1. Lists. It enables the user to read, store, process and print hundreds of items thus not having to spend a long time stating each item for the list.
2. By starting with the first element, the user ensures that the algorithm considers at least one element in its search for the maximum sum, even if all the elements are negative.
3. The for loop iterates through the list from the first element to the last. It uses the i variable as an index, starting from 0 and going up to the length of the lst list. The purpose of the sumz variable is to keep track of the current subarray sum. Within the loop, sumz is updated by adding the current element of the list at index i.
4. The code identifies the maximum subarray sum by comparing the sumz variable with the maxSum variable. The sumz variable represents the sum of the current subarray being considered. If sumz becomes negative, it is reset to 0 because any negative sum will lower the maximum subarray sum. If sumz is greater than maxSum, maxSum is updated with the value of sumz, indicating that a new maximum subarray sum has been found.
5. The time complexity of this code for finding the maximum subarray sum is O(n), where n is the length of the input list. This is because the code contains a single loop that iterates through the list once, and all other operations inside the loop are constant time operations .