**ANSWERS:**

a) The data structure being used to store the numbers in the code is a Python list represented as ‘1st’. This data structure enables us to find the maximum subarray sum because it allows you to store and access a sequence of numbers. Lists provide easy indexing and iteration making it straight forward to implement algorithms that involve processing elements in a specific order.

b) The code initializes `maxSum` with the first element of the list to ensure that it has an initial value. This initialization sets the initial maximum subarray sum to the value of the first element. It is important because it provides a starting point for the algorithm. The algorithm will consider this initial value and update it as it goes through the list to find the maximum subarray sum.

c) The for loop iterates through the list `1st` using the variable `i` to track the current index. The `sumz` variable is used to keep track of the running sum of the subarray as the loop iterates. It is updated within the loop by adding the current element in the list to the running sum. If the running sum becomes negative, it is reset to zero effectively neglecting any previous negative sum. This is done to ensure that negative subarrays do not contribute to the maximum subarray sum.

d) The code identifies the maximum subarray sum by continuously updating the `maxSum` variable. It does this by comparing the running sum with the current `maxSum`. If the running sum becomes greater than the current `maxSum`, it updates `maxSum` with the higher value. This process continues as the loop iterates through the list. Conditions that trigger the update of the `maxSum` variable are:

- If `sumz` becomes greater than `maxSum`, set `maxSum` to `sumz`.

- If `sumz` becomes negative, reset `sumz` to zero.

e) The time complexity of this code for finding the maximum subarray sum is O(n), where n is the length of the list `1st`. The choice of a list data structure and the Kadane's algorithm implementation contributes to the efficiency of the code. Kadane's algorithm is an efficient approach for finding the maximum subarray sum because it iterates through the list only once keeping track of the running sum and the maximum sum. It avoids unnecessary recalculations and provides a linear time complexity which is efficient for large lists. The use of a list data structure is appropriate as it allows easy access to elements by index which is essential for this task.