



[Click on the logo to find **Problem Statement**]

## Intuition

---

The intuition behind this code is to determine whether a given integer is a palindrome, meaning it reads the same forwards and backwards. To achieve this, the integer is converted to a string, which facilitates easy comparison of its digits. By comparing the original string with its reverse, achieved through string slicing, the code checks if the integer is a palindrome. If the original and reversed strings match, indicating that the integer reads the same forwards and backwards, the function returns True; otherwise, it returns False. This approach simplifies palindrome detection by leveraging string manipulation but incurs linear time and space complexity relative to the number of digits in the integer.

## Approach

---

1. Convert the integer to a string: The integer input `x` is converted into a string representation using the `str()` function. This conversion facilitates easier manipulation and comparison of individual digits.
2. Check for palindrome property: The code checks whether the string representation of the integer `x` is equal to its reverse. This is done by comparing the string `x` with its reverse obtained using string slicing (`x[::-1]`).
3. Return the result: If the original string `x` is equal to its reverse, it indicates that the integer is a palindrome. In this case, the function returns True. Otherwise, if the strings are not equal, the function returns False, indicating that the integer is not a palindrome.

## Complexity

---

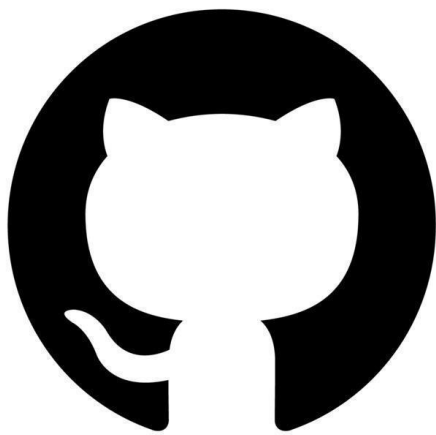
- Time complexity:  $O(n)$
- Space complexity:  $O(n)$

# Code

---

```
class Solution:
    def isPalindrome(self, x: int) -> bool:
        x = str(x)
        if x == x[::-1]:
            return True
        else:
            return False
```

If you want to see more solutions to coding problems, you can visit:



# GitHub