Environments

Announcements

Environments for Higher-Order Functions

Environments Enable Higher-Order Functions

Environments Enable Higher-Order Functions

Functions are first-class: Functions are values in our programming language

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

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!

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

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

square → func square(x) [parent=Global]

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

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

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

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

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

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

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

square → func square(x) [parent=Global]

f1: apply_twice [parent=Global]

f → func square(x) [parent=Global]

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

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

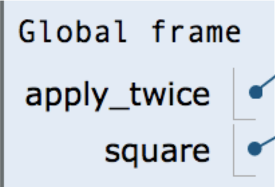
2 Global frame
apply_twice → func apply_twice(f, x) [parent=Global]
square → func square(x) [parent=Global]

1 f1: apply_twice [parent=Global]
f → func square(x) [parent=Global]
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

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)
```



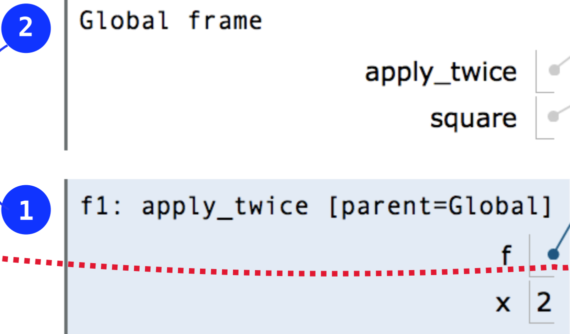
func apply_twice(f, x) [parent=Global]

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)
```



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

func square(x) [parent=Global]

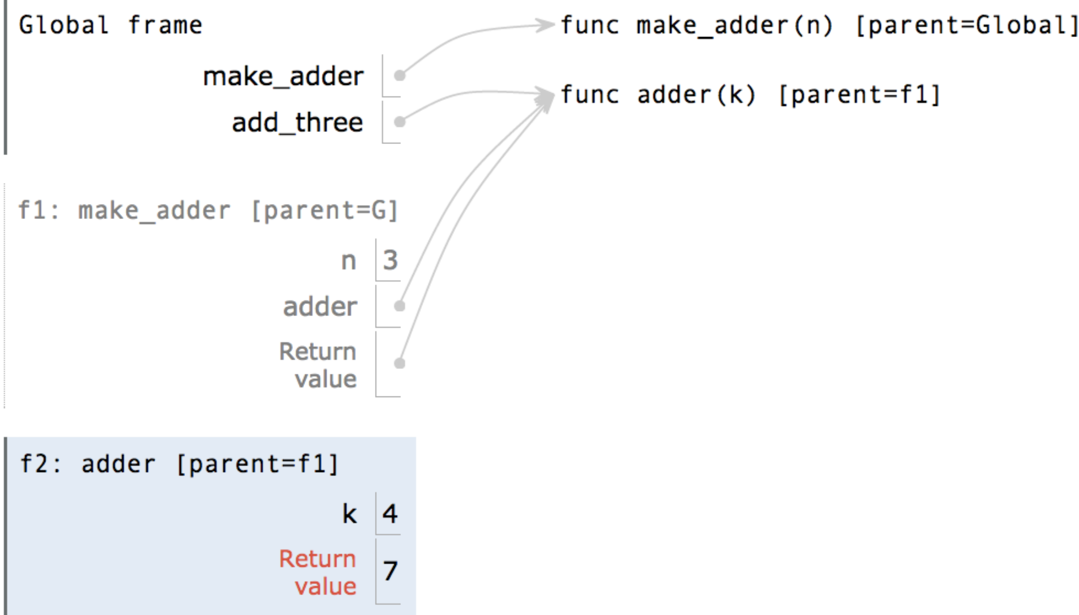
pythontutor.com/composingprograms.html#code=def%20apply_twice%28f,%20x%29%3A%0A%20%20%20%20return%20f%28f%28x%29%29%0A%20%20%20%20%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%20%20%20%20%0Aresult%20%3D%20apply_twice%28square,%202%29&mode=display&origin=composingprograms.js&cumulative=true&py=3&rawInputLstJSON=[]&curInstr=0

Environments for Nested Definitions

(Demo)

Environment Diagrams for Nested Def Statements

```
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)
```

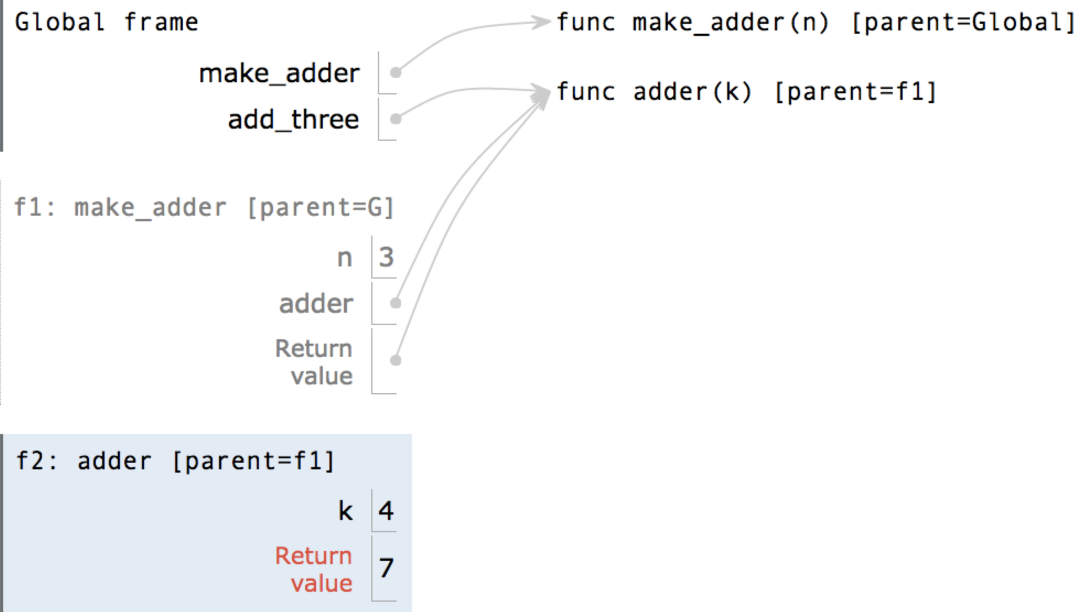


http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

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)
```

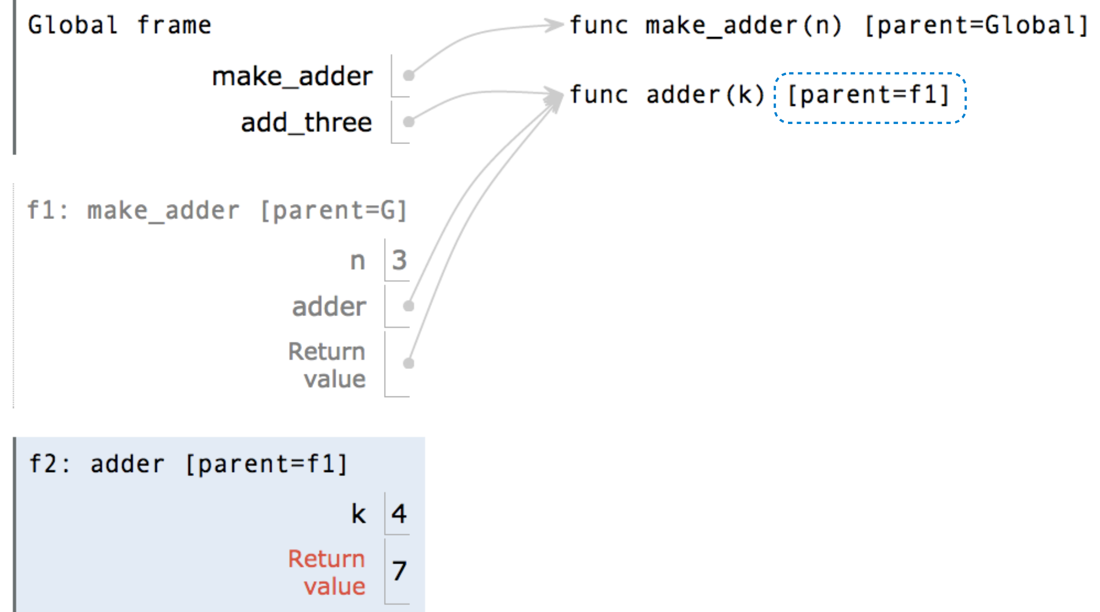


http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

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)
```

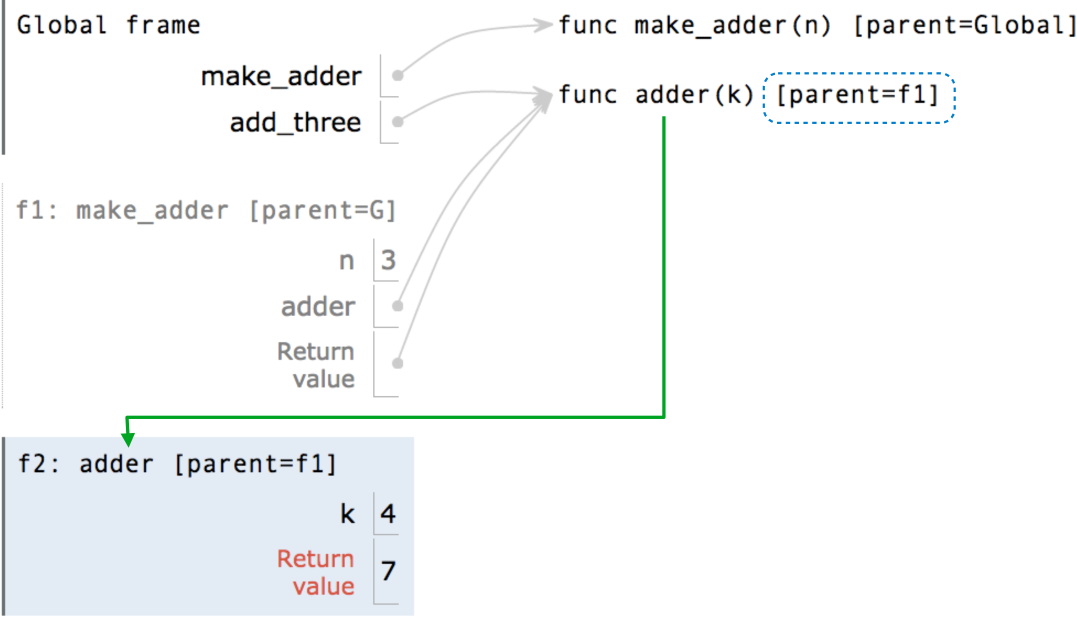


http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

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)
```

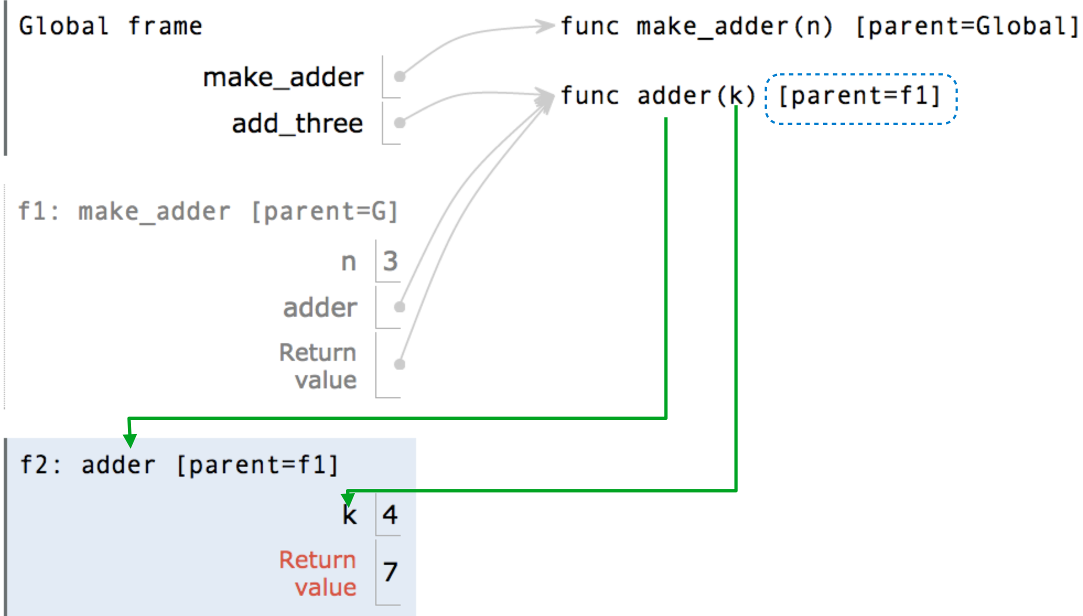


http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

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)
```

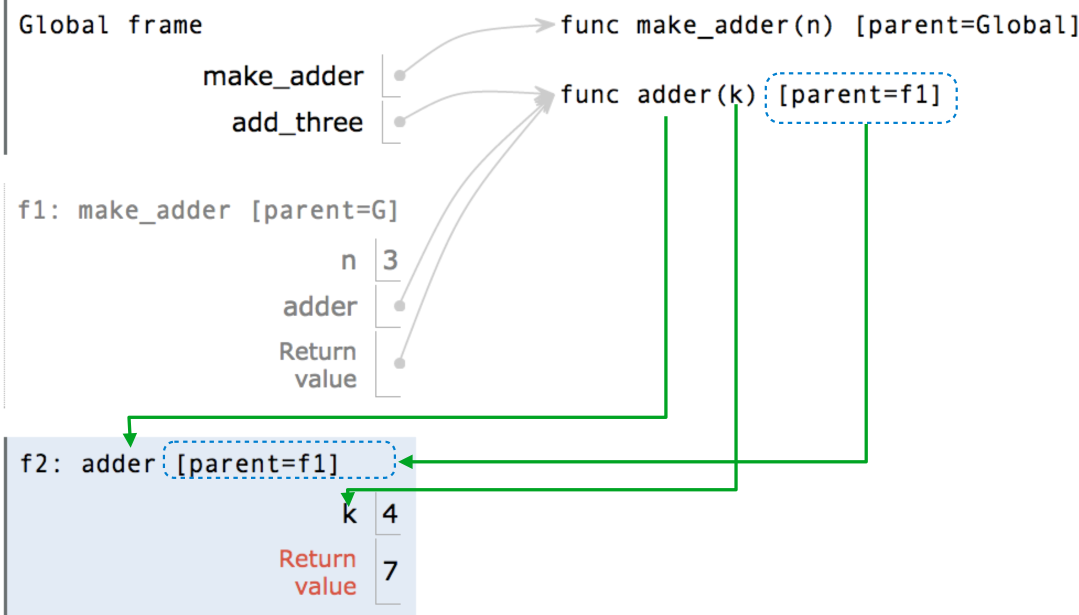


http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

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)
```

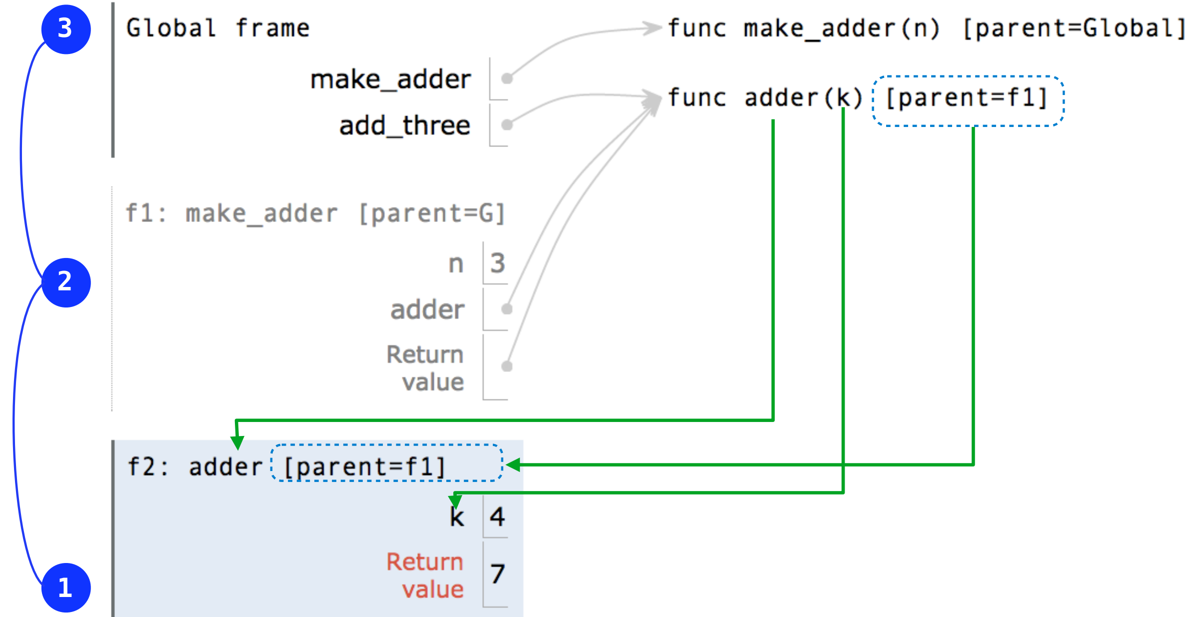


http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

Environment Diagrams for Nested Def Statements

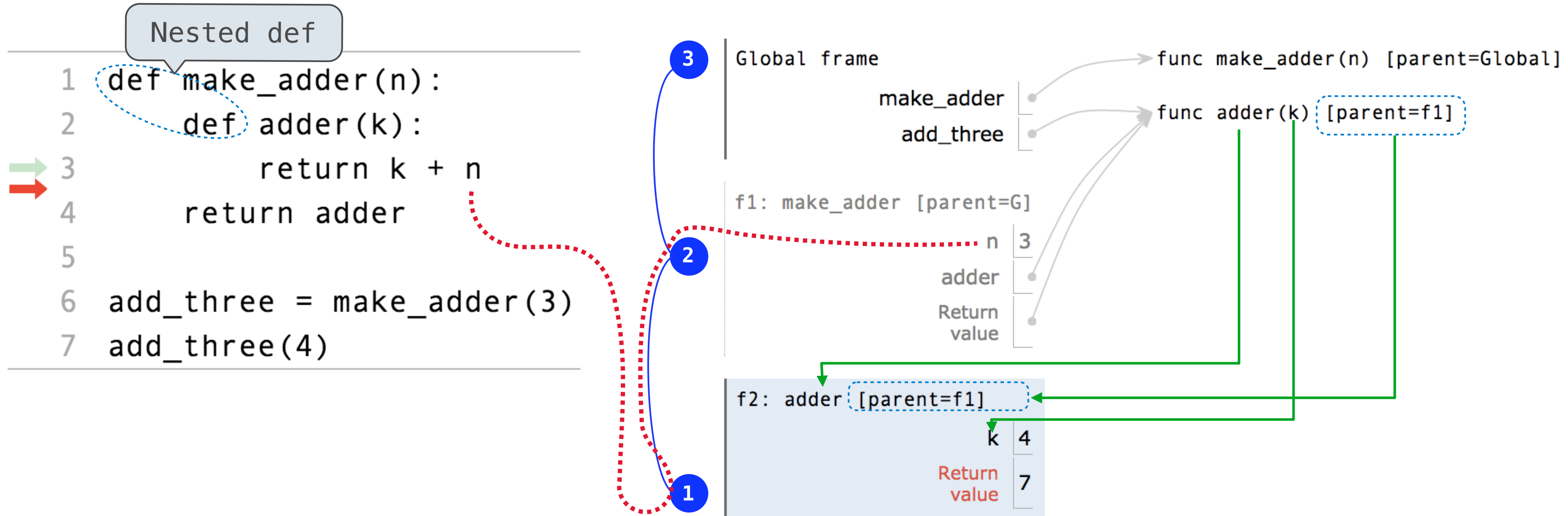
Nested def

```
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)
```



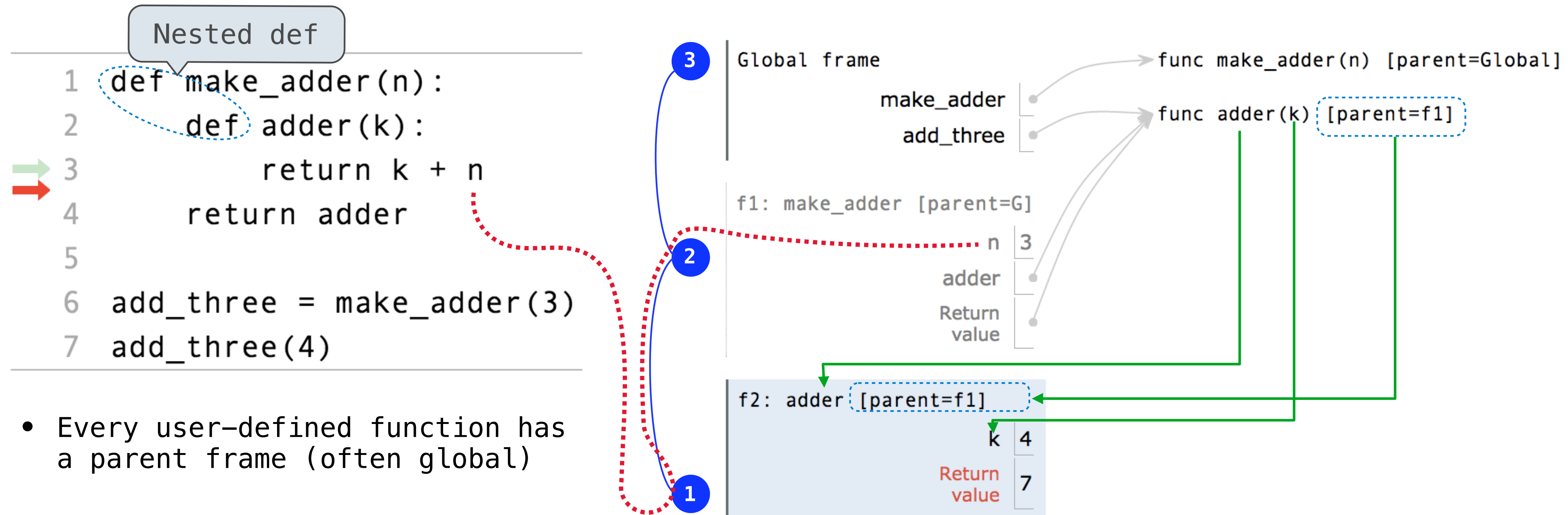
http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

Environment Diagrams for Nested Def Statements



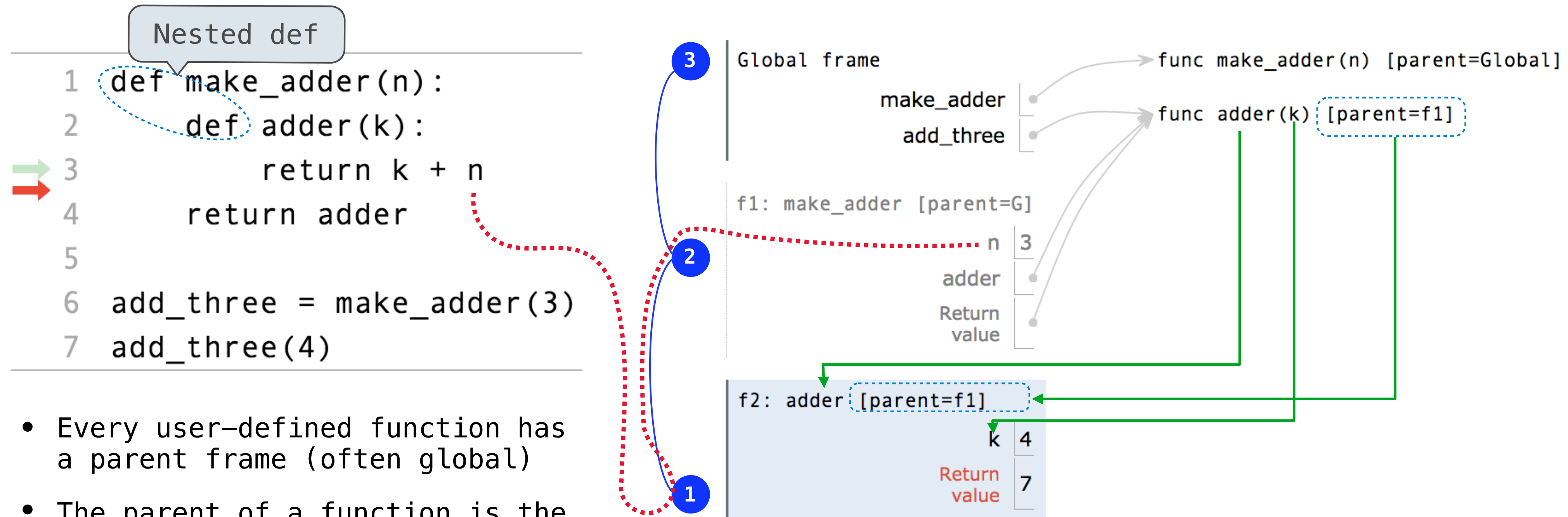
http://pythontutor.com/composingprograms.html#code=def%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Athree_more_than%20%3D%20make_adder%283%29%0Aresult%20%3D%20three_more_than%284%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&r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

Environment Diagrams for Nested Def Statements



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%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

Environment Diagrams for Nested Def Statements

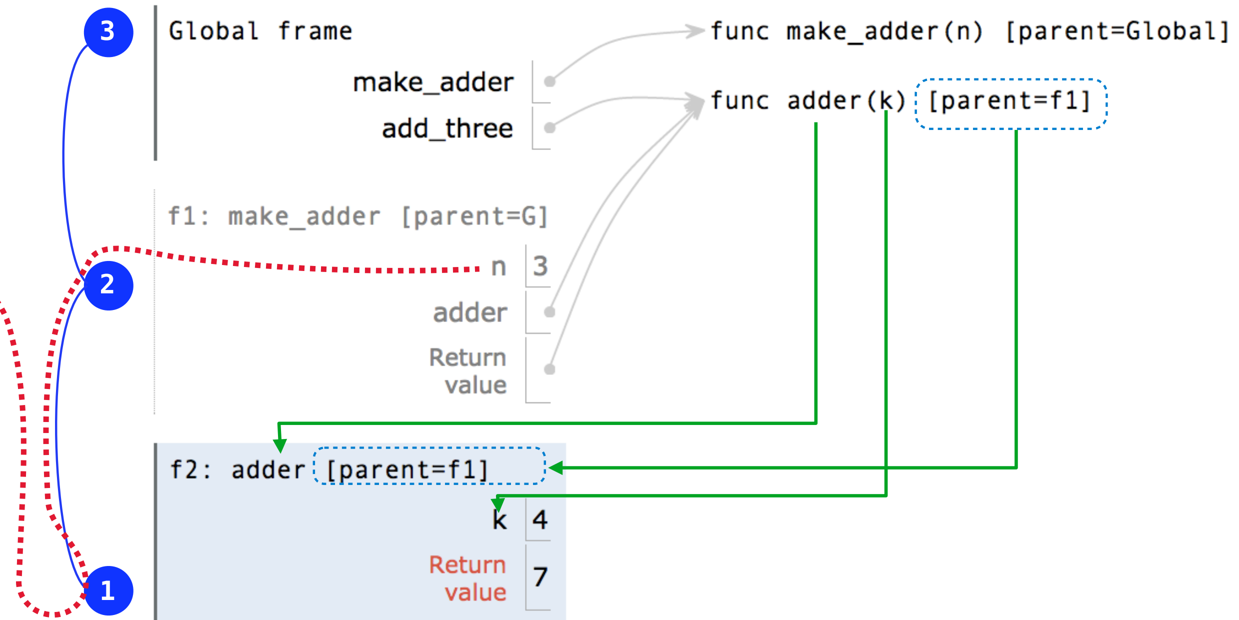


- Every user-defined function has a parent frame (often global)
- The parent of a function is the frame in which it was defined

Environment Diagrams for Nested Def Statements

Nested def

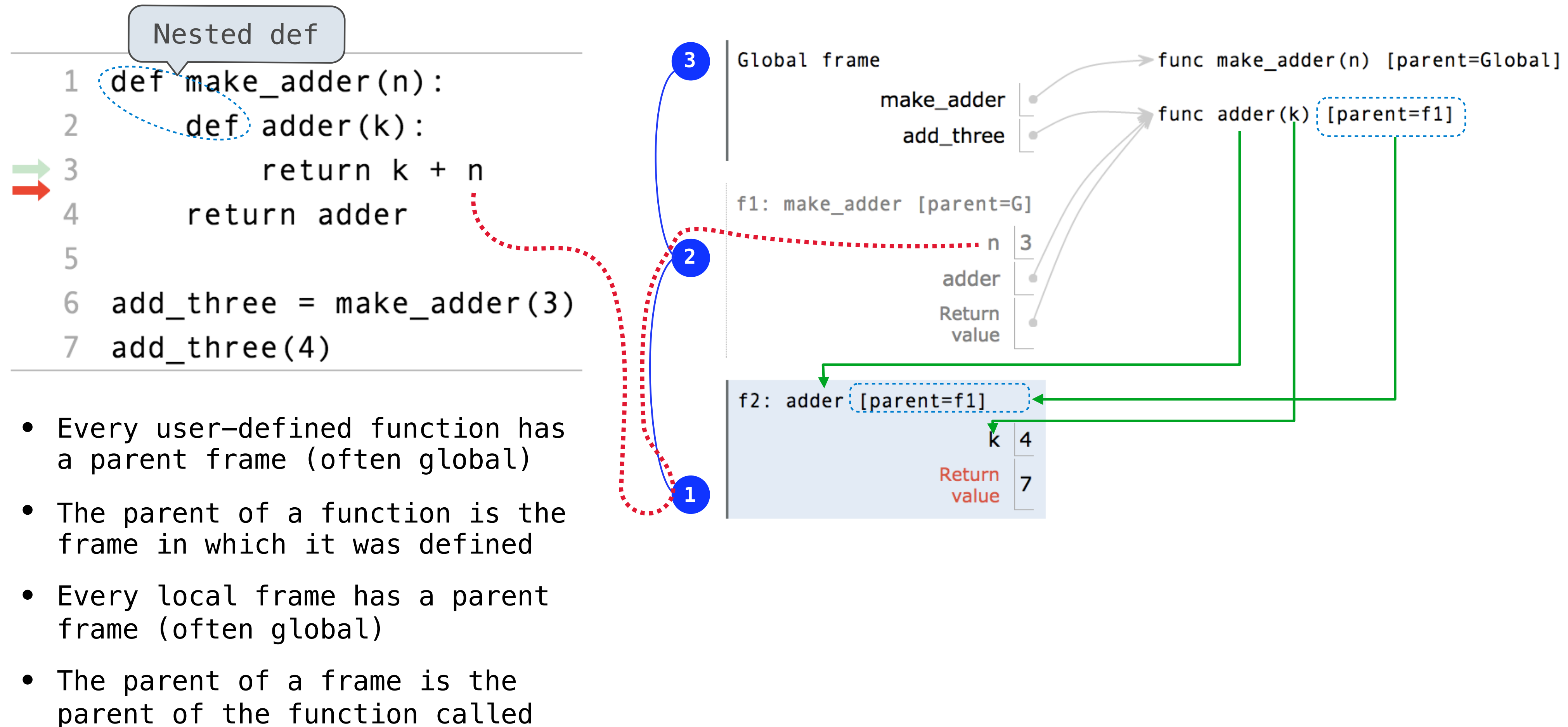
```
1  def make_adder(n):
2      def adder(k):
3          return k + n
4      return adder
5
6  add_three = make_adder(3)
7  add_three(4)
```



- 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)

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%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

Environment Diagrams for Nested Def Statements



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%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

How to Draw an Environment Diagram

When a function is defined:

How to Draw an Environment Diagram

When a function is defined:

Create a function value: func <name>(<formal parameters>) [parent=<label>]

How to Draw an Environment Diagram

When a function is defined:

Create a function value: func <name>(<formal parameters>) [parent=<label>]

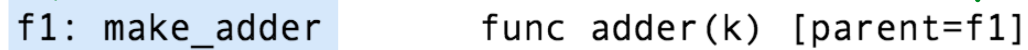
Its parent is the current frame.

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.



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

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:

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.

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>]

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.

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:

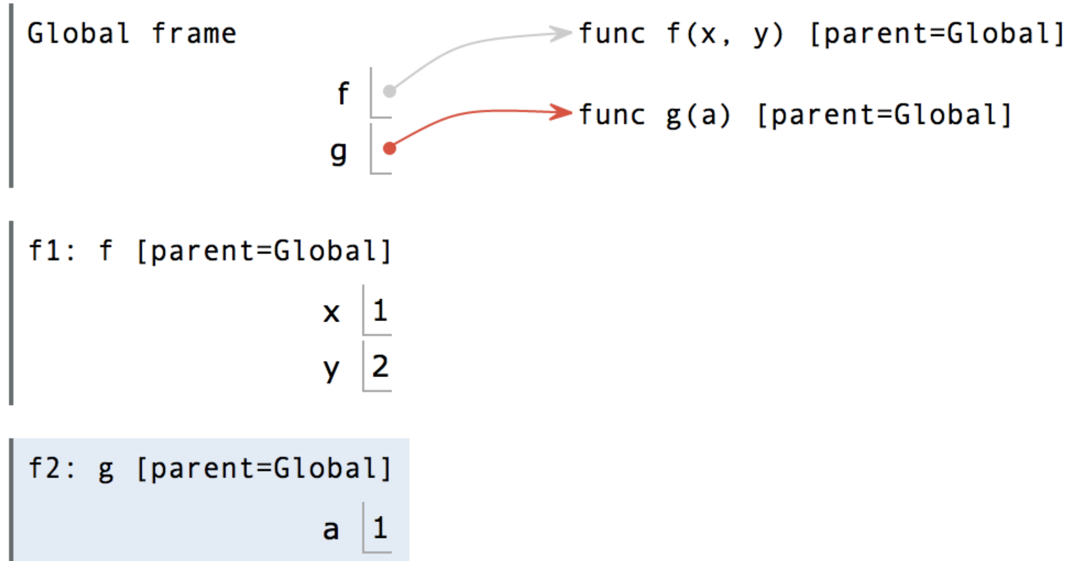
1. Add a local frame, titled with the <name> of the function being called.
2. Copy the parent of the function to the local frame: [parent=<label>]
3. Bind the <formal parameters> to the arguments in the local frame.
4. Execute the body of the function in the environment that starts with the local frame.

Local Names

(Demo)

Local Names are not Visible to Other (Non-Nested) Functions

```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```



http://pythontutor.com/composingprograms.html#code=def%20f%28x,
%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Local Names are not Visible to Other (Non-Nested) Functions

```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```

Global frame
f → func f(x, y) [parent=Global]
g → func g(a) [parent=Global]

f1: f [parent=Global]
x 1
y 2

f2: g [parent=Global]
a 1

1
2

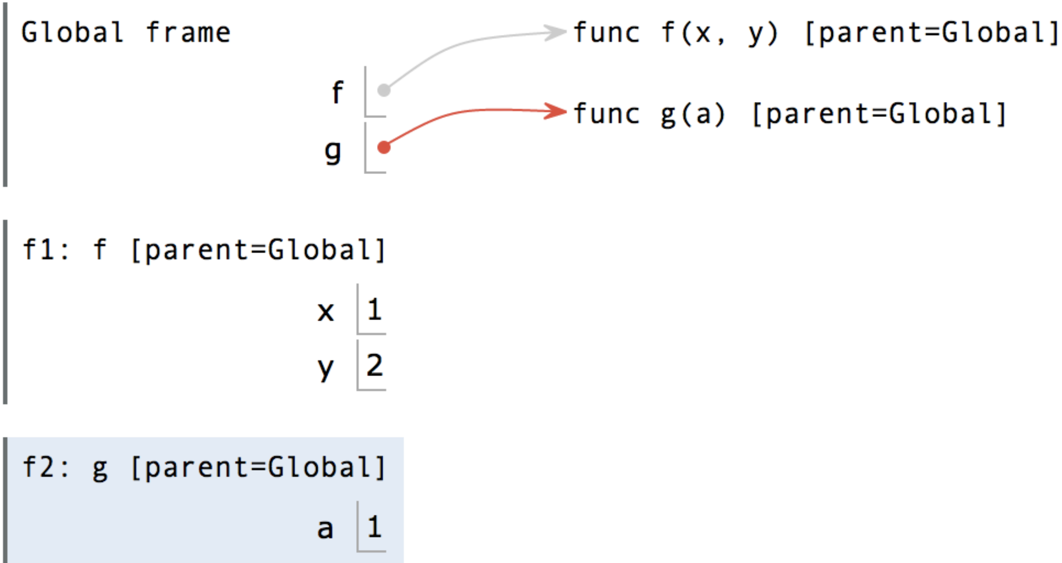
http://pythontutor.com/composingprograms.html#code=def%20f%28x,
%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Local Names are not Visible to Other (Non-Nested) Functions

```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```

2

1

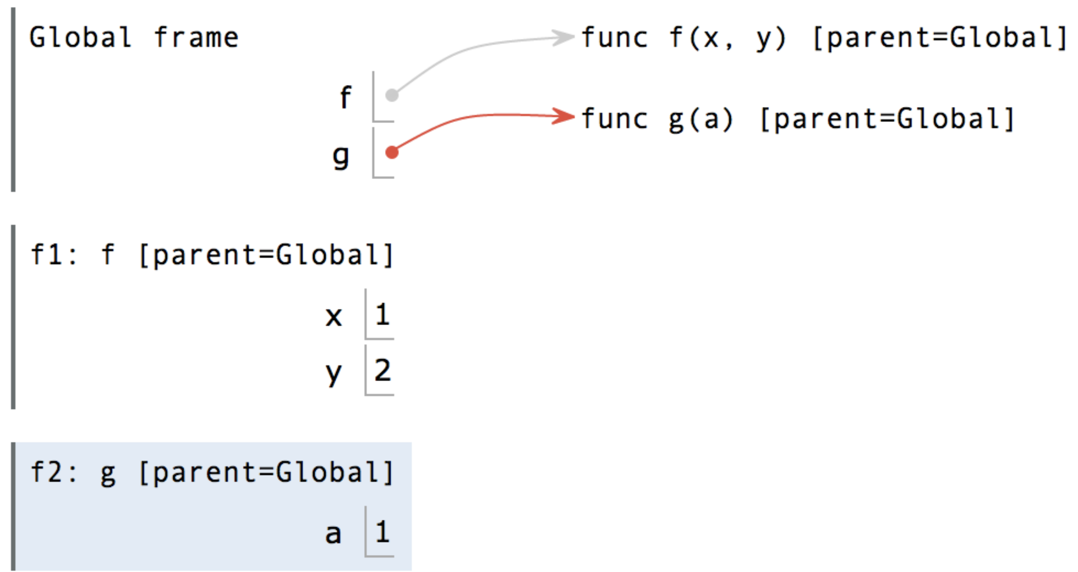


http://pythontutor.com/composingprograms.html#code=def%20f%28x,
%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Local Names are not Visible to Other (Non-Nested) Functions

```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```

“y” is not found



http://pythontutor.com/composingprograms.html#code=def%20f%28x,
%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Local Names are not Visible to Other (Non-Nested) Functions

```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```

“y” is not found, again

“y” is not found

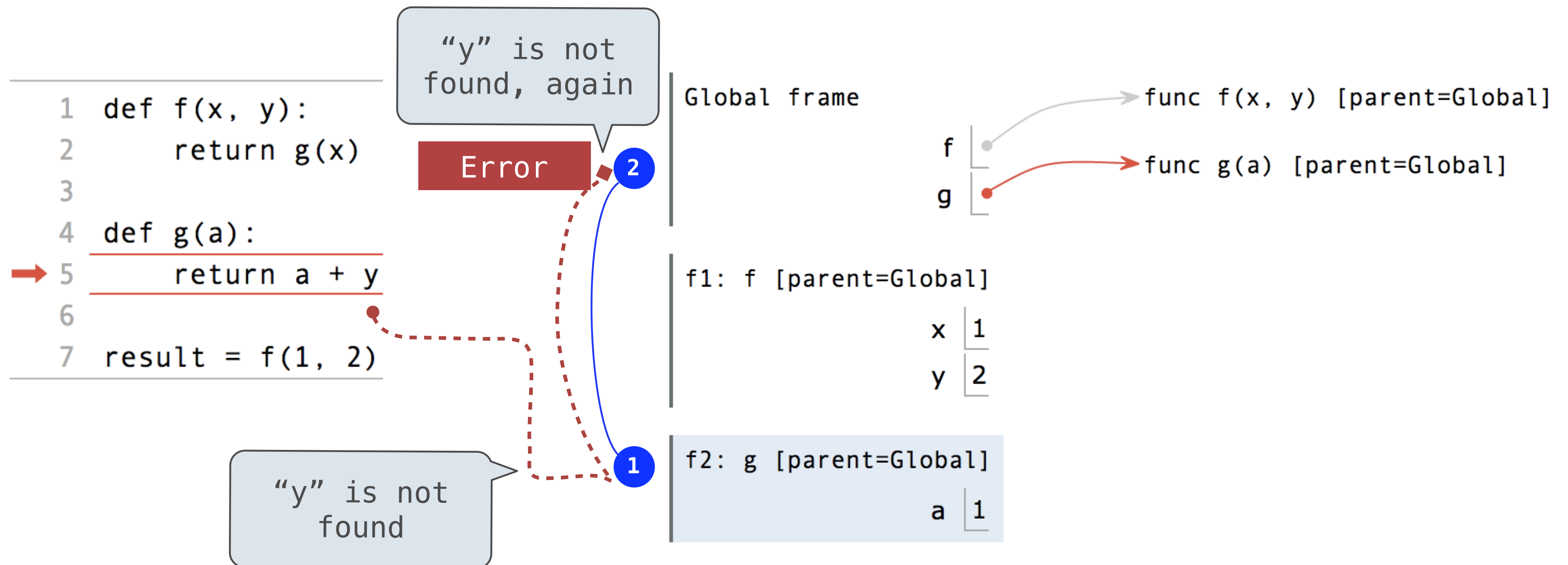
Global frame
f → func f(x, y) [parent=Global]
g → func g(a) [parent=Global]

f1: f [parent=Global]
x 1
y 2

f2: g [parent=Global]
a 1

http://pythontutor.com/composingprograms.html#code=def%20f%28x,
%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Local Names are not Visible to Other (Non-Nested) Functions



http://pythontutor.com/composingprograms.html#code=def%20f%28x,
%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Local Names are not Visible to Other (Non-Nested) Functions

```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```

"y" is not found, again

Error

Global frame

f → func f(x, y) [parent=Global]

g → func g(a) [parent=Global]

f1: f [parent=Global]

x 1

y 2

f2: g [parent=Global]

a 1

"y" is not found

1

2

- An environment is a sequence of frames.

http://pythontutor.com/composingprograms.html#code=def%20f%28x,
%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Local Names are not Visible to Other (Non-Nested) Functions

```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```

“y” is not found, again

Error

2

Global frame

f → func f(x, y) [parent=Global]

g → func g(a) [parent=Global]

f1: f [parent=Global]

x 1

y 2

f2: g [parent=Global]

a 1

1

“y” is not found

- 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.

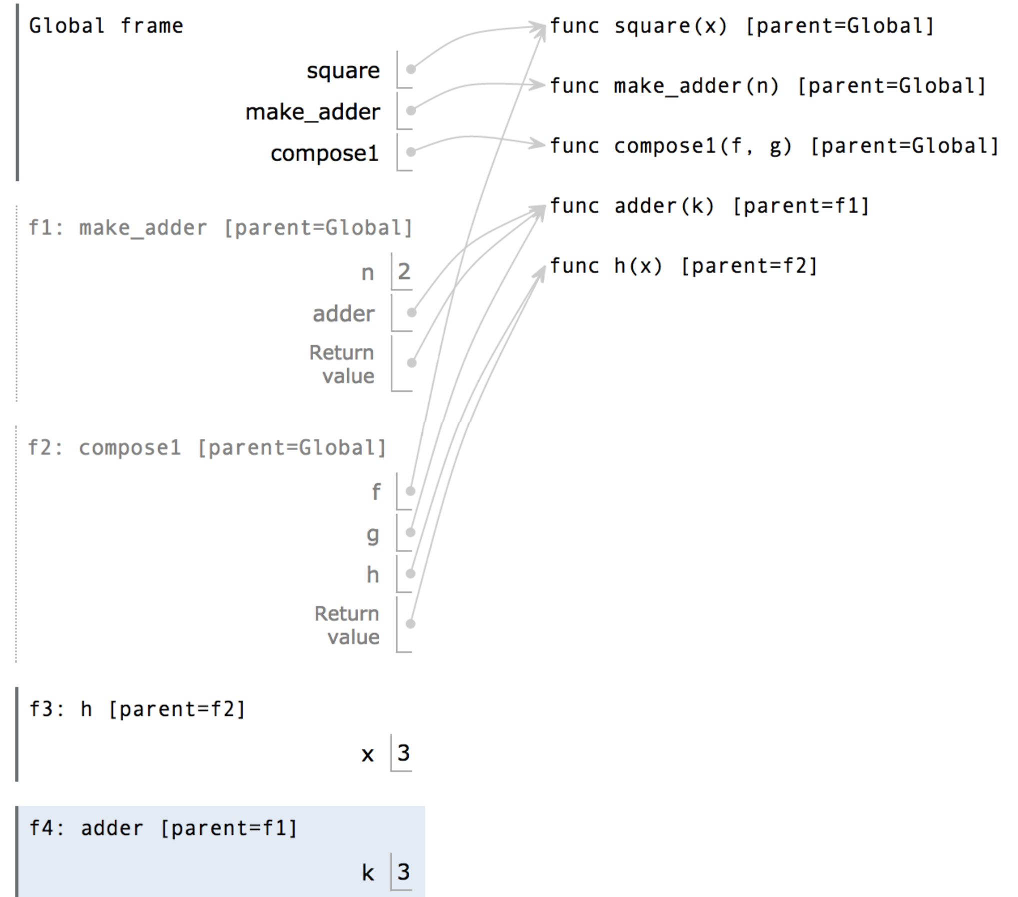
http://pythontutor.com/composingprograms.html#code=def%20f%28x,%20y%29%3A%0A%20%20%20%20return%20g%28x%29%0A%0Adef%20g%28a%29%3A%0A%20%20%20%20return%20a%20%2B%20y%0A%20%20%20%20%0Aresult%20%3D%20f%281,%202%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Function Composition

(Demo)

The Environment Diagram for Function Composition

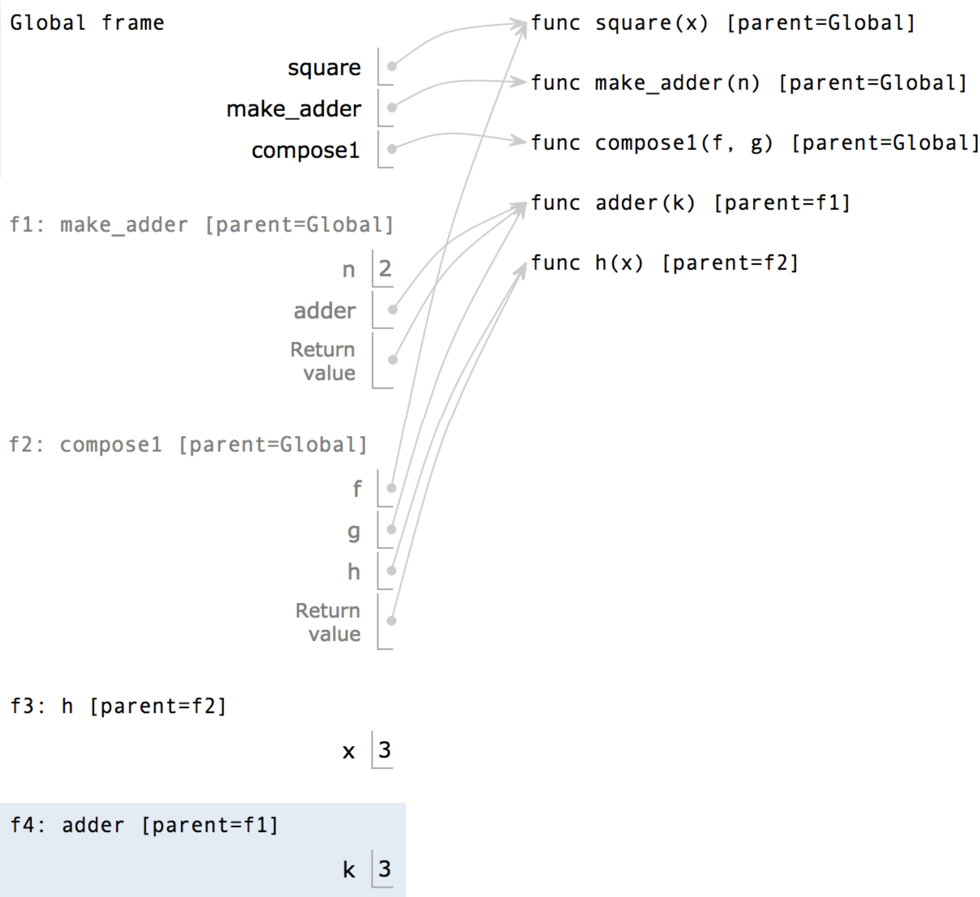
```
 1  def square(x):
 2      return x * x
 3
 4  def make_adder(n):
 5      def adder(k):
 6          return k + n
 7      return adder
 8
 9  def compose1(f, g):
10      def h(x):
11          return f(g(x))
12      return h
13
14  compose1(square, make_adder(2))(3)
```



http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%0Adef%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%20%20%20%20%0Adef%20compose1%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%20%20%20%20%0Acompose1%28square,%20make_adder%282%29%29%283%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

The Environment Diagram for Function Composition

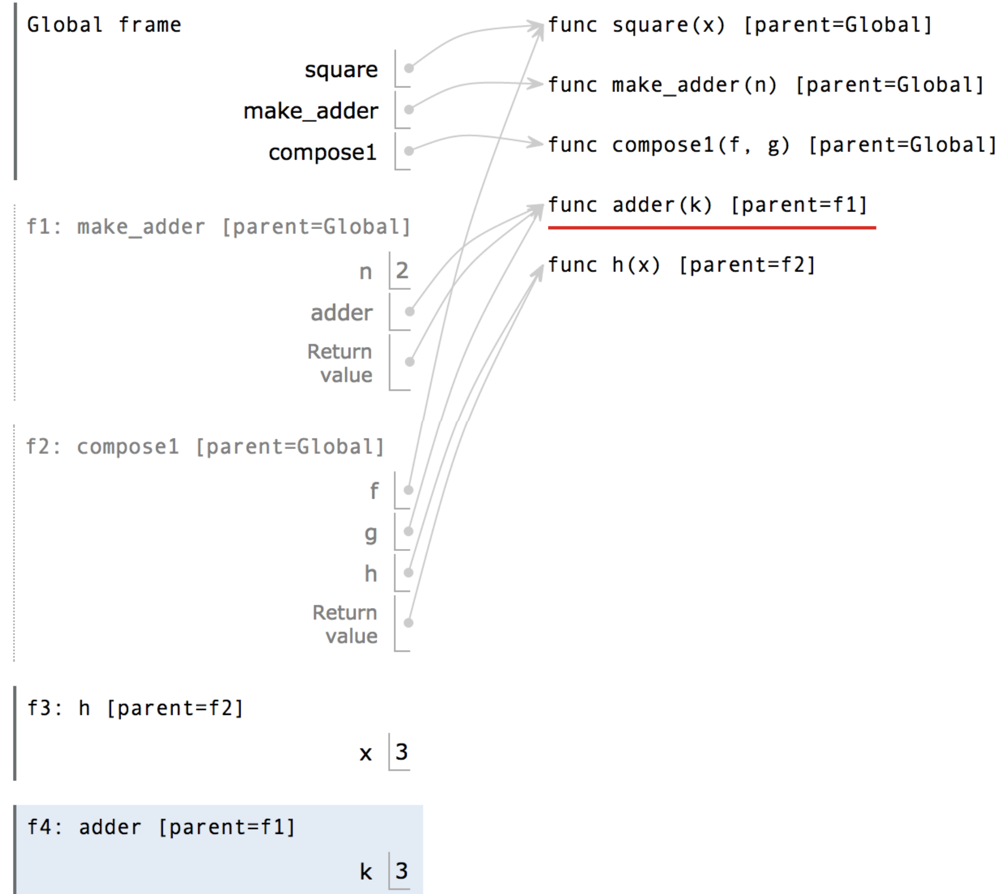
```
1  def square(x):
2      return x * x
3
4  def make_adder(n):
5      def adder(k):
6          return k + n
7      return adder
8
9  def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```



http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%0Adef%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%20%20%20%20%0Adef%20compose1%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%20%20%20%20%0Acompose1%28square,%20make_adder%282%29%29%283%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

The Environment Diagram for Function Composition

```
1  def square(x):
2      return x * x
3
4  def make_adder(n):
5      def adder(k):
6          return k + n
7      return adder
8
9  def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```



http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%0Adef%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%20%20%20%20%0Adef%20compose1%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%20%20%20%20%0Acompose1%28square,%20make_adder%282%29%29%283%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

The Environment Diagram for Function Composition

```
 1  def square(x):
 2      return x * x
 3
 4  def make_adder(n):
 5      def adder(k):
 6          return k + n
 7      return adder
 8
 9  def compose1(f, g):
10      def h(x):
11          return f(g(x))
12      return h
13
14  compose1(square, make_adder(2))(3)
```

Global frame
square
make_adder
compose1

func square(x) [parent=Global]
func make_adder(n) [parent=Global]
func compose1(f, g) [parent=Global]
func adder(k) [parent=f1]
func h(x) [parent=f2]

f1: make_adder [parent=Global]
n 2
adder
Return value

f2: compose1 [parent=Global]
f
g
h
Return value

f3: h [parent=f2]
x 3

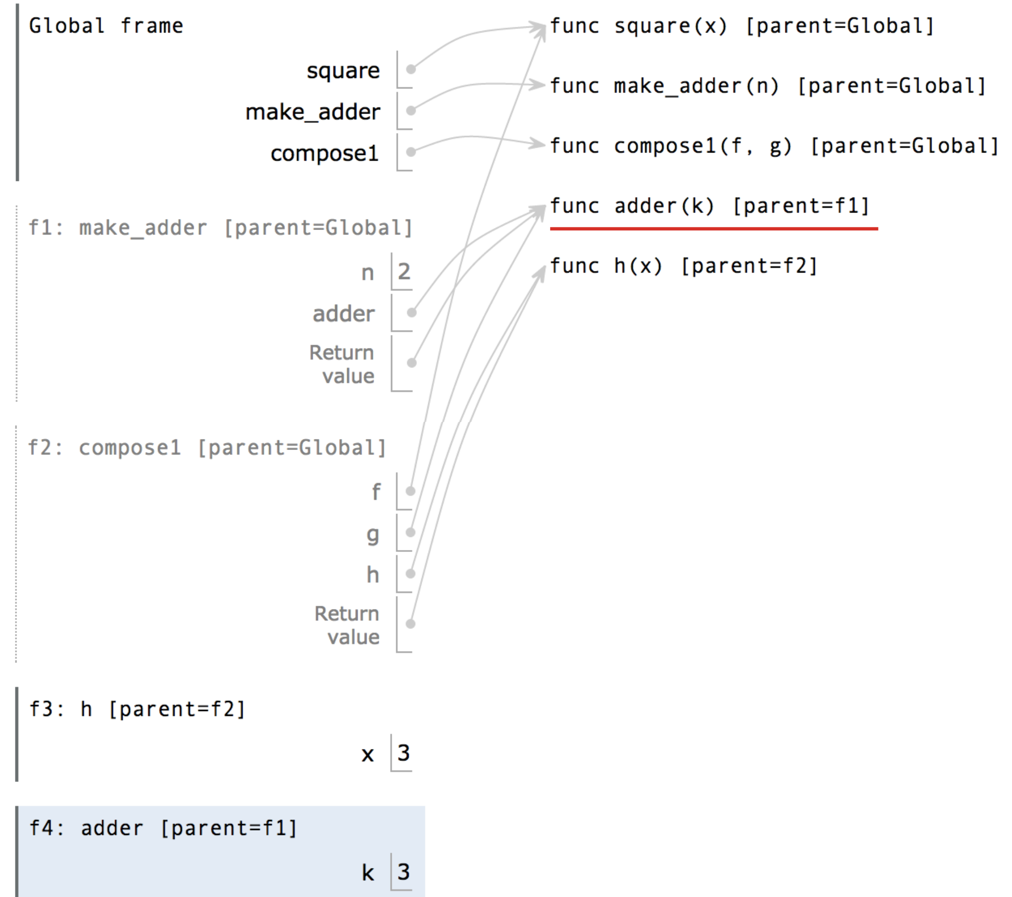
f4: adder [parent=f1]
k 3

http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%0Adef%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%20%20%20%20%0Adef%20compose1%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%20%20%20%20%0Acompose1%28square,%20make_adder%282%29%29%283%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

The Environment Diagram for Function Composition

```
1  def square(x):
2      return x * x
3
4  def make_adder(n):
5      def adder(k):
6          return k + n
7      return adder
8
9  def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```

Return value of make_adder is an argument to compose1

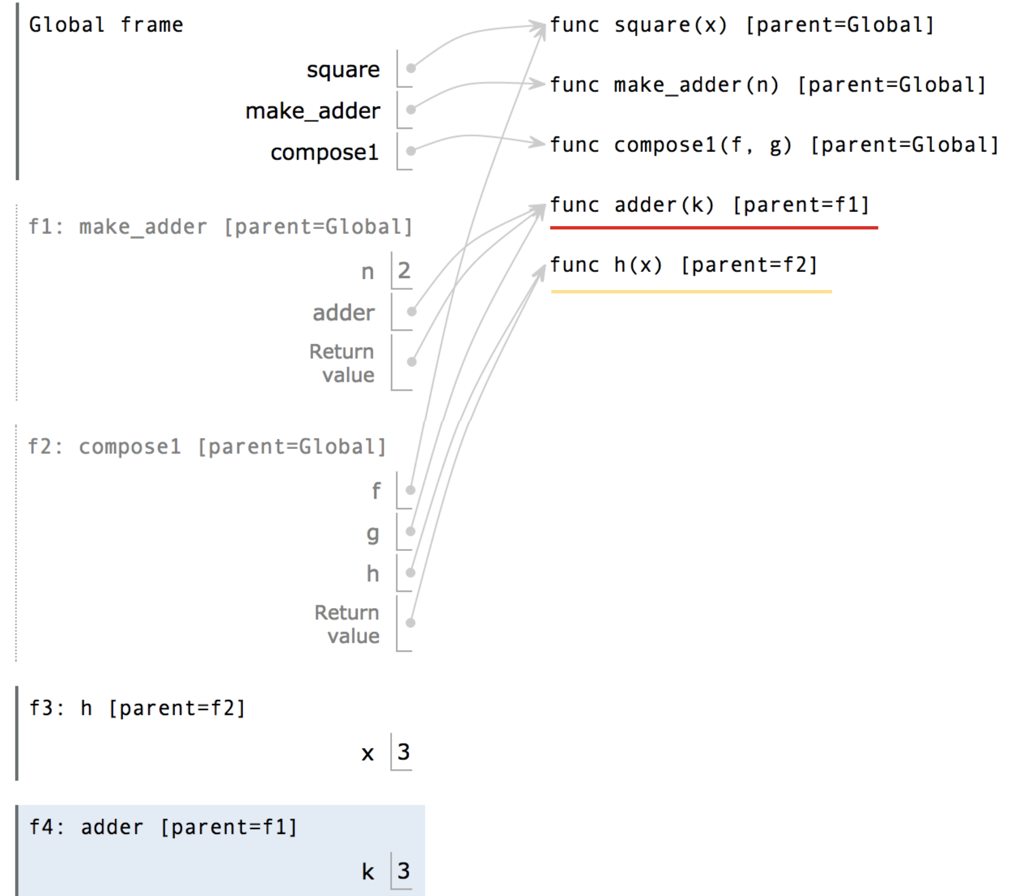


http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%0Adef%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Adef%20compose1%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%20%20%20%20%0Acompose1%28square,%20make_adder%282%29%29%283%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

The Environment Diagram for Function Composition

```
 1  def square(x):
 2      return x * x
 3
 4  def make_adder(n):
 5      def adder(k):
 6          return k + n
 7      return adder
 8
 9  def compose1(f, g):
10      def h(x):
11          return f(g(x))
12      return h
13
14  compose1(square, make_adder(2))(3)
```

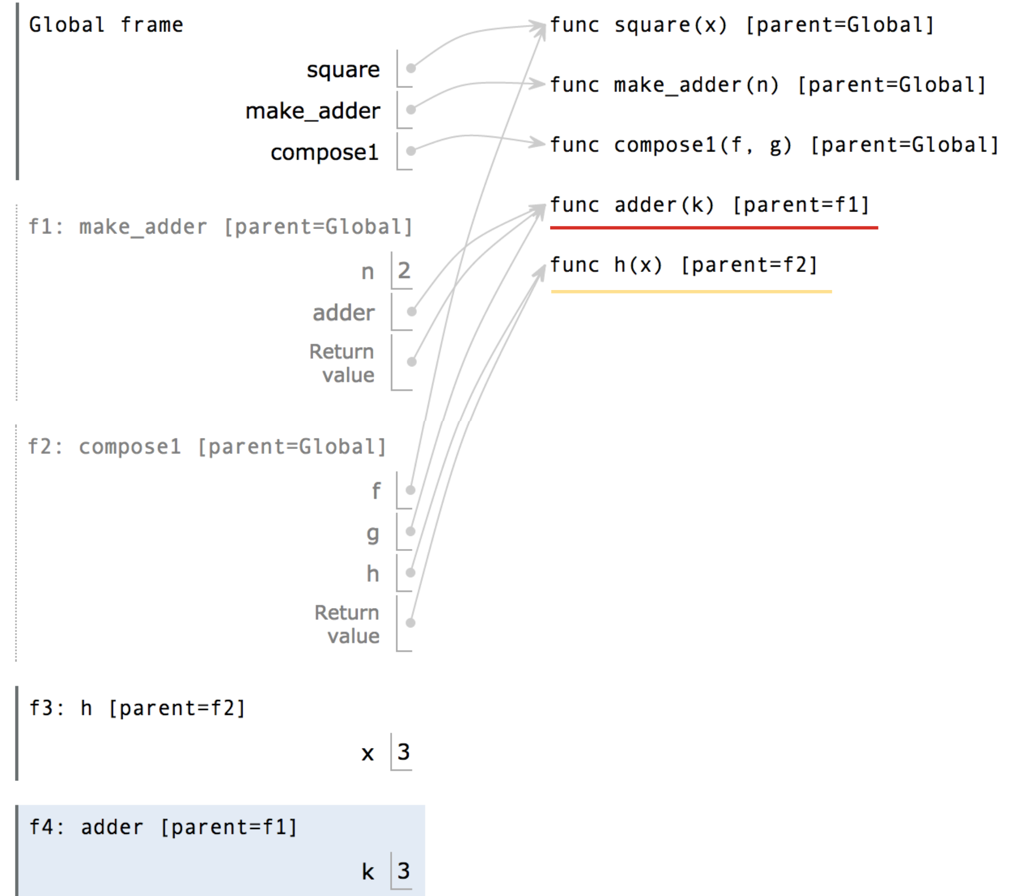
Return value of make_adder is an argument to compose1



The Environment Diagram for Function Composition

```
1  def square(x):
2      return x * x
3
4  def make_adder(n):
5      def adder(k):
6          return k + n
7      return adder
8
9  def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```

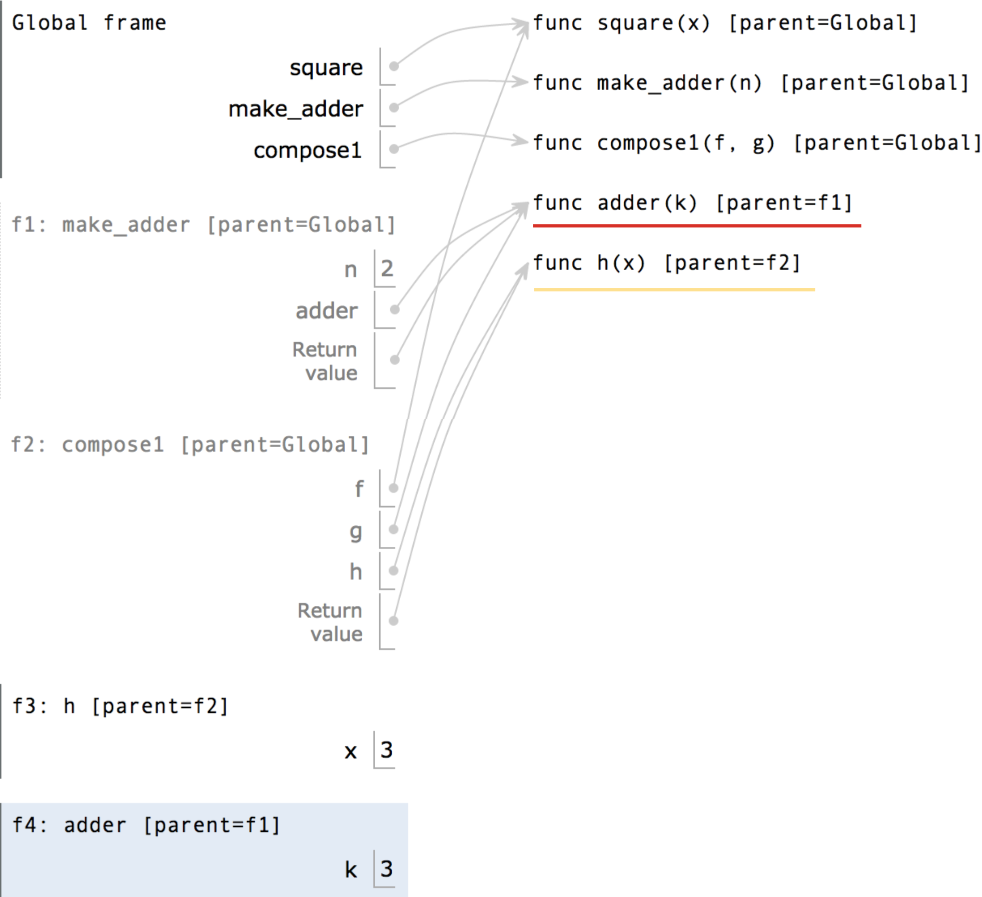
Return value of make_adder is an argument to compose1



The Environment Diagram for Function Composition

```
1  def square(x):
2      return x * x
3
4  def make_adder(n):
5      def adder(k):
6          return k + n
7      return adder
8
9  def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```

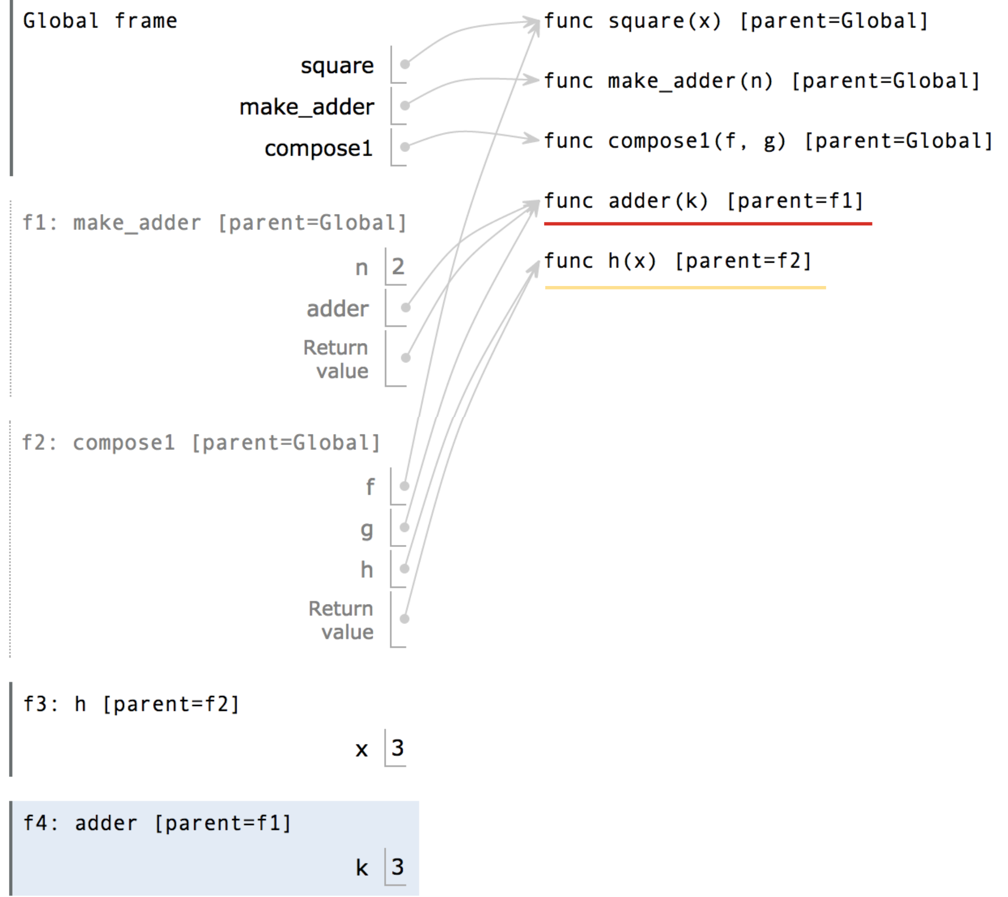
Return value of make_adder is an argument to compose1



The Environment Diagram for Function Composition

```
 1  def square(x):
 2      return x * x
 3
 4  def make_adder(n):
 5      def adder(k):
 6          return k + n
 7      return adder
 8
 9  def compose1(f, g):
10      def h(x):
11          return f(g(x))
12      return h
13
14  compose1(square, make_adder(2))(3)
```

Return value of make_adder is an argument to compose1

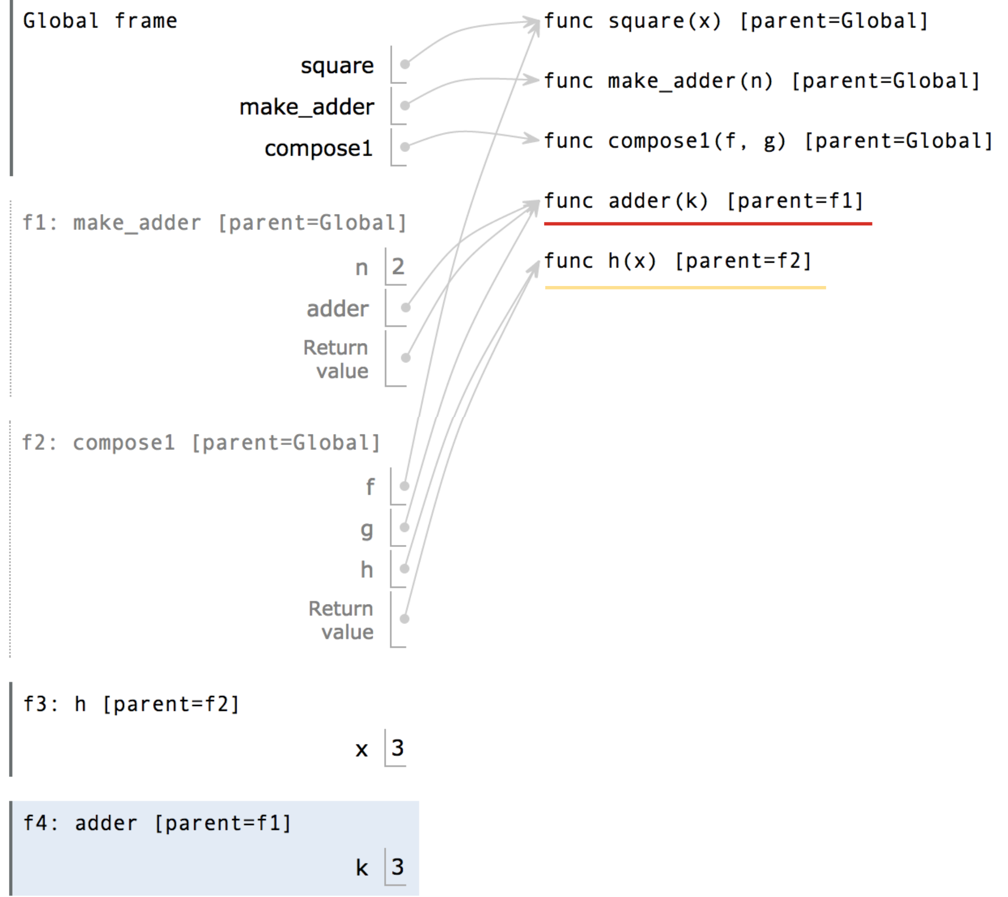


http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%0Adef%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Adef%20compose1%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%20%20%20%20%0Acompose1%28square,%20make_adder%282%29%29%283%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

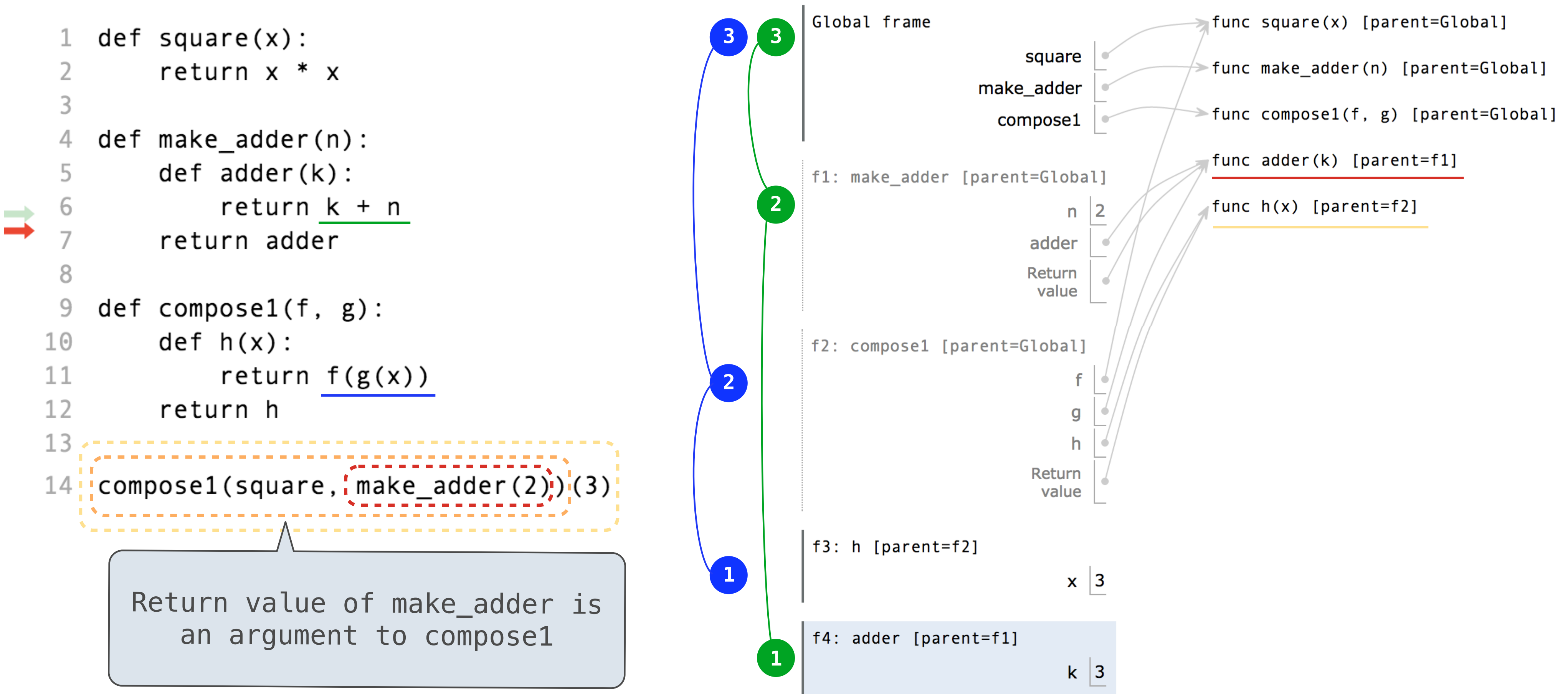
The Environment Diagram for Function Composition

```
 1 def square(x):
 2     return x * x
 3
 4 def make_adder(n):
 5     def adder(k):
 6         return k + n
 7     return adder
 8
 9 def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```

Return value of make_adder is an argument to compose1



The Environment Diagram for Function Composition



http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%0Adef%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%20%20%20%20%0Adef%20compose1%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%20%20%20%20%0Acompose1%28square,%20make_adder%282%29%29%283%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Self-Reference

(Demo)

Returning a Function Using Its Own Name

```
1  def print_sums(n):
2      print(n)
3      def next_sum(k):
4          return print_sums(n+k)
5      return next_sum
6
7  print_sums(1)(3)(5)
```

Frames

Objects

Global frame

print_sums

func print_sums(n) [parent=Global]

func next_sum(k) [parent=f1]

func next_sum(k) [parent=f3]

func next_sum(k) [parent=f5]

f1: print_sums [parent=Global]

n 1

next_sum

Return value

f2: next_sum [parent=f1]

k 3

Return value

f3: print_sums [parent=Global]

n 4

next_sum

Return value

f4: next_sum [parent=f3]

k 5

f5: print_sums [parent=Global]

n 9

next_sum

Return value

http://pythontutor.com/composingprograms.html#code=def%20print_all%28k%29%3A%0A%20%20%20%20print%28k%29%0A%20%20%20%20return%20print_all%0A%20%20%20%20%0Aprint_all%281%29%283%29%285%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

http://pythontutor.com/composingprograms.html#code=def%20print_sums%28n%29%3A%0A%20%20%20%20print%28n%29%0A%20%20%20%20def%20next_sum%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20print_sums%28n%2Bk%29%0A%20%20%20%20return%20next_sum%0A%0Aprint_sums%281%29%283%29%285%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D