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!

(Demo)

Names can be Bound to Functional Arguments

Environment Diagrams for Nested Def Statements

Every user-defined function has a parent frame (often global).
- The parent of a function is the frame in which it was defined.
- Every local frame has a parent frame (often global).
- The parent of a frame is the parent of the function called.
How to Draw an Environment Diagram

When a function is defined:
Create a function value: 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: [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 are not Visible to Other (Non-Nested) Functions

- An environment is a sequence of frames.
- The environment created by calling a top-level function (no def within def) consists of one local frame, followed by the global frame.

The Environment Diagram for Function Composition

- Return value of make_adder is an argument to compose

Self-Reference

- Returning a Function by Its Own Name

Lambda Expressions
Lambda Expressions

```python
>>> x = 10
>>> square = x * x
>>> square = lambda x: x * x
```

An expression: this one evaluates to a number

Also an expression:

A function: evaluates to a function

Important: No "return" keyword!

that returns the value of \( x^2 \)

Must be a single expression

Lambda expressions are not common in Python, but important in general

Lambda expressions in Python cannot contain statements at all!

Lambda Expressions Versus Def Statements

```python
square = lambda x: x * x
```

```python
def square(x):
    return x * x
```

VS

- Both create a function with the same domain, range, and behavior.
- Both functions have as their parent the frame in which they were defined.
- Both bind that function to the name square.
- Only the def statement gives the function an intrinsic name.