1. Two Sum

Status	Done
Coursework Databse	
Level	Easy
Source	
Source 2	
Tags	Leetcode Problems

Problem

> 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 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]
> ```

Solution

My solution

Create a nested for loop using elements from nums list.
To avoid repeating same element put an if loop and compare every element that is chosen
In the else loop put a condition if the outcome of two elements is equal to the target, add the indexes to final list
Return final list from inside the nested if loop

	My score	Best Score
Runtime	9313ms	81ms
Memory	17.1Mb	14.8mb

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

        for i in range(len(nums)):
            for j in range(len(nums)):
                if i == j:
                    continue
                else:
                    if nums[i] + nums[j] == target:
                        final.append(i)
                        final.append(j)
                        return final
```

Leetcode Solution

In this problem, you initialize a **dictionary** (`seen`). This dictionary will keep track of numbers (as `key`) and indices (as `value`).
So, you go over your array (line `#1`) using **`enumerate`** that gives you both index and value of elements in array. As an example, let's do `nums = [2,3,1]` and `target = 3`.
Let's say you're at index `i = 0` and `value = 2`, ok? you need to find `value = 1` to finish the problem, meaning, `target - 2 = 1`. 1 here is the `remaining`.
Since `remaining + value = target`, you're done once you found it, right? So when going through the array, you calculate the `remaining` and check to see whether `remaining` is in the `seen` dictionary (line `#3`).
If it is, you're done! you're current number and the remaining from `seen` would give you the output (line `#4`).
Otherwise, you add your current number to the dictionary (line `#5`) since it's going to be a `remaining` for (probably) a number you'll see in the future assuming that there is at least one instance of answer.

	Score	Best Score
Runtime	75ms	81ms
Memory	17.7mb	14.8mb

```
def twoSum(self, nums: List[int], target: int) -> List[int]:
       seen = {}
       for i, value in enumerate(nums): #1
           remaining = target - nums[i] #2

           if remaining in seen: #3
               return [i, seen[remaining]]  #4
           else:
               seen[value] = i
```