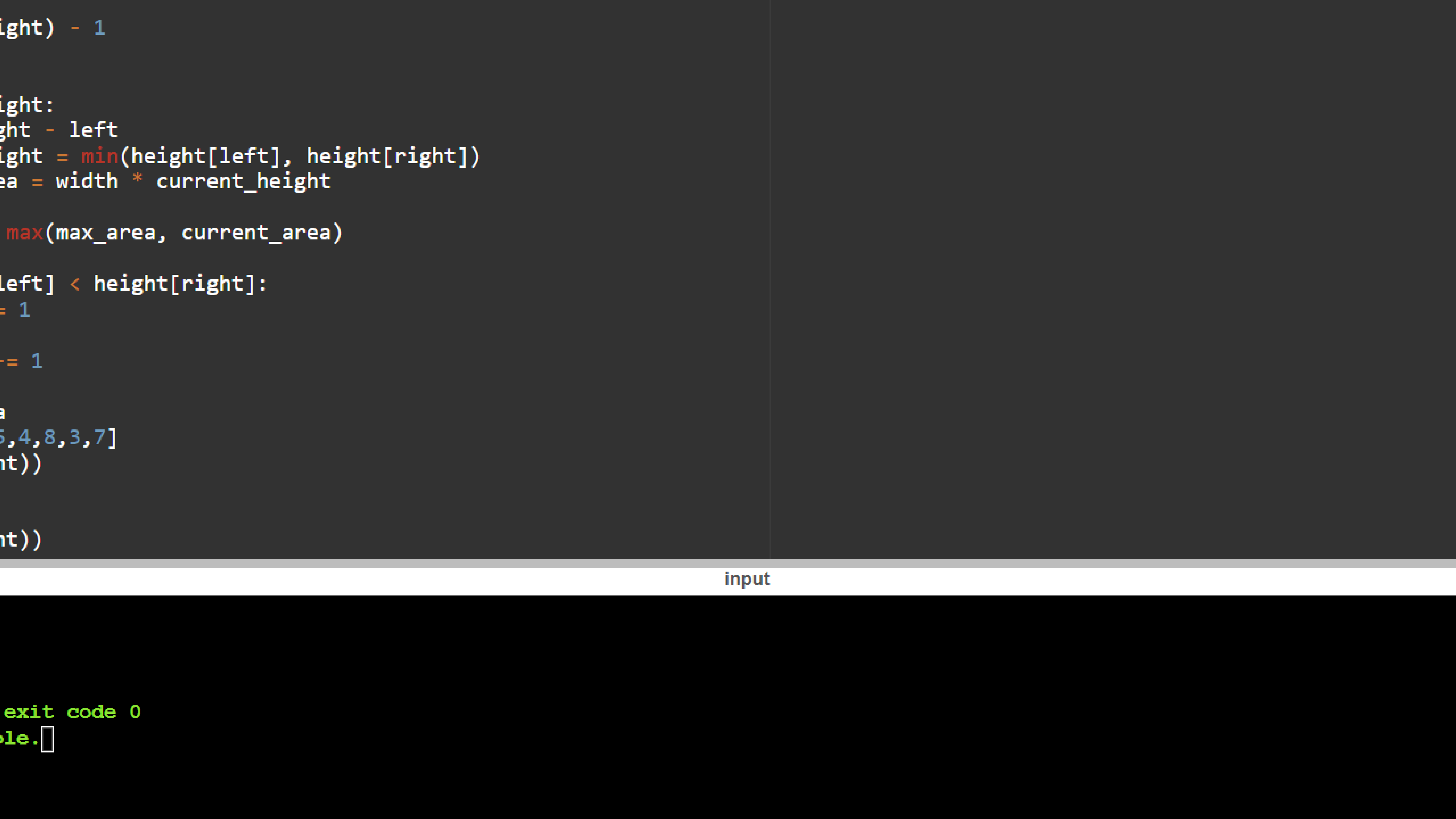
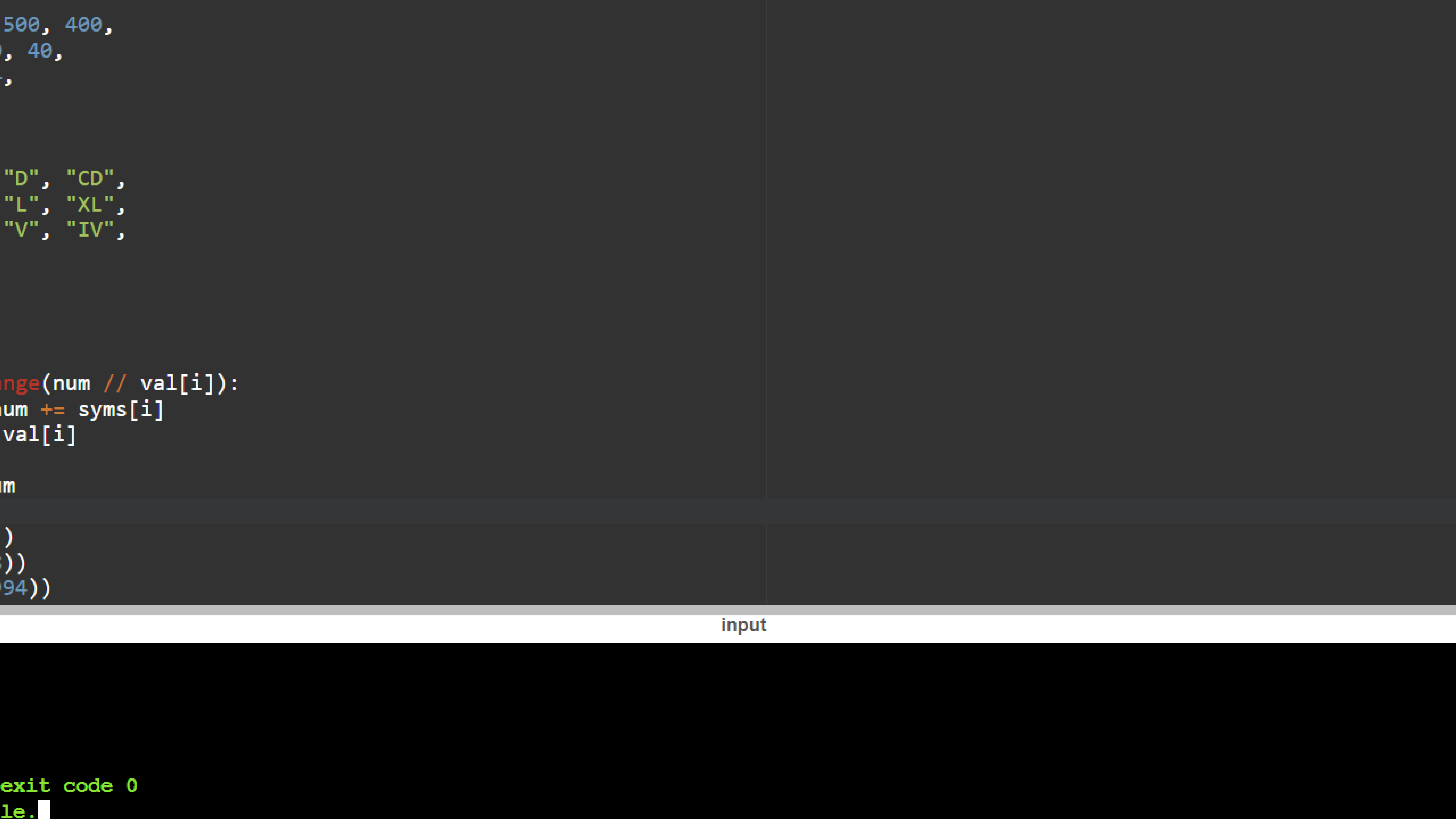


```
ight) - 1


ight:
ght - left
ight = min(height[left], height[right])
ea = width * current_height

 max(max_area, current_area)

left] < height[right]:
= 1

-= 1

a
5,4,8,3,7]
nt))


nt))
```

input

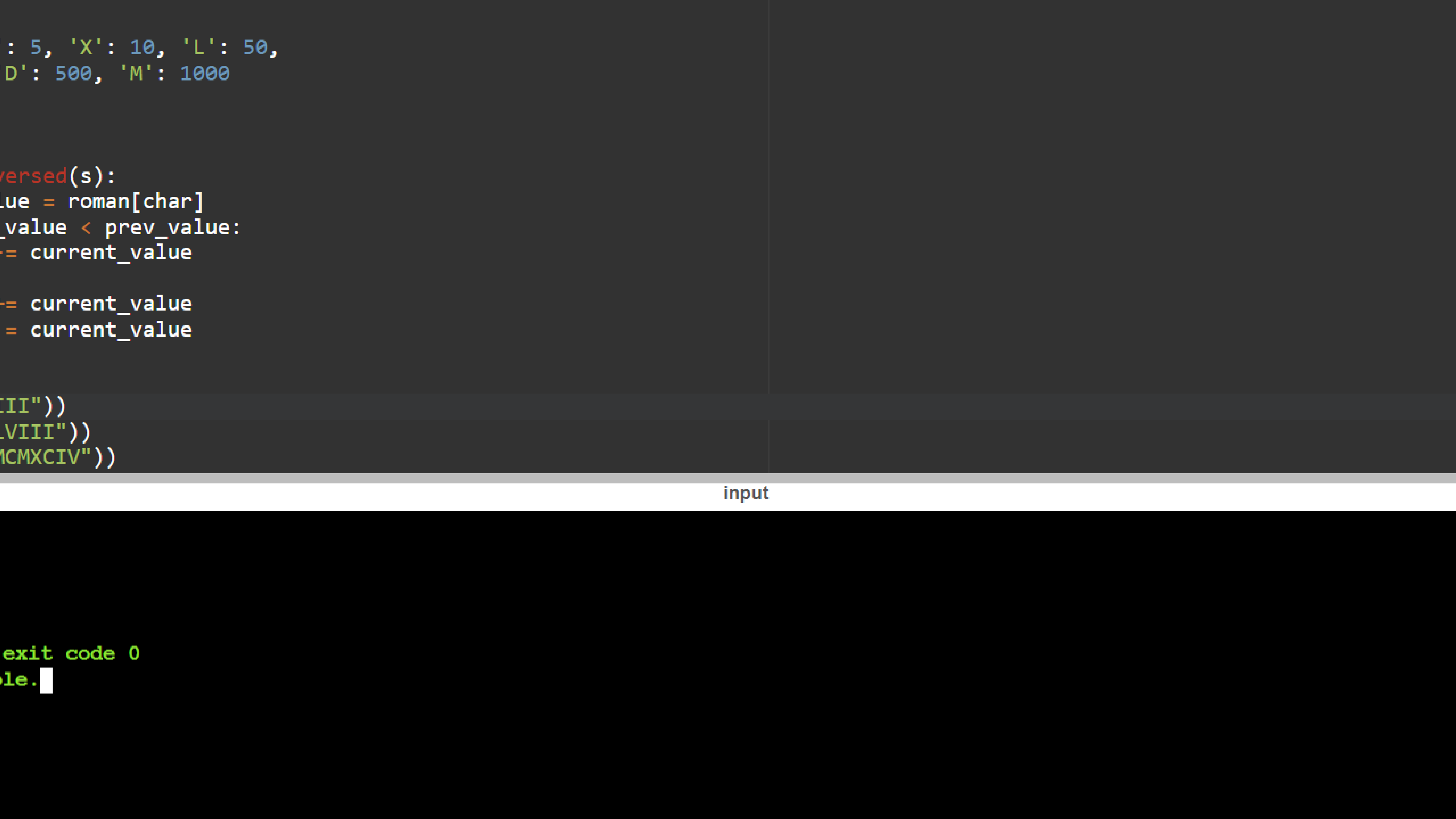
```
 exit code 0
ple.
```



```
 500, 400,
), 40,
,





 "D", "CD",
 "L", "XL",
 "V", "IV",






nge(num // val[i]):
num += syms[i]
 val[i]

um

)
))
94))
```

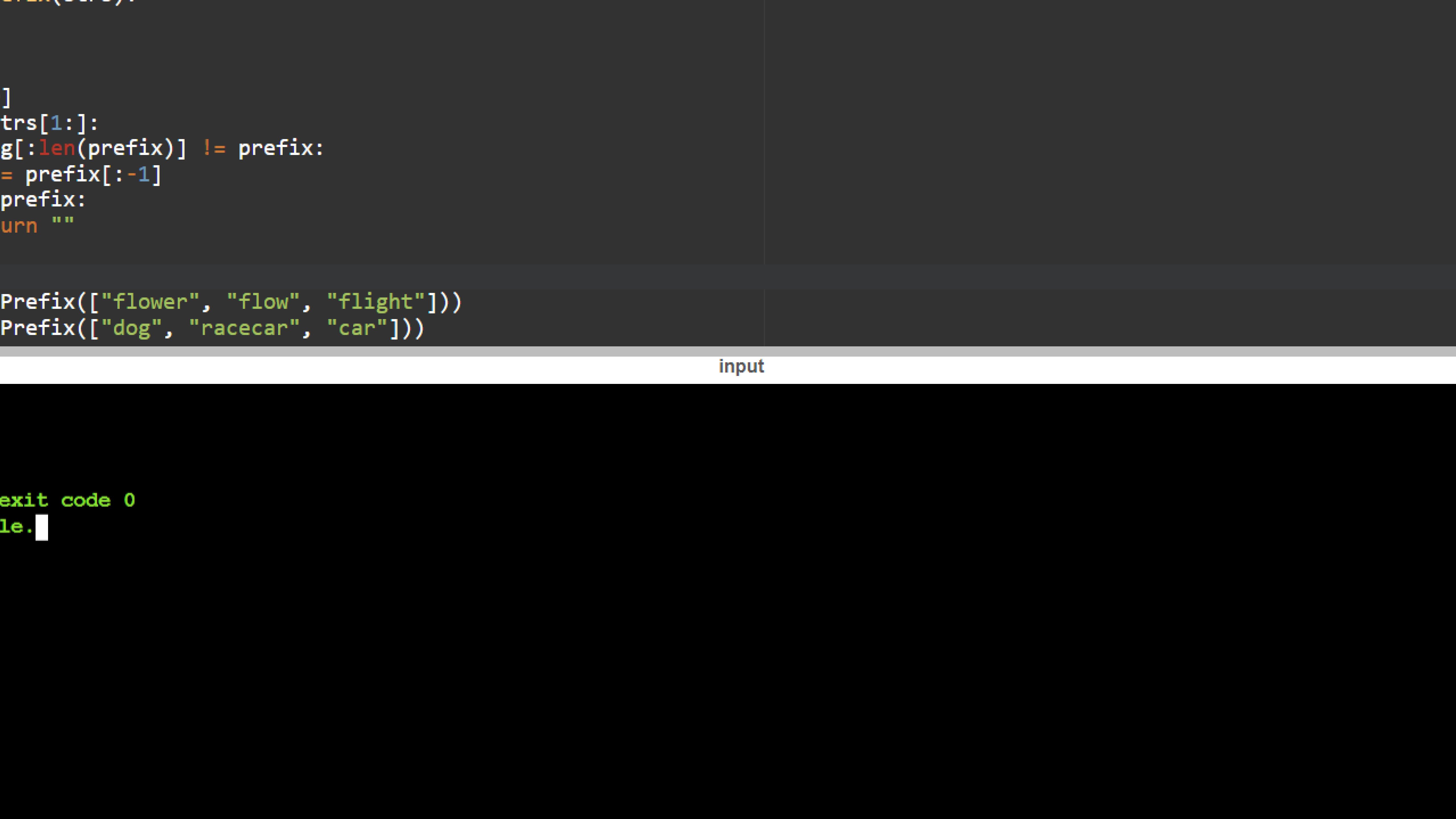
input

```
exit code 0
le.
```



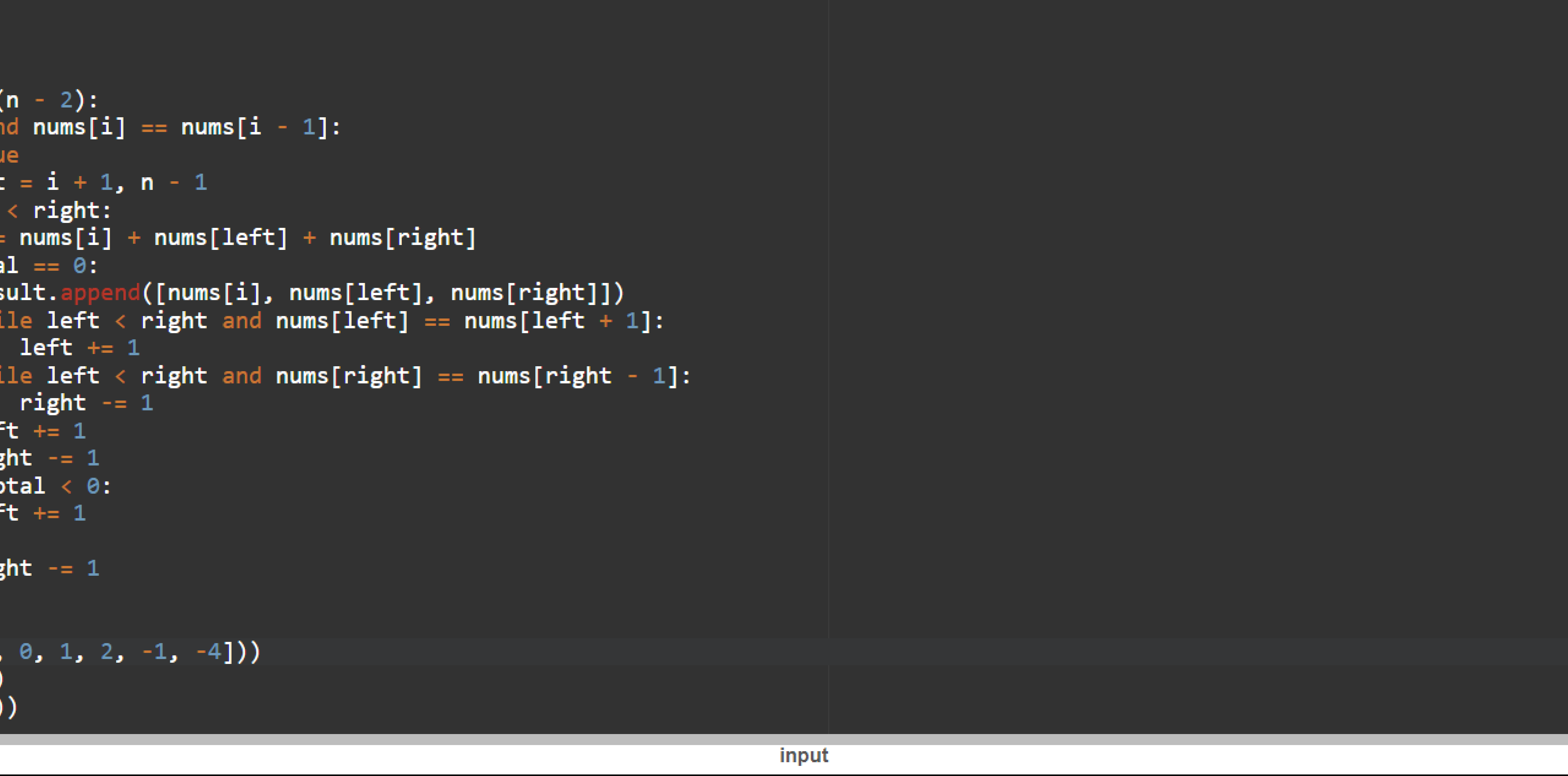
```
': 5, 'X': 10, 'L': 50,
'D': 500, 'M': 1000



versed(s):
lue = roman[char]
_value < prev_value:
-= current_value

+= current_value
 = current_value


III"))
LVIII"))
MCMXCIV"))
```

```
 exit code 0
ole.
```



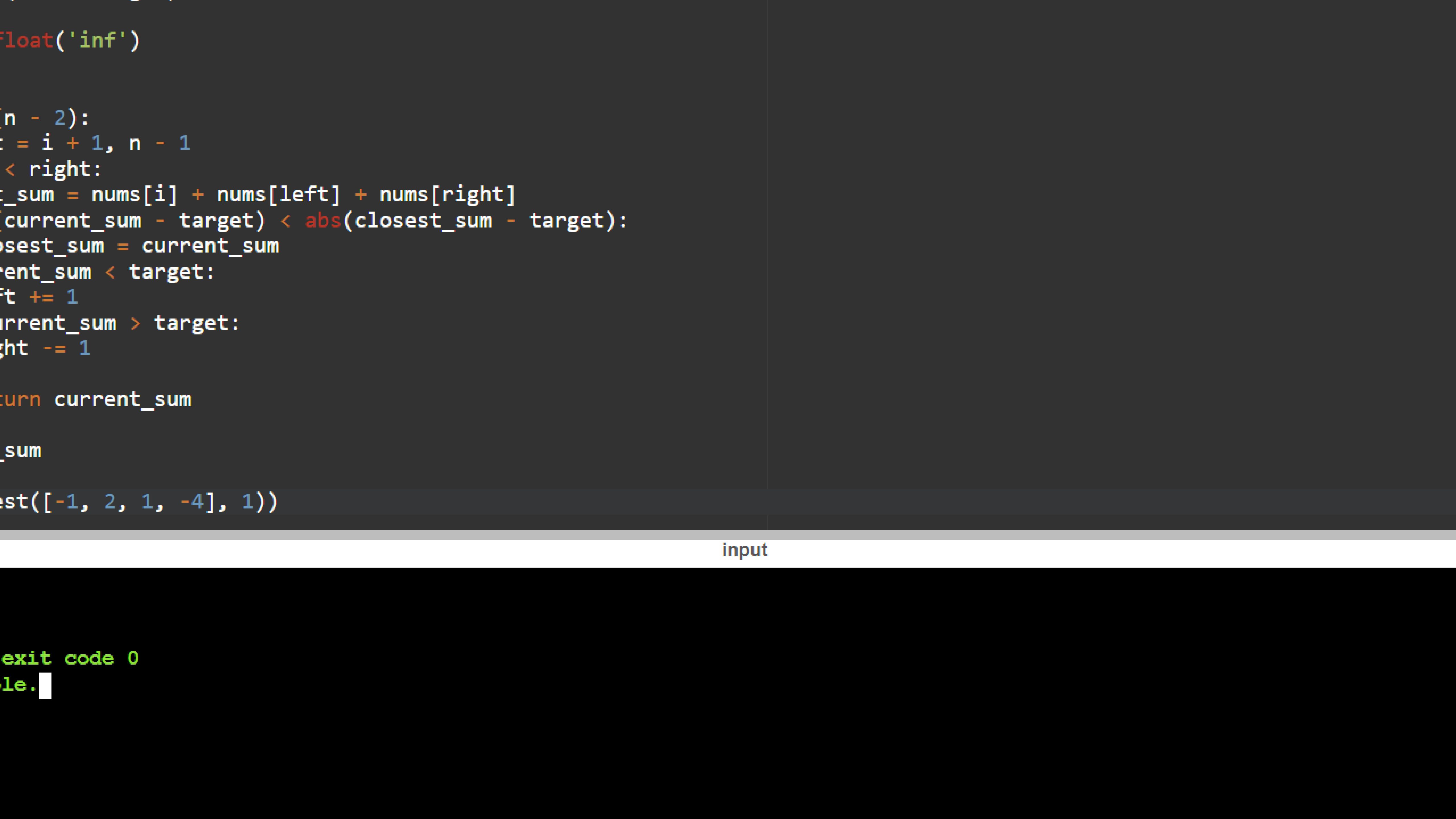
```
efix(strs):




]
trs[1:]:
g[:len(prefix)] != prefix:
= prefix[:-1]
prefix:
urn ""


Prefix(["flower", "flow", "flight"]))
Prefix(["dog", "racecar", "car"]))
```

```
exit code 0
le.
```



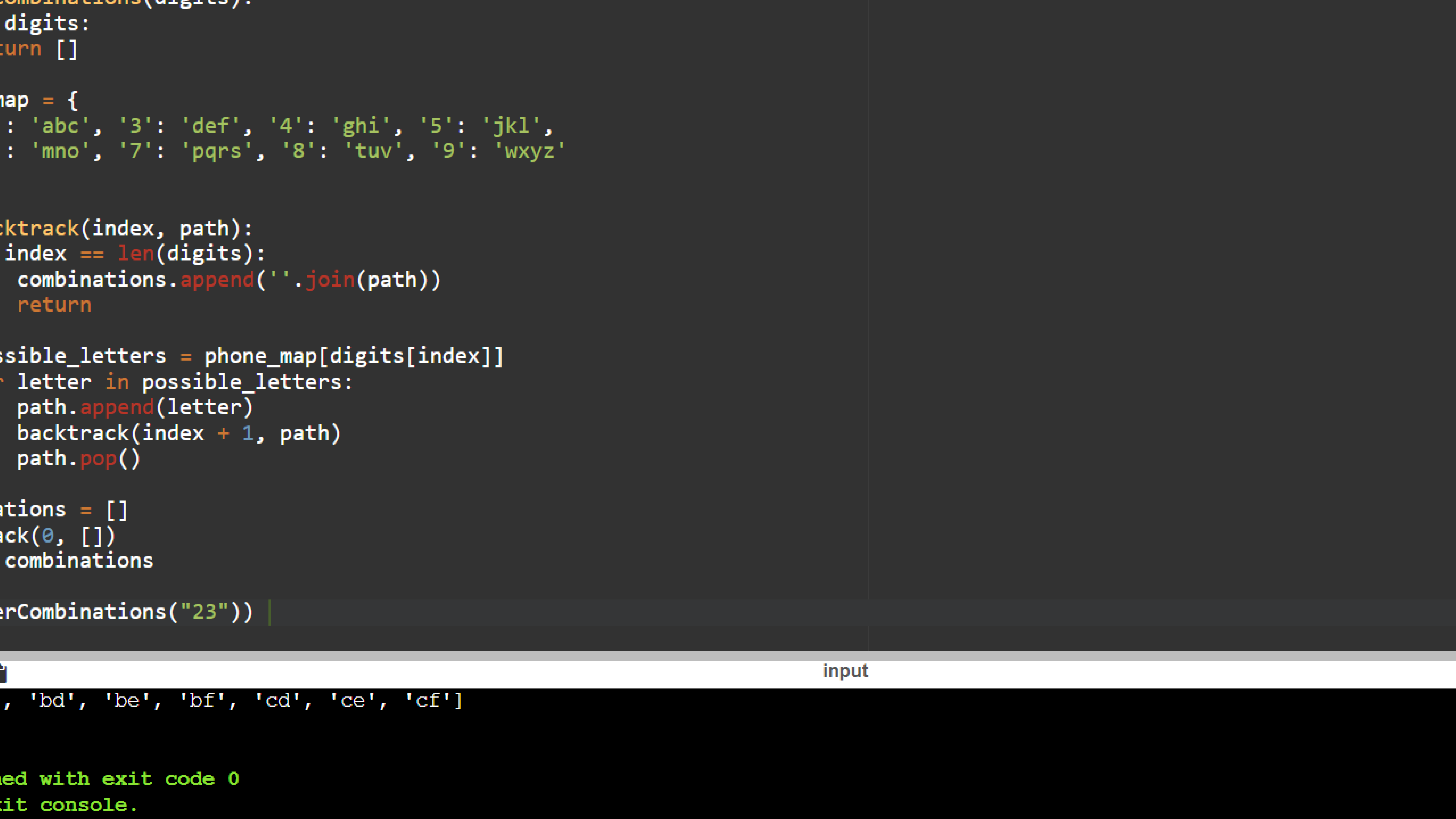
```
(n - 2):
nd nums[i] == nums[i - 1]:
ue
t = i + 1, n - 1
 < right:
= nums[i] + nums[left] + nums[right]
al == 0:
sult.append([nums[i], nums[left], nums[right]])
ile left < right and nums[left] == nums[left + 1]:
  left += 1
ile left < right and nums[right] == nums[right - 1]:
  right -= 1
ft += 1
ght -= 1
otal < 0:
ft += 1

ght -= 1



, 0, 1, 2, -1, -4]))
)
))
```

input



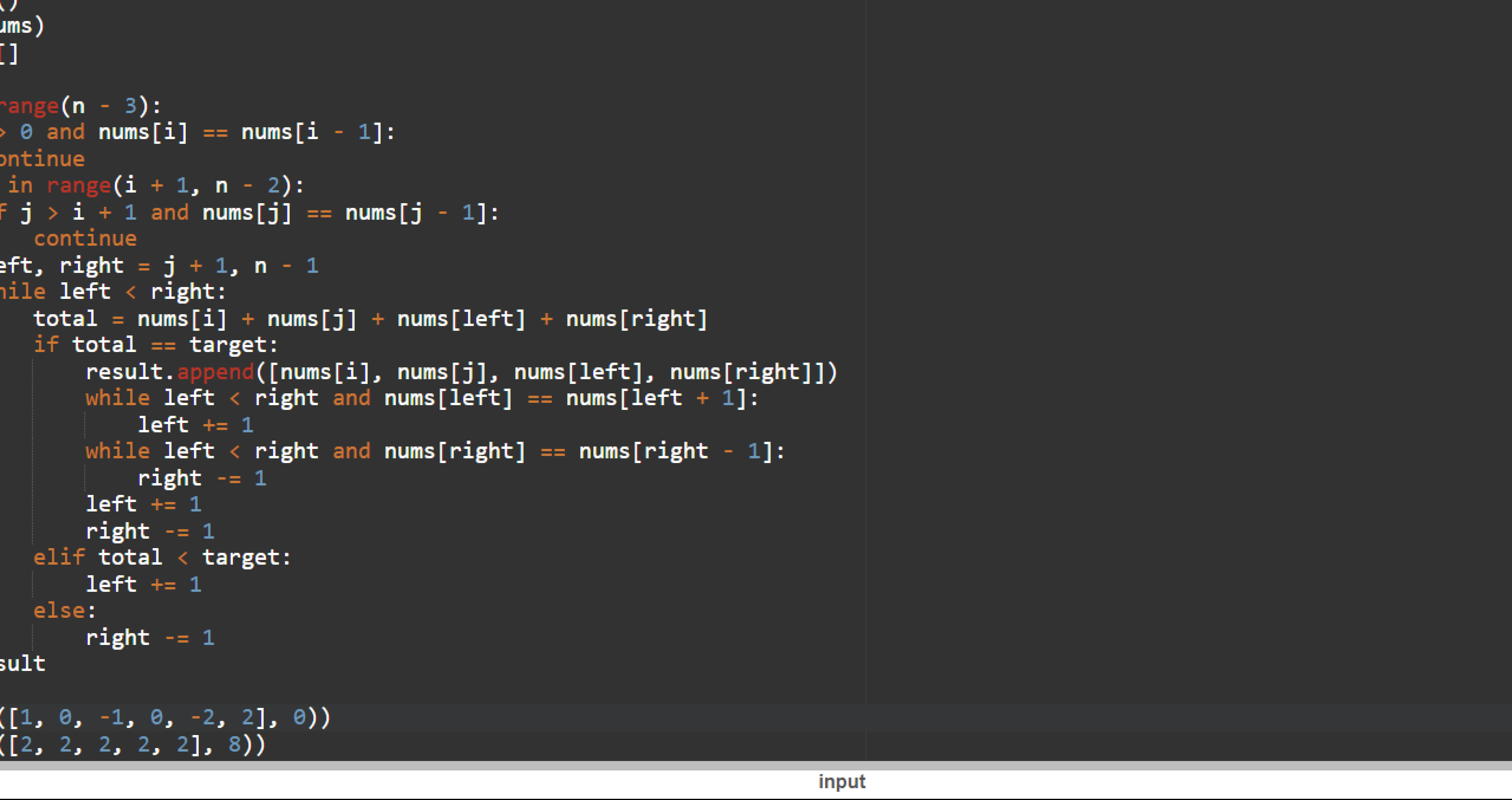
```
float('inf')


(n - 2):
 = i + 1, n - 1
 < right:
t_sum = nums[i] + nums[left] + nums[right]
(current_sum - target) < abs(closest_sum - target):
osest_sum = current_sum
rent_sum < target:
ft += 1
urrent_sum > target:
ght -= 1

turn current_sum

_sum

est([-1, 2, 1, -4], 1))
```

```
exit code 0
le.
```



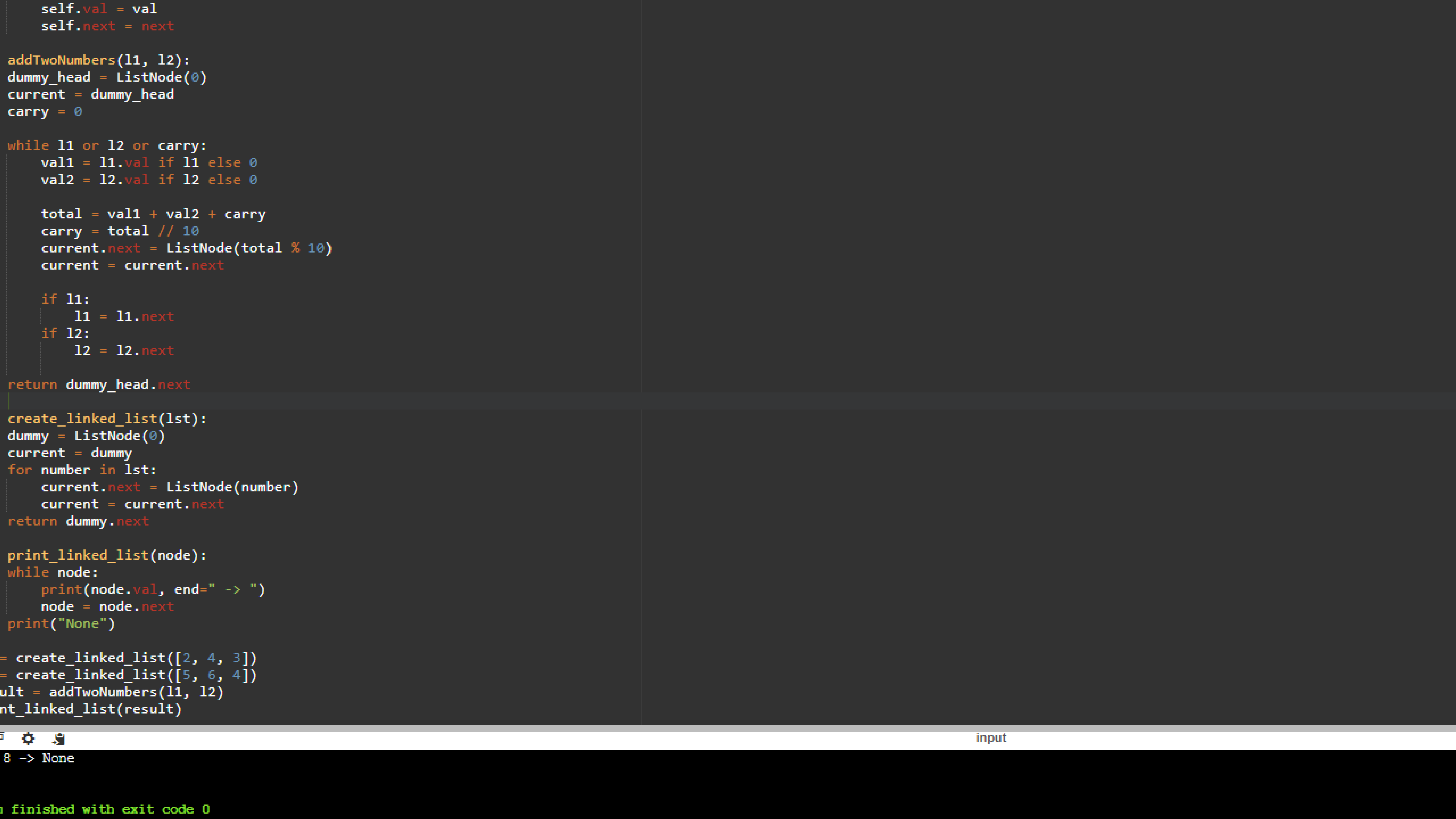
```
ombinations(digits):
 digits:
turn []

ap = {
 : 'abc', '3': 'def', '4': 'ghi', '5': 'jkl',
 : 'mno', '7': 'pqrs', '8': 'tuv', '9': 'wxyz'


cktrack(index, path):
 index == len(digits):
  combinations.append(''.join(path))
  return

ssible_letters = phone_map[digits[index]]
r letter in possible_letters:
  path.append(letter)
  backtrack(index + 1, path)
  path.pop()

ations = []
ack(0, [])
 combinations

erCombinations("23"))
```

```
, 'bd', 'be', 'bf', 'cd', 'ce', 'cf']


ed with exit code 0
it console.
```



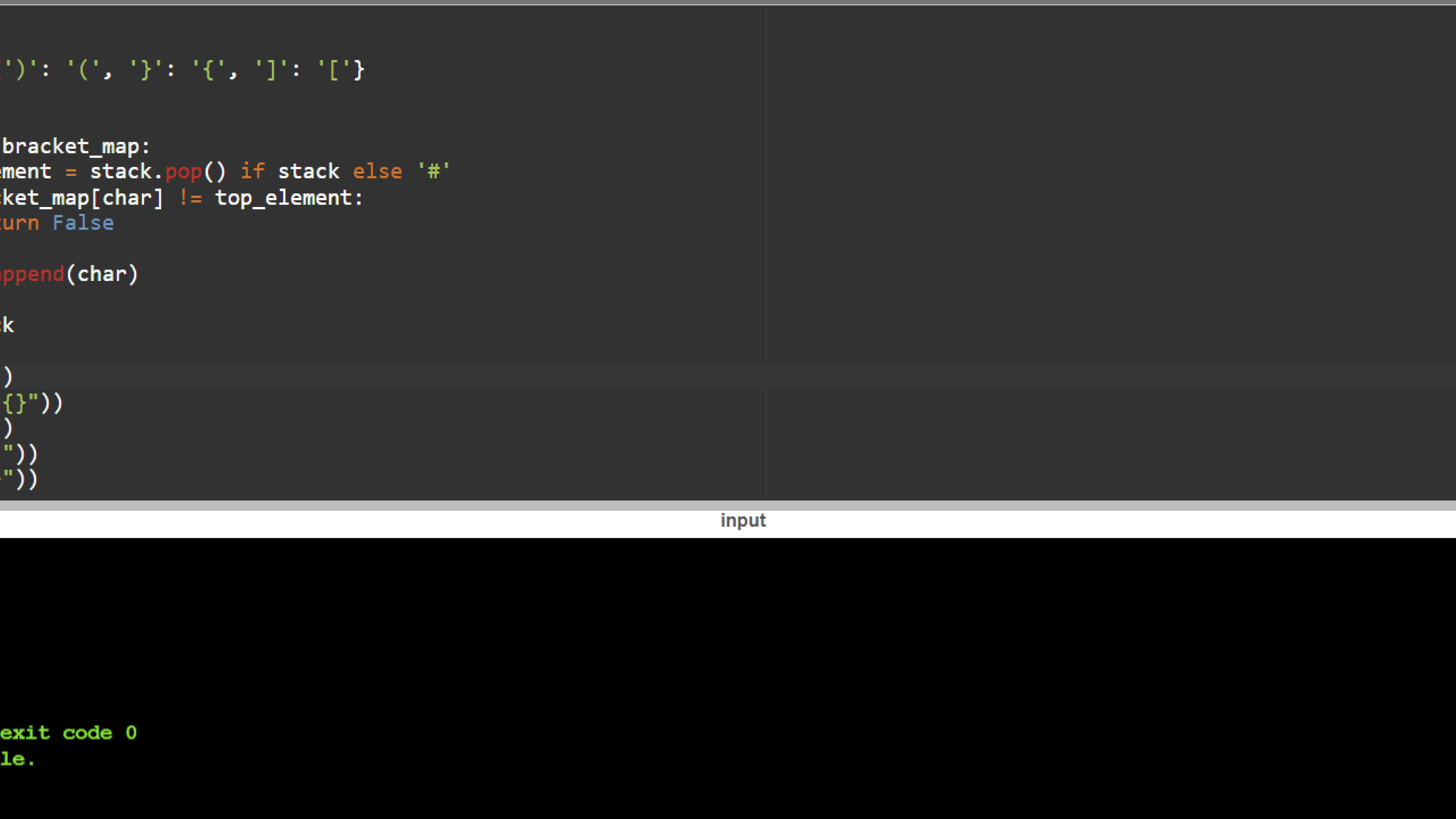
```
()
ums)
[]

range(n - 3):
> 0 and nums[i] == nums[i - 1]:
ontinue
 in range(i + 1, n - 2):
f j > i + 1 and nums[j] == nums[j - 1]:
   continue
eft, right = j + 1, n - 1
hile left < right:
   total = nums[i] + nums[j] + nums[left] + nums[right]
   if total == target:
       result.append([nums[i], nums[j], nums[left], nums[right]])
       while left < right and nums[left] == nums[left + 1]:
           left += 1
       while left < right and nums[right] == nums[right - 1]:
           right -= 1
       left += 1
       right -= 1
   elif total < target:
       left += 1
   else:
       right -= 1
sult

([1, 0, -1, 0, -2, 2], 0))
([2, 2, 2, 2, 2], 8))
```

input



```
    self.val = val
    self.next = next

addTwoNumbers(l1, l2):
dummy_head = ListNode(0)
current = dummy_head
carry = 0

while l1 or l2 or carry:
    val1 = l1.val if l1 else 0
    val2 = l2.val if l2 else 0

    total = val1 + val2 + carry
    carry = total // 10
    current.next = ListNode(total % 10)
    current = current.next

    if l1:
        l1 = l1.next
    if l2:
        l2 = l2.next

return dummy_head.next

create_linked_list(lst):
dummy = ListNode(0)
current = dummy
for number in lst:
    current.next = ListNode(number)
    current = current.next
return dummy.next

print_linked_list(node):
while node:
    print(node.val, end=" -> ")
    node = node.next
print("None")

= create_linked_list([2, 4, 3])
= create_linked_list([5, 6, 4])
ult = addTwoNumbers(l1, l2)
nt_linked_list(result)
```

```
8 -> None


n finished with exit code 0
```



```
')': '(', '}': '{', ']': '['}


bracket_map:
ement = stack.pop() if stack else '#'
ket_map[char] != top_element:
turn False


ppend(char)


k


)
{}"))
)
"))
"))
```

input

```
exit code 0
le.
```