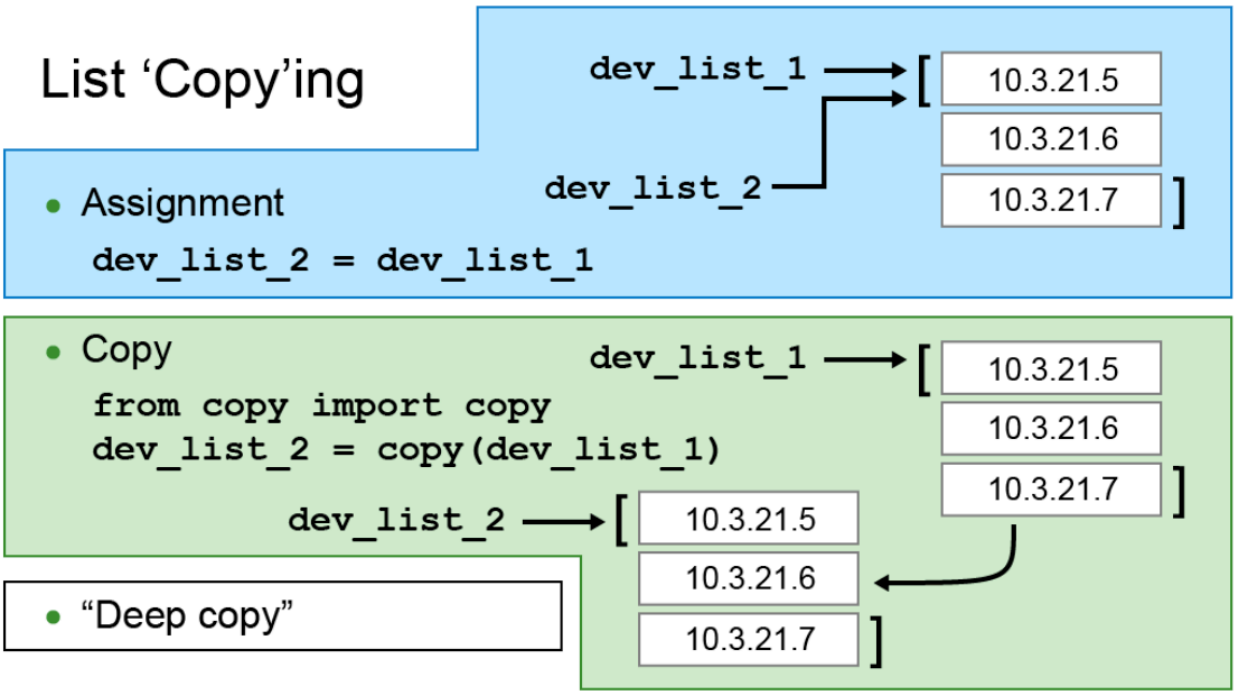

You will need to understand what is meant by 'assignment' and 'copy' in Python.

Assignment

Recall that everything is an object, and that variables are just names that reference, or point at, that object. When you do an assignment in Python, you are just having the new variable reference the original value. As you can see in the figure, when you say:

```
dev_list_2 = dev_list_1
```

You are having dev_list_2 point to the same list that is pointed to by dev_list_1. If you then change the value of items in the list, add items to the list or remove them, both dev_list_1 and dev_list-2 are affected.

Copy (Shallow Copy)

If you want to make a copy of dev_list_1, use the 'copy' function available in Python. In order to use the copy function, you must import it from the Python standard library which is done using the `import` statement.

```
from copy import copy
dev_list_2 = copy(dev_list_1)
```

In this example, you are actually copying the entire list, and all items within it. Using the copy function gives dev_list_2 a completely new copy of dev_list_1. After a copy, changes to dev_list_2 have no effect on dev_list_1, and vice versa.

Note that the copy function is actually doing a 'shallow' copy. A shallow copy, in computer programming, means that the only first level of items is copied. If you happened to have a list that had complex data structures such as a list of lists, or a list of dictionaries, a shallow copy does not make a copy of those referenced items–it only copies the first-level items. If you wish to copy all items, no matter how deeply nested are the data structures, you must do a deep copy.

Copy (Deep Copy)

If you need to do a deep copy on your list, use the `deepcopy` function in Python:

```
from copy import deepcopy
complex_list_2 = deepcopy(complex_list_1)
```

The result is that all data from complex_list_1 is copied to complex_list_2.