**Code Kata 1**

**Problem statement:**

Find the contiguous subarray within an array, A of length N which has the largest sum.

**Input:**

The first and the only argument contains an integer array, A.

**Output:**

Return an integer representing the maximum possible sum of the contiguous subarray.

**Examples:**

Input: A = [1, 2, 3, 4, -10]

Output: 10

Explanation: Subarray [1, 2, 3, 4] has the maximum sum of 10

Input: A = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

Output: 6

Explanation: Subarray [4, -1, 2, 1] has the maximum sum of 6

**Inspiration**

This problem is not just a random puzzle. This and subsequent code katas will have an inspiration section which describes where this problem is encountered and further readings on it.

The Maximum Subarray Problem originated from the problem of Maximum Likelihood Estimation. For example, in a given 2D array of numbers find the 2D subarray with the maximum sum. This is used in computer vision to find the brightest area in an image. This is also used in DNA sequencing to find regions with a particular contiguous sequence of proteins.

The 2D problem's simplified version is the 1D Maximum Subarray Problem.

This problem has the brute-force solution in O(n<sup>3</sup>). It can be optimized to O(n<sup>2</sup>) and the optimal solution is in O(n). Try to arrive at the optimal solution.

**Hints:**

1. For any subarray starting at index 0 and ending at index i the maximum sum possible is either the element at index i or the sum of subarray till (i-1) + element at i.
2. Think in terms of DP for the optimal solution.

**Extras:**

The code kata doesn't end with that problem. Once you have solved the above problem, attempt the following problems:

1. Print the actual subarray with the maximum sum.
2. <https://www.interviewbit.com/problems/flip/>
3. <https://www.interviewbit.com/problems/find-permutation/>

The above two problems use the same underlying principle of the Maximum Subarray Problem.

Once you have solved the above problems, you can then attempt to solve the Maximum 2D Subarray Problem (build over the 1D problem). This will keep you challenged enough for a week.

**Code Format:**

Create a class:

**public class MaximumSubarray**

**{**

**public static int computeMaxSum1DSubarray(List<Integer> oneDArray)**

**{**

**// return the maximum sum possible**

**}**

**public static int computeMaxSum2DSubarray(List<List<Integer>> twoDArray)**

**{**

**// return the maximum sum possible**

**}**

**// If there are multiple subarrays possible, return the first one.**

**public static List<Integer> computeMax1DSubarray(List<Integer> oneDArray)**

**{**

**}**

**// If there are multiple 2D subarrays possible, return the first one.**

**public static List<List<Integer>> computeMax2DSubarray(List<List<Integer>> twoDArray)**

**{**

**}**

**}**

Fill the above class with your solution. This helps us in picking anyone's class, treat it as a blackbox and run tests on it.

Happy Coding!