ASSIGNMENT-1

**NAME:**V.VamsidharReddy

**Reg.no**:192365029

**Course code**:CSA0678

**Couse name**: DAA[Design and analysis of algorithm]

1. Two Sum

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.

**ALGORITHM:**

Initialize an empty hash map (or dictionary) to store the numbers and their corresponding indices.

Iterate through the array of numbers.

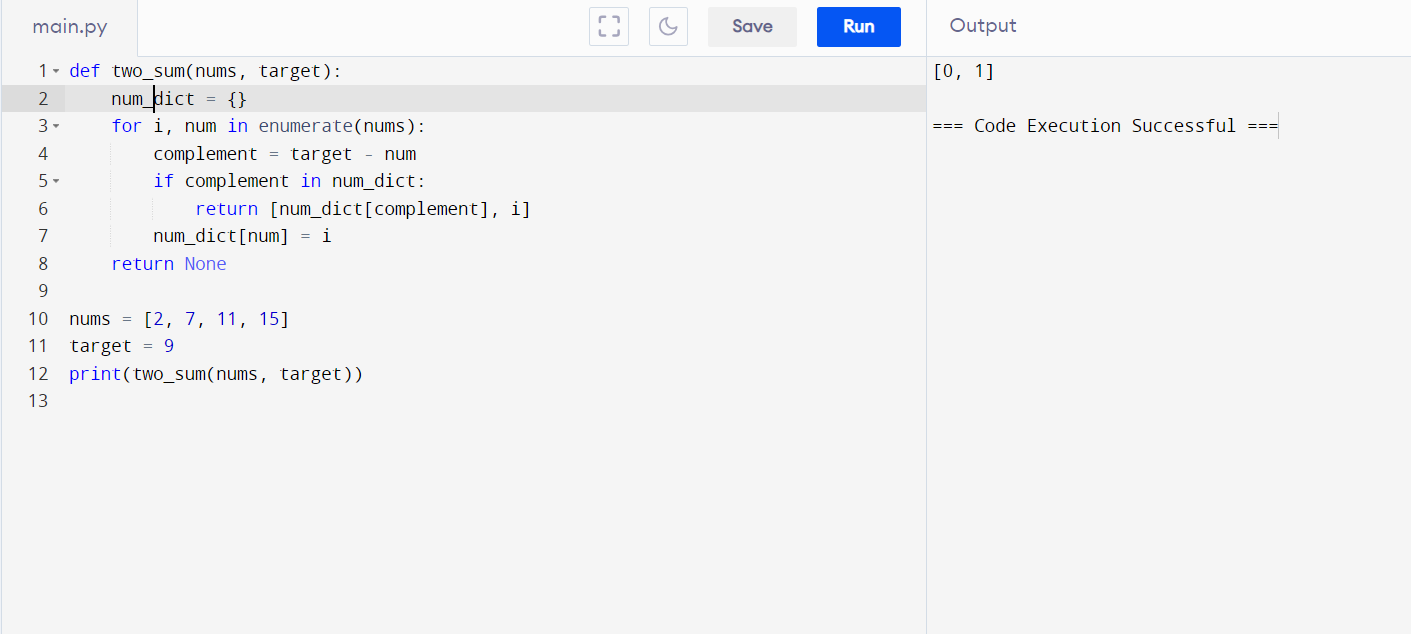
For each number, calculate the complement needed to reach the target by subtracting the current number from the target.

Check if this complement exists in the hash map.

If it does, return the indices of the current number and the complement.

If it does not, add the current number and its index to the hash map.

If no such pair is found by the end of the iteration, return an indication that no solution exists



1. Add Two Numbers

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.

**Algorithm**:

 Initialize a dummy head for the result linked list. This helps simplify the code by avoiding special cases for the head node.

 Initialize a current pointer to build the result linked list and a carry variable to store the carry-over value.

 Traverse both linked lists simultaneously, adding corresponding digits along with the carry.

 If one linked list is shorter than the other, treat its missing nodes as zeros.

 Update the carry for the next iteration.

 If there is a remaining carry after the end of both linked lists, add a new node with this carry value.

 Return the next node of the dummy head, which points to the head of the result linked list.

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def addTwoNumbers(l1, l2):

dummy = ListNode()

current, carry = dummy, 0

while l1 or l2 or carry:

val1, val2 = (l1.val if l1 else 0), (l2.val if l2 else 0)

carry, out = divmod(val1 + val2 + carry, 10)

current.next = ListNode(out)

current = current.next

l1, l2 = (l1.next if l1 else None), (l2.next if l2 else None)

return dummy.next

def create\_linked\_list(lst):

dummy = ListNode()

current = dummy

for number in lst:

current.next = ListNode(number)

current = current.next

return dummy.next

def linked\_list\_to\_list(node):

result = []

while node:

result.append(node.val)

node = node.next

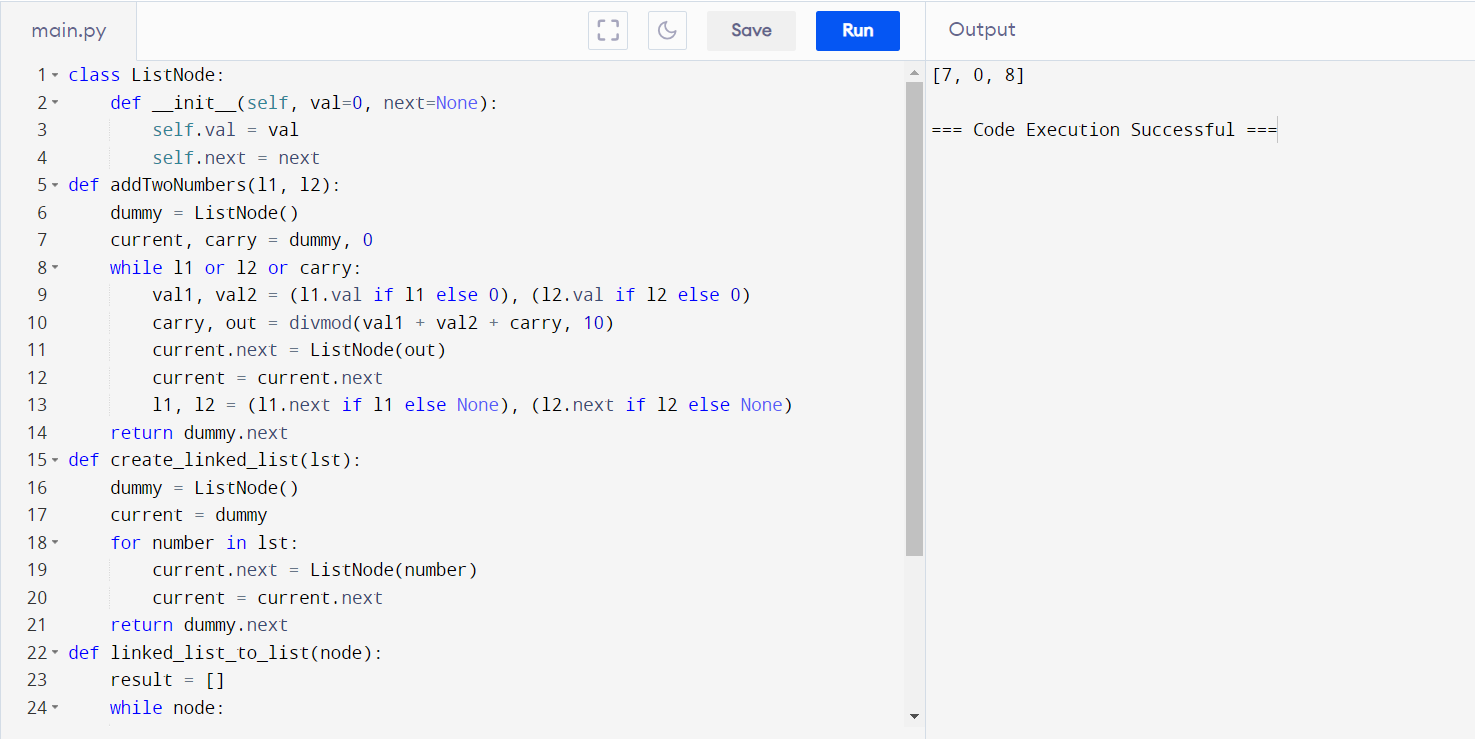
return result

l1 = create\_linked\_list([2, 4, 3])

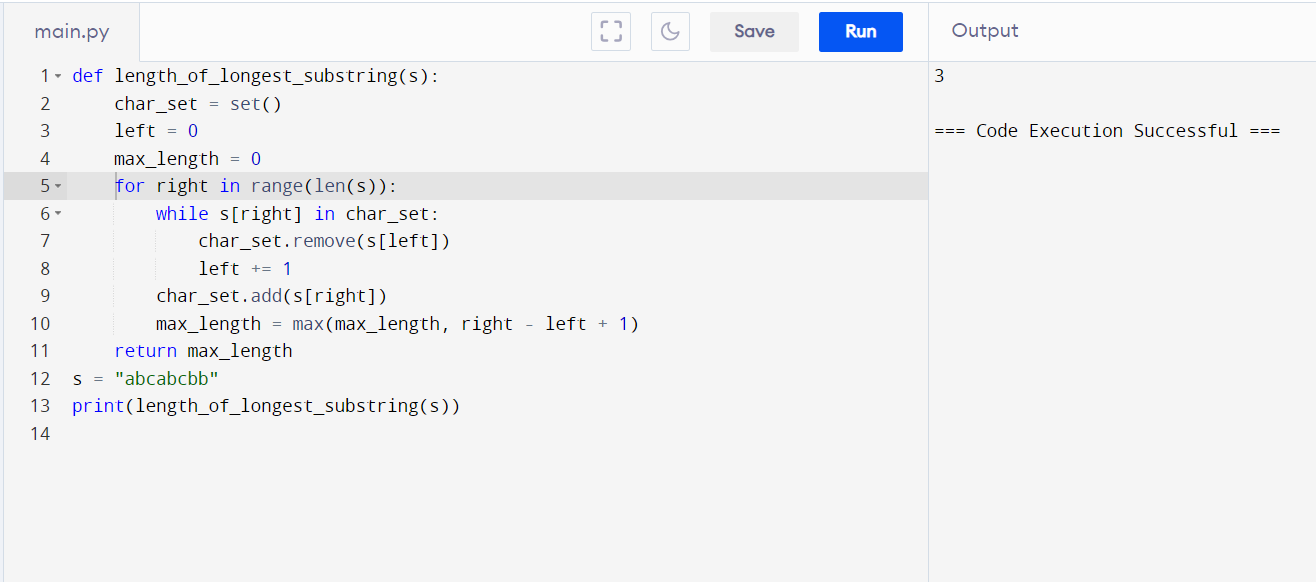
l2 = create\_linked\_list([5, 6, 4])

result = addTwoNumbers(l1, l2)

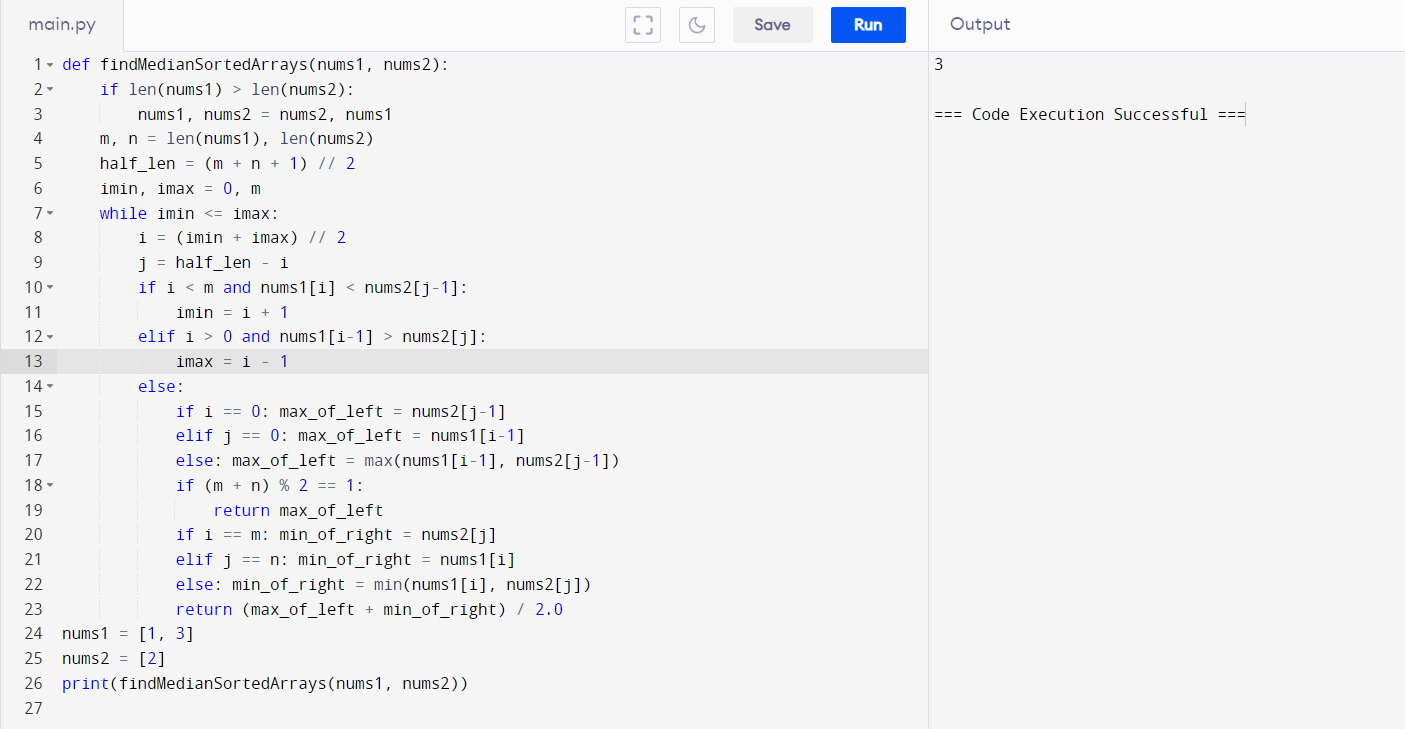
print(linked\_list\_to\_list(result))



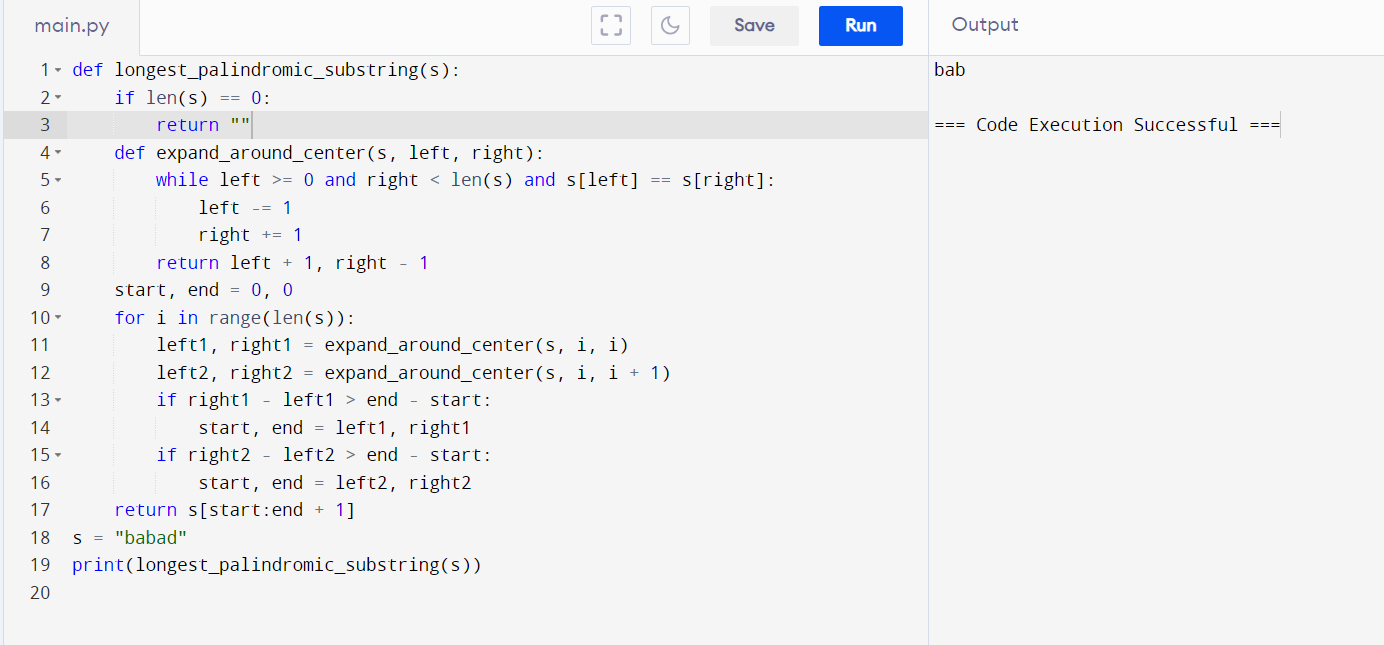
1. Longest Substring without Repeating Characters

Given a string s, find the length of the longest substring without repeating characters.

1. Median of Two Sorted Arrays

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).

1. **Longest Palindromic Substring**

Given a string s, return the longest palindromic substring in s.

1. Zigzag Conversion

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility) P A H N A P L S I I G Y I R And then read line by line: "PAHNAPLSIIGYIR" Write the code that will take a string and make this conversion given a number of rows: string convert(string s, int numRows);

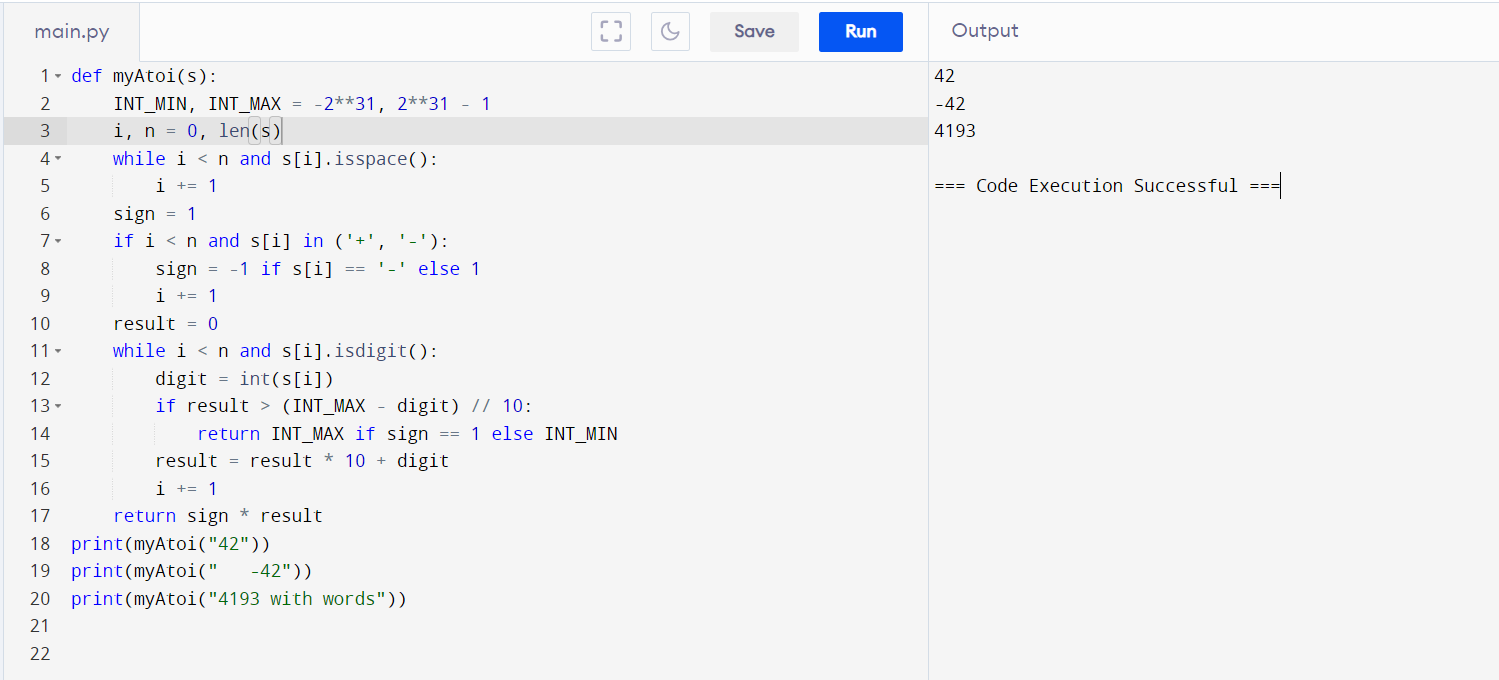


1. Reverse Integer

Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0. Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

1. String to Integer

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer



1. Palindrome Number

Given an integer x, return true if x is a palindrome, and false otherwise.

1. Regular Expression Matching

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*' where: ● '.' Matches any single character. ● '\*' Matches zero or more of the preceding element. The matching should cover the entire input.

t string (not partial).