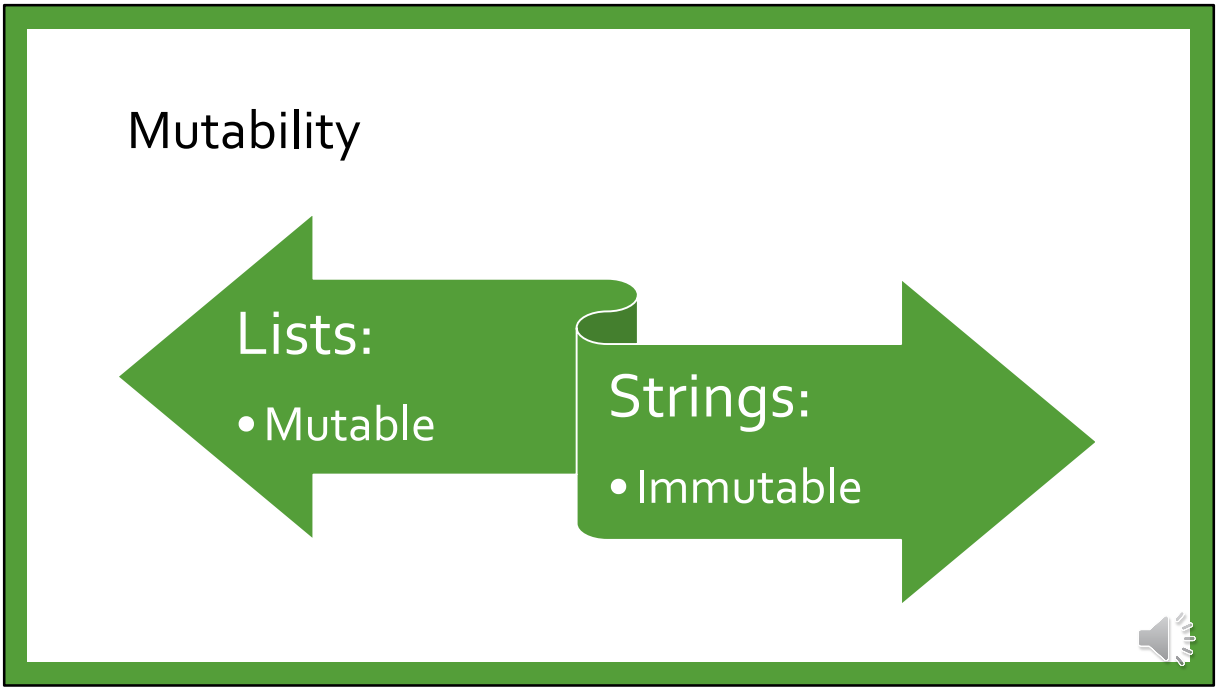
MUTABILITY

An Introduction to Computer Science

Let's learn about Mutability.



Strings and lists are similar because they are both sequences of elements.
However, a big difference between them is their Mutability.
Strings are immutable while lists are mutable.

Immutable Strings

```
name = "Austin Cory Bart"

name.lower()
```

Doesn't change `name`!

```
lowered_name = name.lower()
```

Doesn't change `name`!

```
name = name.lower()
```

Now `name` has been changed

Say you have a string variable.
When you add another string to that variable, you need to assign the result - otherwise it is lost.
Similarly, when you call a string method, you need to assign the result - otherwise it is lost.
This is the idea of immutability - you are never changing the string, you are simply creating new ones.

Mutable Lists

```
grades = [90, 65, 78]

grades.append(98)

grades = grades.append(44)
```

Good! We added 98 to the list

Bad! We added 44, and then overwrote the variable `grades` with None

Now say you have a list variable.
When you append a value to that variable, you must not assign the result back.
The append method modifies the list variable, and then returns None, so if you assign its result you overwrite the list variable.
The append call mutated the list - it changed the data inside the list without affecting the list variable.

Mutability and Parameters

```
def add_x(a_list, a_str):
    a_list.append("X")
    a_str = a_str + " X"

courses = ["Biology", "Music", "Math"]
name = "Ada"
add_x(courses, name)
print(courses)
print(name)
```

`["Biology", "Music", "Math", "X"]`

`"Ada"`

We have seen mutability before when we learned about parameters of functions.
Let's take a look at this function `add_x` that consumes a list and a string, and adds the string value `"X"` to the end.
When the function call is over, only the list has actually been changed.
Even though we were using local versions of each variable, the mutability of the list type meant that the original list was changed.
If we wanted to change the `name` variable, we would have had to return the modified value and assign it back to the `name` variable.