Problem Link:

https://leetcode.com/problems/maximum-subarray/description/

Problem Description:

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

Problem Approach:

There can be many approaches to this problem, but the best and the most optimal one is Kadane's algorithm.

Sample Test Case:

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6

Solution:

We start by iterating through the array with two variables: **maximumSum** and **currentSum**. As the name suggests, the maximumSum will store the maximum sum we found in the elements of the array so far. And the currentSum will keep track of the currently chosen subarray's sum.

We initialize the current sum with 0 (Logically, since we haven't started iterating through the array), and maximumSum with the first element of the array. Reason behind that is because if all the elements of the array are negative, it will consider the first negative value, and will help find the smallest number, which indeed would be the answer to our problem. Otherwise, if we initialize maximumSum with 0, it would always give the answer 0 in case of all negative integer arrays.

Algorithm is quite simple: Iterate through the array, keep track of the currentSum. If the currentSum surpasses the maximumSum, update the maximumSum as currentSum, and if the currentSum becomes less than zero, then there's no point even considering that subarray, since it'll always only drag down any future sums we might calculate.. So we reset the currentSum to zero. This way, we allow the possibility of finding a new subarray that could have a higher positive sum. And when we're done iterating through the array, we can simply return the maximumSum, and that is our answer!

Code (Python):

```
def maxSubArray(self, nums: List[int]) -> int:
        maximumSum = nums[0]
        currentSum = 0
        for i in range(0,len(nums)):
            currentSum += nums[i]
            if currentSum > maximumSum:
                maximumSum = currentSum
            if currentSum < 0:
                 currentSum = 0
        return maximumSum
```