Environments
Announcements
Print and None Review
What does the long expression print?

`s = "Knock"

print(print(print(s, s) or print("Who's There?")), "Who?"

False values in Python: False, 0, '', None (more to come)

To evaluate the expression `<left> or <right>`:
1. Evaluate the subexpression `<left>`.
2. If the result is a true value `v`, then the expression evaluates to `v`.
3. Otherwise, the expression evaluates to the value of the subexpression `<right>`.
Environments for Higher-Order Functions
Names can be Bound to Functional Arguments

1. `def apply_twice(f, x):`
2. `    return f(f(x))`
3. `func apply_twice(f, x) [parent=Global]`
4. `func square(x) [parent=Global]`
5. `Global frame`
6. `apply_twice`
7. `square`

Applying a user-defined function:
- Create a new frame
- Bind formal parameters (f & x) to arguments
- Execute the body: return f(f(x))

```
apply_twice(square, 2)
```
Environments for Nested Definitions

(Demo)
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

```
def make_adder(n):
    def adder(k):
        return k + n
    return adder

add_three = make_adder(3)
add_three(4)
```
How to Draw an Environment Diagram

When a function is defined:

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

\begin{center}
\texttt{f1: make\_adder} \quad \texttt{func adder(k) [parent=f1]}
\end{center}

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.

\star 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.
Twenty-One Environment Diagram

(Demo)
Lambda Expressions

(Demo)
Environment Diagram Practice
def f(x):
    """f(x)(t) returns max(x*x, 3*x) if t(x) > 0, and 0 otherwise."
    y = max(x * x, 3 * x)
    def zero(t):
        if t(x) > 0:
            return y
        return 0
    return zero

# Find the largest positive y below 10 for which f(y)(lambda z: z - y + 10) is not 0.
y = 1
while y < 10:
    if f(y)(lambda z: z - y + 10):
        max = y
    y = y + 1