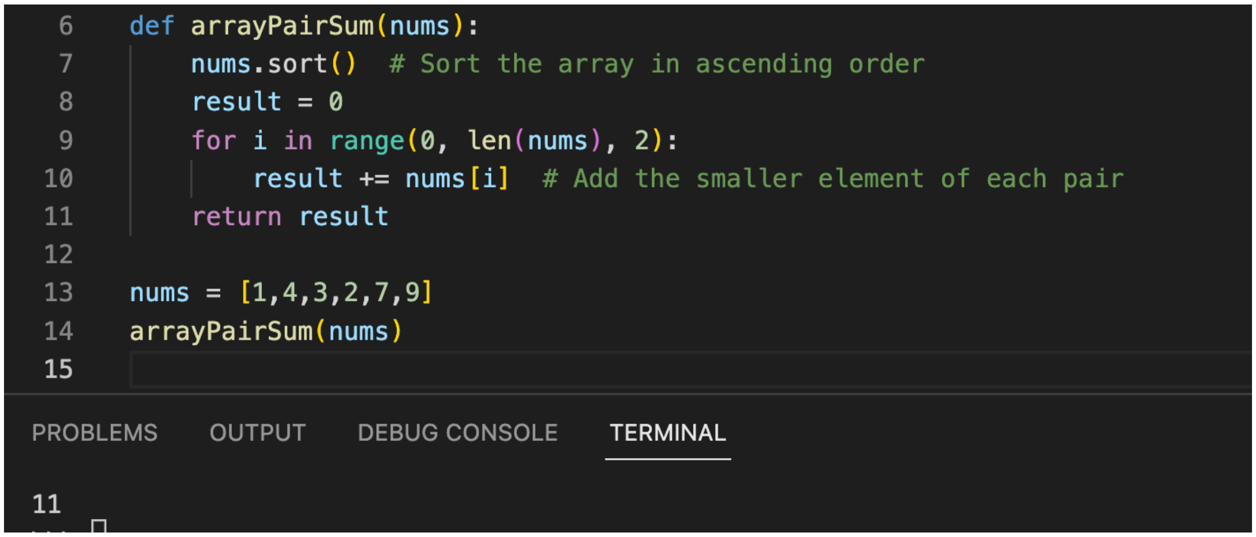
PPT Assignment 2

💡 **Question 1** Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2),..., (an, bn) such that the sum of min(ai, bi) for all i is maximized. Return the maximized sum.

Example 1: Input: nums = [1,4,3,2] Output: 4

Explanation: All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4 So the maximum possible sum is 4



```python
def arrayPairSum(nums):
    nums.sort()  # Sort the array in ascending order
    result = 0
    for i in range(0, len(nums), 2):
        result += nums[i]  # Add the smaller element of each pair
    return result

nums = [1,4,3,2,7,9]
arrayPairSum(nums)
```

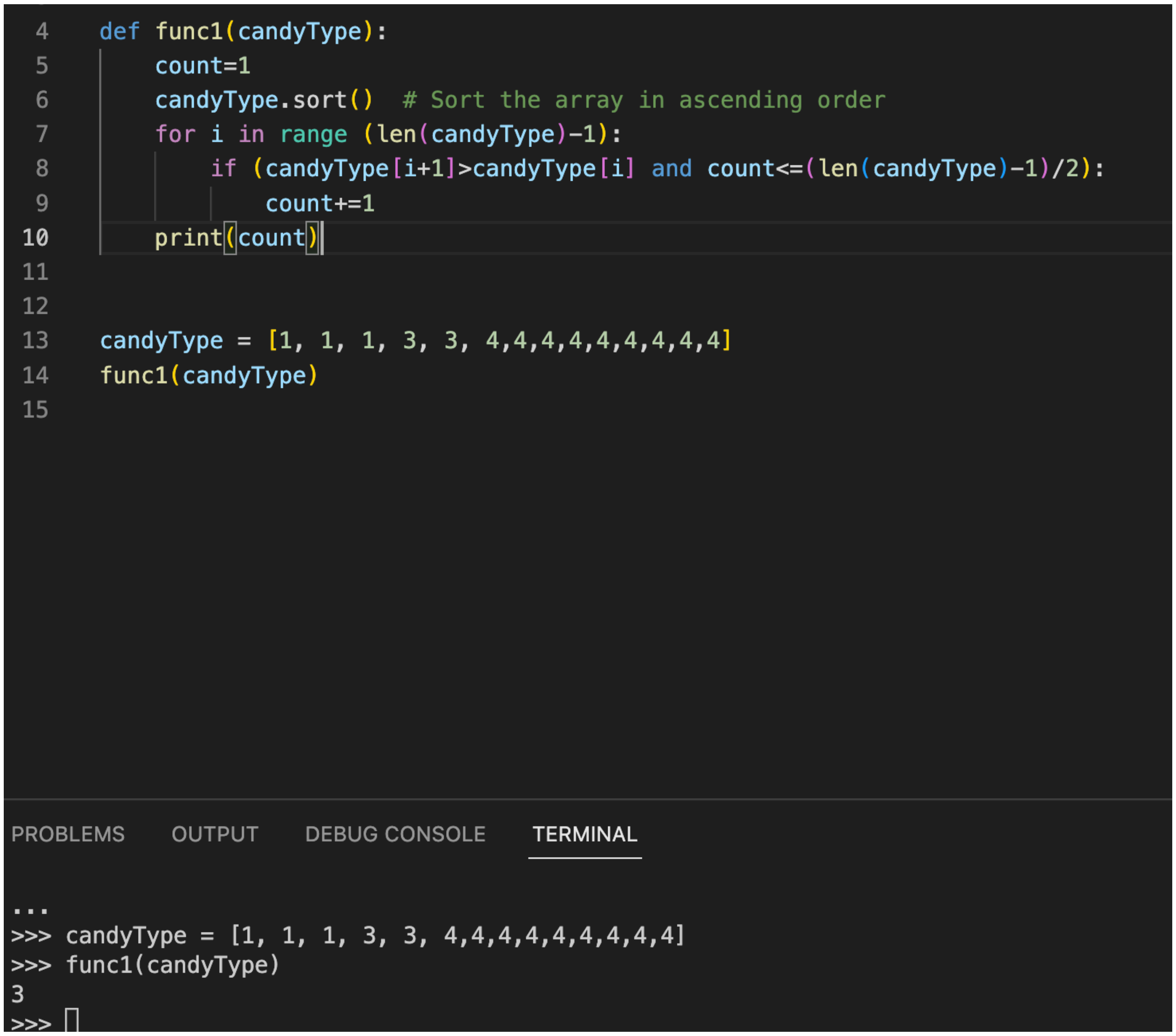
```
11
```

Question 2 Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she started to gain weight, so she visited a doctor.

The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Alice likes her candies very much, and she wants to eat the maximum number of different types of candies while still following the doctor's advice.

Given the integer array candyType of length n, return the maximum number of different types of candies she can eat if she only eats n / 2 of them.

Example 1: Input: candyType = [1,1,2,2,3,3]
Output: 3
Explanation: Alice can only eat 6 / 2 = 3 candies. Since there are only 3 types, she can eat one of each type.



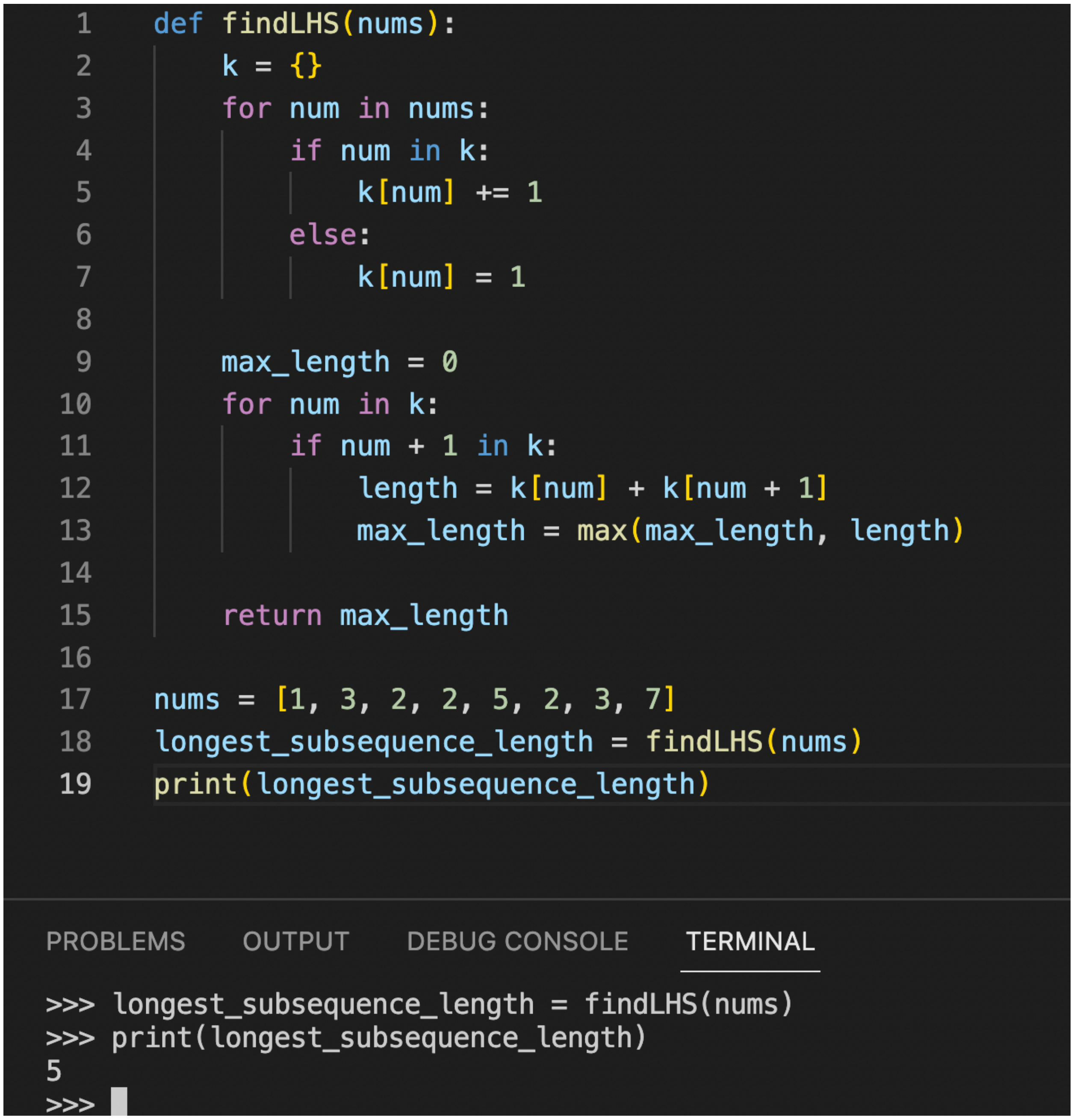
```python
def func1(candyType):
    count=1
    candyType.sort()  # Sort the array in ascending order
    for i in range (len(candyType)-1):
        if (candyType[i+1]>candyType[i] and count<=(len(candyType)-1)/2):
            count+=1
    print(count)


candyType = [1, 1, 1, 3, 3, 4,4,4,4,4,4,4,4,4]
func1(candyType)
```

```
...
>>> candyType = [1, 1, 1, 3, 3, 4,4,4,4,4,4,4,4,4]
>>> func1(candyType)
3
>>> 
```

Question 3 We define a harmonious array as an array where the difference between its maximum value and its minimum value is exactly 1.

Given an integer array nums, return the length of its longest harmonious subsequence among all its possible subsequences.

A subsequence of an array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements.
Example 1: Input: nums = [1,3,2,2,5,2,3,7] Output: 5
Explanation: The longest harmonious subsequence is [3,2,2,2,3]



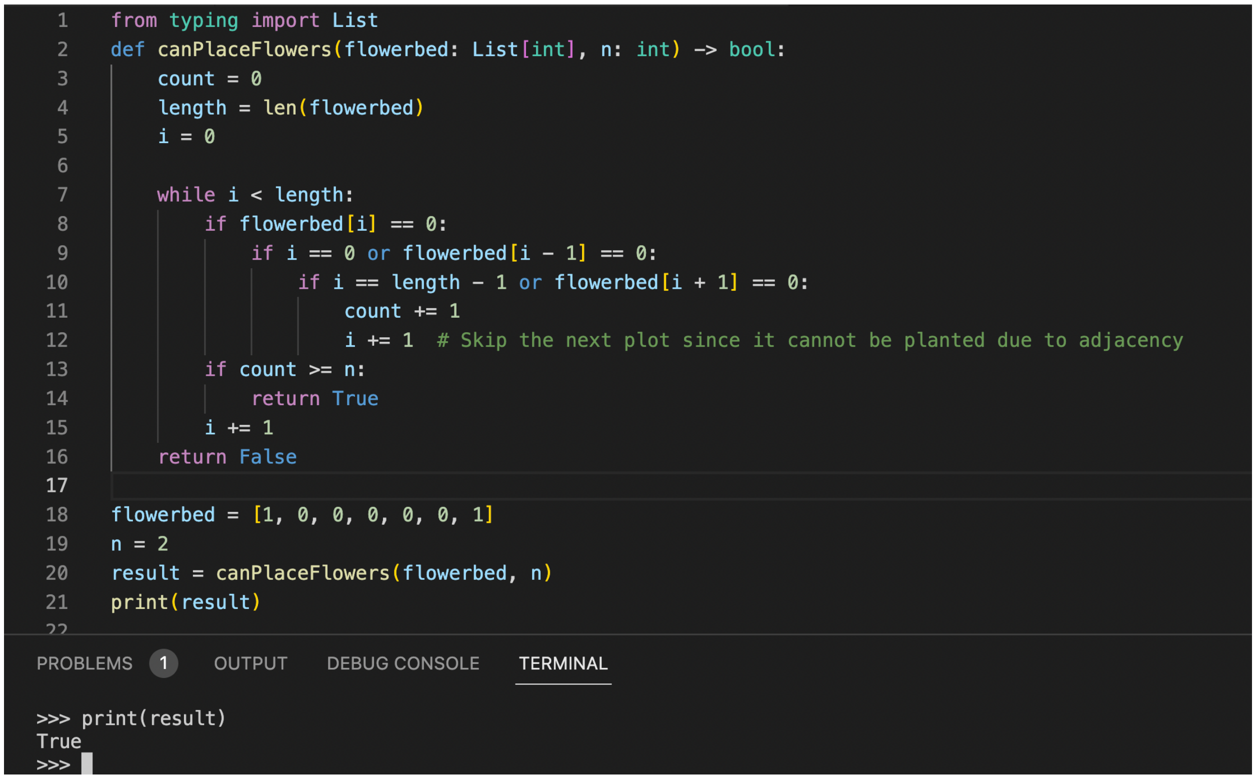
```python
def findLHS(nums):
    k = {}
    for num in nums:
        if num in k:
            k[num] += 1
        else:
            k[num] = 1

    max_length = 0
    for num in k:
        if num + 1 in k:
            length = k[num] + k[num + 1]
            max_length = max(max_length, length)

    return max_length

nums = [1, 3, 2, 2, 5, 2, 3, 7]
longest_subsequence_length = findLHS(nums)
print(longest_subsequence_length)
```

```
>>> longest_subsequence_length = findLHS(nums)
>>> print(longest_subsequence_length)
5
>>>
```

Question 4 You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be planted in adjacent plots.

Given an integer array flowerbed containing 0's and 1's, where 0 means empty and 1 means not empty, and an integer n, return true if n new flowers can be planted in the flowerbed without violating the no-adjacent-flowers rule and false otherwise.

Example 1: Input: flowerbed = [1,0,0,0,1], n = 1 Output: true

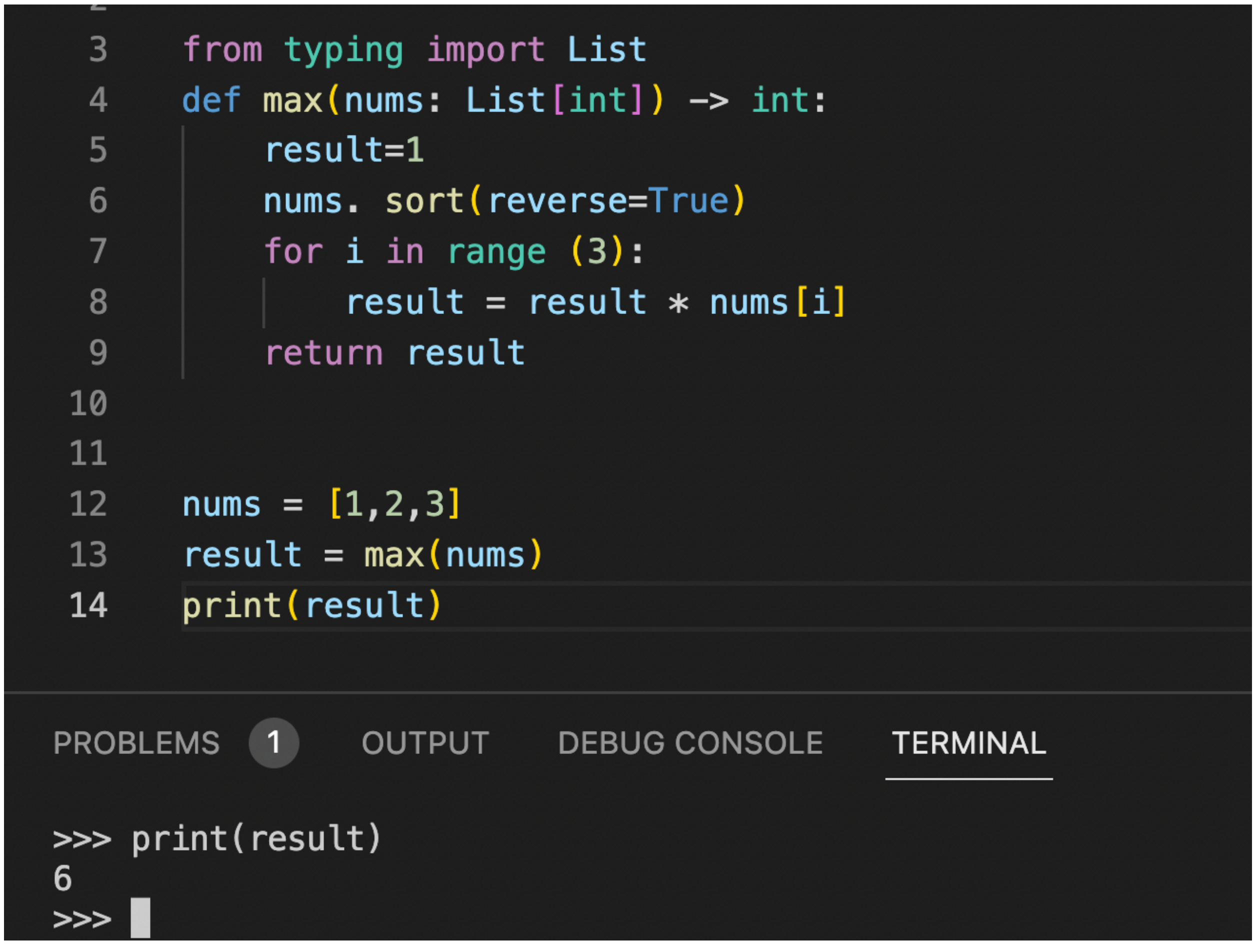


```python
from typing import List
def canPlaceFlowers(flowerbed: List[int], n: int) -> bool:
    count = 0
    length = len(flowerbed)
    i = 0

    while i < length:
        if flowerbed[i] == 0:
            if i == 0 or flowerbed[i - 1] == 0:
                if i == length - 1 or flowerbed[i + 1] == 0:
                    count += 1
                    i += 1  # Skip the next plot since it cannot be planted due to adjacency
        if count >= n:
            return True
        i += 1
    return False

flowerbed = [1, 0, 0, 0, 0, 0, 1]
n = 2
result = canPlaceFlowers(flowerbed, n)
print(result)
```

```
>>> print(result)
True
>>>
```

Question 5 Given an integer array nums, find three numbers whose product is maximum and return the maximum product.

Example 1: Input: nums = [1,2,3] Output: 6

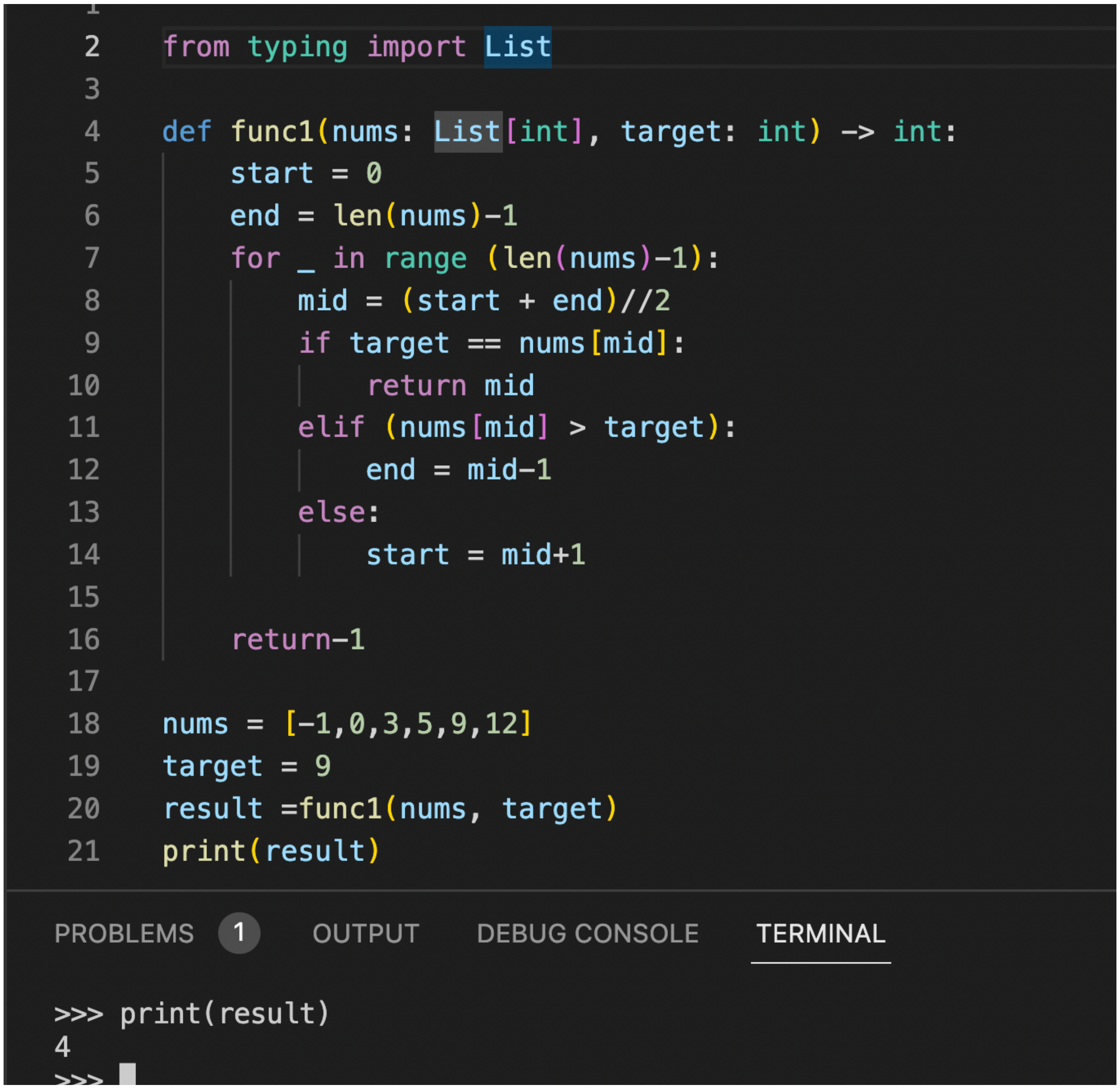


```python
from typing import List
def max(nums: List[int]) -> int:
    result=1
    nums. sort(reverse=True)
    for i in range (3):
        result = result * nums[i]
    return result


nums = [1,2,3]
result = max(nums)
print(result)
```

```
>>> print(result)
6
>>> 
```

Question 6 Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1. You must write an algorithm with O(log n) runtime complexity.

Input: nums = [-1,0,3,5,9,12], target = 9
Output: 4
Explanation: 9 exists in nums and its index is 4

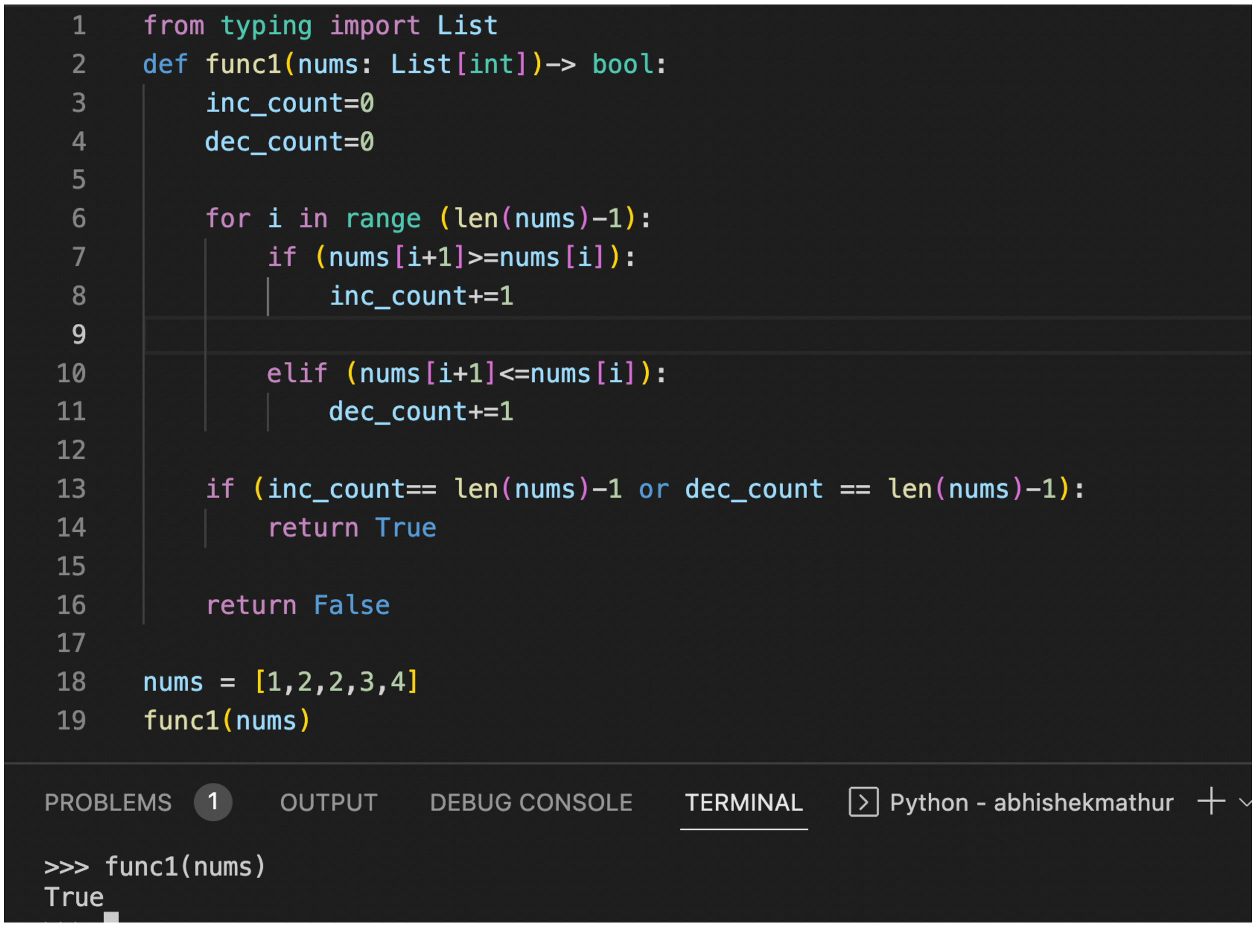


```python

from typing import List

def func1(nums: List[int], target: int) -> int:
    start = 0
    end = len(nums)-1
    for _ in range (len(nums)-1):
        mid = (start + end)//2
        if target == nums[mid]:
            return mid
        elif (nums[mid] > target):
            end = mid-1
        else:
            start = mid+1

    return-1

nums = [-1,0,3,5,9,12]
target = 9
result =func1(nums, target)
print(result)
```

```
>>> print(result)
4
>>> 
```

Question 7 An array is monotonic if it is either monotone increasing or monotone decreasing.
An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j].
An array nums is monotone decreasing if for all i <= j, nums[i] >= nums[j].
Given an integer array nums, return true if the given array is monotonic, or false otherwise.

Example 1: Input: nums = [1,2,2,3] Output: true



```python
from typing import List
def func1(nums: List[int])-> bool:
    inc_count=0
    dec_count=0

    for i in range (len(nums)-1):
        if (nums[i+1]>=nums[i]):
            inc_count+=1

        elif (nums[i+1]<=nums[i]):
            dec_count+=1

    if (inc_count== len(nums)-1 or dec_count == len(nums)-1):
        return True

    return False

nums = [1,2,2,3,4]
func1(nums)
```

```
>>> func1(nums)
True
```

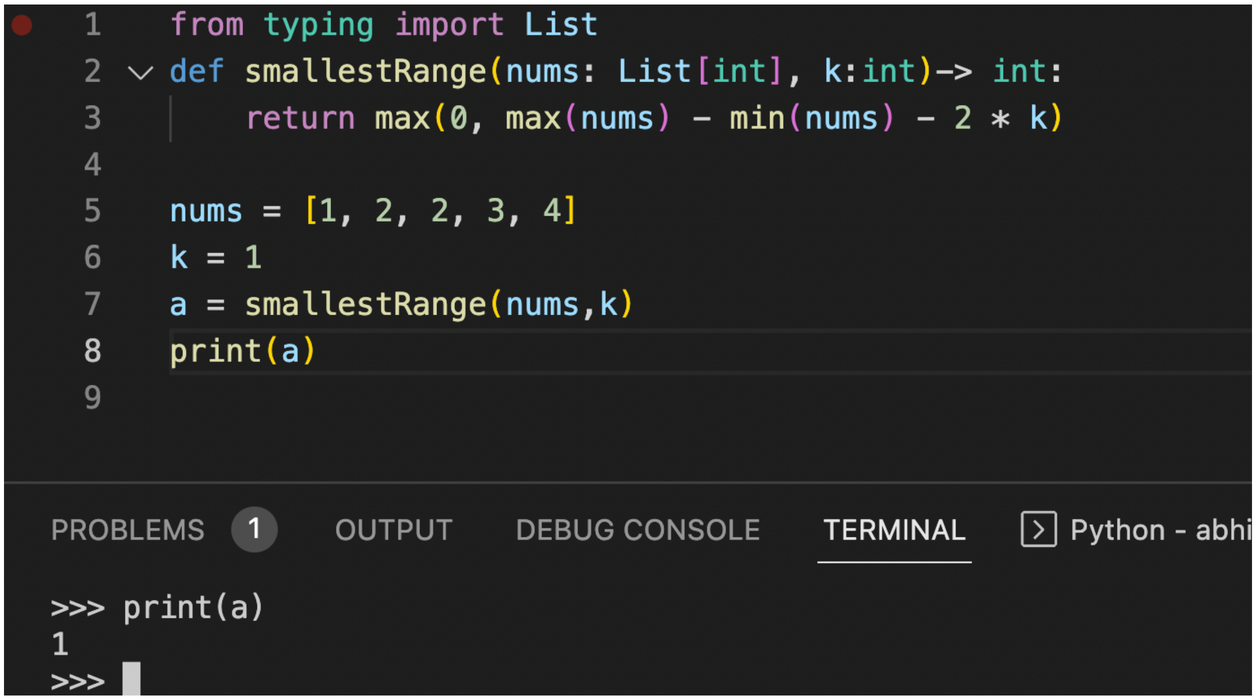
Question 8 You are given an integer array nums and an integer k.

In one operation, you can choose any index i where 0 <= i < nums.length and change nums[i] to nums[i] + x where x is an integer from the range [-k, k].

You can apply this operation at most once for each index i. The score of nums is the difference between the maximum and minimum elements in nums.

Return the minimum score of nums after applying the mentioned operation at most once for each index in it.

Example 1: Input: nums = [1], k = 0 Output: 0 **Explanation:** The score is max(nums) - min(nums) = 1 - 1 = 0.



```python
from typing import List
def smallestRange(nums: List[int], k:int)-> int:
    return max(0, max(nums) - min(nums) - 2 * k)

nums = [1, 2, 2, 3, 4]
k = 1
a = smallestRange(nums,k)
print(a)
```

```
>>> print(a)
1
>>>
```