**Problem #2609: Find the longest Balanced Substring of a Binary String** (Easy)

<https://leetcode.com/problems/find-the-longest-balanced-substring-of-a-binary-string/description/>

**My Solution:**

1. If ‘0’ not in s, then return 0
2. If ‘1’ not in s, then return 0.
3. Initialize maxLen to 0.
4. Iterate I from 0 to len(s) and j from len(s) to -1 in steps of -1.

Consider substr from index I to j.

If length of the substring is greater than maxLen and if if the substring is of even length (it should contain equal number of ‘0’ and ‘1’), then check if the first half consists of ‘0’s and the second half consists of ‘1’s. If so, maxLen becomes the length of the substring.

1. Finally return maxLen

class Solution:

def findTheLongestBalancedSubstring(self, s: str) -> int:

if '0' not in s:

return 0

if '1' not in s:

return 0

maxLen = 0

for i in range(0, len(s)):

for j in range(len(s), -1, -1):

substr = s[i : j]

if len(substr) > maxLen and len(substr) % 2 == 0:

if substr == "0" \* (len(substr)//2) + "1" \* (len(substr)//2):

maxLen = len(substr)

return maxLen