DESCRIPTION:

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].

METHOD 1: Using HashMap to solve the problem

class Solution:

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

hashmap={}

for i,n in enumerate(nums):

diff = target-n

if diff in hashmap:

return [hashmap[diff],i]

hashmap[n]=i

return

Time complexity : O(n)

EXPLANATION:

In python Hash Map is also known as Dictionaries which consists of key\_value pairs as it’s elements. Here keys must be Immutable which means they cannot be changed.

For the 2sum problem first we are creating a hashmap named as hashmap

We can declare it by either

Hashmap = {} OR Hashmap = dict()

After declaring a hashmap now we must iterate throuth the input list so that we can find the target value to be sum of 2 numbers from the list and find their indexes.

Here we used the enumerate() fuction to iterate i,n which are both index and the value at the same time.

Since the target value is some of 2 numbers it can also be said as target – number gives the other number. By this we iterate only once since we already know target value and iterate n value to find whether the other number is present in the list or not.

If the other number is present we return a list giving its index.

If not present it continues its iterations by adding key value pairs to hashmap and checking if the other number is present or not.

METHOD 2: Using nested loops to iterate through the list

class Solution:

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

for i in range(len(nums)):

for j in range(i+1,len(nums)):

if(nums[i]+nums[j]==target):

return [i,j]

This is the basic way of solving this code. The time complexity here is more compared to HashMap method.

Time complexity: O(n^2)

Here we use if condition to check whether the condition is satisfied or not. If satisfied then return the index as a list if not it coninues the iteration.