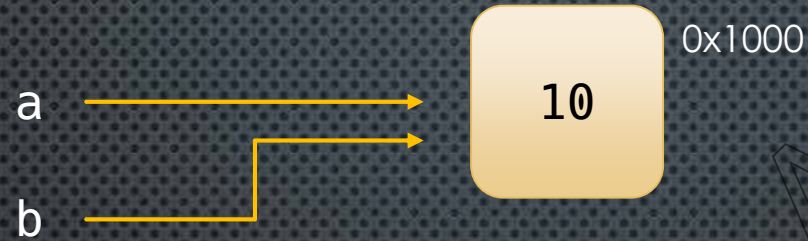


# SHARED REFERENCES AND MUTABILITY

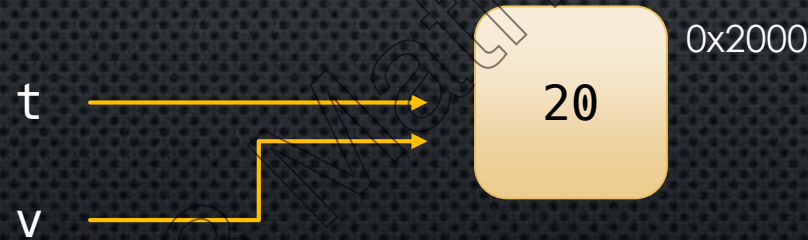


The term **shared reference** is the concept of two variables referencing the **same** object in memory (i.e. having the same memory address)

```
a = 10  
b = a
```



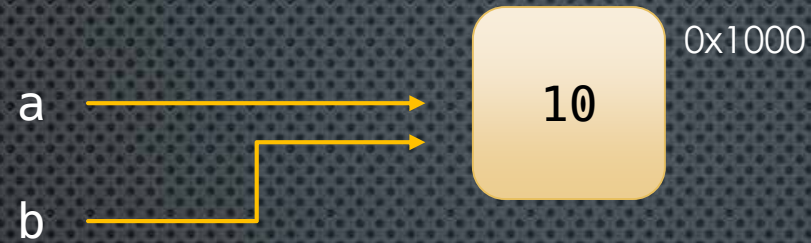
```
def my_func(v):  
    ...  
  
t = 20  
my_func(t)
```



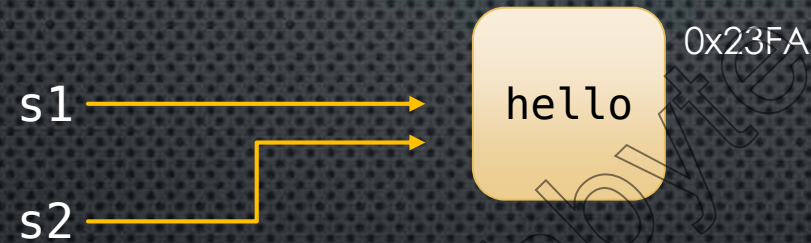


In fact, the following may surprise you:

```
a = 10  
b = 10
```



```
s1 = 'hello'  
s2 = 'hello'
```



In both these cases, Python's memory manager decides to automatically re-use the memory references!!

We'll revisit this again soon

Is this even safe? **Yes**

The integer object 10, and the string object 'hello' are **immutable** – so it is safe to set up a shared reference

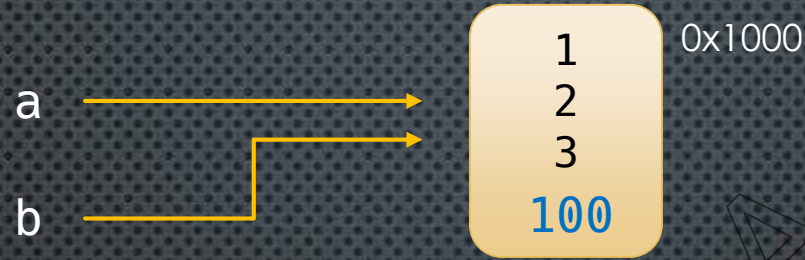


When working with **mutable** objects we have to be more careful

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

```
b = a
```

```
b.append(100)
```



With mutable objects, the Python memory manager will never create shared references

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

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

