FUNCTION ARGUMENTS
AND
MUTABILITY
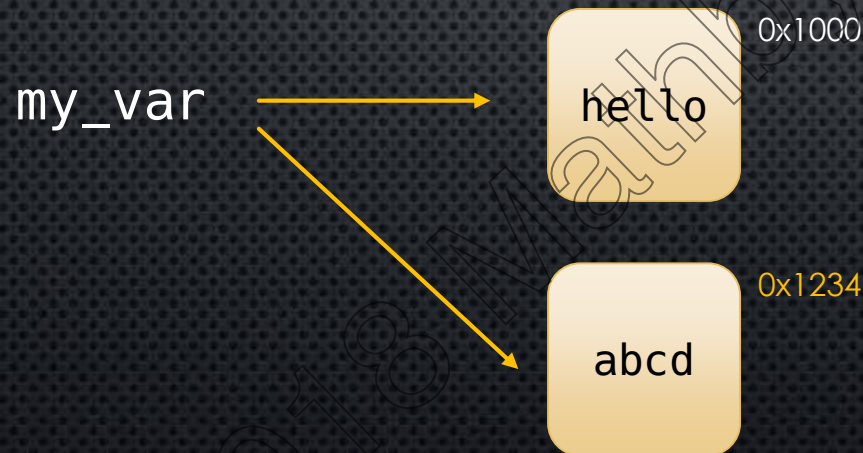
In Python, Strings (`str`) are immutable objects.

Once a string has been created, the contents of the object can never be changed

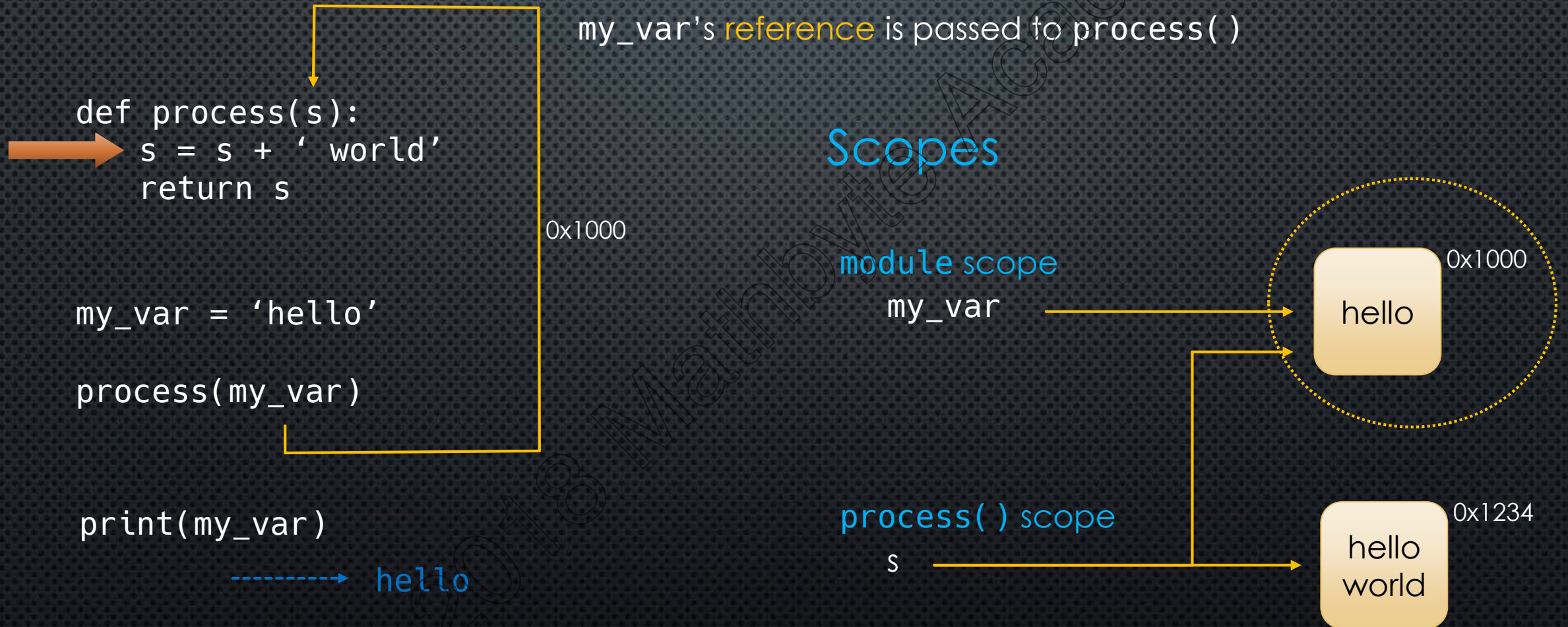In this code: `my_var = 'hello'`

 the only way to modify the "value" of `my_var` is to re-assign my_var to another object

```
my_var  ──────────►  ┌──────────┐ 0x1000
                     │          │
            \        │  hello   │
             \       │          │
              \      └──────────┘
               \
                \     ┌──────────┐ 0x1234
                 \    │          │
                  ──► │  abcd    │
                     │          │
                     └──────────┘
```
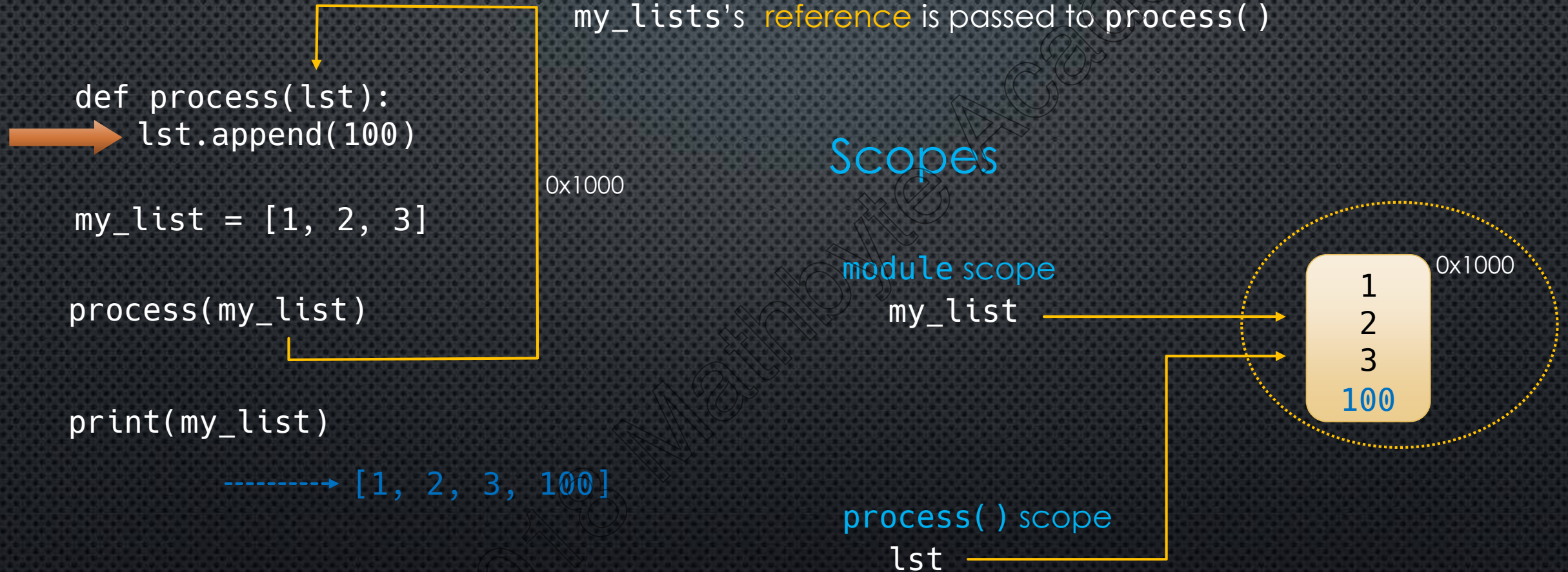
Immutable objects are safe from unintended side-effects

but watch out for immutable collection objects that contain mutable objects

my_var's reference is passed to process()

```
def process(s):
    s = s + ' world'
    return s



my_var = 'hello'


process(my_var)



print(my_var)
            --------->  hello
```

0x1000

Scopes

module scope
        my_var -----> | hello | 0x1000

process() scope
        s -----> | hello world | 0x1234

Mutable objects are not safe from unintended side-effects

my_list's reference is passed to process()

```python
def process(lst):
    lst.append(100)


my_list = [1, 2, 3]


process(my_list)


print(my_list)
```

0x1000

---------> [1, 2, 3, 100]

Scopes

module scope
my_list

0x1000
1
2
3
100

process() scope
lst

Immutable collection objects that contain mutable objects

my_tuple's reference is passed to process()

```python
def process(t):
    t[0].append(3)

0x1000

my_tuple = ([1,2], 'a')

process(my_tuple)

print(my_tuple)

            ([1, 2, 3], 'a')
```

Scopes

module scope
my_tuple

0x1000
[1, 2, 3]
'a'

process() scope
t