Exercise 1:

1/ **What is nested list?**

**A nested list** is simply a list that occurs as an element of another list (which may of course itself be an element of another list, etc.). Common reasons nested lists arise are: They're matrices (a list of rows, where each row is itself a list, or a list of columns where each column is itself a list)

2/ **Can a list store both integers and strings in it?**

Yes, a list can store both integers and strings in it.

The following list contains a string, a float, an integer, and (amazingly) another list:

1 zs = ["hello", 2.0, 5, [10, 20]]

A list within another list is said to be **nested**.

Exercise 2:

Using slices to delete list elements can be error-prone. Python provides an alternative that is more readable. The del statement removes an element from a list:

>>> a = ["one", "two", "three"]

>>> del a[1]

>>> a

[’one’, ’three’]

As you might expect, del causes a runtime error if the index is out of range.

You can also use del with a slice to delete a sublist:

>>> a\_list = ["a", "b", "c", "d", "e", "f"]

>>> del a\_list[1:5]

>>> a\_list

[’a’, ’f’]

As usual, the sublist selected by slice contains all the elements up to, but not including, the second index.

Example:

>>> myList

['Yes', 'The', 'earth', 'revolves', 'around', 'sun', ['a', 'true'], 'statement', 'for', 'sure']

>>> myList.remove("statement")

>>> myList

['The', 'earth', 'revolves', 'around', 'sun', ['a', 'true'], 'for', 'sure']

Here we see that remove can be used to easily delete elements in list.

Here is how a sub list can be deleted :

>>> myList.remove(["a", "true"])

>>> myList

['The', 'earth', 'revolves', 'around', 'sun', 'for', 'sure']

So we see that the sub list was deleted from the original list.

If it is required to access the last element and then to delete it, this can be done through pop method.

>>> myList.pop()

'sure'

>>> myList

['The', 'earth', 'revolves', 'around', 'sun', 'for']

So we see that the value was displayed and  deleted simultaneously.