Problem:

Given an array of integers (positive and negative) find the largest continuous sum.

Solution:

If the array is all positive, then the result is simply the sum of all numbers. The negative numbers in the array will cause us to need to begin checking sequences.

The algorithm is, we start summing up the numbers and store in a current sum variable. After adding each element, we check whether the current sum is larger than maximum sum encountered so far. If it is, we update the maximum sum. As long as the current sum is positive, we keep adding the numbers. When the current sum becomes negative, we start with a new current sum. Because a negative current sum will only decrease the sum of a future sequence. Note that we don’t reset the current sum to 0 because the array can contain all negative integers. Then the result would be the largest negative number.

**def** large\_cont\_sum(arr):

*# Check to see if array is length 0*

**if** len(arr)==0:

**return** 0

*# Start the max and current sum at the first element*

max\_sum=current\_sum=arr[0]

*# For every element in array*

**for** num **in** arr[1:]:

*# Set current sum as the higher of the two*

current\_sum=max(current\_sum+num, num)

*# Set max as the higher between the currentSum and the current max*

max\_sum=max(current\_sum, max\_sum)

**return** max\_sum

My Soln:

**def large\_cont\_sum(arr):**

**max1 = 0**

**sum1 = 0**

**if len(arr)==0:**

**return 0**

**for i in range(len(arr)):**

**if sum1 <= 0:**

**if arr[i] <= 0:**

**max1 = max(arr[i],max1)**

**else:**

**sum1 = arr[i]**

**max1 = max(sum1,max1)**

**else:**

**sum1 = sum1 + arr[i]**

**max1 = max(sum1,max1)**

**return max1**