Problem Statement

Given an integer x, return **`true`** if x is a palindrome, and **`false`** otherwise.

Examples:

1. **Input:** $x = 121$
   - **Output:** **`true`**
   - **Explanation:** 121 reads as 121 from left to right and from right to left.
2. **Input:** $x = -121$
   - **Output:** **`false`**
   - **Explanation:** From left to right, it reads -121. From right to left, it becomes 121-. Therefore, it is not a palindrome.
3. **Input:** $x = 10$
   - **Output:** **`false`**
   - **Explanation:** Reads 01 from right to left. Therefore, it is not a palindrome.

Constraints:

- $-2^{31} \le x \le 2^{31} - 1$

What is a Palindrome?

A palindrome is a sequence that reads the same forwards and backwards, such as "121" or "racecar".

Explanation Like You're 16

Understanding the Problem

You need to figure out if a number looks the same when you read it forwards and backwards. For example:

- **121** is a palindrome because if you flip it around, it still looks like **121**.
- **-121** is not a palindrome because flipping it around gives **121-**, which is different.
- **10** is not a palindrome because flipping it around gives **01**, which is different.

Step-by-Step Solution

:Negative Numbers .4

- Any negative number can't be a palindrome because the .minus sign will be at the end when reversed

:Single Digit Numbers .5

- Any single digit number (like 0, 1, 2, ..., 9) is always a palindrome because it looks the same forwards and .backwards

:Reversing the Number .6

- Instead of converting the number to a string, we can reverse its digits and then compare the reversed number to the .original

Reversing the Number

.Start with the original number .7
.Extract the last digit of the number .8
.Add this digit to a new number that we're building (start with 0) .9
.Remove the last digit from the original number .10
.Repeat until the original number is 0 .11
.Compare the reversed number to the original .12

Code Explanation

:Here’s the code that does this

```
:class Solution
    :def isPalindrome(self, x: int) -> bool
        Step 1: Negative numbers are not palindromes #
        :if x < 0
            return False
    # Step 2: Single digit numbers are always palindromes
    if 0 <= x < 10:
        return True

    # Step 3: Reverse the number
    original = x  # Keep the original number
    reversed_num = 0  # This will store the reversed number

    while x != 0:
        pop = x % 10  # Get the last digit
        x //= 10  # Remove the last digit
        reversed_num = reversed_num * 10 + pop  # Build the reversed number

    # Step 4: Compare the original number to the reversed number
    return original == reversed_num
```

Detailed Steps in the Code

13. **Check for Negative Numbers:**
    - If x is negative, return `False` because negative numbers are not palindromes.
14. **Check for Single Digit Numbers:**

- If xx is between 0 and 9, return `True` because single-digit .numbers are palindromes

.15 **:Reverse the Number**

- .Save the original number in a variable called `original`
- .Initialize `reversed_num` to 0
- :Use a loop to reverse the digits of xx
  - .`pop = x % 10` gets the last digit of xx
  - .`x //= 10` removes the last digit from xx
  - `reversed_num = reversed_num * 10 + pop` builds the .reversed number

.16 **:Compare Original and Reversed Numbers**

- If the reversed number is the same as the original number, .return `True`; otherwise, return `False`

Final Thoughts

This approach ensures that we correctly determine if a number is a palindrome without converting it to a string, making it efficient and easy .to understand