Bubble sorting

```
#Bubble Sort
def func_bubblesort(list01):

    complete = False
    while complete == False: # this means that the loop will cotninue until all items are in correct palce rather than just looping once (alternative for i in range (0, (len(list01) -1)): )
        complete = True
        position = 0

        for x in range(0,(len(list01) -1)): # Why did craig and dave put - 2 here??? it doesn't work!
            if list01[position] > list01[position+1]: # equal items not re-ordered   # swaps positions of right is greater than left
                temp = int(list01[position]) # temporary variable as it is to be overwritten
                list01[position] = list01[position+1]
                list01[position+1] = temp
                complete = False

            #else no swaps are made so complete is true and lsit is fully sorted
            # we could indeed put else: and then compelte = True here but this iterates and so is inefficient we don't neeed to
            # say that it is complete every tiem we go through so instead it is put at the beginning so if no swaps are made it must be complete
            position = position + 1

    return list01

def main_proc():
    list01 = [9,8,2,55,3,2,6,3,0]
    print("Unsorted list is: " , list01)

    sorted_list = func_bubblesort(list01)
    print("The sorted list is:", str(sorted_list))

#loops program
response = "Y"
while response == "Y":
    main_proc()
    response = str(input("\nContinue? (y/n)"))
    response = response.upper()
```

Bubble sorting is a method of sorting a list in to ascending or descending order. Every item in the list is checked with the item next to it, if the item next to it is of a lower value then the two items swap positions. This means that after only one iteration the highest number in the list will be the last item in the list. However, the algorithm must continue to check through the list since this will not fully sort the list, there may be a higher number to the left or several places back of an element as this comparison was not made. The algorithm must iterate n times where n comparisons are made. In total there are n^2 iterations unless efficiency is improved.