**Problem #1013: Partition Array into Three Parts with Equal Sum**

<https://leetcode.com/problems/partition-array-into-three-parts-with-equal-sum/description/>

**My Soltuion:**

1. Let total be the sum of the array elements of arr.
2. Let partTotal be total divided by (integer division)
3. If sum of arr is not evenly divided by 3 I.e. there is a non-zero remainder, then False.
4. Let n be the length of arr.
5. Initialize firstTotal to 0 and firstIndex to -1.
6. Iterate through arr. Add the array elements to firstTotal. If firstTotal is equal to partTotal then set firstIndex to that index and break from the loop.

If firstIndex is -1, then return False.

1. Set secondTotal to 0 and secondIndex to -1 Iterate through arr from firstIndex + 1.

Add arr value to secondTotal. If secondTotal is equal to partTotal, then secondIndex is the current index i. Then break from the for loop.

If secondIndex is -1 or secondIndex is n – 1, then return False.

1. Return True if we have not returned False so far.

class Solution:

def canThreePartsEqualSum(self, arr: List[int]) -> bool:

total = sum(arr)

partTotal = sum(arr)//3

if sum(arr) % 3 != 0:

return(False)

n = len(arr)

firstTotal = 0

firstIndex = -1

for i in range(n):

firstTotal += arr[i]

if firstTotal == partTotal:

firstIndex = i

break

if firstIndex == -1:

return(False)

secondTotal = 0

secondIndex = -1

for i in range(firstIndex + 1, n - 1):

secondTotal += arr[i]

if secondTotal == partTotal:

secondIndex = i

break

if secondIndex == -1 or secondIndex == n - 1:

return(False)

return(True)