CS61A Lecture 6

Amir Kamil
UC Berkeley
February 4, 2013
Locally Defined Functions

The inner definition is executed each time the outer function is called.

Example: [http://goo.gl/pnU8f](http://goo.gl/pnU8f)
Locally defined functions can be returned
They have access to the frame in which they are defined

A function that returns a function

```python
def make_adder(n):
    """Return a function that adds n to its argument.
    >>> add_three = make_adder(3)
    >>> add_three(4)
    7
    """
    def adder(k):
        return add(n, k)
    return adder
```

The name add_three is bound to a function

A local def statement

Can refer to names in the enclosing function
Functions are first-class: they can be manipulated as values in Python

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

Higher order functions:
- Express general methods of computation
- Remove repetition from programs
- Separate concerns among functions
Environment of Function Application

The environment in which a function is applied consists of:

- A *new* local frame *each* time the function is *applied*

- The environment in which the function was *defined*
  - Before, this was just the global frame
  - For a locally-defined function, this includes all local frames in the definition environment, plus the global frame
Environment for Non-Nested Function

The environment of a function application is a new local frame plus the environment in which the function was defined.

Example: http://goo.gl/73anC
Every user-defined function has a parent frame.

The parent frame of a function is the frame in which it was defined.

Every local frame has a parent frame.

The parent of a local frame is the parent of the function called.
The Structure of Environments

A frame extends the environment that begins with its parent

A three-frame environment

A two-frame environment

The global environment: the environment with only the global frame

When a frame or function has no label

[parent=___]

then its parent is always the global frame
How to Draw an Environment Diagram

When defining a function:

Create a function value with signature
<name>(<formal parameters>)

For nested definitions, label the parent as the first frame of the current environment

Bind <name> to the function value in the first frame of the current environment

When calling a function:

1. Add a local frame labeled with the <name> of the function
2. If the function has a parent label, copy it to this frame
3. Bind the <formal parameters> to the arguments in this frame
4. Execute the body of the function in the environment that starts with this frame
Environment for Function Composition

Example: http://goo.gl/5zcug
Lambda Expressions

```python
>>> ten = 10

>>> square = x * x
An expression: this one evaluates to a number

>>> square = lambda x: x * x
Also an expression: evaluates to a function

>>> square(4)
16
A function with formal parameter x and body "return x * x"

Lambda expressions are rare in Python, but important in general
```
Evaluation of Lambda vs. Def

\[ \text{lambda } x: x \times x \quad \text{VS} \quad \text{def } \text{square}(x): \text{return } x \times x \]

Execution procedure for def \textit{statements}:
1. Create a function value with signature \texttt{<name>(<formal parameters>)}
   and the current frame as parent
2. Bind \texttt{<name>} to that value in the current frame

Evaluation procedure for lambda \textit{expressions}:
1. Create a function value with signature \texttt{\lambda(<formal parameters>)}
   and the current frame as parent
2. Evaluate to that value
Lambda vs. Def Statements

\[ \text{square} = \text{lambda } x: \ x \star \ x \quad \text{VS} \quad \text{def square}(x): \ \text{return } x \star x \]

Both create a function with the same arguments & behavior

Both of those functions are associated with the environment in which they are defined

Both bind that function to the name "square"

Only the def statement gives the function an intrinsic name