1. To make this Pythagorean Theorem function, the function will take in two values, a & b, and then apply the Pythagorean Theorem c = sqrt(a^+b^2) which will return c. I use the operator x**y which is x^y to get the a^2+b^2 and to get the square root function.

   Code:

```
def pythagoreamTheorem(a, b):
    c = (a**2+b**2)**0.5 # the c = sqrt(a^2+b^2) formula
    return c
```

```
print(pythagoreamTheorem(3,6))
print(pythagoreamTheorem(2,2))
print(pythagoreamTheorem(3,4))
```

   Result:

```
6.708203932499369
2.8284271247461903
5.0
```

2. To make the list mangler function, I started out by creating a new list the same length of the list taken in. This is to prevent the original input from being changed. The function will cycle through every element in using a for loop. It will check if the element is even or odd. If the element is even, the value of that element will be multiplied by two and put in the same index of the new list. If the element is odd it will do the same thing but multiply the value by three.

   Code:

```
def list_mangler(list):
    newList = [0]*len(list) #creates new list so you do not change old list
    for i in range(len(list)): #checks every value to see if they are even or odd
        if isEven(list[i]): # If even multiply by 2
            newList[i] = list[i]*2 #add new value to same spot in new list
        else:#if odd multiply by 3
            newList[i] = list[i]*3
    return newList #return new list
```

```
list1 = [67,78.99,110,45,34]
list2 = [400, 34,56,19,26,74]
list3 = [45,83,39,193]
print(list_mangler(list1))
print(list_mangler(list2))
print(list_mangler(list3))
```

   Result:

```
[201, 236.96999999999997, 220, 135, 68]
[800, 68, 112, 57, 52, 148]
[135, 249, 117, 579]
```

3. To make this function, I started by making a sorted new list which is a copy of the old list. This is to avoid changing the original list. It then deletes the first n (number of grades to drop) elements which will delete the n lowest grades. It will then cycle through the grade thresholds to decide the letter grade.

   Code:

```
def grade_calc(grades, to_drop):
    newGrades = grades.copy() # create new list  so that the original list is not affected
    newGrades.sort() # sort new list in ascending order
    del newGrades[0:to_drop] #delete first to_drop elements of the list which eliminates the n lowest number of elements
    mean = statistics.mean(newGrades) #gets the mean of the list
    if mean < 60: # will cycle through each letter grade threshold
        final_grade = "Your final grade is a F"
    elif mean < 70:
        final_grade = "Your final grade is a D"
    elif mean < 80:
        final_grade = "Your final grade is a C"
    elif mean < 90:
        final_grade = "Your final grade is a B"
    else:
        final_grade = "Your final grade is a A"
    return final_grade # return final letter grade
```

```
list1 = [67,78.99,110,45,34]
list2 = [100, 34,56,19,26,74]
list3 = [45,83,39,193]
print(grade_calc(list1,1))
print(grade_calc(list2,2))
print(grade_calc(list3,3))
```

   Results:

```
Your final grade is a C
Your final grade is a D
Your final grade is a A
```

4. To create the function, I started by creating to sublist for odd and even elements. The function will cycle through each element, check if the element is even or odd, and then place them in the correct list.

   Code:

```
def odd_even_filter(list):
    evenList = [] # create 2 list for even and odd numbers
    oddList = []
    for i in range(len(list)): #cycle through each element in list
        if isEven(list[i]): #if element is even, append element to even list
            evenList.append(list[i])
        else:
            oddList.append(list[i]) #if element is odd, append element to odd list
    sublists = [evenList, oddList] #make list made of lists
    return sublists #return list made of lists


list1 = [67,78.99,110,44.22,34]
list2 = [100, 34,56,19,26,74]
list3 = [45,83,39,193]
print(odd_even_filter(list1))
print(odd_even_filter(list2))
print(odd_even_filter(list3))
```

Result:

```
[[110, 34], [67, 78.99, 44.22]]
[[100, 34, 56, 26, 74], [19]]
[[], [45, 83, 39, 193]]
```