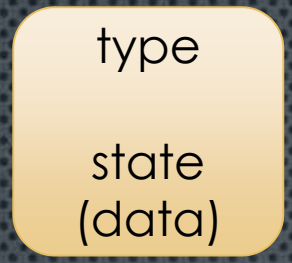
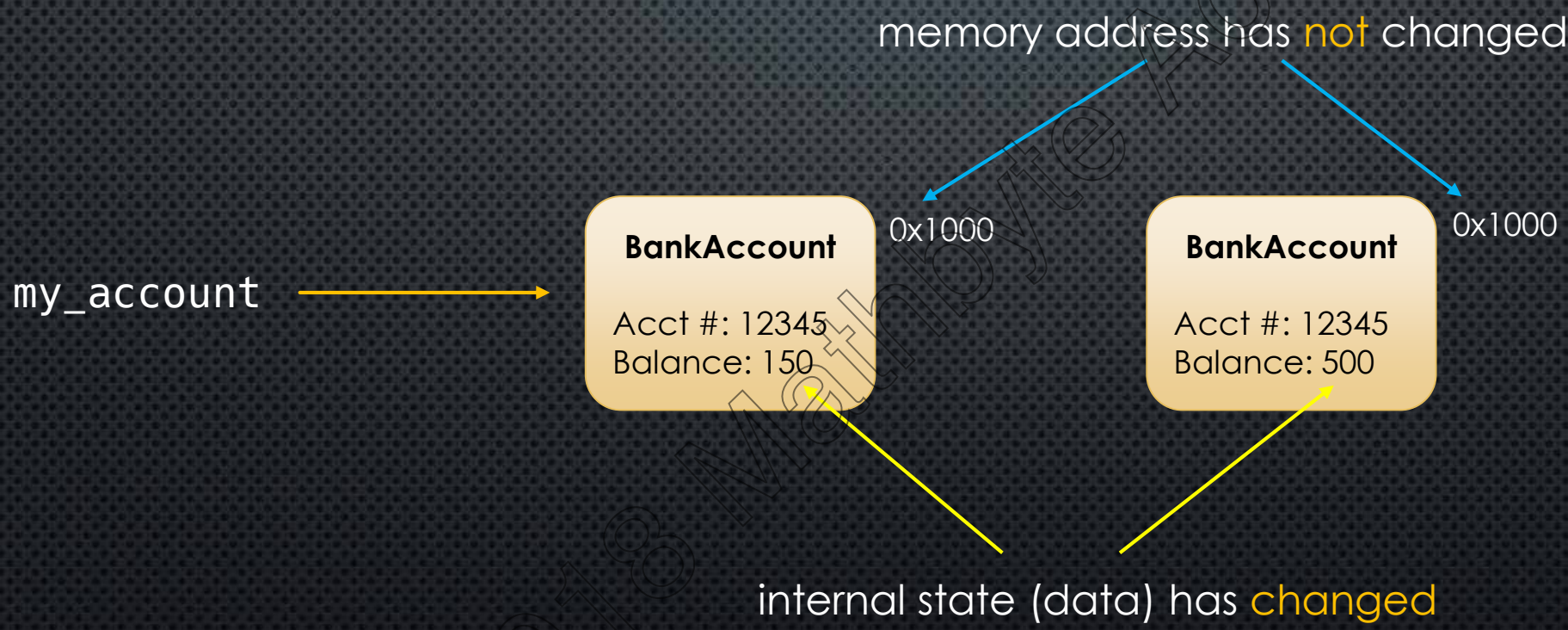
OBJECT MUTABILITY

Consider an object in memory:



Changing the data inside the object is called modifying the internal state of the object.



Object was mutated

→ fancy way of saying the internal data has changed

An object whose internal state can be changed, is called

Mutable

An object whose internal state cannot be changed, is called

Immutable

Examples in Python

Immutable

- Numbers (int, float, Booleans, etc)
- Strings
- Tuples
- Frozen Sets
- User-Defined Classes

Mutable

- Lists
- Sets
- Dictionaries
- User-Defined Classes

```
t = (1, 2, 3)
```

Tuples are immutable: elements cannot be deleted, inserted, or replaced

In this case, both the container (tuple), and all its elements (ints) are immutable

But consider this:

```
a = [1, 2]
b = [3, 4]
```

Lists are mutable: elements can be deleted, inserted, or replaced

```
t = (a, b)        t = ([1, 2], [3, 4])

a.append(3)
b.append(5)       t = ([1, 2, 3], [3, 4, 5])
```

In this case, although the tuple is immutable, its elements are not.

The object references in the tuple did not change

but the referenced objects did mutate!

```
t = (1, 2, 3)
```

tuple is immutable

these are references to immutable object (int)

```
t = ([1, 2], [3, 4])
```

tuple is immutable

these are references to a mutable object (list)