# Two sum (naive solution)

```
def twoSum(A, target):
  for i in range(len(A)-1):
    for j in range(i, len(A)):
      if A[i] + A[j] == target:
        return [i, j]
  return  [-1, -1]
```

- Runtime: $O(n^2)$
- Space: $O(1)$

Can we do better by using more space in exchange for lower time complexity?

## Two sum (better solution)

Avoid nested for-loops

```
def twoSum(A, target):
  diffs = {}
  for i in range(len(A)):
    diffs[target - A[i]] = i
  for i in range(len(A)):
    if A[i] in diffs:
      if diffs[A[i]] != i:
        return [i, diffs[A[i]]]
  return [-1, -1]
```

- Runtime: O(n)
- Space: O(n)

Have traded space for runtime - worth the performance improvement.

In general, space is much cheaper than runtime.