

Announcements

String Representations

String Representations

An object value should behave like the kind of data it is meant to represent

For instance, by producing a string representation of itself

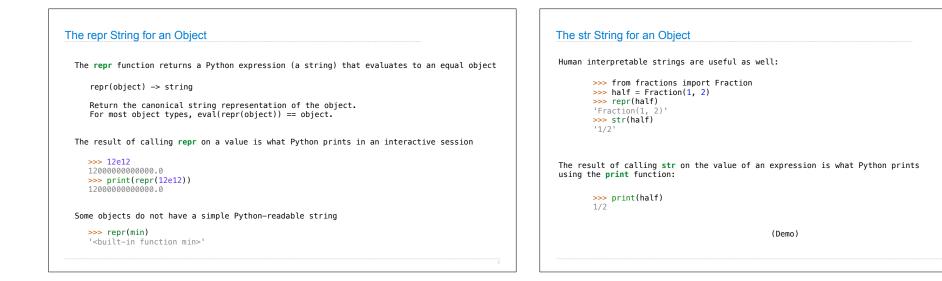
Strings are important: they represent language and programs

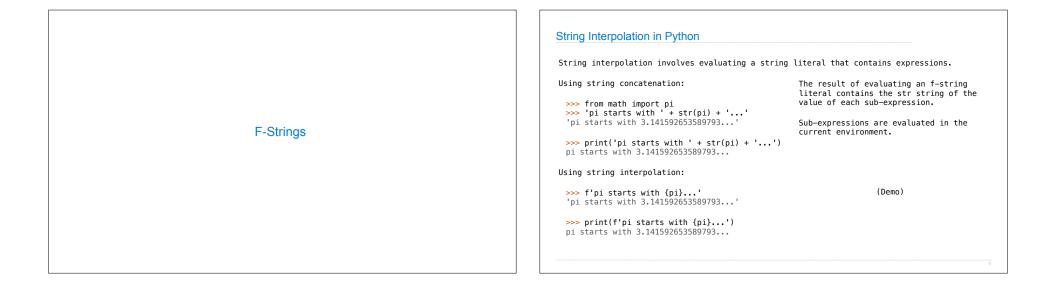
In Python, all objects produce two string representations:

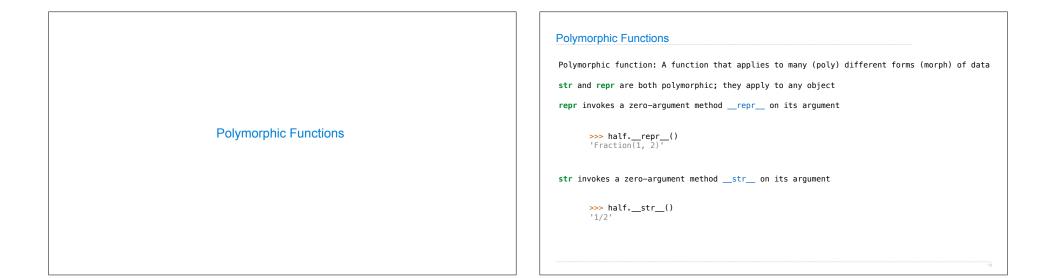
• The **str** is legible to humans

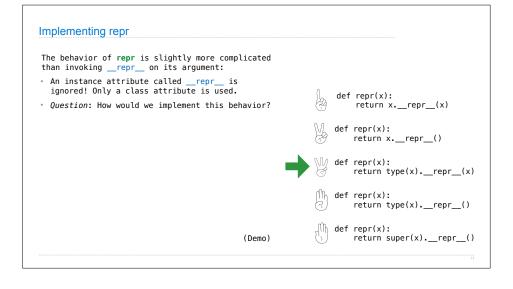
• The **repr** is legible to the Python interpreter

The **str** and **repr** strings are often the same, but not always









Interfaces

Message passing: Objects interact by looking up attributes on each other (passing messages)

The attribute look-up rules allow different data types to respond to the same message

A ${\bf shared}\ {\bf message}\ ({\rm attribute}\ {\rm name})\ {\rm that}\ {\rm elicits}\ {\rm similar}\ {\rm behavior}\ {\rm from}\ {\rm different}\ {\rm object}\ {\rm classes}\ {\rm is}\ {\rm a}\ {\rm powerful}\ {\rm method}\ {\rm of}\ {\rm abstraction}$

An interface is a set of shared messages, along with a specification of what they mean

Example:

Classes that implement $_repr_$ that returns a Python–evaluatable string implements an interface for producing machine–readable string representations

Special Method Names	Special Method Names in Python
	Certain names are special because they have built-in behavior
	These names always start and end with two underscores
	init Method invoked automatically when an object is constructed
	repr Method invoked to display an object as a Python expression
	add Method invoked to add one object to another
	bool Method invoked to convert an object to True or False
	float Method invoked to convert an object to a float (real number)
	<pre>>>> zero, one, two = 0, 1, 2 >>> one + two 3 >>> bool(zero), bool(one) (False, True)</pre> <pre> Same behavior using methods</pre> >>> zero, one, two = 0, 1, 2 >>> oneadd(two) 3 >>> zerobool_(), onebool_() (False, True)

Special Methods

Adding instances of user-defined classes invokes either the __add__ or __radd__ method

>>> Ratio(1, 3) + Ratio(1, 6)
Ratio(1, 2)

>>> Ratio(1, 3).__add__(Ratio(1, 6))
Ratio(1, 2)

>>> Ratio(1, 6).__radd__(Ratio(1, 3))
Ratio(1, 2)

http://getpython3.com/diveintopython3/special-method-names.html

http://docs.python.org/3/reference/datamodel.html#special_method_names

(Demo)

Generic Functions

A polymorphic function might take two or more arguments of different types

Type Dispatching: Inspect the type of an argument in order to select behavior

Type Coercion: Convert one value to match the type of another

>>> Ratio(1, 3) + 1 Ratio(4, 3)

>>> 1 + Ratio(1, 3) Ratio(4, 3)

>>> from math import pi
>>> Ratio(1, 3) + pi
3.4749259869231266

(Demo)