← course home (/table-of-contents)

Mutable vs Immutable Objects

A **mutable** object can be changed after it's created, and an **immutable** object can't.

For example, lists are mutable in Python:

Python 2.7

```
int_list  = [4, 9]

int_list[0] = 1
# int_list is now [1, 9]
```

And tuples are immutable:

Python 2.7

```
int_tuple = (4, 9)

int_tuple[0] = 1
# Raises: TypeError: 'tuple' object does not support item assignment
```

Strings can be mutable or immutable depending on the language.

Strings are immutable in Python:

Python 2.7

```
test_string = 'mutable?'

test_string[7] = '!'
# Raises: TypeError: 'str' object does not support item assignment
```

But in some other languages, like Ruby, strings are mutable:

```ruby
test_string = 'mutable?'

test_string[7] = '!'
# test_string is now 'mutable!'
```

Mutable objects are nice because you can make changes **in-place**, without allocating a new object. But be careful—whenever you make an in-place change to an object, *all* references to that object will now reflect the change.

← course home (/table-of-contents)

Next up: Rectangular Love → (/question/rectangular-love?course=fc1&section=general-programming)