Problem 1

Complete the function below that takes a Python list as an argument. You can assume the list will only contain numbers. Use a `for` - or `while` -loop to compute the total (i.e. sum) of all the entries in `alist` . The function should return only the total. For example, calling the function with

```
list_sum([1, 2, 3])
```

will return `6` .

Do not use the built in Python command `sum()` , the tests will fail if you try.

In [ ]:
```
def list_sum(alist):
    total = 0
    for i in alist:
        total += i
    return total

    #return #sum of all entries in list
```

In [ ]:
```
list_sum([1, 2, 3])
```

Out[ ]: 6

Problem 2

Complete the function below that takes a Python list as an argument. You can assume the list will only contain numbers. Use a `for` - or `while` -loop to compute the cumulative sum at each entry of the input list. The function should return a list, where each value in the list is the sum of the value in the equivalent index in `alist` and all those before it in order. For example, calling the function with

```
cumulative_sum([1, 2, 3])
```

will return `[1, 3, 6]` . And

```
cumulative_sum([1, 3, 5, 8])
```

will return `[1, 4, 9, 17]` .

In [ ]:
```
def cumulative_sum(alist):
    cu_list = []
    length = len(alist)
    cu_list = [sum(alist[0:x:1]) for x in range(0, length+1)]
    return cu_list[1:]
#    return #list containing cumulative sums
```

In [ ]:
```
cumulative_sum([1, 2, 3])
```

Out[ ]: [1, 3, 6]

In [ ]:
```
def cumulative_sum2(alist):
    new_list = []
    j = 0
    for i in range(0,len(alist)):
        j += alist[i]
        new_list.append(j)
    return new_list
```

In [ ]:
```
cumulative_sum2([2, 5, 6])
```

Out[ ]: [2, 7, 13]