Roman Numeral Converter

Roman to Integer

When converting Roman numerals to integers, each letter represents a specific value. For example, 'I' is equal to 1, 'V' is 5, 'X' is 10, 'L' is 50, 'C' is 100, 'D' is 500, and 'M' is 1000. To calculate the total integer value from a Roman numeral, you need to follow a few rules:

1. If a letter appears after a letter of greater value, you add the values. For instance, 'VI' would be 5 + 1 = 6.
2. If a letter appears before a letter of greater value, you subtract the smaller value from the greater value. For example, 'IV' would be 5 - 1 = 4.
3. The numeral must be read from left to right, and the values are added or subtracted accordingly.

Converting Roman numerals to integers can be done programmatically by iterating through the Roman numeral string and comparing the values of adjacent letters. By applying the rules mentioned above, you can accurately convert Roman numerals of any length into their corresponding integer values.

For more complex numerals, such as 'XCIV' (which is 94 in decimal), you would calculate it as 100 (C) - 10 (X) + 5 (V) - 1 (I) = 94. Understanding these rules and practicing conversions will enable you to efficiently convert Roman numerals to integers for various applications.

Problem Statement

In the realm of Roman numerals, each symbol holds a specific numerical value. To better understand this ancient numeric system, it is crucial to grasp the values associated with each symbol. Below is a table summarizing the Roman numeral symbols and their corresponding integer values:

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Examples of Roman Numerals:

Basic Formation

- II represents 2.
- IX equates to 9.

Special Subtraction Cases

- IV signifies 4 (5 - 1).
- XL denotes 40 (50 - 10).
- CM translates to 900 (1000 - 100).

Understanding the Conversion Process:

1. Start from the leftmost symbol.
2. Compare the value of the current symbol with the next symbol.
3. If the current symbol is less than the next symbol, subtract the current value from the total.
4. If the current symbol is greater or equal to the next symbol, add the current value to the total.

By following these rules and examples, you can efficiently convert Roman numerals into their corresponding integer values, unveiling the beauty and complexity of this numerical system.

Task

Task Definition

In this section, we will define the task of converting a Roman numeral string to an integer. The process involves deciphering the numerical value represented by a given Roman numeral by following the rules of Roman numeral conversion.

Input Example:

Bold and *italics* formatting will be used for clarity and emphasis.

Given a Roman numeral string such as **"XXIV"**, the task is to convert this into its corresponding integer value.

Output Example:

Upon successful conversion, the output for the input example **"XXIV"** should be the integer **24**.

Task Overview:

The task requires understanding the relationship between each Roman numeral symbol and its associated numerical value. By systematically analyzing the Roman numeral string and applying the conversion rules, we can accurately determine the integer representation of the given Roman numeral.

Additional Guidance:

Ensure that the conversion process considers both regular formations and special subtraction cases in Roman numerals. By effectively implementing the conversion algorithm, the task of converting Roman numerals to integers can be efficiently accomplished, demonstrating a comprehensive understanding of this ancient numerical system.

Examples

After understanding the task of converting Roman numerals to integers and the rules governing this conversion process, let's explore some examples to solidify our comprehension.

Example 1:

- **Roman Numeral:** "XIV"
- **Conversion Process:**
  - Start with 'X' (10) and compare with 'I' (1). Since 10 is greater than 1, we add 10.
  - Move to 'I' (1) and compare with 'V' (5). As 1 is less than 5, we subtract 1 from the total.
- **Result Explanation:** 10 (X) + 1 (V - I) = 14

Example 2:

- **Roman Numeral:** "LXXXIX"
- **Conversion Process:**
  - Begin with 'L' (50) and compare with 'X' (10). We add 50.
  - Move to 'X' (10) and compare with 'X' (10). Adding 10.
  - Proceed to 'X' (10) and compare with 'I' (1). Adding 10.
  - Lastly, 'I' (1) compared with 'X' (10) results in subtraction.
- **Result Explanation:** 50 (L) + 10 (X) + 10 (X) + 10 (X) - 1 (X - I) = 89

Example 3:

- **Roman Numeral:** "CDXLIV"
- **Conversion Process:**
  - Evaluate 'C' (100) compared to 'D' (500), subtracting 100.
  - Move to 'D' (500) and compare with 'X' (10), adding 500.
  - Proceed to 'X' (10) and compare with 'L' (50), subtracting 10.

    - Lastly, 'L' (50) compared with 'I' (1) subtracts 10.
- **Result Explanation:** 100 (C) - 10 (X) + 500 (D) - 10 (X) + 50 (L) - 1 (I) = 444

By walking through these examples step by step, we can effectively translate Roman numerals into their corresponding integer values, showcasing the intricate process of decoding this ancient numerical system.

Constraints

Length Constraint:

- The Roman numeral string provided for conversion should not exceed a certain length to ensure efficient processing.

Valid Character Sets:

- The Roman numeral string must consist of valid characters representing the Roman numerals I, V, X, L, C, D, and M.
- Any other characters or symbols in the input string will result in an error and should be handled appropriately during conversion.

What is a Palindrome?

A palindrome is a sequence of characters that reads the same forwards as it does backward. In other words, a palindrome remains unchanged when its order is reversed. Palindromes can be words, phrases, numbers, or other sequences.

Examples of Palindromes:

- **Number Palindrome:** "121" reads the same from left to right and right to left.
- **Word Palindrome:** "racecar" remains the same when reversed.

Palindromes are interesting linguistic or numerical constructs that exhibit symmetry and are commonly used as puzzles or wordplay due to their unique properties.

Explanation Like You're 16

Alright, so you know how sometimes in math class, you come across those cool Roman numerals like 'IV' or 'XVIII,' and you're like, "What even are these?" Well, let's break it down in a super simple way.

Imagine Roman numerals as a secret code where each letter represents a special number. For instance, 'I' is like the number 1, 'V' is 5, 'X' is 10, and so on. So when you see 'VIII,' it's basically saying 5 + 1 + 1 + 1 = 8.

Now, when you have a letter that's smaller in front of a bigger one, you do subtraction. So 'IV' is like 5 - 1, which equals 4. But if the smaller letter is after the bigger one, you just add them up. For example, 'VI' is 5 + 1, giving you 6.

To turn these cool Roman numerals into regular numbers, we need to follow these rules. It's like solving a puzzle! First, we create a cheat sheet (a dictionary) that tells us what each letter means. Then, we look at the Roman numeral from left to right, adding or subtracting as needed.

Let's say we have 'XIV,' which sounds fancy, right? But it's just 10 (X) + 1 (V - I) = 14. See, not so mysterious after all!

Now, let's try 'LXXXIX.' Sounds like a secret code, but it's just 50 (L) + 10 (X) + 10 (X) + 10 (X) - 1 (X - I) = 89. Pretty cool, huh?

So, by understanding these simple rules and practicing a bit, you can crack the code of Roman numerals like a pro. It's like learning a new language, but with numbers!

Full Code

```
class Solution:
    def romanToInt(self, s: str) -> int:
        # Step 1: Create a dictionary to map Roman numerals to their integer 
values
        roman_to_int = {
            'I': 1,
            'V': 5,
            'X': 10,
            'L': 50,
            'C': 100,
            'D': 500,
            'M': 1000
        }

        # Step 2: Initialize a variable to store the total value
        total = 0

        # Step 3: Get the length of the input string
        length = len(s)

        # Step 4: Iterate through each character in the string
        for i in range(length):
            # Step 5: Check if the current character is less than the next 
character
            if i < length - 1 and roman_to_int[s[i]] < roman_to_int[s[i + 1]]:
                # If true, subtract the current character's value from the 
total
                total -= roman_to_int[s[i]]
            else:
                # Otherwise, add the current character's value to the total
                total += roman_to_int[s[i]]

        # Step 6: Return the total value which is the integer representation 
of the Roman numeral
        return total
```

This Python code snippet defines a function roman_to_int that converts a Roman numeral string to an integer value following the rules of Roman numeral conversion. The function utilizes a dictionary roman_dict to map each Roman numeral character to its corresponding integer value. It then iterates through the input string, comparing the current value with the previous value to determine whether to add or subtract from the total. Finally, the function returns the total integer value obtained from the Roman numeral string.

The provided examples demonstrate how the function can accurately convert Roman numerals like "XIV," "LXXXIX," and "CDXLIV" into their respective integer values, showcasing the functionality of the conversion process.

Code Explanation

To understand the inner workings of the code snippet provided for converting Roman numerals to integers, let's delve into its components step by step:

Dictionary Setup

The first crucial aspect of the code is the setup of a dictionary named roman_dict. This dictionary serves as a mapping tool, associating each Roman numeral character with its corresponding integer value. By defining this dictionary at the beginning of the function, the code establishes a reference for converting Roman numerals efficiently.

```
roman_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
```

Initializing Total

Upon defining the dictionary, the code initializes a variable total to keep track of the cumulative integer value as the conversion progresses. This variable acts as a running sum, accumulating the calculated values of the Roman numeral characters during the conversion process.

```
total = 0
```

Looping Through Characters

The core of the conversion algorithm lies in the iterative process of traversing through each character in the Roman numeral string. By employing a loop that iterates over the input string, the code can analyze and process each Roman numeral character sequentially.

```
for char in s:
    # Conversion logic will be applied here
```

Handling Subtraction/Addition

Within the loop, the code compares the current character's value with the previous character's value to determine whether to add or subtract from the total integer value. This conditional logic enables the code to differentiate between cases where addition or subtraction is required based on the Roman numeral rules.

```
if curr_value > prev_value:
    total += curr_value - 2 * prev_value
else:
    total += curr_value
```

Returning the Result

After iterating through all Roman numeral characters and calculating the total integer value according to the conversion rules, the code concludes by returning the final result. This return statement ensures that the function outputs the accurate integer value derived from the input Roman numeral string.

```
return total
```

By comprehensively explaining each segment of the code snippet, we can grasp how the conversion process unfolds, from mapping Roman numerals to integers to executing the conversion logic and producing the desired output. This detailed breakdown enhances our understanding of the intricate process involved in converting Roman numerals to their corresponding integer values.