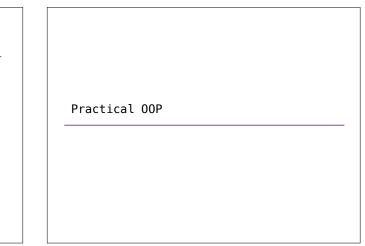
Lecture 17: Mutable Linked Lists

Brian Hou July 20, 2016

Announcements

- Homework 6 is due today at 11:59pm
- Project 3 is due 7/26 at 11:59pm
- Earn 1 EC point for completing it by 7/25
- Quiz 5 tomorrow at the beginning of lecture
 May cover mutability, OOP I (Monday)
- Project 1 revisions due 7/27 at 11:59pm

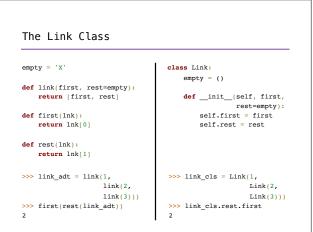
Roadmap Introduction Functions Data Mutability	This week (Objects), the goals are: • To learn the paradigm of object-oriented programming • To study applications of, and
Objects Interpretation Paradigms	problems that be solved using, OOP
Applications	

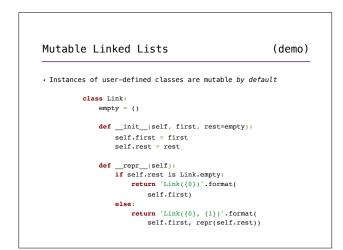


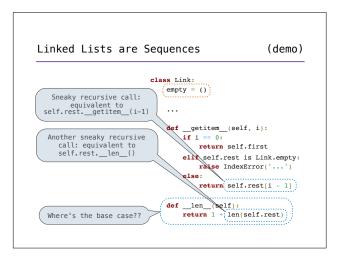
Checking Types (and	Accounts)	(demo)
 We often check the type of an object to determine what operations it permits The type built-in function 	<pre>>>> a = Account('Brian') >>> ch = CheckingAccount(>>> type(a) == Account True</pre>	'Brian')
returns the class that its argument is an instance of • The isinstance built-in function returns whether	<pre>>>> type(ch) == Account False >>> type(ch) == CheckingA True</pre>	ccount
<pre>its first argument (object) is an instance of the second argument (class) or a subclass</pre>	<pre>>>> isinstance(a, Account True >>> isinstance(ch, Account True</pre>	·
<pre>• isinstance(obj, cls) is usually preferred over type(obj) == cls</pre>	<pre>>>> isinstance(a, Checkin False >>> isinstance(ch, Checki True</pre>	· · ·

Python's Magic Methods	(demo
• How does the Python interpreter disp	lay values?
• First, it evaluates the expression	to some value
• Then, it calls repr on that value	and prints that string
 How do magic methods work? Are integers objects too? (Yep!) Are objects too? (Yep!) 	<pre>>>> x = Rational(3, 5 >>> y = Rational(1, 3 >>> y Rational(1, 3) >>> rep(y) 'Rational(1, 3)' >>> print(repr(y)) Rational(1, 3) >>> x * y Rational(1, 5) >>> xmul_(y) Rational(1, 5)</pre>

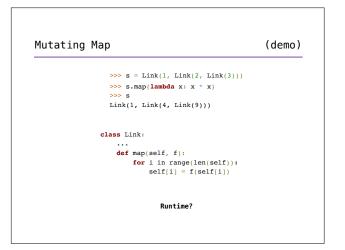


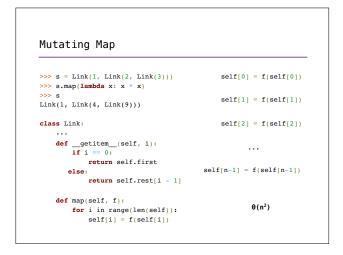


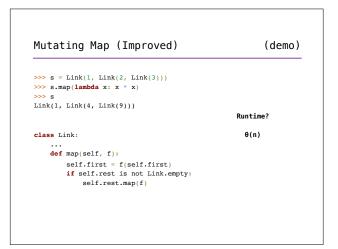


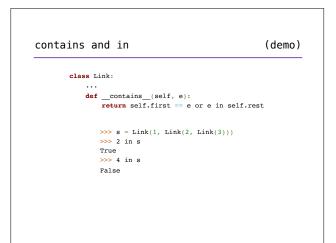


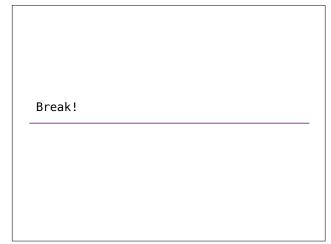
Thesetitem Magic Method	(demo)
>>> s = Link(1, Link(2, Link(3)))	
>>> s[1] = 3	
>>> s	
Link(1, Link(3, Link(3)))	
class Link:	
<pre>defsetitem_(self, i, val):</pre>	
if i == 0:	
<pre>self.first = val</pre>	
elif self.rest is Link.empty:	
<pre>raise IndexError('')</pre>	
else:	
<pre>self.rest[i - 1] = val</pre>	

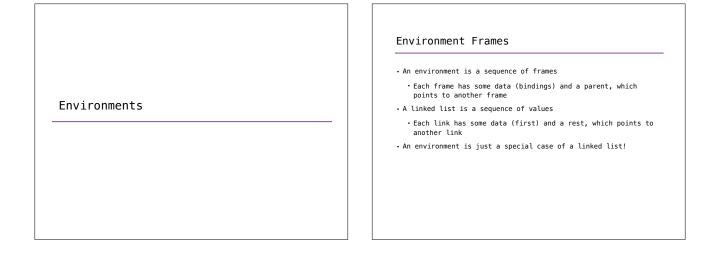


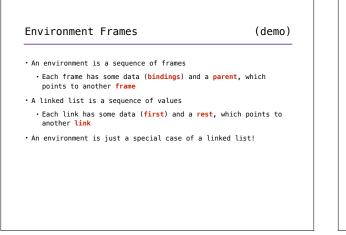


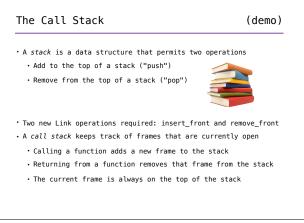


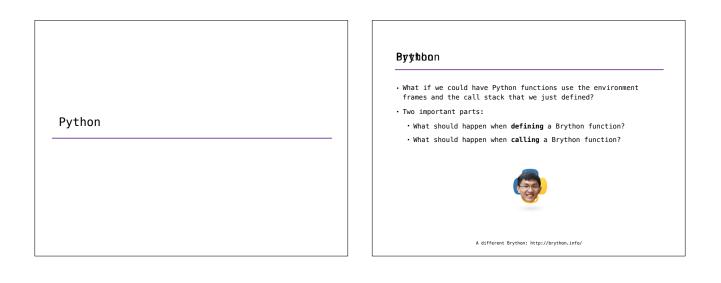


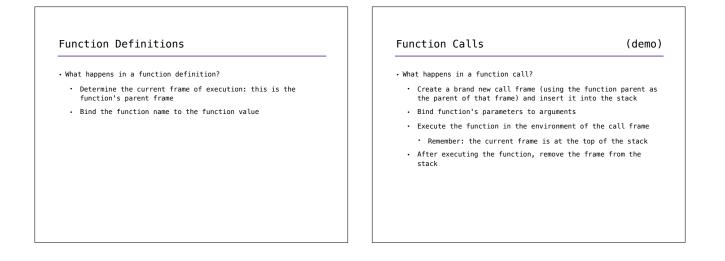












Summary

- $\boldsymbol{\cdot}$ Linked lists are one way to store sequential data
- An object-based implementation of the linked list abstraction allows for easy mutability

No more crazy nonlocal stuff!

- Implementing magic methods lets us hook into convenient Python syntax and built-in functions
- Linked lists can be used to implement some of the core ideas of this course!