Given an integer array *nums*, find the sum of the elements between indices i and j ($i \leq j$), inclusive.

Example:

> Given *nums* = [-2, 0, 3, -5, 2, -1]
>
> sumRange(0, 2) → 1
> sumRange(2, 5) → -1
> sumRange(0, 5) → -3

Note:

- You may assume that the array does not change.
- There are many calls to *sumRange* function.

Answer:

Here is one technique used often in 1-dim dp problems, i.e. *Cumulative Sum* or *CS*. For example, we do *CS* on [1, 2, 3], then we get [1, 3, 6].

The main purpose of *CS* is to find the sum of elements between two indices in $\mathcal{O}(1)$ time.

Obviously, sumRang(i, j) = *cnums*[j] - *cnums*[i - 1] where *cnums* is the array after *Cumulative Sum*-ing with the setting that *cnums*[- 1] = 0.

Code:

```
class NumArray(object):
    def __init__(self, nums):
        csums = [0] + nums
        for i in range(len(nums)):
            csums[i + 1] += csums[i]
            self.csums = csums

    def sumRange(self, i, j):
        csums = self.csums
        assert -1 < i < len(csums) and -1 < j < len(csums)
        return csums[j + 1] - csums[i]
```

Question:

- Does ”the array doesn’t change” mean that it can’t be modified?
- If it can’t be modified, is it suitable to use another array which is the modified version?