Solutions Solved Exercises Lists

Problem 1:

`sorted` will return a sorted version of the list, but not change the original list.
`sort( )` will sort the original list.

```
    lista = ['air', 'water', 'fire', 'earth', 'ether']
    print(sorted(lista))
    lista.sort()
    print(lista)
```

Problem 2:

```
    lista = list(range(1, 101))
    lista.extend(list(range(501,601)))
    print(lista)
```

Problem 3:
I do not check for error conditions (e.g. the list is empty or contains a zero), because the Python errors are appropriate in these cases.

```
def geometric_mean(lista):
    product = 1
    for element in lista:
        product *= element
    return product**(1/len(lista))

def harmonic_mean(lista):
    denominator = 0
    for element in lista:
        denominator += 1/element
    return len(lista)/denominator
```

Problem 4.1:
We use the standard pattern, i.e., we define an empty result list into which we add the elements of the list, but only, if they were not already added.

```
def elements(a_list):
    result = [ ]
    for element in a_list:
        if element not in result:
            result.append(element)
    return result
```

Problem 4.2:

```
def minus(list_one, list_two):
    result = []
    for element in list_one:
        if element not in list_two:
            result.append(element)
    return result
```

Problem 4.3:

We use the standard pattern. We copy the elements of the list into the result list, but only if they are not already there:

```
def intersection(list_one, list_two):
    result = []
    for element in list_one:
        if element in list_two:
            result.append(element)
    return result
```

Problem 4.4:

We could try to count the number of occurrences of each element, running through the list several times. To do this, we first write a function `count(an_element, a_list),` that counts the number of occurrences of this element in the list. We put those with two occurrences into the result list:

```
def count(an_element, a_list):
    count = 0
    for element in a_list:
        if element == an_element:
            count += 1
    return count

def twice(a_list):
    result = [ ]
    for element in a_list:
        if count(element, a_list) == 2:
            if element not in result:
                result.append(element)
    return result
```

This is not a very good solution for long lists. For each element in the list, we run through the whole list again and look at all elements. Thus, if there are 1000 elements in the list, for each of them, we look at all 1000 elements, meaning that we are checking for equality a million times. If the list has 100,000 elements, then we would check for equality 10,000,000,000 and this would take a real long time.

A different solutions makes three auxiliary lists: one for elements seen once or more, one for elements seen at least twice, and one for elements seen at least three times. We still use the standard pattern. We go through the list. If the element is not in the first list, we put it there. This must be the first time we see it. If the element is in the first list, but not in the second list, then we have seen it for the second time. So we put it into the second list. Finally, if we see an element that is in the first, the second, and not in the third list, then we see this element for the third time. Finally, we make a list of all elements in the second list but not in the third list and return it. This turns out to be much faster for long lists, because the checking for membership is done internally very efficiently.

```
def twice_a(a_list):
    once_or_more = []
    twice_or_more = []
    thrice_or_more = []
    for element in a_list:
        if element not in once_or_more:
            once_or_more.append(element)
        elif element not in twice_or_more:
            twice_or_more.append(element)
        else:
            thrice_or_more.append(element)
    result = []
    for element in twice_or_more:
        if element not in thrice_or_more:
            result.append(element)
    return result
```