Environments

Announcements
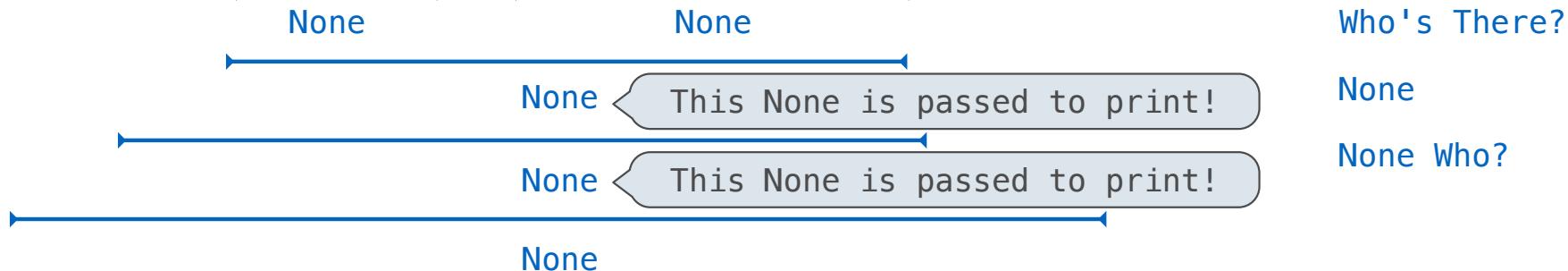
Print and None Review

What does the long expression print?

```
s = "Knock"

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

None   None

None

None   *This None is passed to print!*

None   *This None is passed to print!*

None

Knock Knock

Who's There?

None

None Who?

```
False values in Python:    False, 0, '', None    (more to come)
```
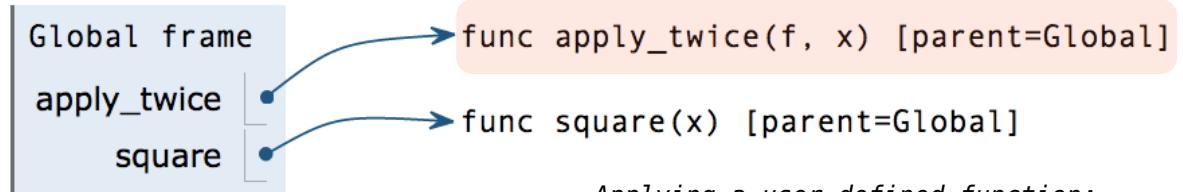
To evaluate the expression **<left> or <right>**:
1. Evaluate the subexpression **<left>**.
2. If the result is a true value **v**, then the expression evaluates to **v**.
3. Otherwise, the expression evaluates to the value of the subexpression **<right>**.

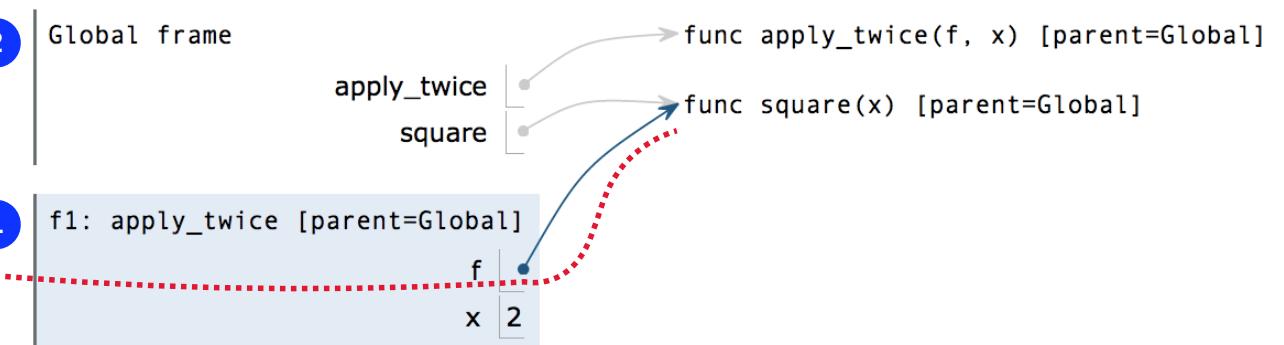# Environments for Higher-Order Functions

Names can be Bound to Functional Arguments

```
1  def apply_twice(f, x):
2      return f(f(x))
3
4  def square(x):
5      return x * x
6
7  result = apply_twice(square, 2)
```

Global frame

apply_twice

square

func apply_twice(f, x) [parent=Global]

func square(x) [parent=Global]

Applying a user-defined function:

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

```
1  def apply_twice(f, x):
2      return f(f(x))
3
4  def square(x):
5      return x * x
6
7  result = apply_twice(square, 2)
```

Global frame

apply_twice

square

func apply_twice(f, x) [parent=Global]

func square(x) [parent=Global]

f1: apply_twice [parent=Global]

f

x 2

pythontutor.com/composingprograms.html#code=def%20apply_twice%28f,%20x%29%3A%0A%20%20%20%20return%20f%28f%28x%29%29%0A%20%20%20%20%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%20%20%20%20%0Aresult%20%3D%20apply_twice%28square,%202%29&mode=display&origin=composingprograms.js&cumulative=true&py=3&rawInputLstJSON=[]&curInstr=0
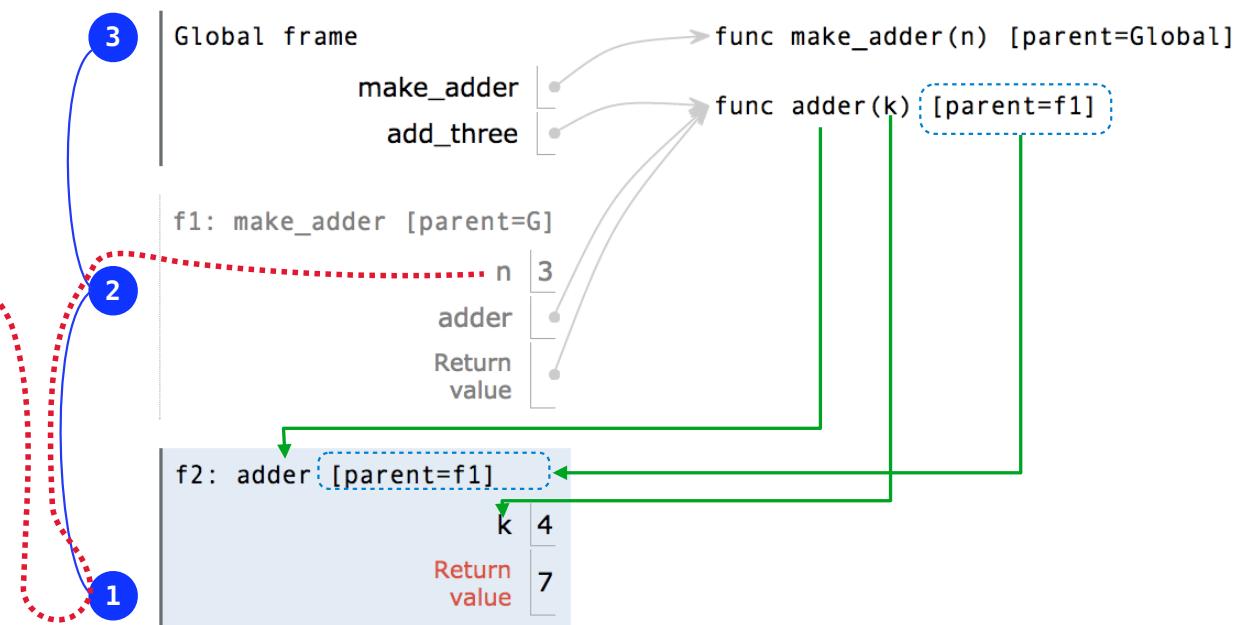
Environments for Nested Definitions

(Demo)

Environment Diagrams for Nested Def Statements

Nested def

```
1  def make_adder(n):
2      def adder(k):
3          return k + n
4      return adder
5
6  add_three = make_adder(3)
7  add_three(4)
```

- 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

Global frame

make_adder
add_three

func make_adder(n) [parent=Global]
func adder(k) [parent=f1]

f1: make_adder [parent=G]

n 3
adder
Return value

f2: adder [parent=f1]

k 4
Return value 7

http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

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.

f1: make_adder   func adder(k) [parent=f1]

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.

Twenty-One Environment Diagram

(Demo)

https://pythontutor.com/cp/composingprograms.html#code=def%20play%28strategy0,%20strategy1,%20goal%3D21%29%3A%0A%20%20%20%20n,
%20who%20%3D%200,%20%0A%20%20%20%20while%20n%20%3C%20goal%3A%0A%20%20%20%20%20%20%20%20if%20who%20%3D%3D%200%3A%0A%20%20%20%20%20%20%20%20%20%20%20%20n%20%3D%20n%20%2B%20strategy0%28n%29%0A%20%20%20%20%20%20%20%20%20%20%20%20who%20%3D%201%0A%20%20%20%20%20%20%20%20elif%20who%20%3D%3D%201%3A%0A%20%20%20%20%20%20%20%20%20%20%20%20n%20%3D%20n%20%2B%20strategy1%28n%29%0A%20%20%20%20%20%20%20%20%20%20%20%20who%20%3D%200%0A%20%20%20%20%20%20%20%20return%20who%0A%0Adef%20two_strat%28n%29%3A%0A%20%20%20%20return%202%0A%0Adef%20noisy_strat%28who,%20s%29%3A%0A%20%20%20%20def%20strat%28n%29%3A%0A%20%20%20%20%20%20%20%20choice%20%3D%20s%28n%29%0A%20%20%20%20%20%20%20%20print%28'Player',%20who,%20'added',%20choice,%20'to',%20n,%20'to%20reach',
%20choice%20%2B%20n%29%0A%20%20%20%20%20%20%20%20return%20choice%0A%20%20%20%20return%20strat%0A%0Aplay%28noisy_strat%280,%20two_strat%29,%20noisy_strat%281,%20two_strat%29%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Lambda Expressions

(Demo)

Environment Diagram Practice

• **The Diagram**
• **Annotations**

```
1: def f(x):
2:     """f(x)(t) returns max(x*x, 3*x)
3:     if t(x) > 0, and 0 otherwise.
4:     """
5:     y = max(x * x, 3 * x)
6:     def zero(t):
7:         if t(x) > 0:
8:             return y
9:         return 0
10:    return zero
11:
12: # Find the largest positive y below 10
13: # for which f(y)(lambda z: z - y + 10)
14: # is not 0.
15: y = 1
16: while y < 10:
17:     if f(y)(lambda z: z - y + 10):
18:         max = y
19:         y = y + 1
```

Global frame

f	→ func f(x) [p=G]
y	1 2
max	1 → func max(...) [p=G]

f1: f   [parent=___G___]

x	1
y	3
zero	→ func zero(t) [p=f1]
Return Value	

f2: zero   [parent=___f1___]

t	→ func λ <ln 17>(z) [p=G]
Return Value	3

f3: λ <ln 17>   [parent=___G___]

z	1
Return Value	10

f4: f   [parent=___G___]

x	2
Return Value	