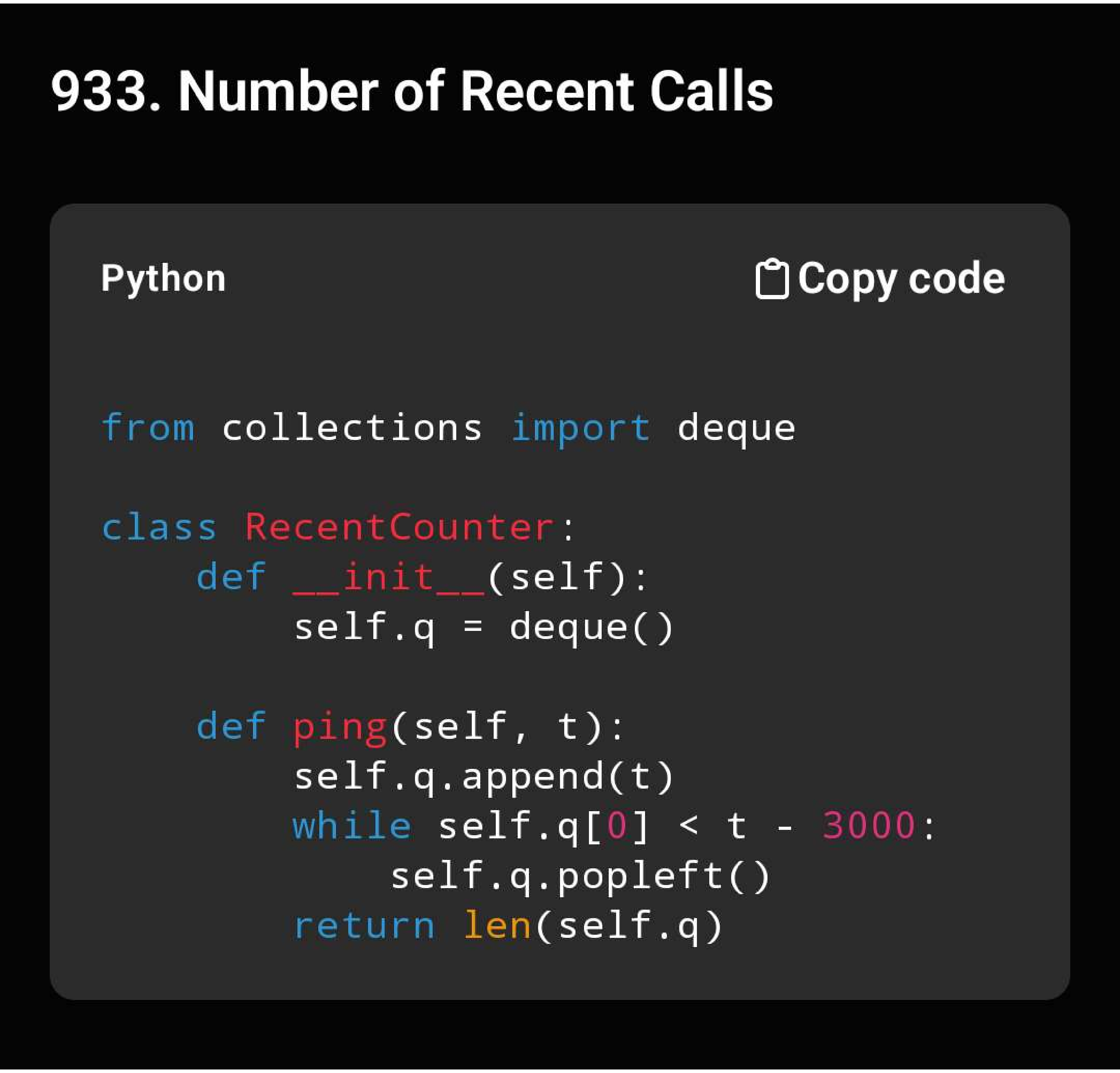


933. Number of Recent Calls

```python
from collections import deque

class RecentCounter:
    def __init__(self):
        self.q = deque()

    def ping(self, t):
        self.q.append(t)
        while self.q[0] < t - 3000:
            self.q.popleft()
        return len(self.q)
```

225. Implement Stack Using Queues

Python

Copy code

```python
from collections import deque

class MyStack:
    def __init__(self):
        self.q = deque()

    def push(self, x):
        self.q.append(x)
        for _ in range(len(self.q) -
1):

self.q.append(self.q.popleft())

    def pop(self):
        return self.q.popleft()

    def top(self):
        return self.q[0]

    def empty(self):
        return not self.q
```

Queue

232. Implement Queue Using Stacks

Python Copy code

```python
class MyQueue:
    def __init__(self):
        self.s1, self.s2 = [], []

    def push(self, x):
        self.s1.append(x)

    def pop(self):
        self.peek()
        return self.s2.pop()

    def peek(self):
        if not self.s2:
            while self.s1:

self.s2.append(self.s1.pop())
        return self.s2[-1]

    def empty(self):
        return not self.s1 and not
self.s2
```

155. Min Stack

Python Copy code

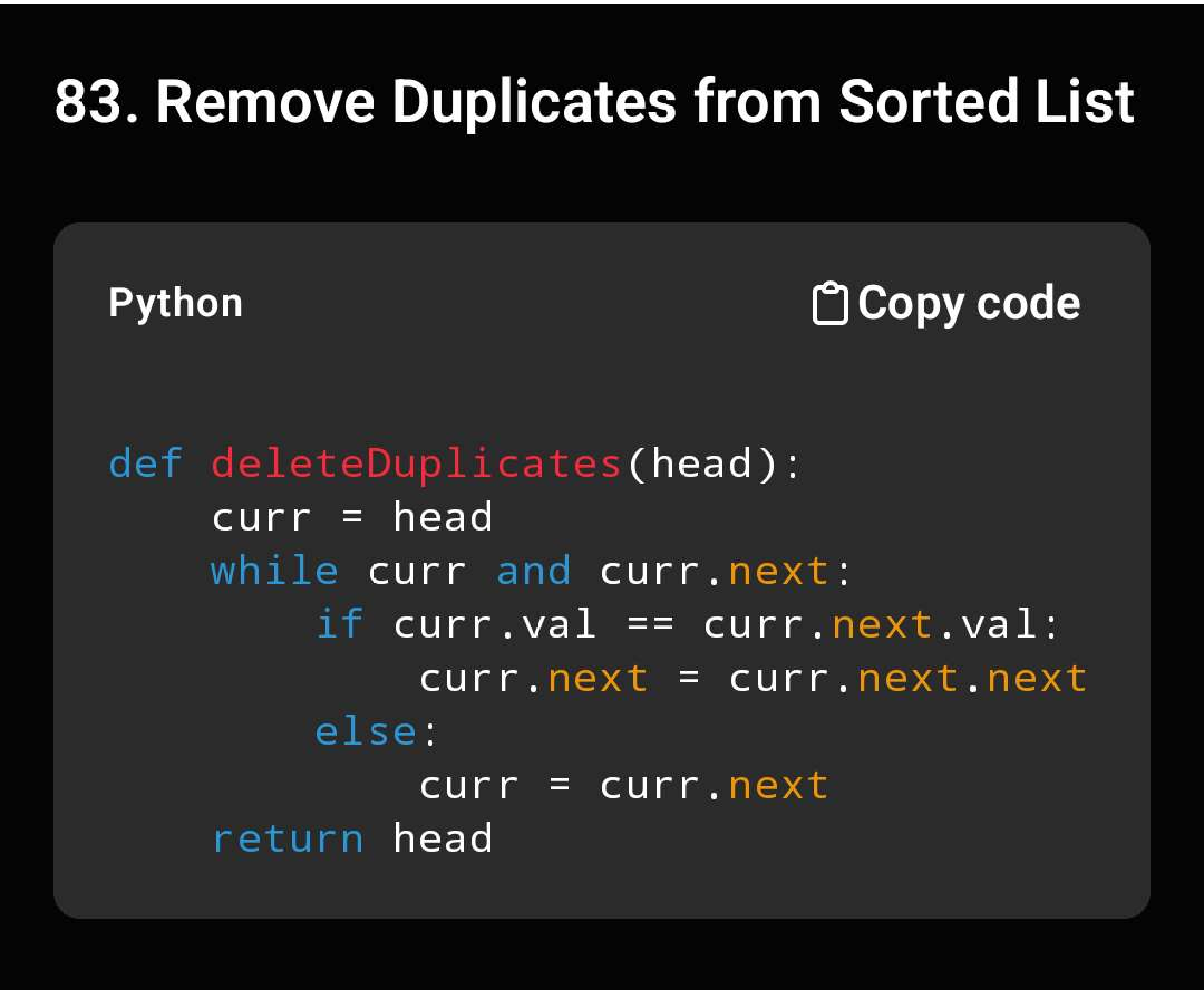
```python
class MinStack:
    def __init__(self):
        self.stack = []
        self.min_stack = []

    def push(self, val):
        self.stack.append(val)
        if not self.min_stack or val
<= self.min_stack[-1]:
            self.min_stack.append(val)

    def pop(self):
        if self.stack.pop() ==
self.min_stack[-1]:
            self.min_stack.pop()

    def top(self):
        return self.stack[-1]

    def getMin(self):
        return self.min_stack[-1]
```

Stack

150. Evaluate Reverse Polish Notation

Python Copy code

```python
def evalRPN(tokens):
    stack = []
    for t in tokens:
        if t in '+-*/':
            b, a = stack.pop(),
stack.pop()
            if t == '+':
stack.append(a + b)
            elif t == '-':
stack.append(a - b)
            elif t == '*':
stack.append(a * b)
            else: stack.append(int(a /
b))
        else:
            stack.append(int(t))
    return stack[0]
```



83. Remove Duplicates from Sorted List

```python
def deleteDuplicates(head):
    curr = head
    while curr and curr.next:
        if curr.val == curr.next.val:
            curr.next = curr.next.next
        else:
            curr = curr.next
    return head
```

19. Remove Nth Node From End

Python

Copy code

```python
def removeNthFromEnd(head, n):
    dummy = ListNode(0, head)
    fast = slow = dummy
    for _ in range(n):
        fast = fast.next
    while fast.next:
        fast = fast.next
        slow = slow.next
    slow.next = slow.next.next
    return dummy.next
```

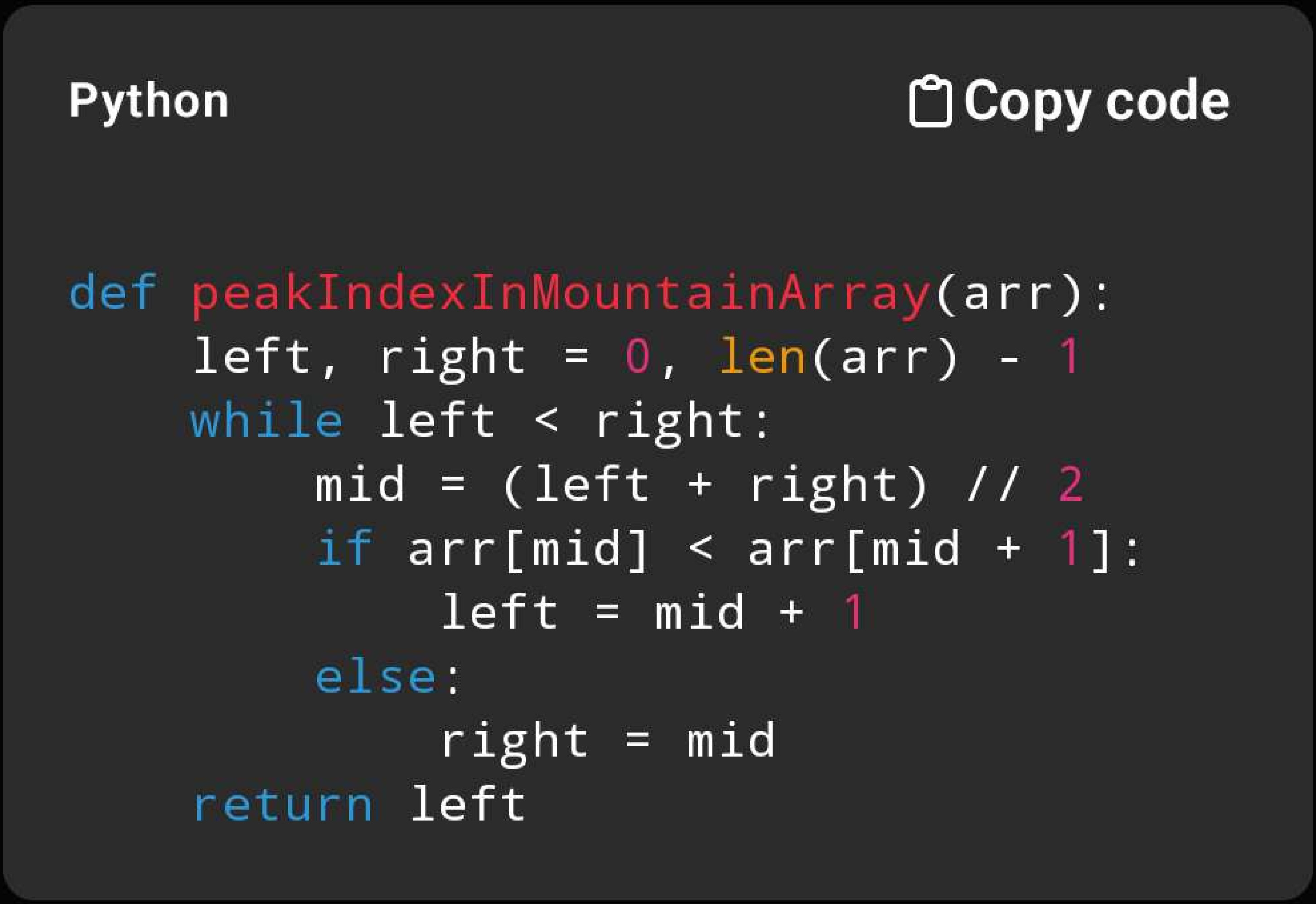
Linked List

203. Remove Linked List Elements

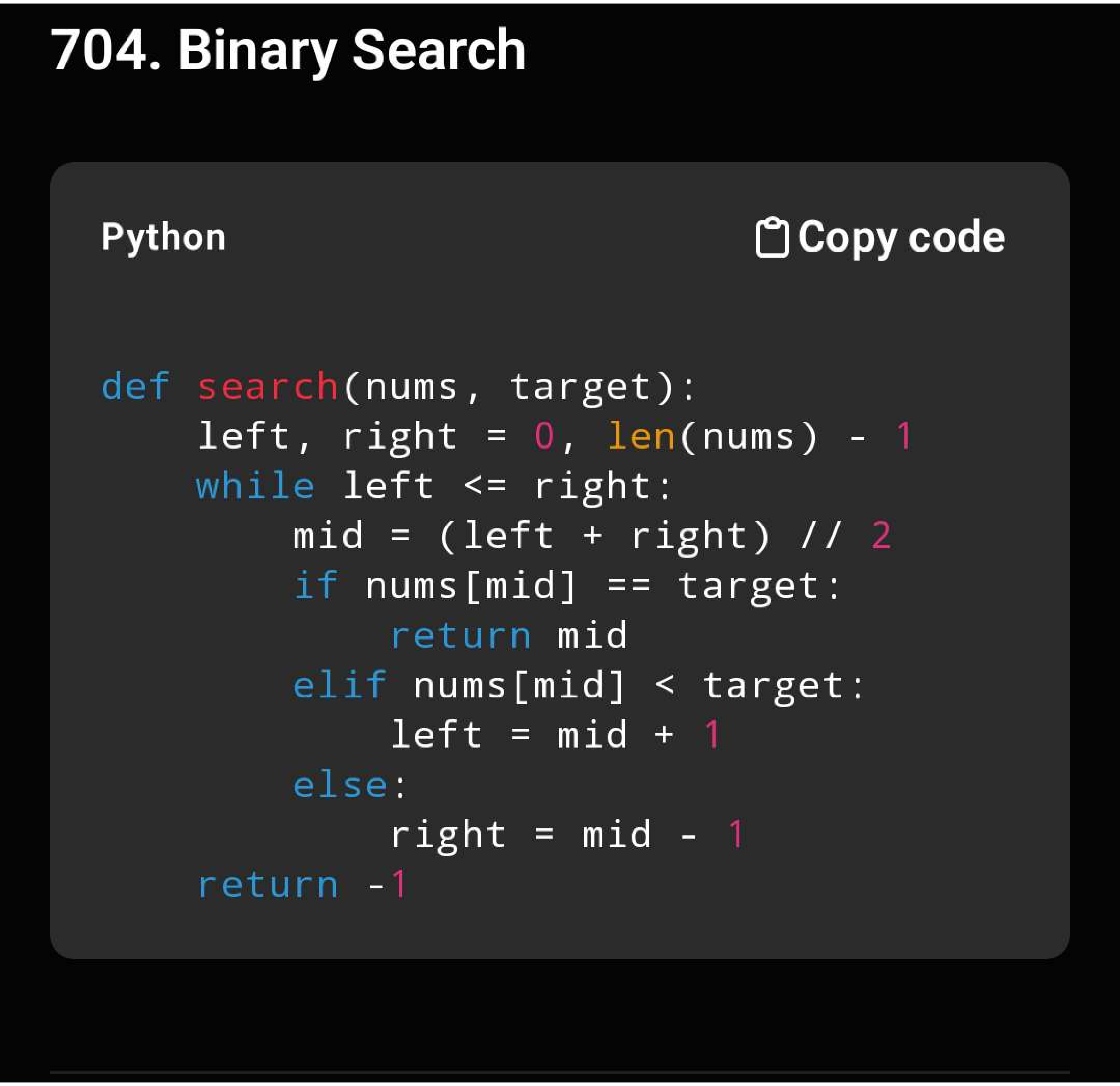
Python Copy code

```python
def removeElements(head, val):
    dummy = ListNode(0)
    dummy.next = head
    curr = dummy
    while curr.next:
        if curr.next.val == val:
            curr.next = curr.next.next
        else:
            curr = curr.next
    return dummy.next
```

852. Peak Index in a Mountain Array

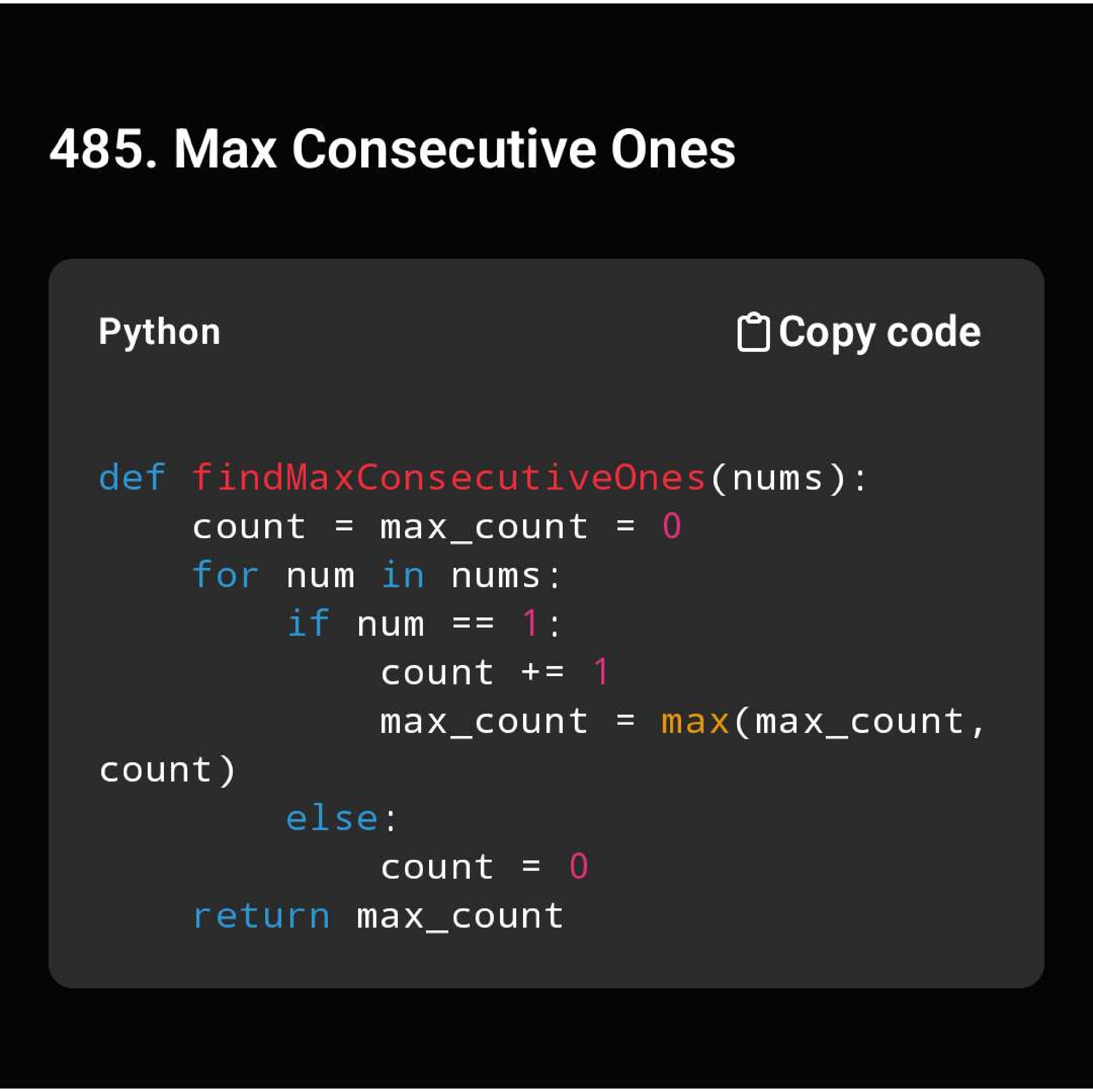


```python
def peakIndexInMountainArray(arr):
    left, right = 0, len(arr) - 1
    while left < right:
        mid = (left + right) // 2
        if arr[mid] < arr[mid + 1]:
            left = mid + 1
        else:
            right = mid
    return left
```



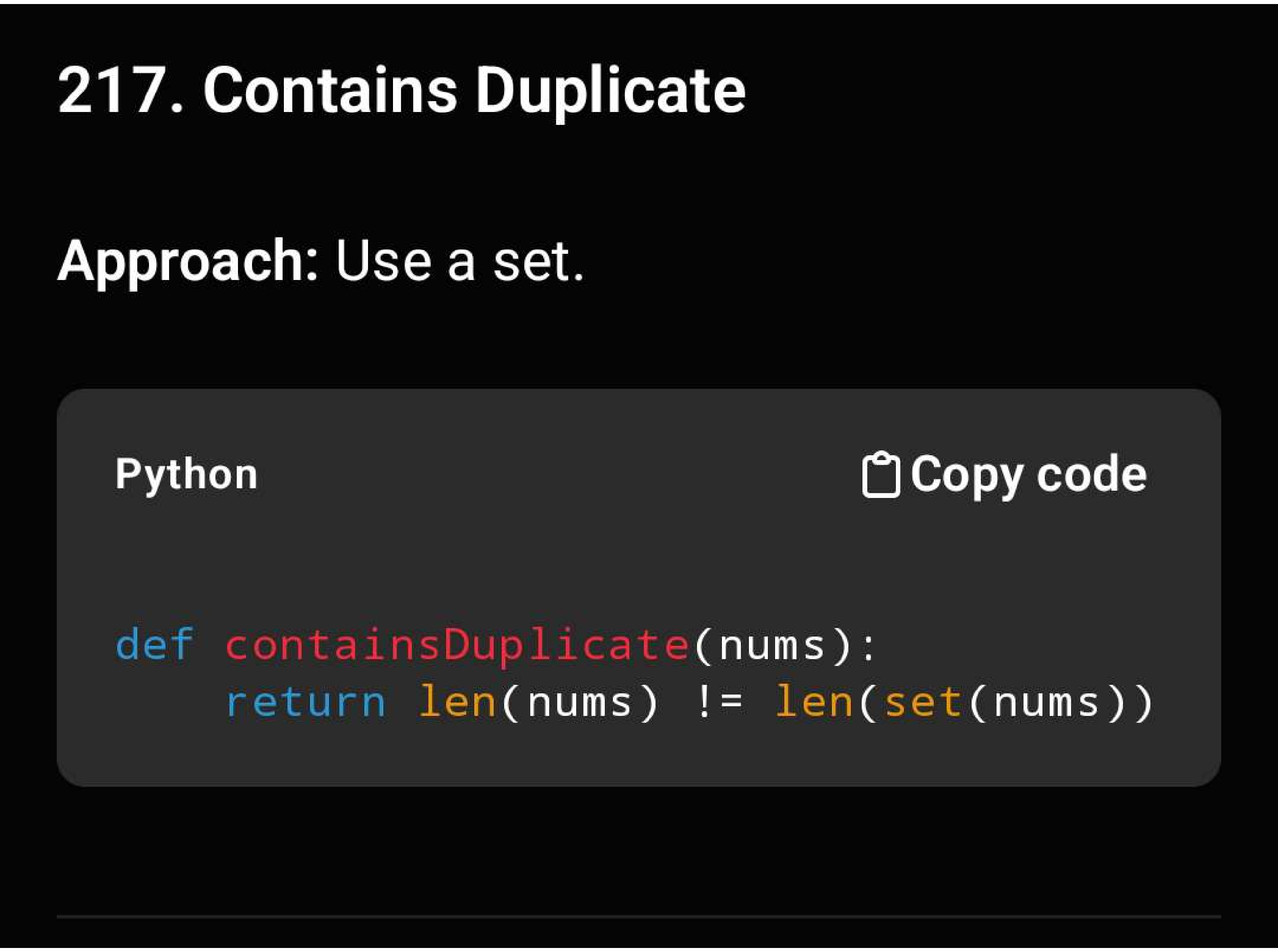
704. Binary Search

```python
def search(nums, target):
    left, right = 0, len(nums) - 1
    while left <= right:
        mid = (left + right) // 2
        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```



485. Max Consecutive Ones

```python
def findMaxConsecutiveOnes(nums):
    count = max_count = 0
    for num in nums:
        if num == 1:
            count += 1
            max_count = max(max_count,
count)
        else:
            count = 0
    return max_count
```



217. Contains Duplicate

Approach: Use a set.

Python

Copy code

```python
def containsDuplicate(nums):
    return len(nums) != len(set(nums))
```

35. Search Insert Position

Approach: Binary search.

Python Copy code

```python
def searchInsert(nums, target):
    left, right = 0, len(nums) - 1
    while left <= right:
        mid = (left + right) // 2
        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return left
```

Time Complexity: O(log n)

27. Remove Element

Approach: Use two pointers to overwrite unwanted values.

Python   Copy code

```python
def removeElement(nums, val):
    k = 0
    for i in range(len(nums)):
        if nums[i] != val:
            nums[k] = nums[i]
            k += 1
    return k
```

Time Complexity: O(n)

Arrays

1. Two Sum

Approach: Use a hash map to store visited numbers and their indices.

Python Copy code

```python
def twoSum(nums, target):
    seen = {}
    for i, num in enumerate(nums):
        diff = target - num
        if diff in seen:
            return [seen[diff], i]
        seen[num] = i
```

Time Complexity: O(n)