

Lists and Mutability

We thought we understood everything there was to know about lists. Let's see.

```
In [1]: x = 34
        y = x
        y = 20
        print("x =", x, ", y =", y)

x = 34    y = 20
```

```
In [2]: mammals = ['cat', 'kangaroo', 'horse']
        fish = ['tuna', 'shark', 'catfish']
```

```
In [3]: animals = [mammals, fish]
        print(animals)

[['cat', 'kangaroo', 'horse'], ['tuna', 'shark', 'catfish']]
```

```
In [4]: mammals[0] = 'mule'
        print("mammals = ", mammals)
        print("animals = ", animals)

mammals = ['mule', 'kangaroo', 'horse']
animals = [['mule', 'kangaroo', 'horse'], ['tuna', 'shark', 'catfish']]
```

Let's see this in the tutor. Visit www.pythontutor.com/visualize.html (<http://www.pythontutor.com/visualize.html>)

```
In [5]: mammals_2 = mammals
        mammals_2[0] = 'dog'
        print("mammals_2 = ", mammals_2)
        print("mammals = ", mammals)
        print("animals = ", animals)

mammals_2 = ['dog', 'kangaroo', 'horse']
mammals = ['dog', 'kangaroo', 'horse']
animals = [['dog', 'kangaroo', 'horse'], ['tuna', 'shark', 'catfish']]
```

So, it's the same list with different names! But can't we make a copy of the list?

```
In [6]: mammals_3 = mammals[:] # this creates a copy!
```

```
In [7]: mammals_3[0] = 'mouse'
        print("mammals_3 = ", mammals_3)
        print("mammals = ", mammals)
        print("animals = ", animals)

mammals_3 = ['mouse', 'kangaroo', 'horse']
mammals = ['dog', 'kangaroo', 'horse']
animals = [['dog', 'kangaroo', 'horse'], ['tuna', 'shark', 'catfish']]
```

Let's see what's going on in the tutor!

But it gets better (or worst, depending on how you look at it) ...

```
In [8]: s1 = ['student 1', 'James', 'Brown']
        s2 = ['student 2', 'Chris', 'Black']
        s3 = ['student 3', 'Phil', 'Blue']
```

Beyond Basic Programming - Intermediate Python

recluze.net/learn

```
In [9]: students = [s1, s2, s3]
        print(students)
```

```
[['student 1', 'James', 'Brown'], ['student 2', 'Chris', 'Black'], ['student 3', 'Phil', 'Blue']]
```

```
In [10]: students_copy = students[:]
         print(students_copy)
```

```
[['student 1', 'James', 'Brown'], ['student 2', 'Chris', 'Black'], ['student 3', 'Phil', 'Blue']]
```

```
In [11]: students_copy[0][0] = 'student 1 changed'
         print(students_copy)
         print(students)
```

```
[['student 1 changed', 'James', 'Brown'], ['student 2', 'Chris', 'Black'], ['student 3', 'Phil', 'Blue']]
[['student 1 changed', 'James', 'Brown'], ['student 2', 'Chris', 'Black'], ['student 3', 'Phil', 'Blue']]
```

Ok, that was unexpected. Let's go back to the tutor.

Deep Copying

```
In [12]: import copy
```

```
In [13]: students_3 = copy.deepcopy(students)
```

```
In [14]: students_3[0][0] = 'student 1 changed again '
         print(students_3)
         print('-----')
         print(students)
```

```
[['student 1 changed again ', 'James', 'Brown'], ['student 2', 'Chris', 'Black'], ['student 3', 'Phil', 'Blue']]
-----
[['student 1 changed', 'James', 'Brown'], ['student 2', 'Chris', 'Black'], ['student 3', 'Phil', 'Blue']]
```

Passing lists as arguments

Finally, let's see what happens when we pass a list to a function.

```
In [15]: def change_var(x):
         x = x + 1

         def change_list(lst):
             lst.append(24)
```

```
In [16]: x = 25
         change_var(x)
         print(x) # our global x remains the same
```

```
25
```

```
In [17]: print(s2)
         change_list(s2)
         print(s2) # but the list is affected!
```

```
['student 2', 'Chris', 'Black']
['student 2', 'Chris', 'Black', 24]
```

Beyond Basic Programming - Intermediate Python

recluze.net/learn

Now this should be understandable. Let's go back to the tutor.

And that's why we have tuples as immutable .. so that called functions are guaranteed to not change them.