'`: Since `'t'` is not in `"aeiou"`, it passes the filter. The generator yields `'t'`.
   - `'c'`: Since `'c'` is not in `"aeiou"`, it passes the filter. The generator yields `'c'`.
   - `'o'`: `'o'` is a vowel; it does not pass the filter and is not yielded by the generator.
   - `'d'`: Since `'d'` is not in `"aeiou"`, it passes the filter. The generator yields `'d'`.
   - `'e'`: `'e'` is a vowel; it does not pass the filter and is not yielded by the generator.
5. The characters that pass the filter ('l', 't', 'c', 'd') are joined by the `join` function into a new string.
6. The resulting string is `"ltcd"`, which is the original string `"leetcode"` without the vowels.
7. The `removeVowels` method returns the string `"ltcd"`.

Our example has successfully demonstrated how each character is considered individually, and only non-vowel characters are concatenated to form the final string. The simplicity of the generator expression makes this method both efficient and easy to read.

## Solution Implementation

### Python

```python
class Solution:
    def removeVowels(self, string: str) -> str:
        # Initialize a list to store characters that are not vowels
        non_vowel_chars = []

        # Iterate over each character in the input string
        for char in string:
            # Check if the character is not a vowel
            if char.lower() not in "aeiou":
                # If it's not a vowel, append it to the list
                non_vowel_chars.append(char)

        # Join the non-vowel characters back into a string and return it
        return "".join(non_vowel_chars)
```

### Java

```java
class Solution {
    // Method to remove all vowels from a given string
    public String removeVowels(String s) {
        // Use StringBuilder to build the result string efficiently
        StringBuilder resultBuilder = new StringBuilder();

        // Iterate over each character in the string
        for (int i = 0; i < s.length(); i++) {
            // Retrieve the current character
            char currentChar = s.charAt(i);

            // Check if the current character is not a vowel
            if (!(currentChar == 'a' || currentChar == 'e' || currentChar == 'i' ||
                  currentChar == 'o' || currentChar == 'u')) {
                // If it's not a vowel, append it to the resultBuilder
                resultBuilder.append(currentChar);
            }
        }

        // Convert the StringBuilder to a String and return it
        return resultBuilder.toString();
    }
}
```

### C++

```cpp
class Solution {
public:
    // Function to remove vowels from the string
    string removeVowels(string s) {
        string result; // Initialize an empty string to store the result

        // Iterate through each character in the input string
        for (char& c : s) {
            // Check if the character is not a vowel
            if (!(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')) {
                result += c; // If it's not a vowel, add it to the result string
            }
        }

        return result; // Return the resultant string without vowels
    }
};
```

### TypeScript

```typescript
// This function removes all vowels from a given string
function removeVowels(s: string): string {
    // Use a regular expression to find all vowels (both lowercase and uppercase)
    // in the string and replace them with an empty string
    return s.replace(/[aeiouAEIOU]/g, '');
}
```

```python
class Solution:
    def removeVowels(self, string: str) -> str:
        # Initialize a list to store characters that are not vowels
        non_vowel_chars = []

        # Iterate over each character in the input string
        for char in string:
            # Check if the character is not a vowel
            if char.lower() not in "aeiou":
                # If it's not a vowel, append it to the list
                non_vowel_chars.append(char)

        # Join the non-vowel characters back into a string and return it
        return "".join(non_vowel_chars)
```

## Time and Space Complexity

### Time Complexity

The time complexity of traversing the string and checking each character if it's a vowel or not is `O(n)`, where `n` is the length of the string. No nested loops or complex operations are involved, so the time complexity is linear with the size of the input string.

### Space Complexity

The space complexity is also `O(n)`, primarily due to the space required to build the output string. Since the output string may potentially contain almost all characters from the input string (in the case when the input contains few or no vowels), the space required grows linearly with the input size.