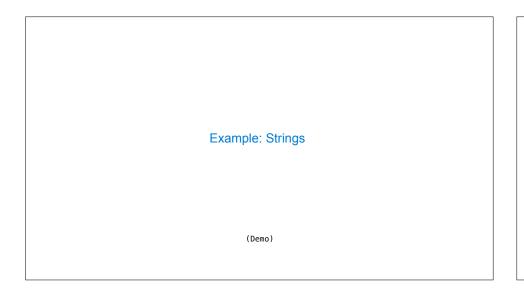
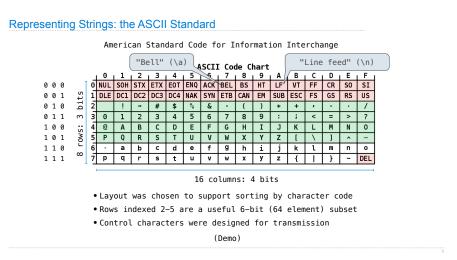
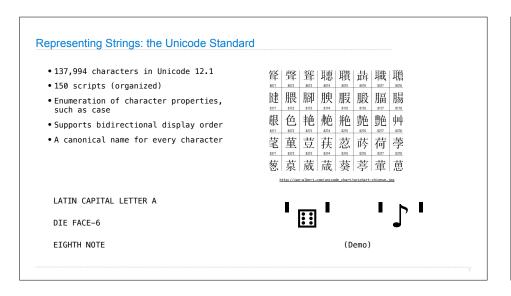
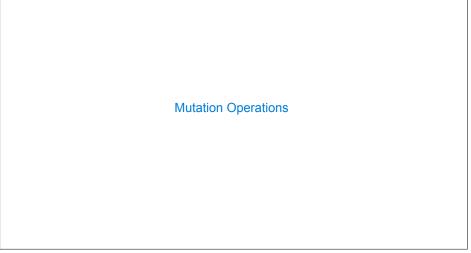
Mutability	Announcements
Objects (Demo)	Objects  • Objects represent information  • They consist of data and behavior, bundled together to create abstractions  • Objects can represent things, but also properties, interactions, & processes  • A type of object is called a class; classes are first-class values in Python  • Object-oriented programming:  • A metaphor for organizing large programs  • Special syntax that can improve the composition of programs  • In Python, every value is an object  • All objects have attributes  • A lot of data manipulation happens through object methods  • Functions do one thing; objects do many related things









# Some Objects Can Change

#### [Demo]

First example in the course of an object changing state

The same object can change in value throughout the course of computation



All names that refer to the same object are affected by a mutation

Only objects of mutable types can change: lists & dictionaries

{Demo}

# **Tuples**

(Demo)

# Mutation Can Happen Within a Function Call

A function can change the value of any object in its scope

# Tuples are Immutable Sequences

Immutable values are protected from mutation

The value of an expression can change because of changes in names or objects

An immutable sequence may still change if it contains a mutable value as an element

# Mutation

# Sameness and Change

- \*As long as we never modify objects, a compound object is just the totality of its pieces
- \*A rational number is just its numerator and denominator
- \*This view is no longer valid in the presence of change
- \*A compound data object has an "identity" in addition to the pieces of which it is composed
- •A list is still "the same" list even if we change its contents
- \*Conversely, we could have two lists that happen to have the same contents, but are different

```
>>> a = [10]
>>> b = [10]
>>> a = [10]
>>> b = a
>>> a == b
                                    >>> a == b
True
                                    True
>>> a.append(20)
                                    >>> b.append(20)
>>> a
                                    >>> a
[10, 20]
                                    [10]
>>> b
                                    >>> b
[10, 20]
                                    [10, 20]
>>> a == b
                                    >>> a == b
True
                                    False
```

# **Identity Operators**

#### Identity

<exp0> is <exp1>

evaluates to True if both <exp0> and <exp1> evaluate to the same object

#### Equality

<exp0> == <exp1>

evaluates to True if both <exp0> and <exp1> evaluate to equal values

Identical objects are always equal values

(Demo)

# Mutable Default Arguments are Dangerous

A default argument value is part of a function value, not generated by a call

