**Python Questions:**  
 1. Write a Python function to find all unique pairs of integers in a list that sum up to a given target value.

<https://leetcode.com/problems/two-sum/description/>

class Solution:  
 def twoSum(self, nums: List[int], target: int) -> List[int]:  
 d = {}  
 for i,v in enumerate(nums):  
 diff = target-v  
 if diff in d.keys():  
 return [i,d[diff]]  
 d[v] = i  
 return []

2. Given a string, write a function to check if it’s a palindrome, ignoring spaces, punctuation, and case sensitivity.

def str\_palin(s):  
 s\_new = re.sub(r'[^a-zA-Z0-9]','',s)  
 return s\_new==s\_new[::-1]

3. Explain the difference between deep copy and shallow copy in Python. When would you use each?

**✅ Shallow Copy**

* Creates a **new object**, but **references the same nested objects** (inner lists, dicts, etc.).
* Only **one level** is copied; the rest is shared.

**✅ Deep Copy**

* Creates a **new object**, and also **recursively copies all nested objects**.
* Fully independent from the original.

import copy

original = [[1, 2], [3, 4]]

shallow = copy.copy(original)

deep = copy.deepcopy(original)

original[0][0] = 99

print("Original:", original) # [[99, 2], [3, 4]]

print("Shallow:", shallow) # [[99, 2], [3, 4]] (shared inner list)

print("Deep:", deep) # [[1, 2], [3, 4]] (fully independent)

4. What are decorators in Python, and how do they work? Provide an example of a scenario where a decorator would be useful.

A **decorator** is a function that **modifies the behavior** of another function **without changing its code**. It wraps a function to add extra functionality (like logging, timing, checking permissions, etc.).

def my\_decorator(func):

def wrapper():

print("Before the function call")

func()

print("After the function call")

return wrapper

@my\_decorator

def say\_hello():

print("Hello!")

say\_hello()