Task 2:

a) When we change b[-1] to "apples", the value of a[-1] also changes. Because a and b are pointing to the same object. Assignment in python doesn't create a new object. It creates a reference to the existing object. When coding b = a you assign address of a to b. Therefore, changing data in b will also "change" a.

b) When we change b[-1] to "apples", the value of a[-1] doesn't change. Because a and b are not pointing to the same object in memory. Because b is a shallow copy of only values of a. When coding b = a[:] it just creates a new object in memory. So, they don't point to the same address. Therefore, changing data in one of them do not change the other.

c) When we change b[-1][-1] to "apples", the value of a[-1][-1] also changes. However, a and b are not pointing to the same object in memory, but nested list in a and b are pointing to the same object. Memory location of the nested list are the same. Therefore, changing data in the nested list inside a list (in this case b) modifies data in the other list (in this case a).

d) deepcopy() creates a new object in memory. Therefore, a and b, and their elements are not pointing to the same object in memory even they are nested lists. Therefore changing data in the nested list inside a list (in this case b) doesn’t modify data in the other list (in this case a) while using deepcopy()