EQUILIBRIUM INDEX:

Index such that sum of elements till the index(including it) is equal to the sum of remaining elements.

Approach 1

For each index test whether it is equilibrium index or not. For each i check sum till that index and sum of remaining elements and check whether both sums are equal or not. For checking each index - O(n) For finding sum we have to scan the array - O(n) Complexity: Time : O(n^2) Space : O(1)

Approach 2

First find the cumulative sum list of the array elements. Then for each index check sum till the point and compare it with other sum like below: cum_sum[i] == cum_sum[-1]-cum_sum[i] if condition is satisfied equilibrium index is i Complexity: finding cum_sum - O(n). Scanning it O(n) Time : O(n) Space: O(n)

In [2]:
```
def findEquilibriumIndex(arr):
    cum_sum = []
    curr_sum = 0
    for i in arr:
        curr_sum+=i
        cum_sum.append(curr_sum)
    for ix,ele in enumerate(cum_sum):
        if ele == cum_sum[-1] - ele:
            return ix
    return -1
```

In [19]:
```
arr = [1,5,10,9,1,2,4]
findEquilibriumIndex(arr)
#ans 2
```

Out[19]: 2

In [20]:
```
arr = [1,5,10,9,1,2]
findEquilibriumIndex(arr)
#ans -1
```

Out[20]: -1

In [21]:
```
arr = [1,1,5,10,9,1,2,5]
findEquilibriumIndex(arr)
#ans 3
```

Out[21]: 3

You can also do this approach by reducing space complexity to O(1) as below:

In [26]:
```
def findEquilibriumIndex_opt(arr):
    total_sum = 0
    curr_sum = 0
    for i in arr:
        total_sum+=i
    for ix in range(len(arr)):
        curr_sum += arr[ix]
        if curr_sum == total_sum - curr_sum:
            return ix
    return -1
```

In [27]:
```
arr = [1,5,10,9,1,2]
findEquilibriumIndex_opt(arr)
#ans -1
```

Out[27]:
```
-1
```

In [28]:
```
arr = [1,1,5,10,9,1,2,5]
findEquilibriumIndex_opt(arr)
#ans 3
```

Out[28]:
```
3
```

In [ ]:
```

```