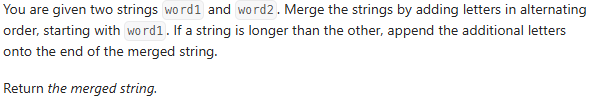
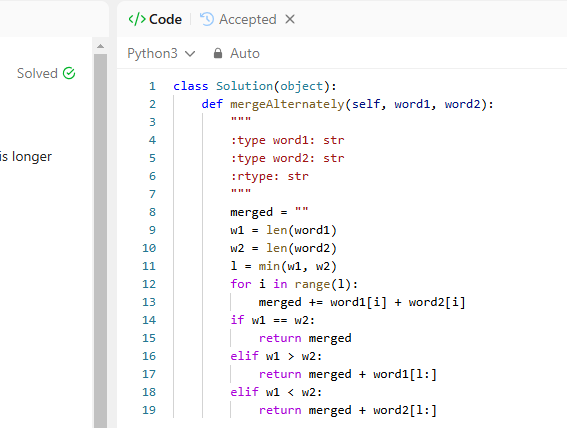
Task 1. [Merge Strings Alternately](https://leetcode.com/problems/merge-strings-alternately/)





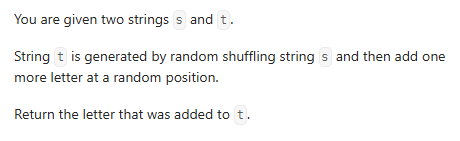
**Problem Solution:**

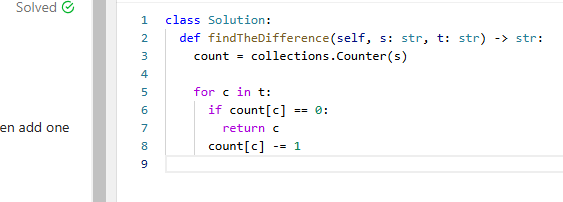
I iterated through both strings up to the length of the shorter one, appending characters alternately. If one string was longer, I added the remaining part to the result and returned the merged string.

Test results

Task 2. [Find the Difference](https://leetcode.com/problems/find-the-difference/)

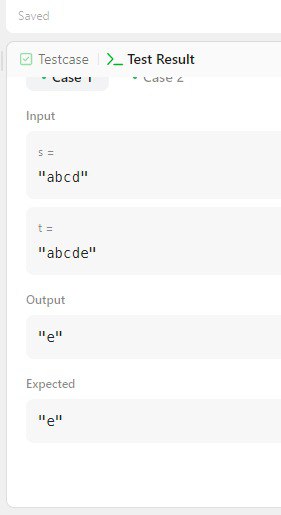




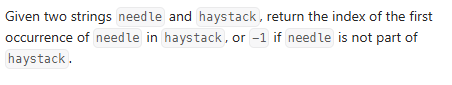
**Problem Solution:**

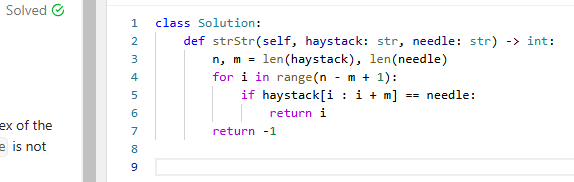
I used a Counter to store occurrences of characters in s. Then, I iterated through t, checking if a character was missing or appeared more times. The extra character was returned as the result.

Test results

Task 3. [Find the Index of the First Occurrence in a String](https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/)

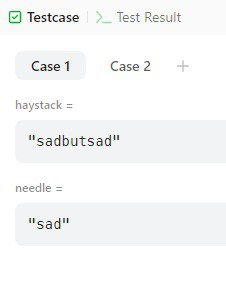
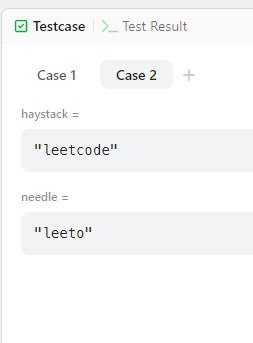




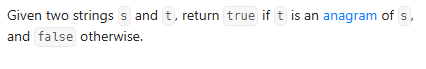
**Problem Solution:**

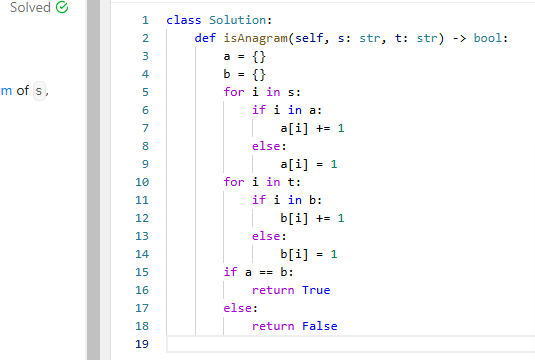
I iterated through haystack and checked substrings of the same length as needle. If I found a match, I returned the index; otherwise, I returned -1.

Test results



Task 4. Valid Anagram

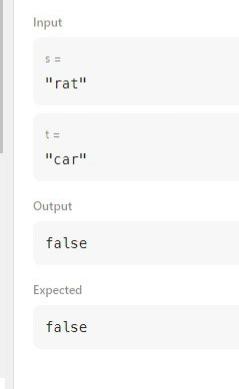
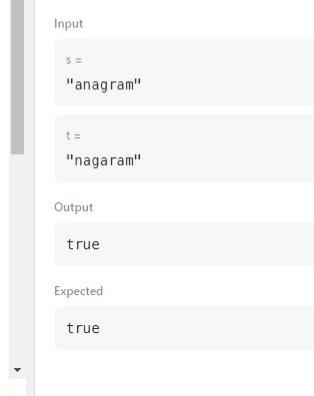




**Problem Solution:**

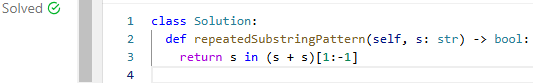
Given two strings s and t, return true if t is an anagram of s, and false otherwise.

I created two dictionaries to store character frequencies for both strings. Then, I iterated through each character in both strings, updating their respective frequency counts. Finally, I compared the dictionaries to determine if they were equal.

Results of test

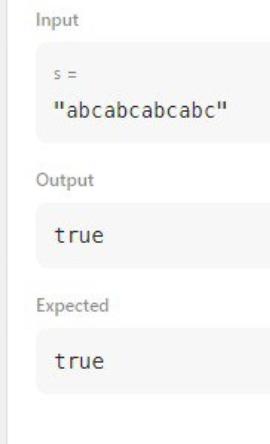
Task 5. [Repeated Substring Pattern](https://leetcode.com/problems/repeated-substring-pattern/)



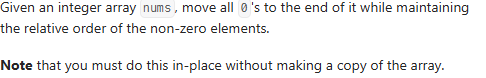


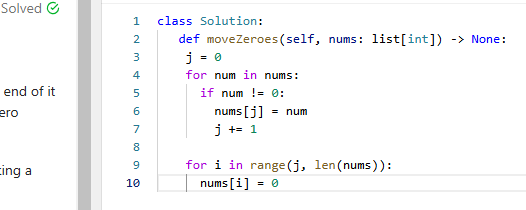
**Problem Solution:**

I used string manipulation by checking if s exists within (s + s) [1: -1]. This approach leverages the properties of repeated patterns in strings.

Results of test

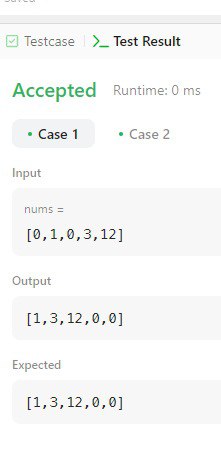
Task 6. Move Zeroes



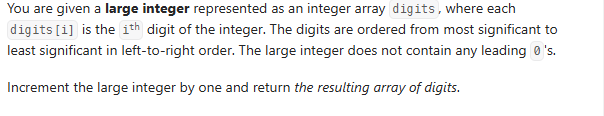


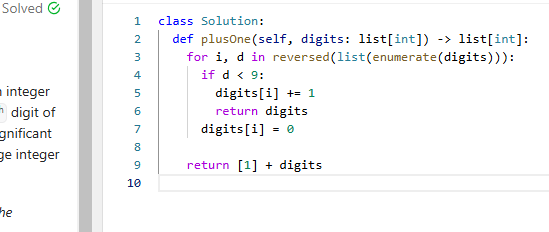
**Problem Solution:**

I used two pointers to shift non-zero elements to the front. Then, I filled the remaining positions with zeroes. This approach ensures in-place modifications and minimizes operations.

Results of test

Task 7. Plus One

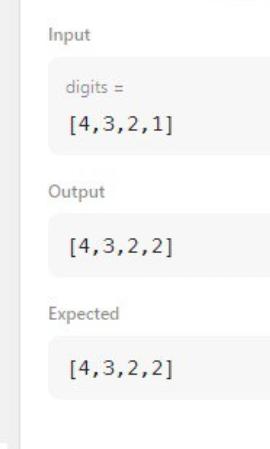




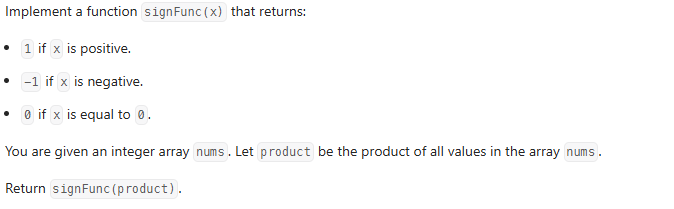
**Problem Solution:**

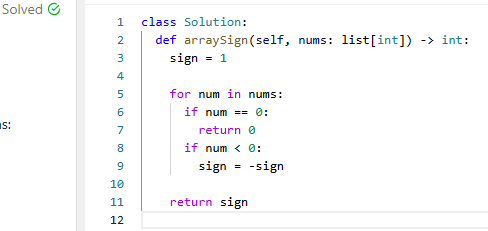
I iterated through digits in reverse. If a digit was less than 9, I increased it and returned the array. If all digits were 9, I converted them to 0 and added 1 at the beginning.

Results of test



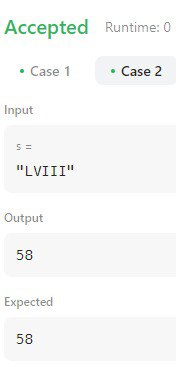
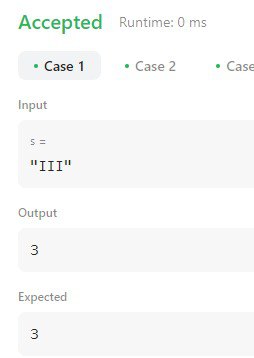
Task 8. [Sign of the Product of an Array](https://leetcode.com/problems/sign-of-the-product-of-an-array/)



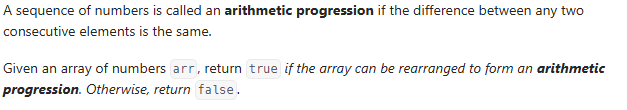


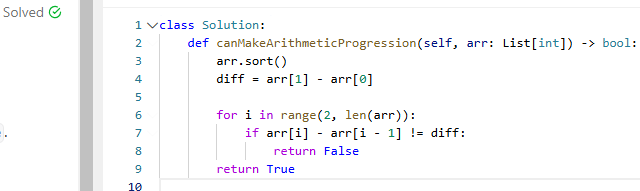
**Problem Solution:**

I iterated through the array and checked for zeros. If a zero was found, I returned 0. Otherwise, I counted negative numbers. If the count was even, the product was positive; otherwise, it was negative.

Results of test

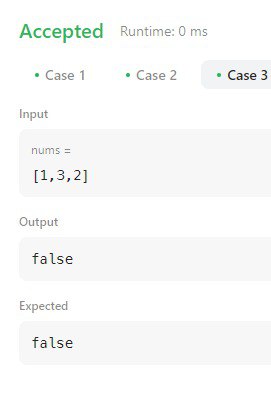
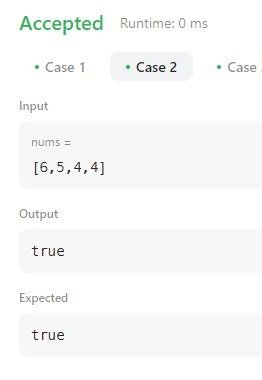
Task 9. [Can Make Arithmetic Progression from Sequence](https://leetcode.com/problems/can-make-arithmetic-progression-from-sequence/)



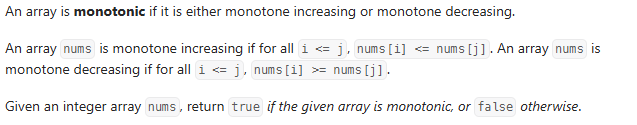


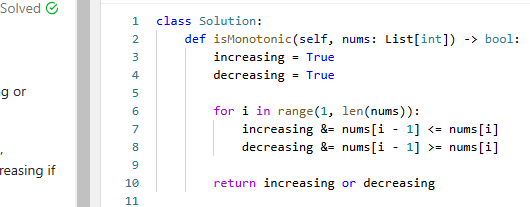
**Problem Solution:**

I sorted the array and calculated the common difference. Then, I iterated through the array to check if each consecutive difference matched the expected difference.

Results of test

Task 10. Monotonic Array

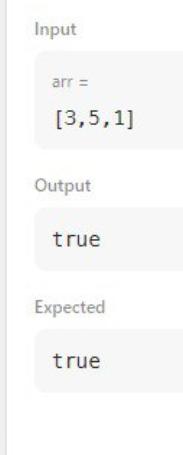




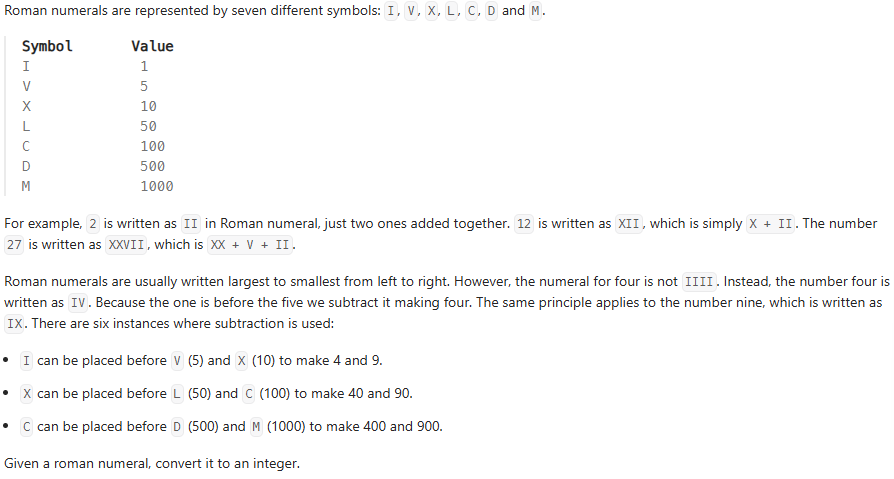
**Problem Solution:**

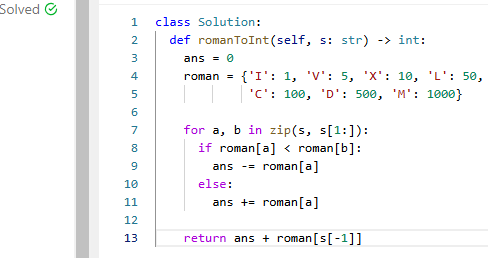
I iterated through the array while tracking whether it was increasing or decreasing. If both conditions failed, the array was not monotonic.

Results of test



Task 11. Roman to integer.





**Problem Solution:** I used a dictionary to store values of Roman numerals. Then, I iterated through the string, checking for subtraction cases and summing the values accordingly.

Results of test

