Environments Enable Higher-Order Functions

Functions are first-class: Functions are values in our programming language.

Higher-order function: A function that takes a function as an argument value or
A function that returns a function as a return value.

Environment diagrams describe how higher-order functions work!

Environment Diagrams for Nested Definitions

Environments for Nested Definitions

Names can be Bound to Functional Arguments

Environment Diagrams for Nested Def Statements

Environments for Higher-Order Functions

Office Hours: You Should Go!

You are not alone!

http://cs61a.org/office-hours.html
How to Draw an Environment Diagram

When a function is defined:
Create a function value: \(\text{func <name>(<formal parameters>) [parent=<label>]}\)
Its parent is the current frame.

Bind <name> to the function value in the current frame.

When a function is called:
1. Add a local frame, titled with the <name> of the function being called.
2. Copy the parent of the function to the local frame: \([\text{parent=<label>}\)]
3. Bind the <formal parameters> to the arguments in the local frame.
4. Execute the body of the function in the environment that starts with the local frame.

Local Names

Local Names are not Visible to Other (Non-Nested) Functions

The Environment Diagram for Function Composition

Lambda Expressions

Lambda Expressions versus Def Statements

Lambda expressions are not common in Python, but important in general.
Lambda expressions in Python cannot contain statements at all!