Question 1

1. List

This data structure is suitable for the task because it maintains the order of elements, allows for indexing, and is flexible. This makes it suitable for calculating the Maximum Sub Array sum in a sequential list of numbers.

1. Initializing maxSum with the first element of the list sets a starting point for comparison and ensures that the algorithm handles negative numbers correctly. It is a critical step in correctly identifying the maximum subarray sum.
2. The for loop iterates through the list from the first element to the last. The sumz variable keeps track of the current sub array sum and is updated by adding the current element. If sumz becomes negative, it is reset to 0 to ensure accurate tracking of the maximum sub array sum.
3. The code identifies the maximum sub array sum by iterating and updating the maxSum variable. The maxSum is updated when the following condition is met: if sumz > maxSum, indicating that the current sub array sum is greater than any previously encountered sum. Additionally, if sumz becomes negative ( sumz < 0), it is reset to 0 to start a new potential subarray. maxSum holds the maximum sub array sum when the loop completes.

e)

The code's time complexity for finding the maximum subarray sum using Kadane's algorithm is O(n), where 'n' is the number of elements in the input list. The choice of a Python list data structure and Kadane's algorithm ensures a simple and efficient solution with a linear time complexity, making it suitable for handling large lists with good performance.

Objectives

1. C
2. B
3. B
4. D
5. C
6. B
7. A
8. D
9. A
10. B