Two Sum

Difficulty	Easy
Category	Arrays
Question	https://leetcode.com/problems/two-sum/
Solution	https://youtu.be/KLlXCFG5TnA
Status	Done

Question

> Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.
>
> You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.
>
> You can return the answer in any order.

Example

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

Idea

💡 Use a simple iterative approach to form sub arrays and check the sum

💡 Use hash map to instantly check for difference value, map will add index of last occurrence of a num, don’t use same element twice;

Solution

```
# SIMPLE - ITERATIVE APPROACH

class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:

        for i in range(0, len(nums)):
            for j in range(i+1, len(nums)):
                if nums[i]+nums[j] == target:
                    return [i,j]
        return []
```

```
# HASH MAP

class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        prevMap = {}  # A dictionary to store {number: index} mapping

        for i, n in enumerate(nums):
            diff = target - n  # Calculate the difference needed to reach the target
            if diff in prevMap:
                # If the difference (diff) is already in the dictionary, it means we've found a pair
                # that adds up to the target, so we return the indices of these two numbers.
                return [prevMap[diff], i]
            prevMap[n] = i  # Store the current number and its index in the dictionary

        # If no solution is found, return an empty list
        return []
```