You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example 1:**

**Input:** l1 = [2,4,3], l2 = [5,6,4] **Output:** [7,0,8] **Explanation:** 342 + 465 = 807.

**Example 2:**

**Input:** l1 = [0], l2 = [0] **Output:** [0]

**Example 3:**

**Input:** l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9] **Output:** [8,9,9,9,0,0,0,1]

**Constraints:**

The number of nodes in each linked list is in the range [1, 100].  
0 <= Node.val <= 9 It is guaranteed that the list represents a number that does not have leading zeros.

**Note:**Create a GitHub file for the solution and add the file link the the answer section below.

Solution :

class Solution:

def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:

dummy = ListNode(0)

curr = dummy

carry = 0

while l1 is not None or l2 is not None:

l1\_val = l1.val if l1 is not None else 0

l2\_val = l2.val if l2 is not None else 0

sum\_val = l1\_val + l2\_val + carry

carry = sum\_val // 10

curr.next = ListNode(sum\_val % 10)

curr = curr.next

if l1 is not None:

l1 = l1.next

if l2 is not None:

l2 = l2.next

if carry > 0:

curr.next = ListNode(carry)

return dummy.next