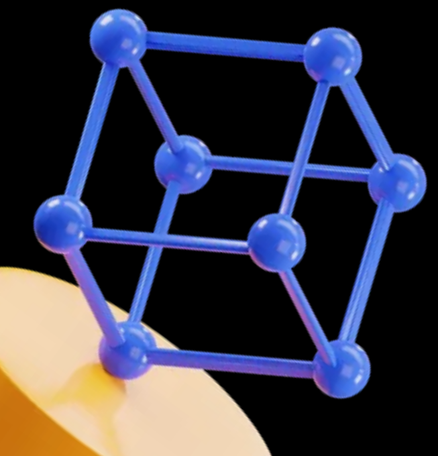
Mark, Andrew, Brayden

HKD EASY SECTION #3

Sliding Windows

October 21st, 2026



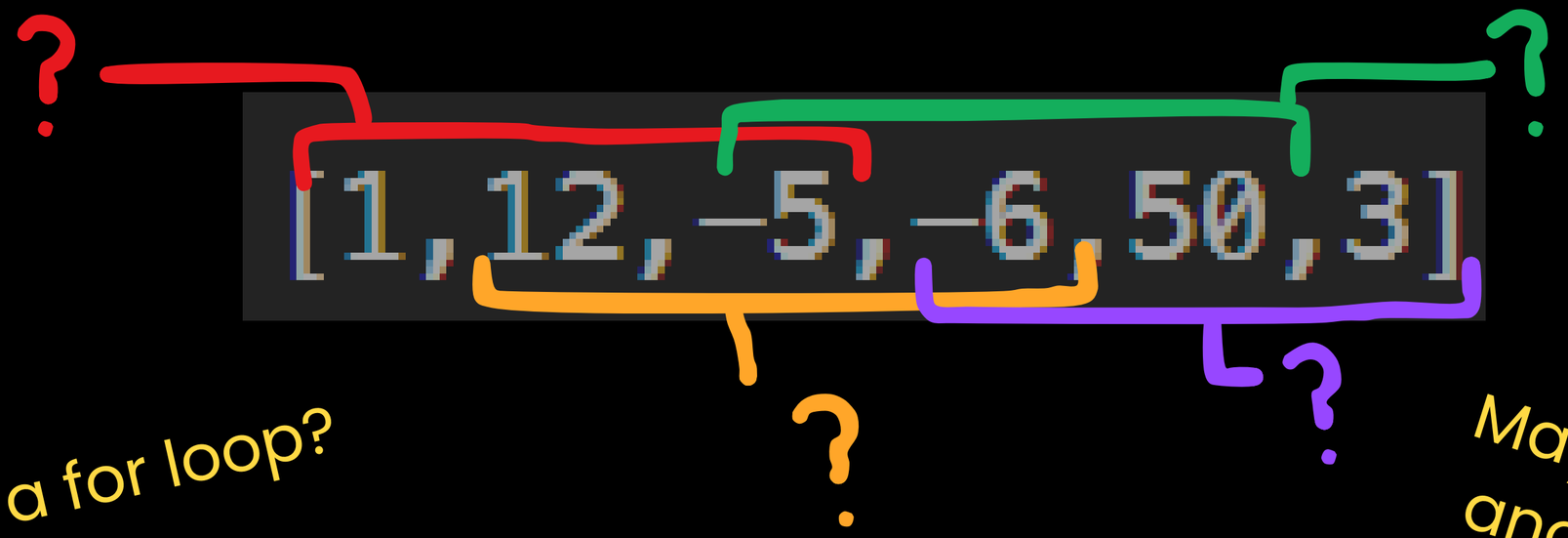
Attendance and Announcements



- The Hacker Devil will continue providing us with pizza as long as you scan its QR code.
- Don’t make it angry
- This is all it has

Before The Concept...

How might you get the maximum sum of 3 connected elements in this array?





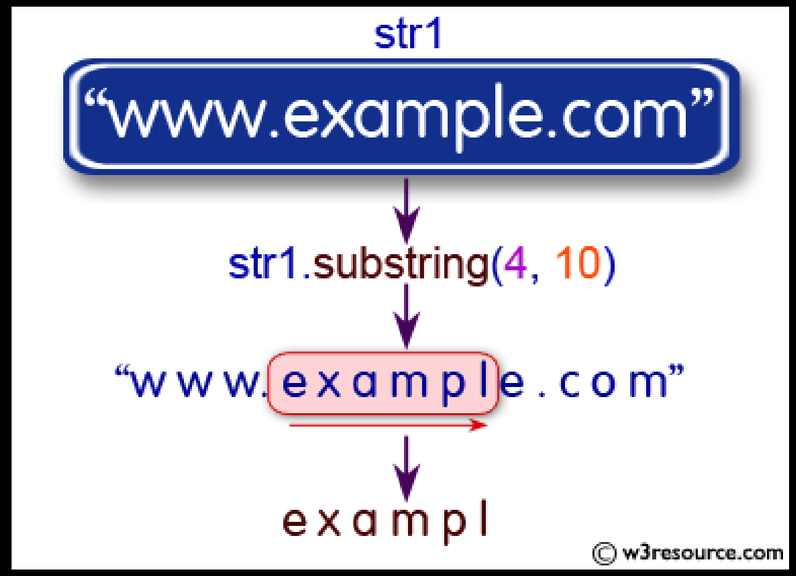
Today's Concept

Sliding Windows!

A sliding window is a pattern that is used to solve problems that involve contiguous (connected) subarrays or substrings

If you are in CSE 110, you already learned about substrings!

Basically the same thing as a subarray



Today's Concept

Sliding Windows!

The purpose of a sliding window is to optimize nested loops $O(n^2)$ into a single $O(n)$ loop.

From this

```python
class Solution:
    def findMaxAverage(self, nums: list[int], k: int) -> float:
        max_sum = 0.0
        for i in range(len(nums) - k + 1):
            current_sum = 0
            for j in range(i, i + k):
                current_sum += nums[j]
            max_sum = max(max_sum, current_sum)
        return max_sum / k
```

To this!

```python
class Solution:
    def findMaxAverage(self, nums: list[int], k: int) -> float:
        window_sum = sum(nums[:k])
        max_sum = window_sum
        for i in range(k, len(nums)):
            window_sum += nums[i]
            window_sum -= nums[i - k]
            max_sum = max(max_sum, window_sum)
        return max_sum / k
```

Today's Concept

How might you get the maximum sum of 3 connected elements in this array?

```
[1,12,-5,-6,50,3]
```


```
[1,12,-5,-6,50,3]
```

```
[1,12,-5,-6,50,3]
```

Today's Concept

How might you get the maximum sum of 3 connected elements in this array?

```
[1,12,-5,-6,50,3]
```

1+12-5=8

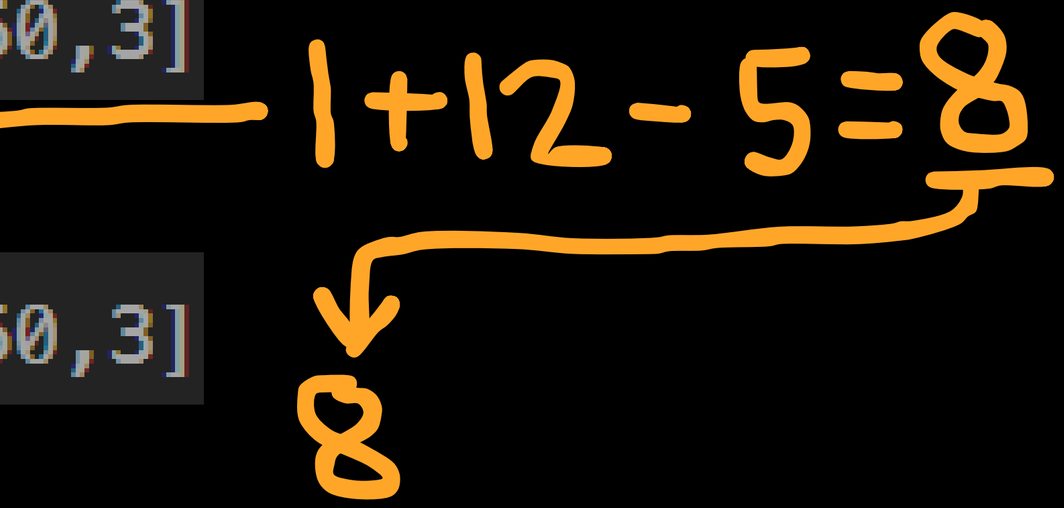

```
[1,12,-5,-6,50,3]
```

```
[1,12,-5,-6,50,3]
```

Today's Concept

How might you get the maximum sum of 3 connected elements in this array?

```
[1,12,-5,-6,50,3]
```

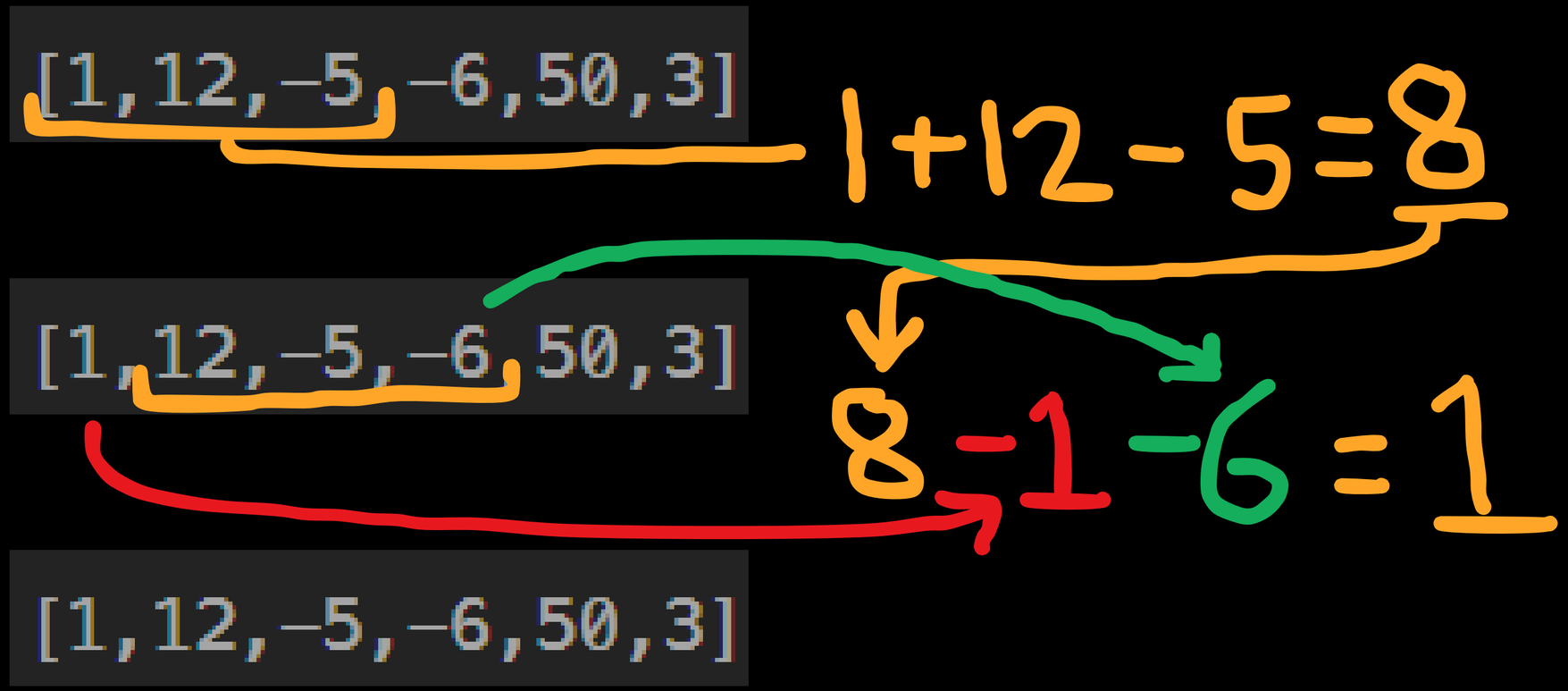



```
[1,12,-5,-6,50,3]
```

```
[1,12,-5,-6,50,3]
```

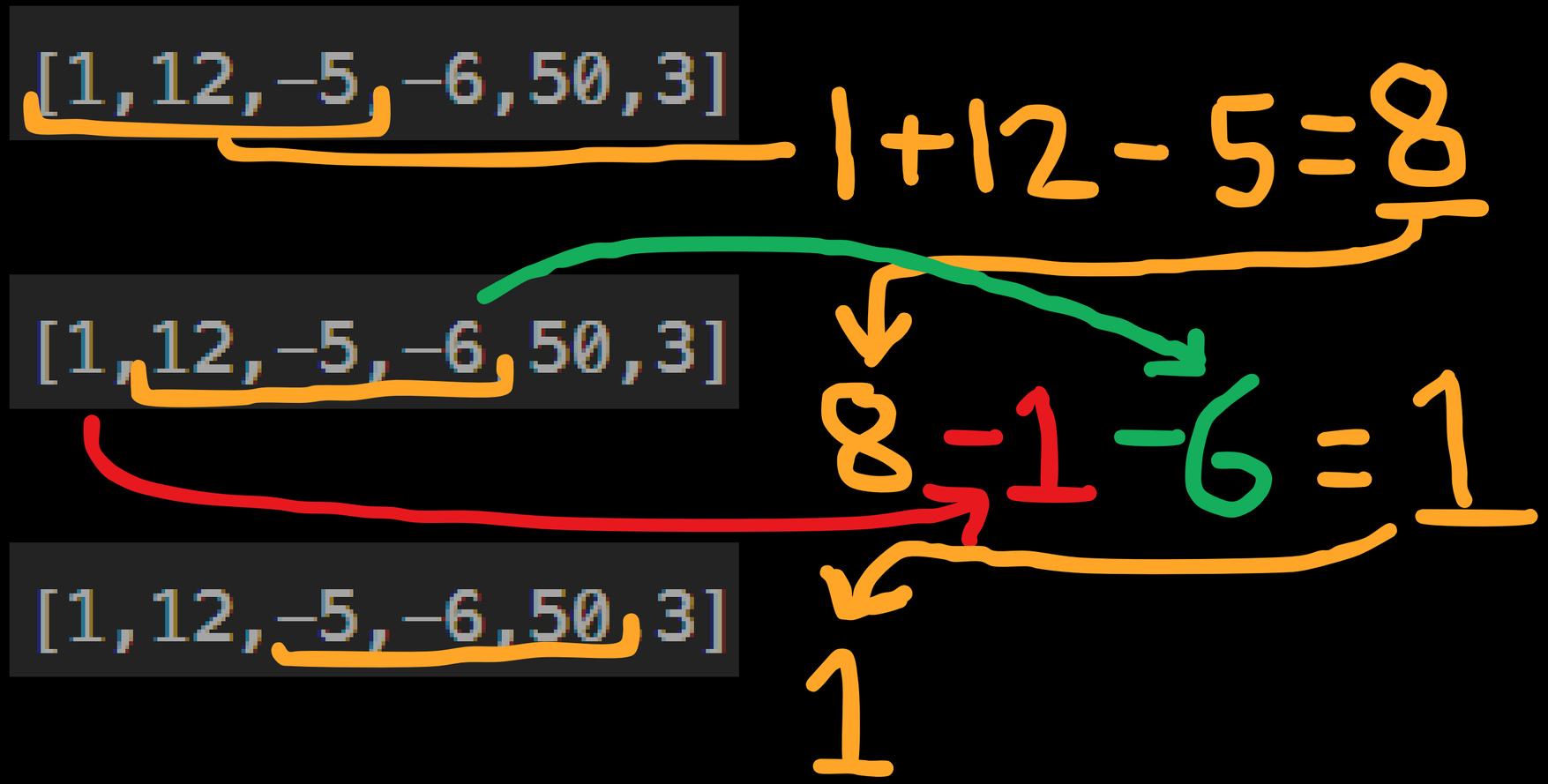
Today's Concept

How might you get the maximum sum of 3 connected elements in this array?



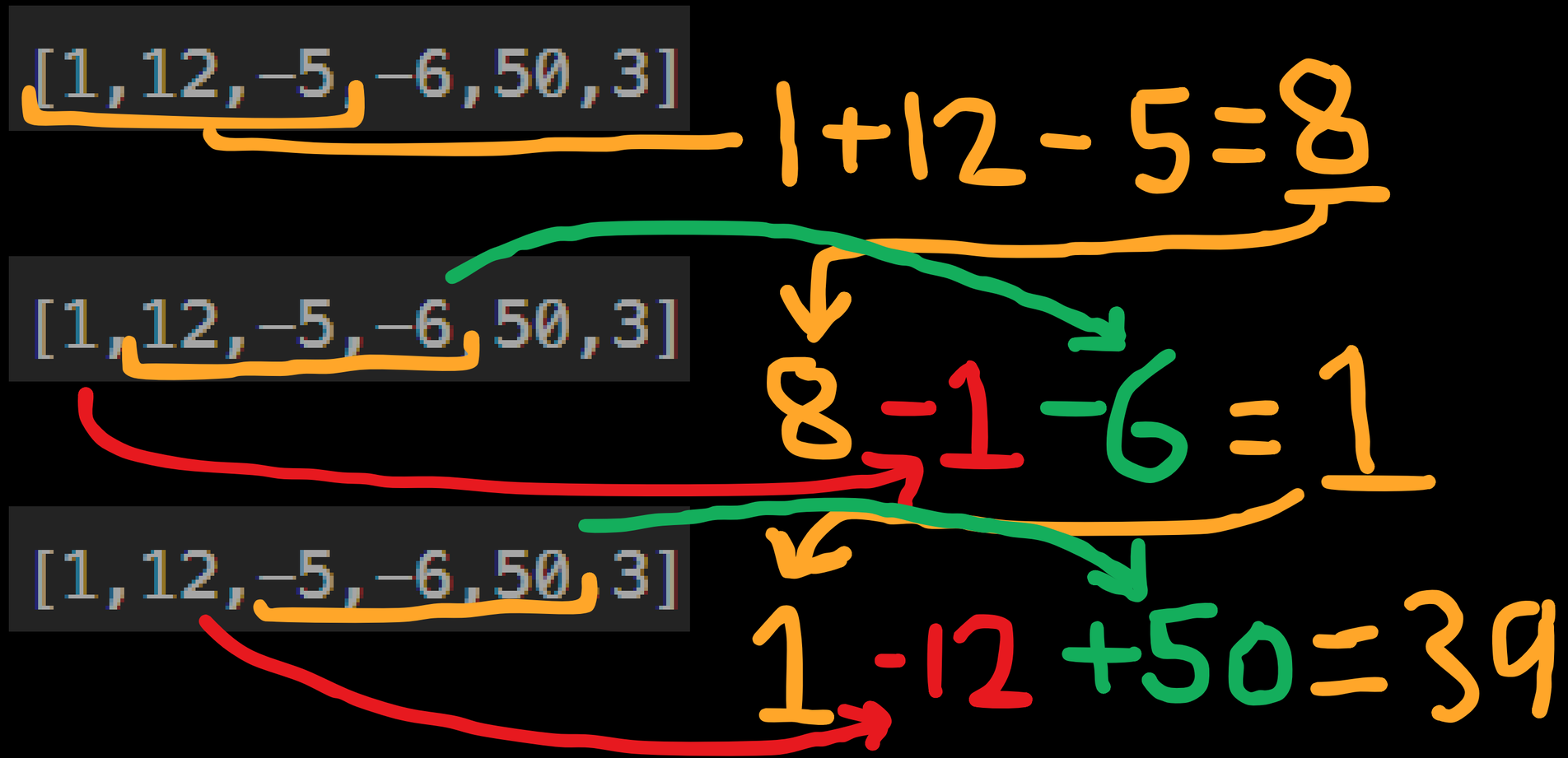
Today's Concept

How might you get the maximum sum of 3 connected elements in this array?



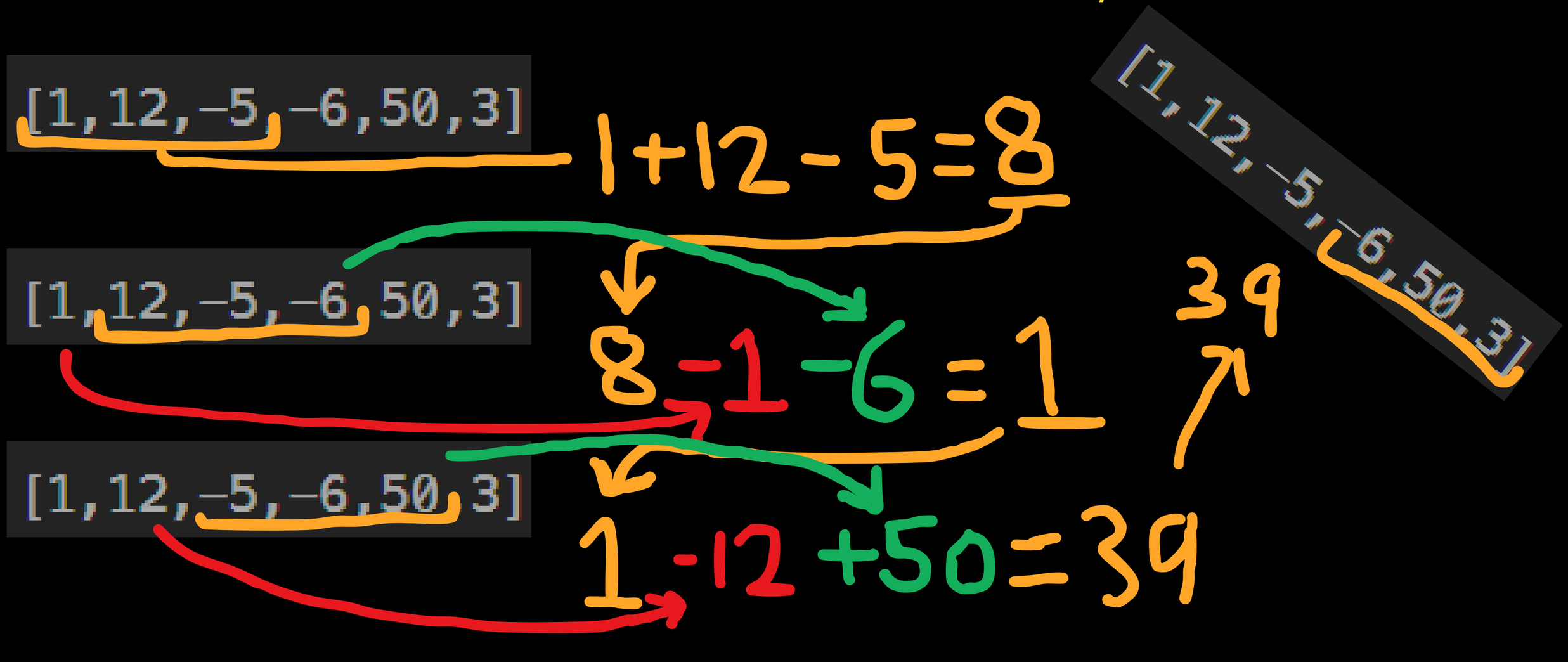
Today's Concept

How might you get the maximum sum of 3 connected elements in this array?



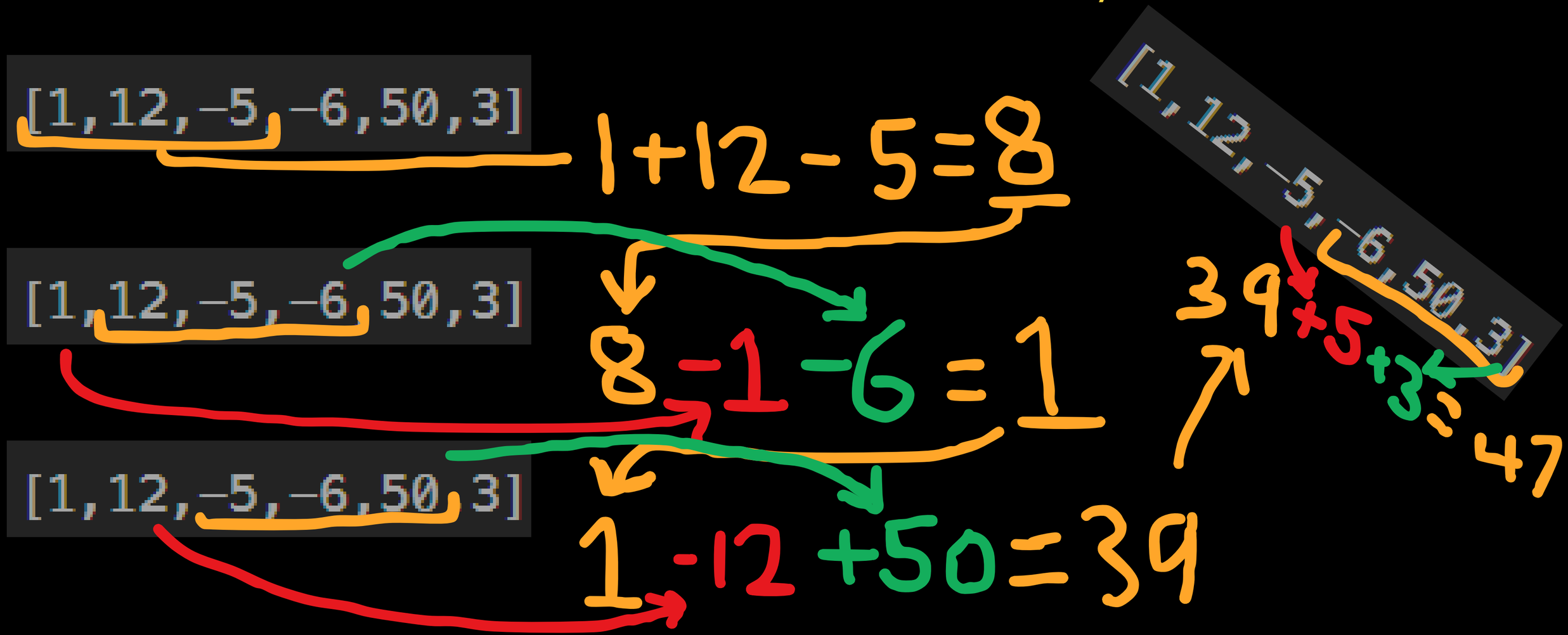
Today's Concept

How might you get the maximum sum of 3 connected elements in this array?



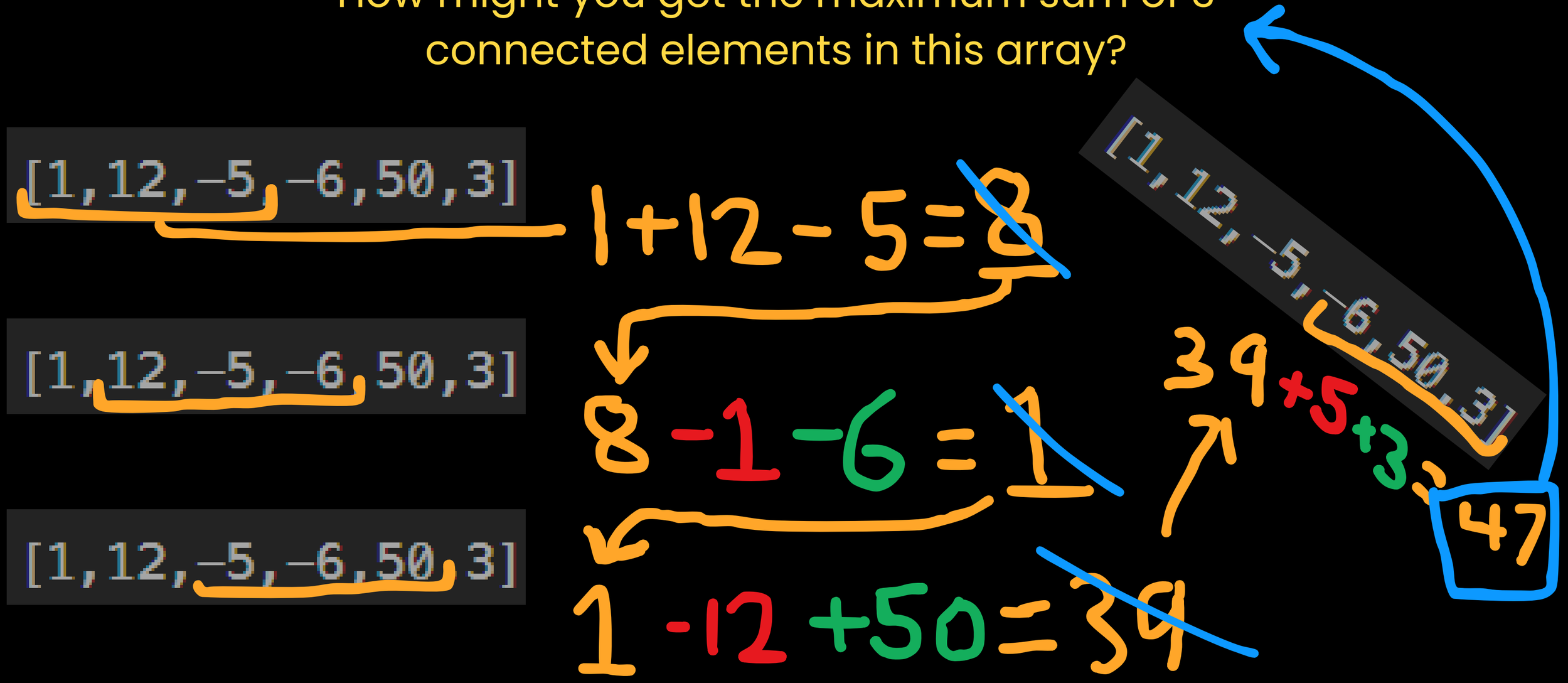
Today's Concept

How might you get the maximum sum of 3 connected elements in this array?



Today's Concept

How might you get the maximum sum of 3 connected elements in this array?



Problem 1

643. Maximum Average Subarray I

Easy | Topics | Companies

You are given an integer array `nums` consisting of `n` elements, and an integer `k`.

Find a contiguous subarray whose **length is equal to** `k` that has the maximum average value and return *this value*. Any answer with a calculation error less than `10^-5` will be accepted.

Example 1:

> **Input:** nums = [1,12,-5,-6,50,3], k = 4
> **Output:** 12.75000
> **Explanation:** Maximum average is (12 - 5 - 6 + 50) / 4 = 51 / 4 = 12.75

Example 2:

> **Input:** nums = [5], k = 1
> **Output:** 5.00000

Problem Simplification

We are given an array:

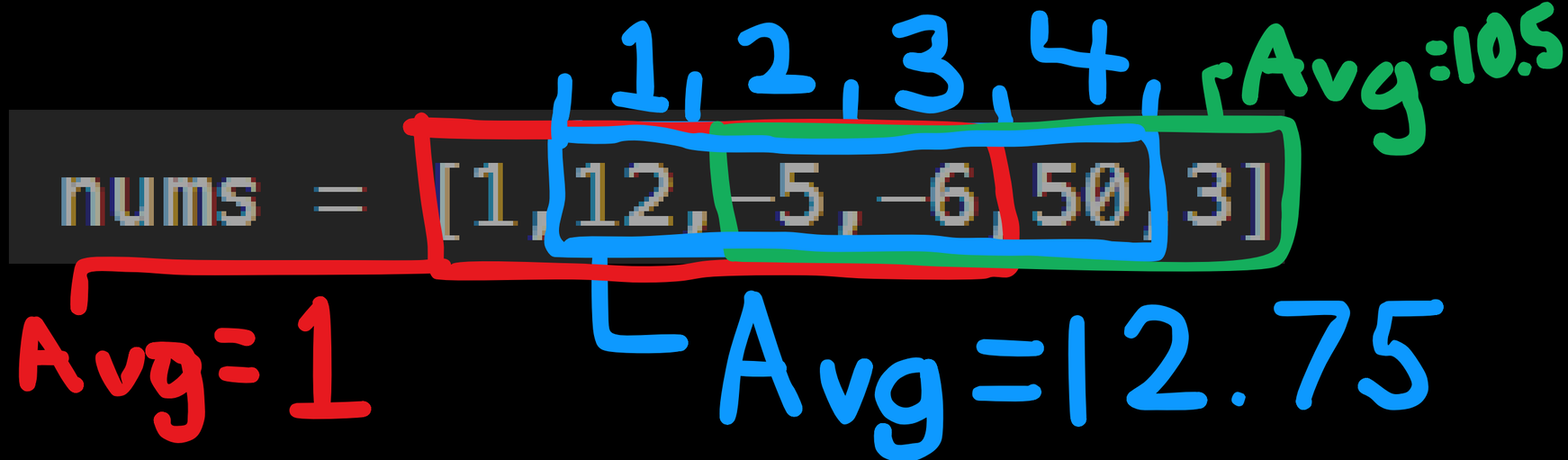
```
nums = [1,12,-5,-6,50,3]
```

And the size of the “window”

```
k = 4
```

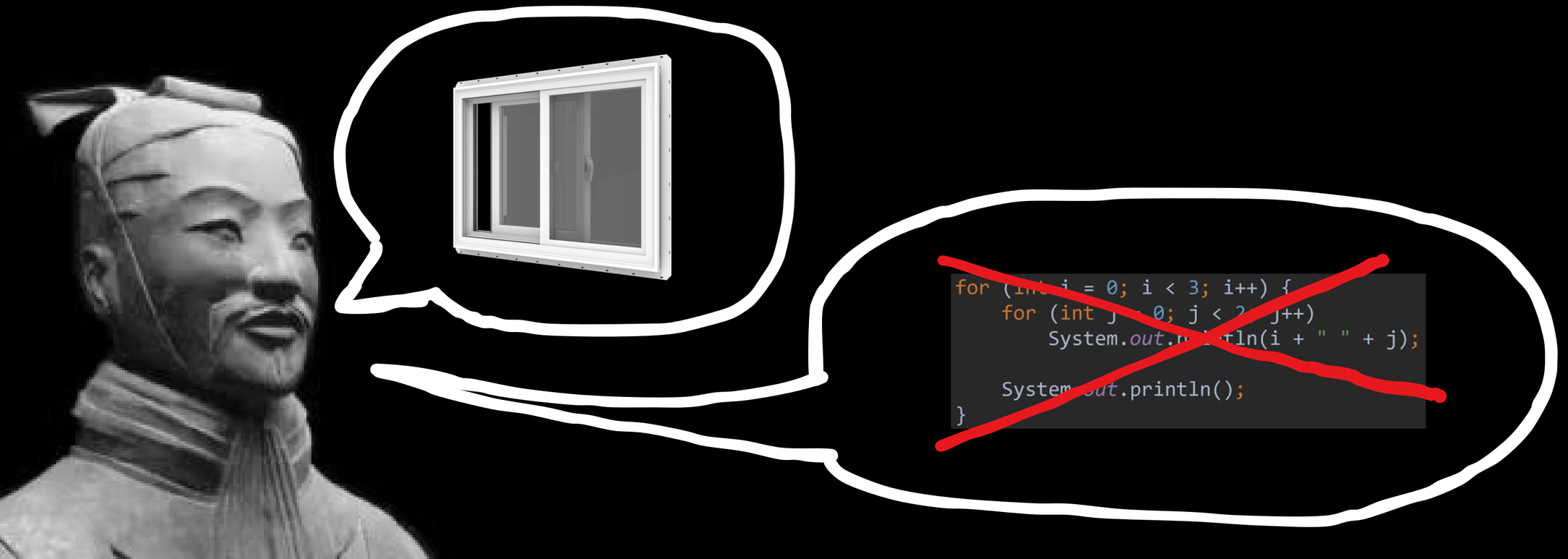
We must write an algorithm that returns the maximum average value that fits within the “window”

Output: 12.75000
Explanation: Maximum average is (12 - 5 - 6 + 50) / 4 = 51 / 4 = 12.75



Try it youself!

And remember your training...



Problem 1 solution

C

```c
double findMaxAverage(int* nums, int numsSize, int k) {
    int sum = 0;

    for(int i = 0; i < k; i++){
        sum += nums[i];
    }
    int max = sum;

    for(int i = k; i < numsSize; i++){
        sum -= nums[i - k];
        sum += nums[i];

        if(sum > max) max = sum;
    }
    return (double)max / k;
}
```

Python

```python
class Solution:
    def findMaxAverage(self, nums: List[int], k: int) -> float:
        window_sum = sum(nums[:k])
        max_sum = window_sum

        for i in range(k, len(nums)):
            window_sum += nums[i] - nums[i - k]
            max_sum = max(max_sum, window_sum)

        return max_sum / k
```

Problem 1 solution

Java

```java
class Solution {
    public double findMaxAverage(int[] nums, int k) {
        int sum = 0;
        for(int i = 0; i < k; i++)
        {
            sum += nums[i];
        }
        int maxSum = sum;
        for(int i = k; i < nums.length; i++)
        {
            sum = sum - nums[i - k] + nums[i];
            if(sum > maxSum) maxSum = sum;
        }
        return (double) maxSum / k;
    }
}
```

```cpp
class Solution {
public:
    double findMaxAverage(vector<int>& nums,
        double sum = 0;
        double maxSum = 0;

        for (int i = 0; i < nums.size(); i++)
            sum += nums[i];

            if (i == k - 1) {
               maxSum = sum;
            } else if (i >= k) {
                sum -= nums[i - k];
                maxSum = max(maxSum, sum);
            }
        }
        return maxSum/k;
    }
};
```

Hacker Devils

Presented by Mark, Andrew, Brayden

THANK YOU

for your time and attention

