

Algorithms and structures programming

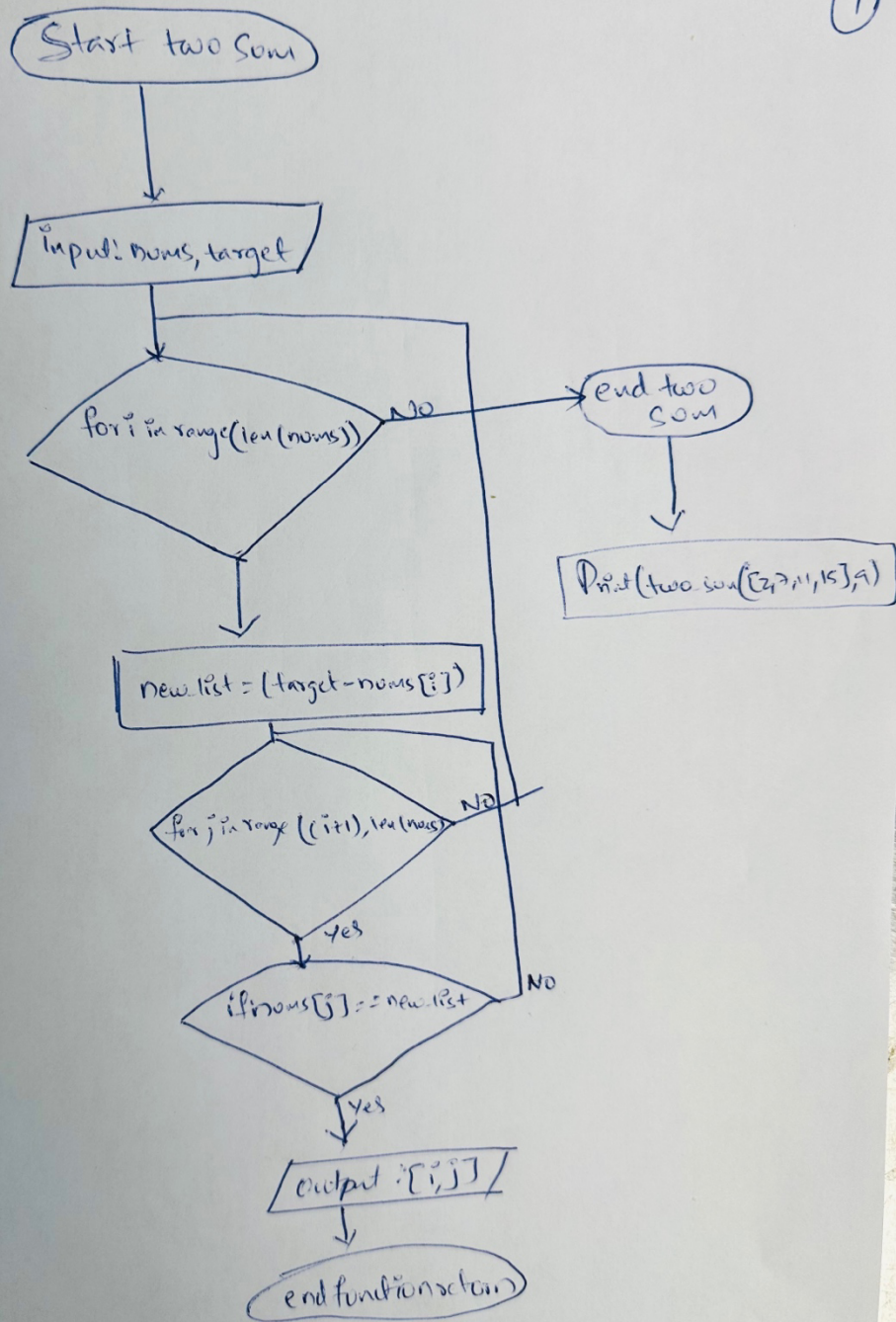
Rakesh Kasha
19695

Python Program:

```
def two_sum(nums, target):  
  
    for i in range(len(nums)):  
  
        new_list = target - nums[i]  
  
        for j in range(i + 1, len(nums)):  
  
            if nums[j] == new_list:  
                return [i, j]  
  
print(two_sum([2,7,11,15], 9))  
print(two_sum([2,3,4],6))  
print(two_sum([3,3],6))
```

Flowchart:

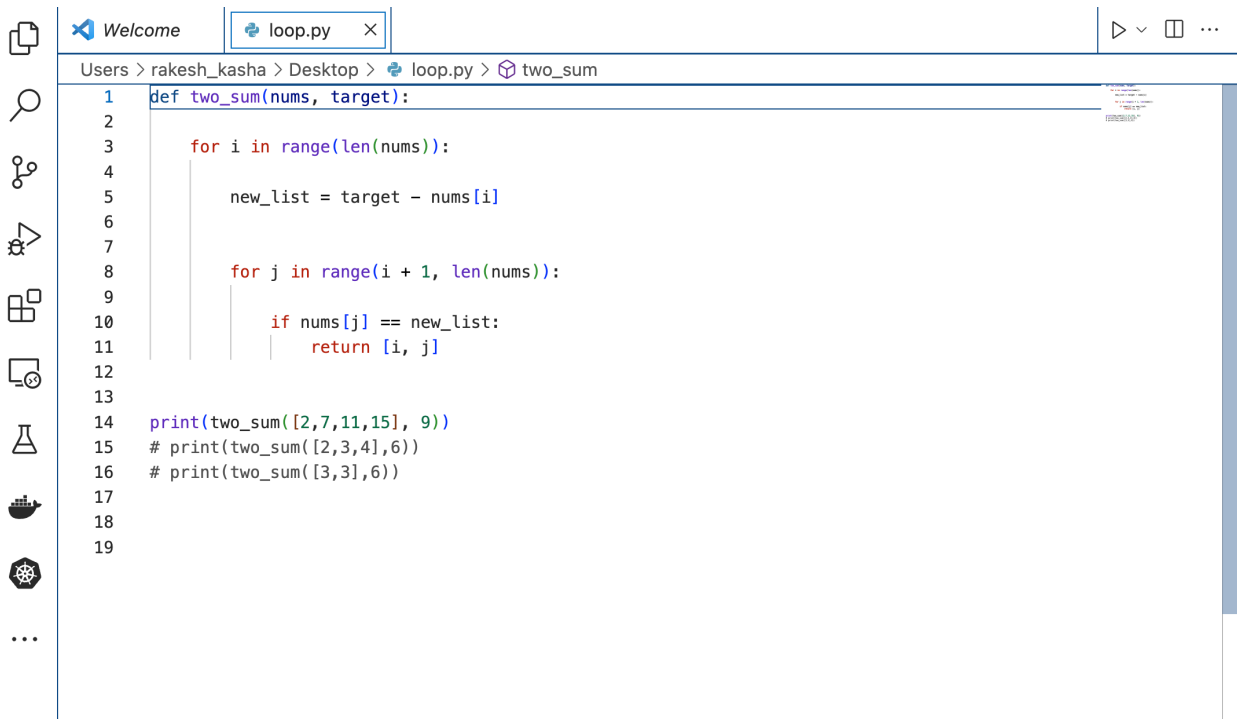
①



Trace Table:

Step	Nums	Target	I	J	New_list	Nums[j]==new_list	screen
1	[2,7,11,15]	9					
2			0				
3					9-2=7		
4				1			
5						True	
6							[0,1]

Test Case:



The screenshot shows a code editor with a file named 'loop.py' open. The code defines a function 'two_sum(nums, target)' that uses nested loops to find two numbers in a list that add up to a target. Below the function, there are three print statements testing the function with different inputs. The editor interface includes a sidebar with various icons and a top bar with file tabs and a run button.

```

1 def two_sum(nums, target):
2
3     for i in range(len(nums)):
4
5         new_list = target - nums[i]
6
7
8         for j in range(i + 1, len(nums)):
9
10            if nums[j] == new_list:
11                return [i, j]
12
13
14 print(two_sum([2,7,11,15], 9))
15 # print(two_sum([2,3,4],6))
16 # print(two_sum([3,3],6))
17
18
19

```