Gannon Traynor

There are a couple problems I see instantly with this block of code; I broke them down into edge cases and efficiency

Find Edge Cases: Firstly, we should assume a string with one or zero characters are a palindrome. The function right now is assuming the input is only one word with all lowercase characters but, there can be strings with capitals and spaces that is still a palindrome. To fix this we must normalize the data and remove every space and change every capital letter to a lowercase in the string.

Efficiency: Right now, the function is iterating through a string twice, once to create a reversed string and once to check if the two stings are equal. There is a way to do this by only iterating through the string once, (with only half the comparisons) Create a placeholder variable that will track which element in the string we are looking at from end of the string, call it “back” initiate it as one less than the strings length. In a loop with “front” initiated as 0 and a range equal to half the length of the string. Then compare the “front” and “back” character in the string, if they aren’t the same return false immediately, otherwise increment “front” by one and decrement “back” by one. Repeat until you return false or exit the loop if you exit the loop return true.

```
def isPalindrome(s):

    if len(s) < 2: return True

    s.replace(" ", "")

    back = len(s) - 1
    half = len(s) % 2
    for front in range(half):
        if s[front].lower() != s[back].lower():
            return False
        back = back -1

    return True
```