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

Locally Defined Functions

Functions as Return Values

Higher-Order Functions

Environment of Function Application

Environment for Non-Nested Function
Environment for Nested Function

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.

How to Draw an Environment Diagram

When defining a function:
1. Create a function value with signature `<name>(<formal parameters>)`.
2. For nested definitions, label the parent as the first frame of the current environment.
3. 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.

Lambda Expressions

>>> ten = 10
>>> square = x * x
>>> square = lambda x: x * x

Lambda expressions are rare in Python, but important in general.

Evaluation of Lambda vs. Def

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

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

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

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

The Greek letter lambda