61A Lecture 5

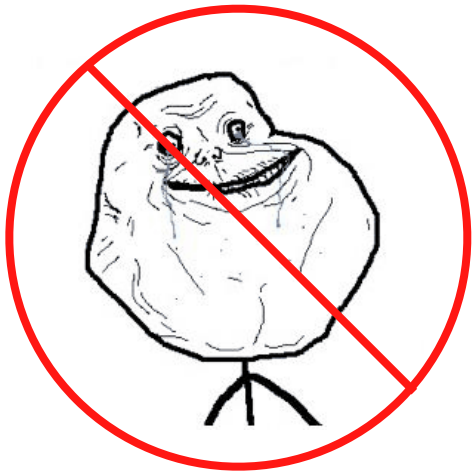
Wednesday, September 5

Office Hours: You Should Go!

You are not alone!

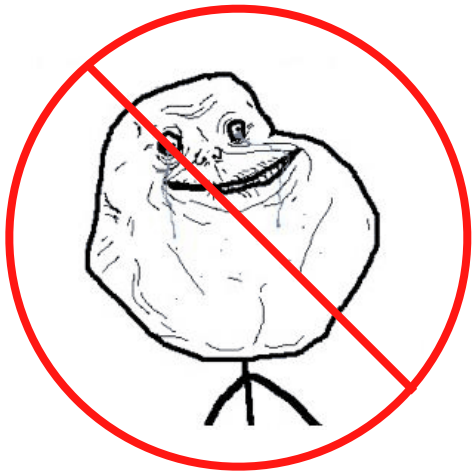
Office Hours: You Should Go!

You are not alone!



Office Hours: You Should Go!

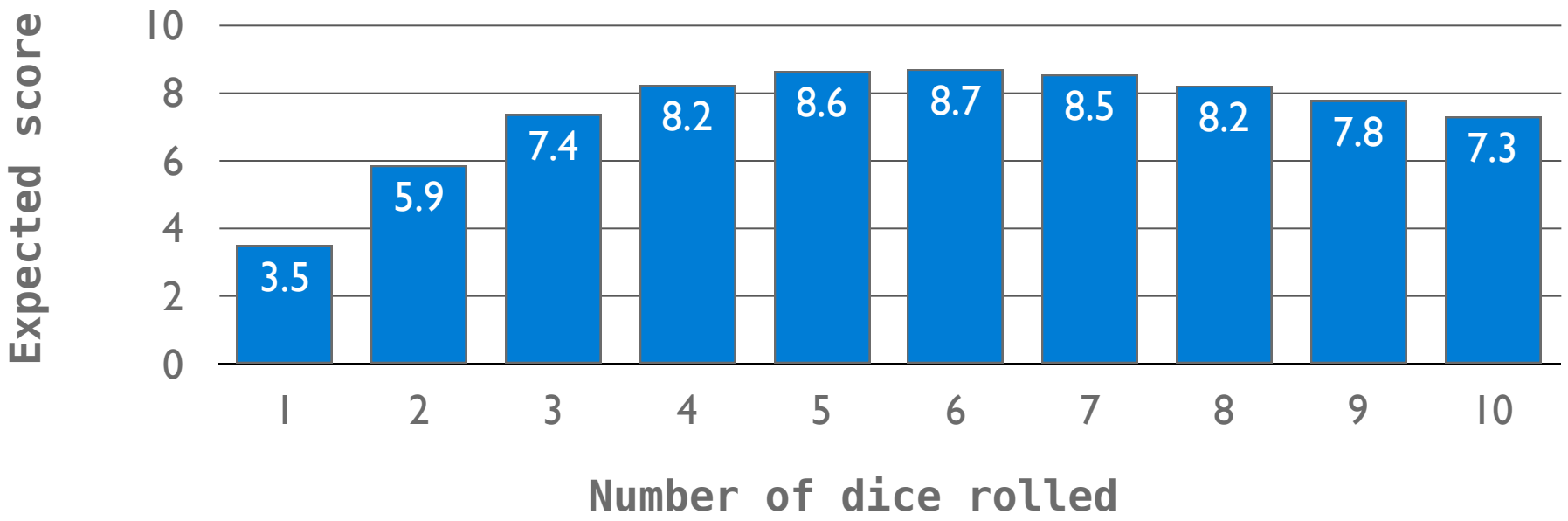
You are not alone!



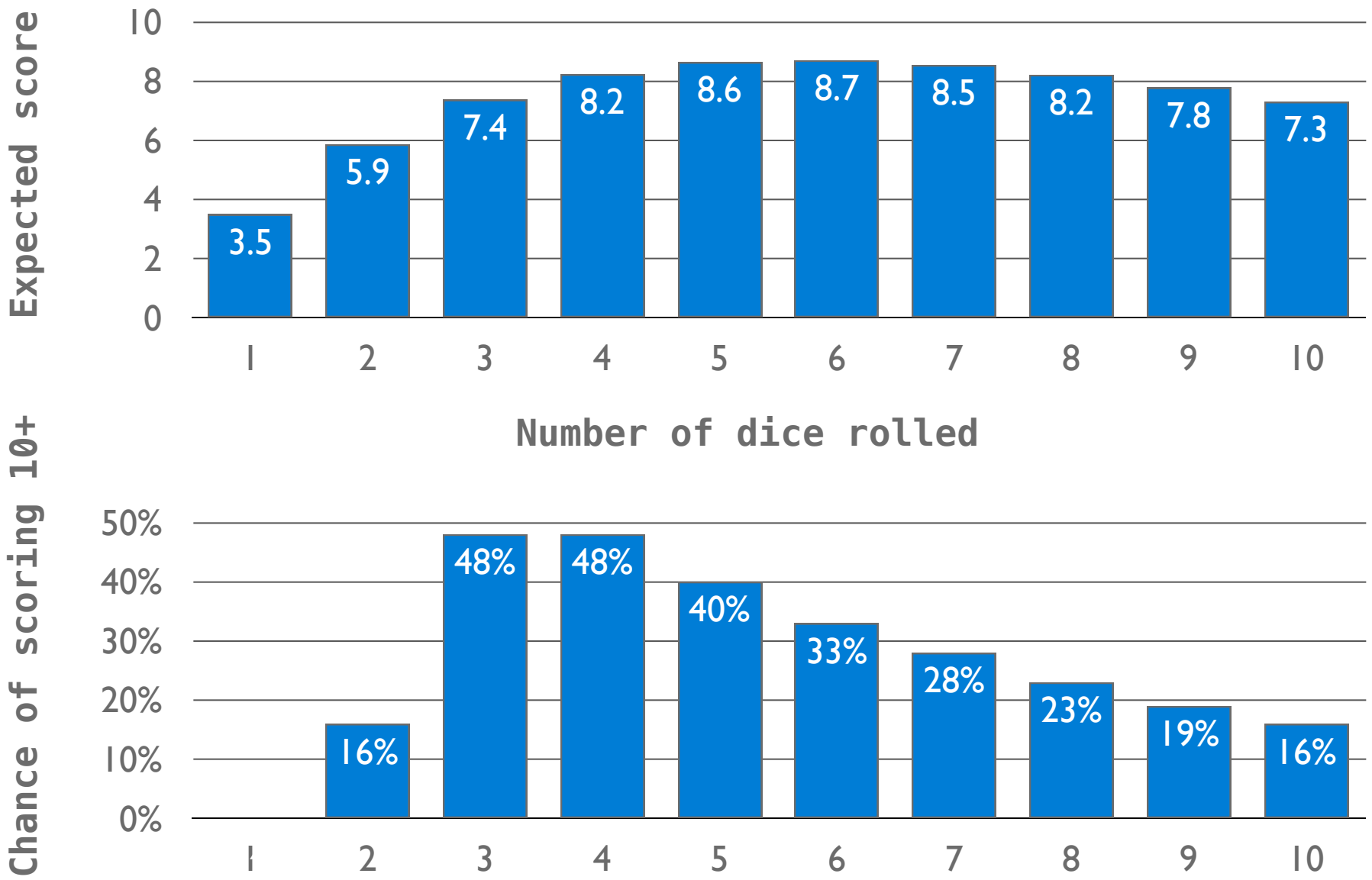
http://inst.eecs.berkeley.edu/~cs61a/fa12/staff.html

The Game of Hog

The Game of Hog



The Game of Hog



Environments Enable Higher-Order Functions

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

Environments Enable Higher-Order Functions

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

Functions as arguments:

Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current environment model handles that already!

Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current environment model handles that already!

We'll discuss an example today

Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current environment model handles that already!

We'll discuss an example today

Functions as return values:

Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current environment model handles that already!

We'll discuss an example today

Functions as return values:

We need to extend our model a little

Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current environment model handles that already!

We'll discuss an example today

Functions as return values:

We need to extend our model a little

Functions need to know where they were defined

Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current environment model handles that already!

We'll discuss an example today

Functions as return values:

We need to extend our model a little

Functions need to know where they were defined

Almost everything stays the same

Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current environment model handles that already!

We'll discuss an example today

Functions as return values:

We need to extend our model a little

Functions need to know where they were defined

Almost everything stays the same

(demo)

Names Bound to Functional Arguments

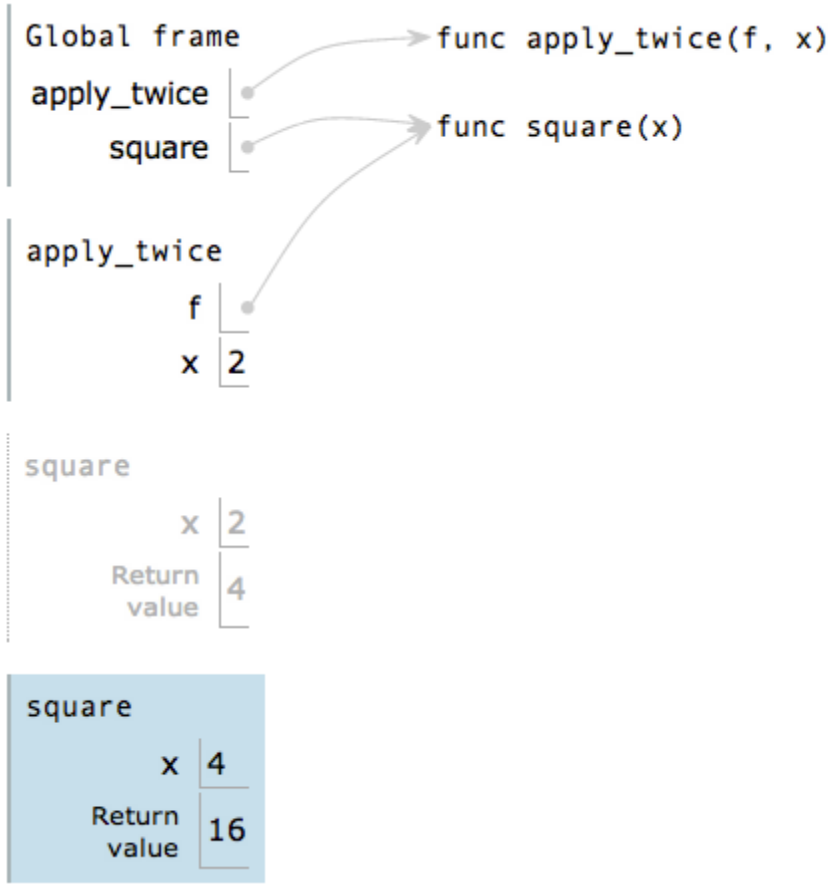
Names Bound to Functional Arguments

```
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)
```

Example: http://goo.gl/Gbtc5

Names Bound to Functional Arguments

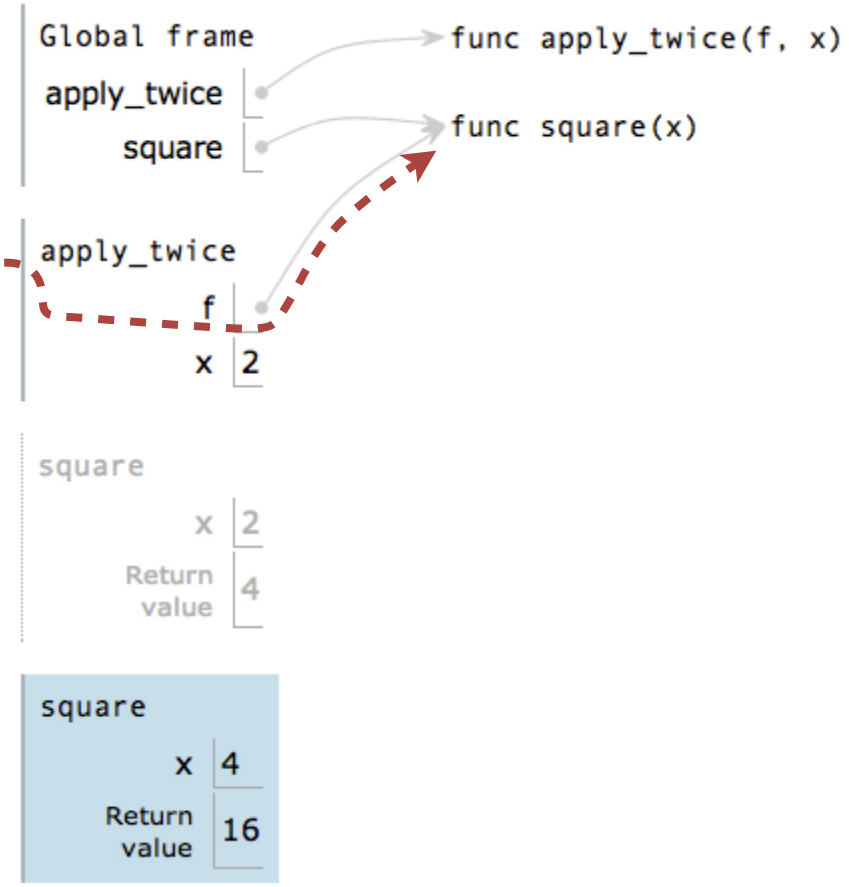
```
1 def apply_twice(f, x):
2     return f(f(x))
3
4 def square(x):
5     return x * x
6
7 result = apply_twice(square, 2)
```



Example: http://goo.gl/Gbtc5

Names 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)
```



Example: http://goo.gl/Gbtc5

Non-Nested Functions Calls Have One Local Frame

Example: http://goo.gl/tgT5H

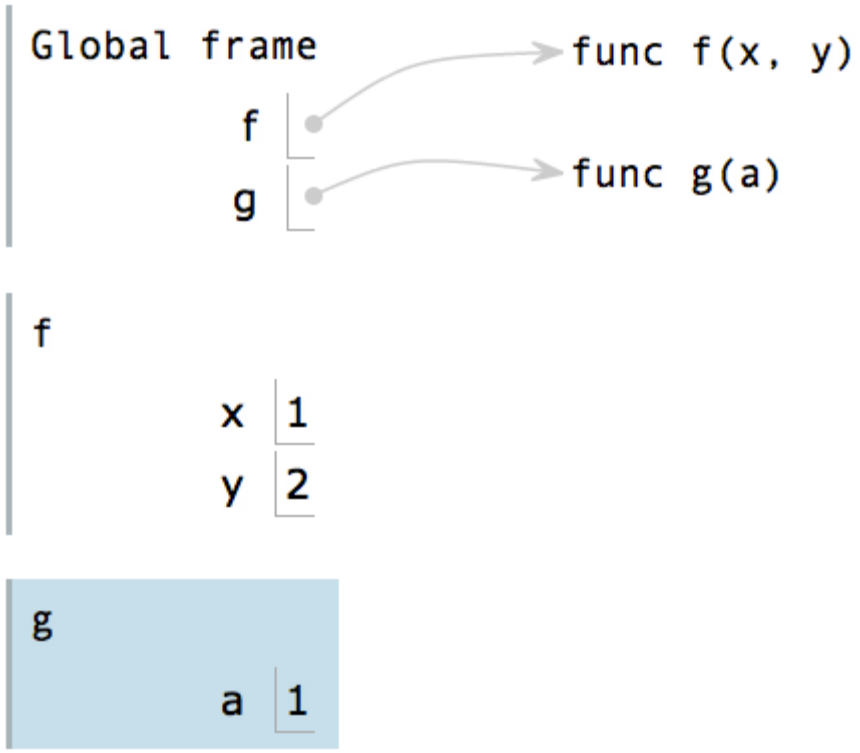
Non-Nested Functions Calls Have One Local Frame

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

Example: http://goo.gl/tgT5H

Non-Nested Functions Calls Have One Local Frame

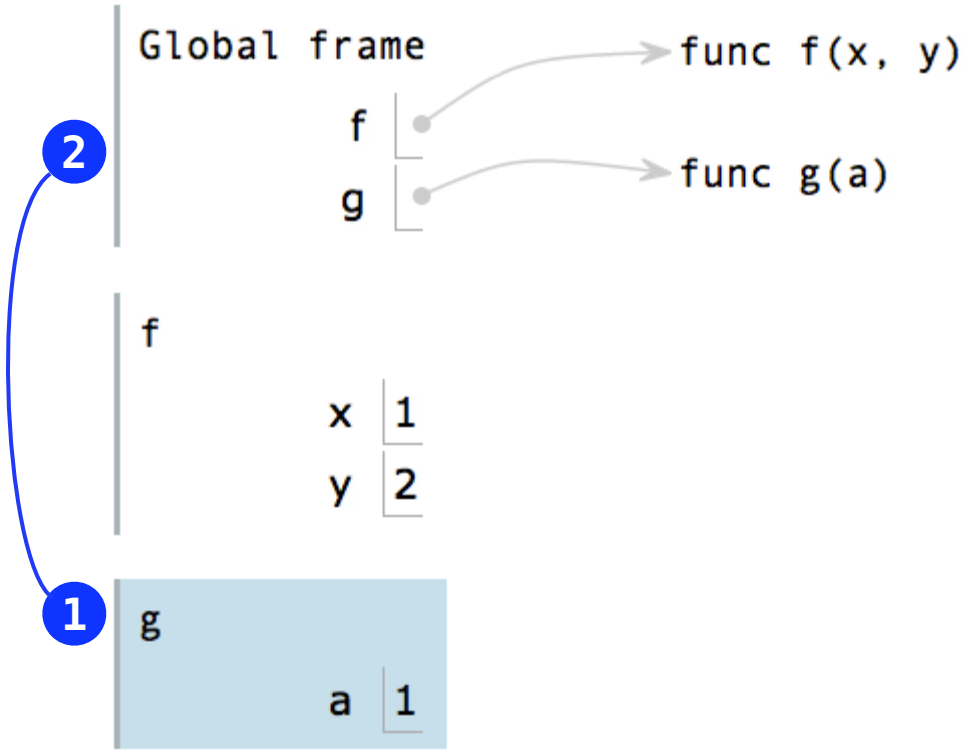
```
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```



Example: http://goo.gl/tgT5H

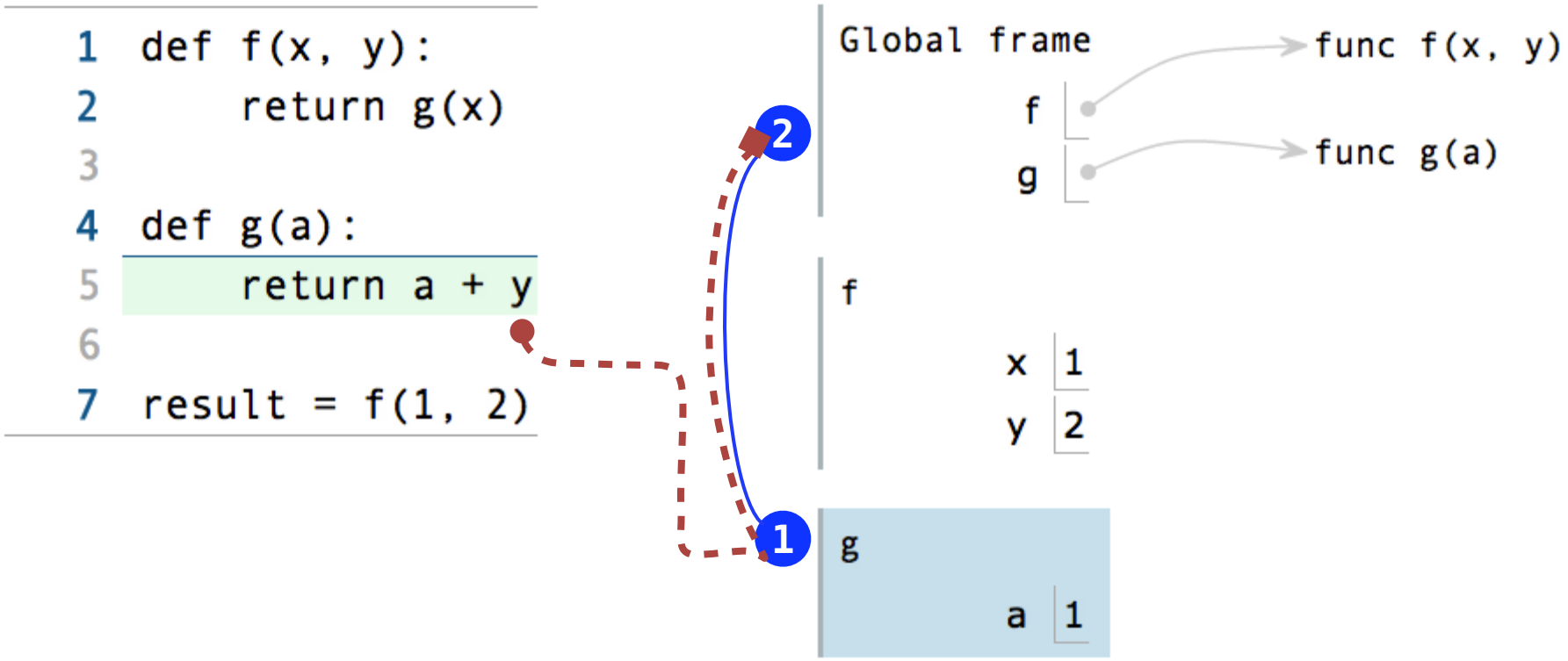
Non-Nested Functions Calls Have One Local Frame

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



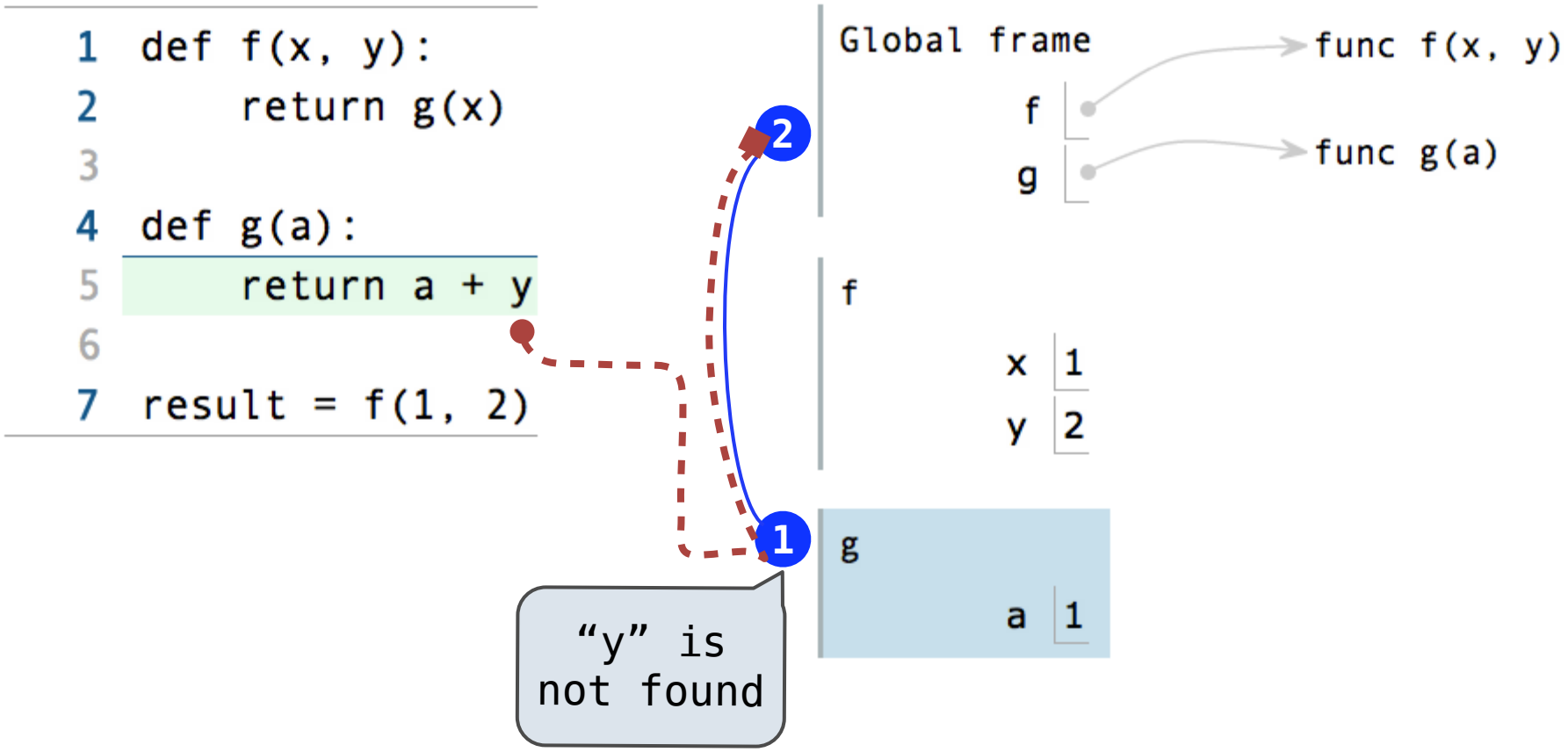
Example: http://goo.gl/tgT5H

Non-Nested Functions Calls Have One Local Frame



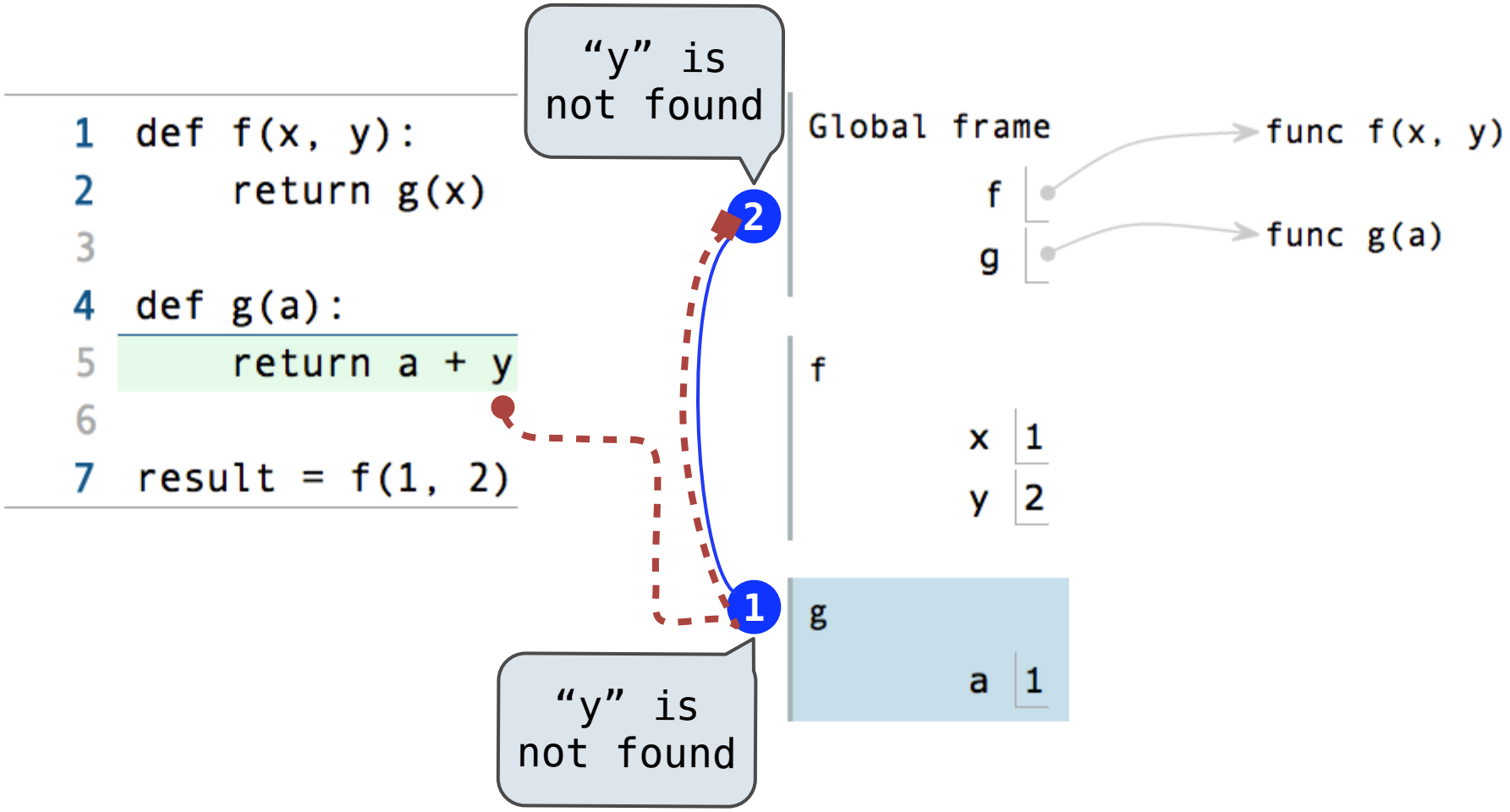
Example: http://goo.gl/tgT5H

Non-Nested Functions Calls Have One Local Frame



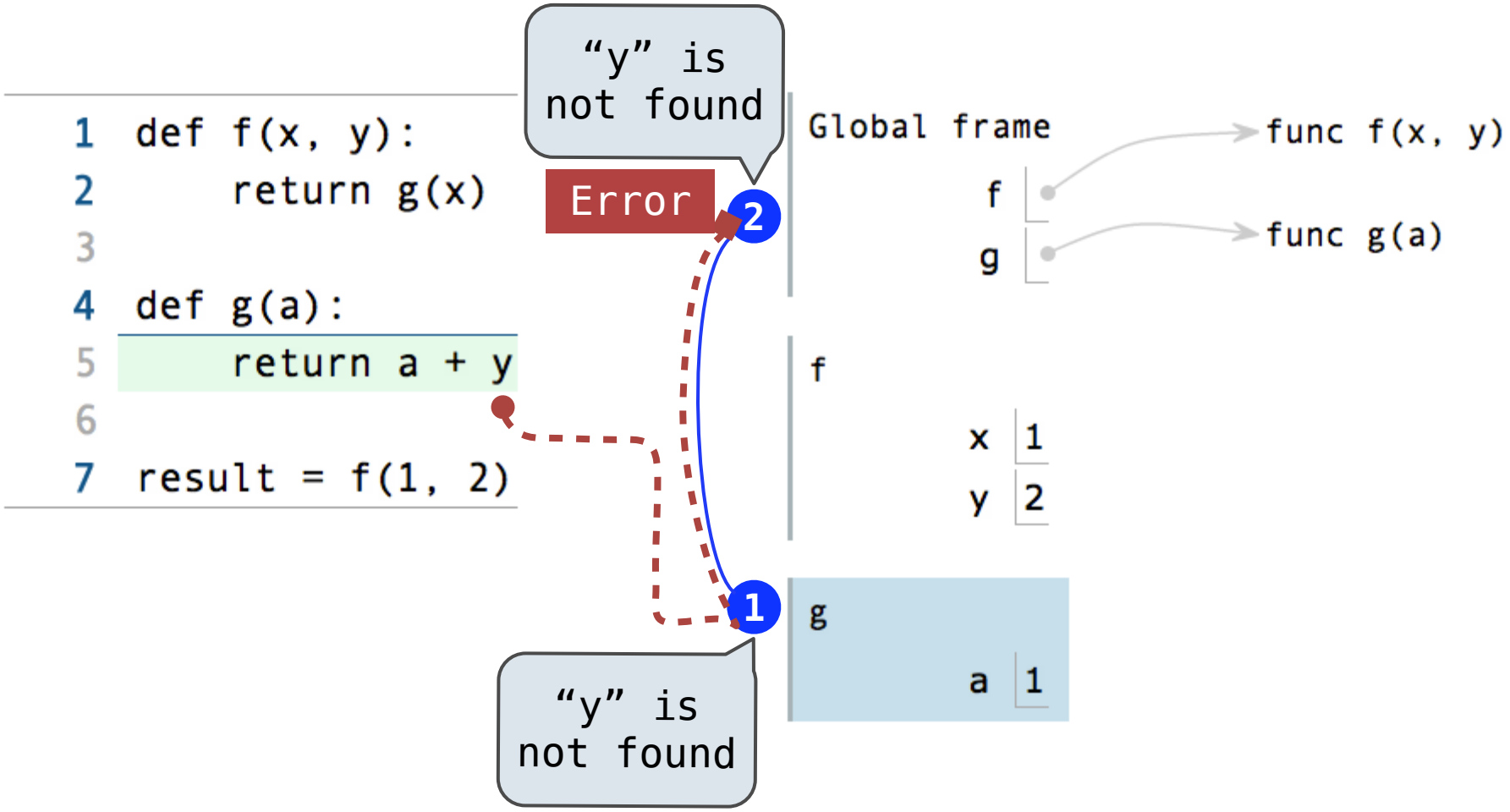
Example: http://goo.gl/tgT5H

Non-Nested Functions Calls Have One Local Frame



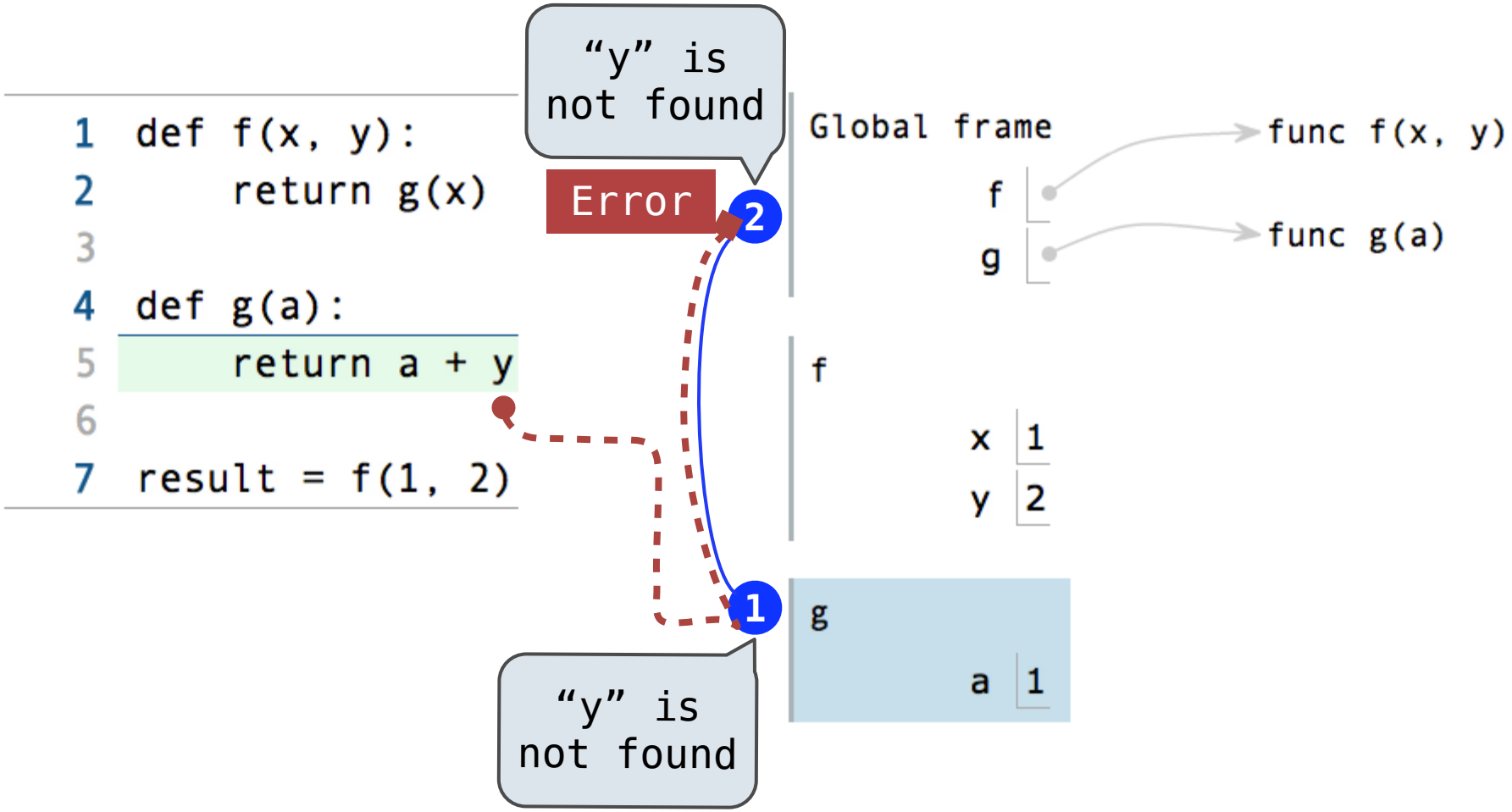
Example: http://goo.gl/tgT5H

Non-Nested Functions Calls Have One Local Frame



Example: http://goo.gl/tgT5H

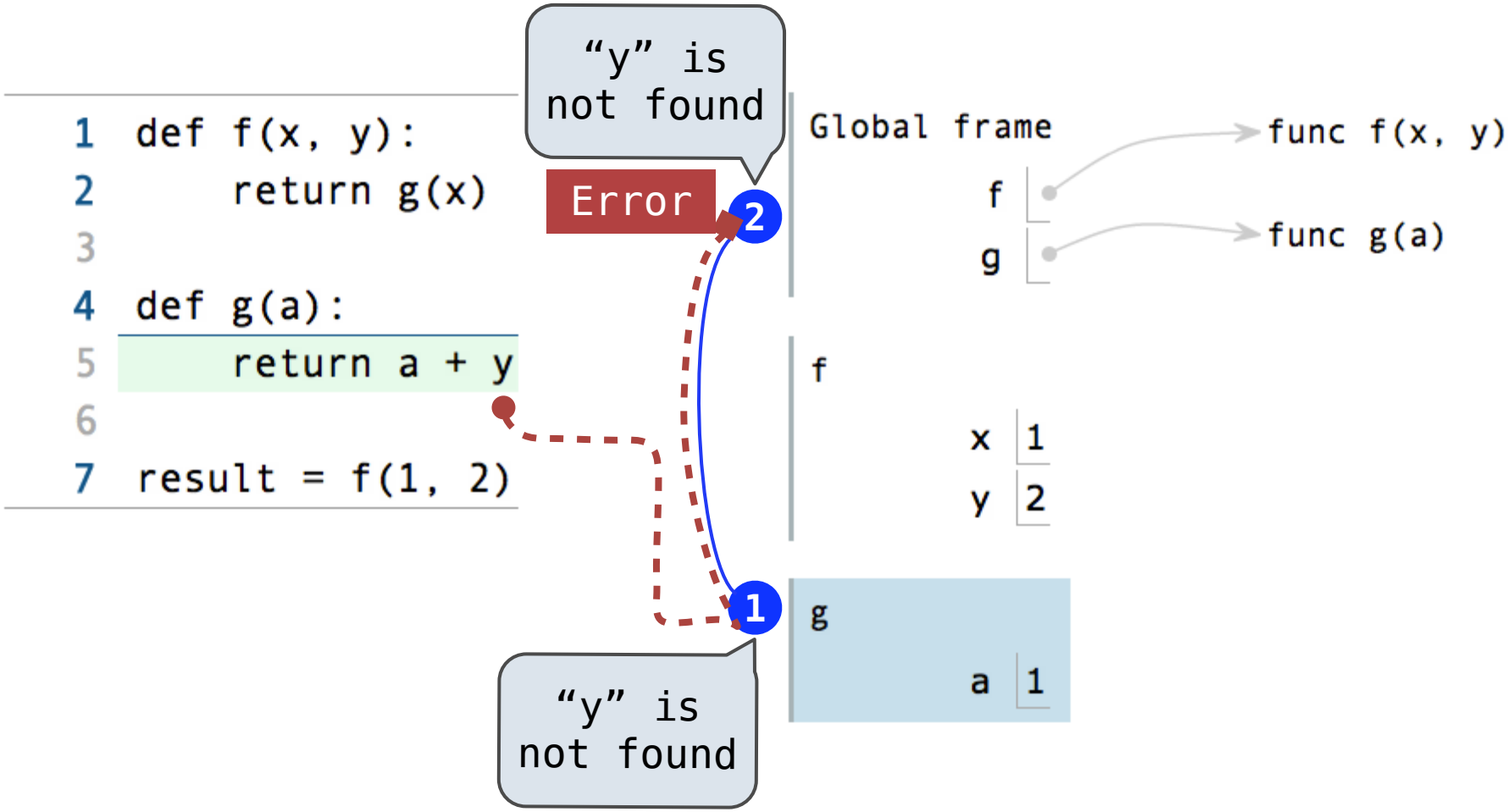
Non-Nested Functions Calls Have One Local Frame



- An environment is a sequence of frames

Example: http://goo.gl/tgT5H

Non-Nested Functions Calls Have One Local Frame



- An environment is a sequence of frames
- An environment for a non-nested function (no def within def) consists of one local frame, followed by the global frame

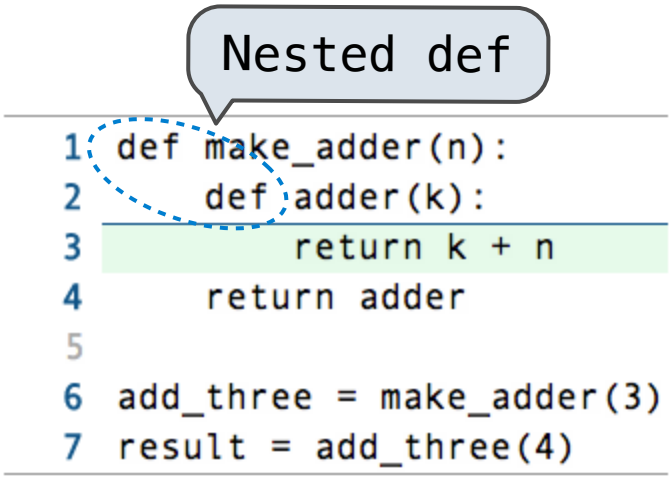
Example: http://goo.gl/tgT5H

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 result = add_three(4)
```

Example: http://goo.gl/L9G2q

Environment Diagrams for Nested Def Statements

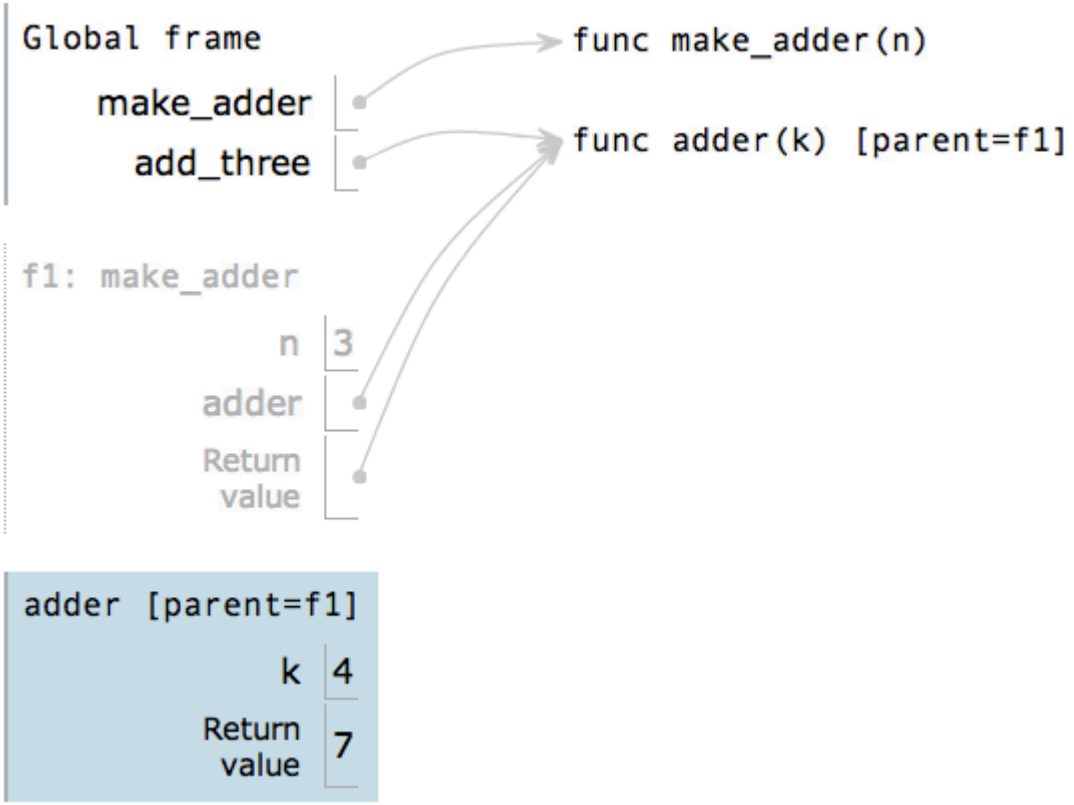


Example: http://goo.gl/L9G2q

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 result = add_three(4)
```

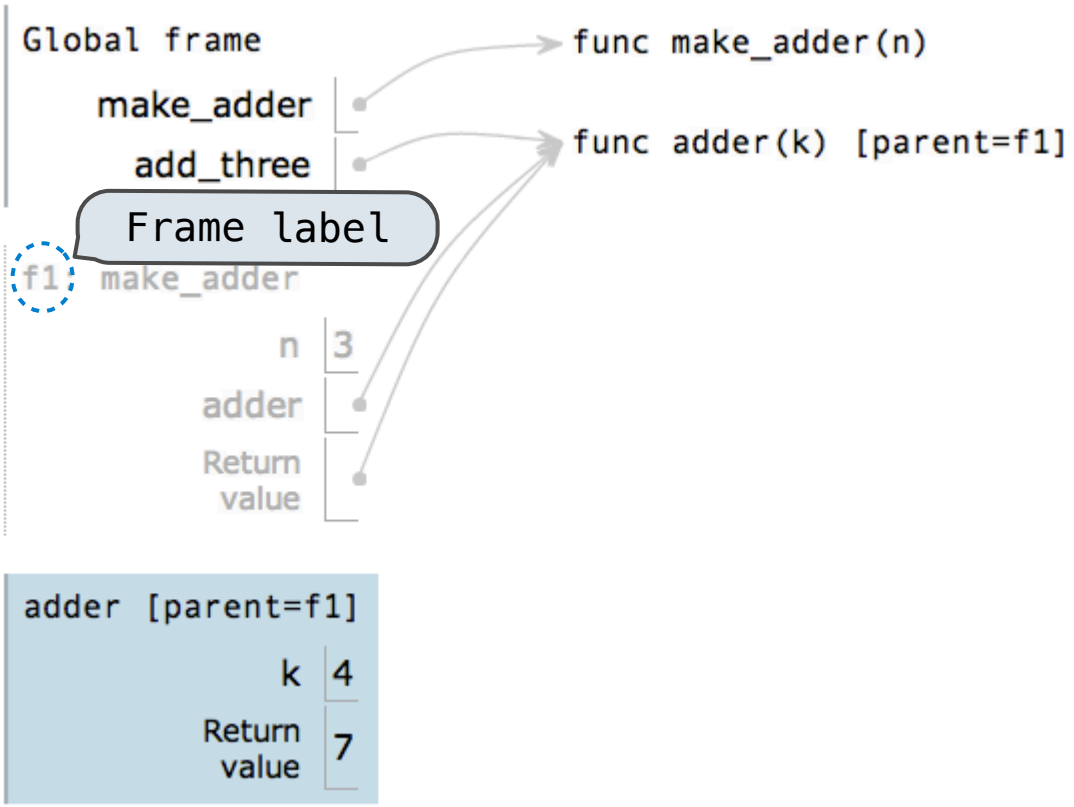


Example: http://goo.gl/L9G2q

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 result = add_three(4)
```

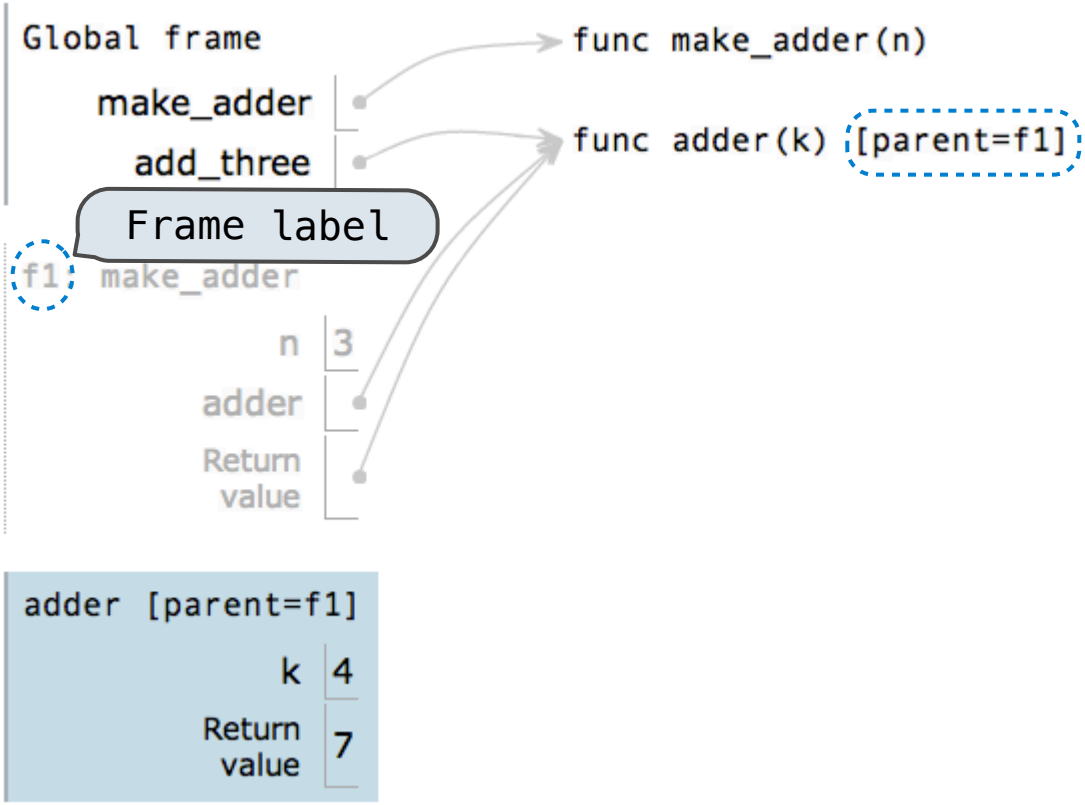


Example: http://goo.gl/L9G2q

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 result = add_three(4)
```

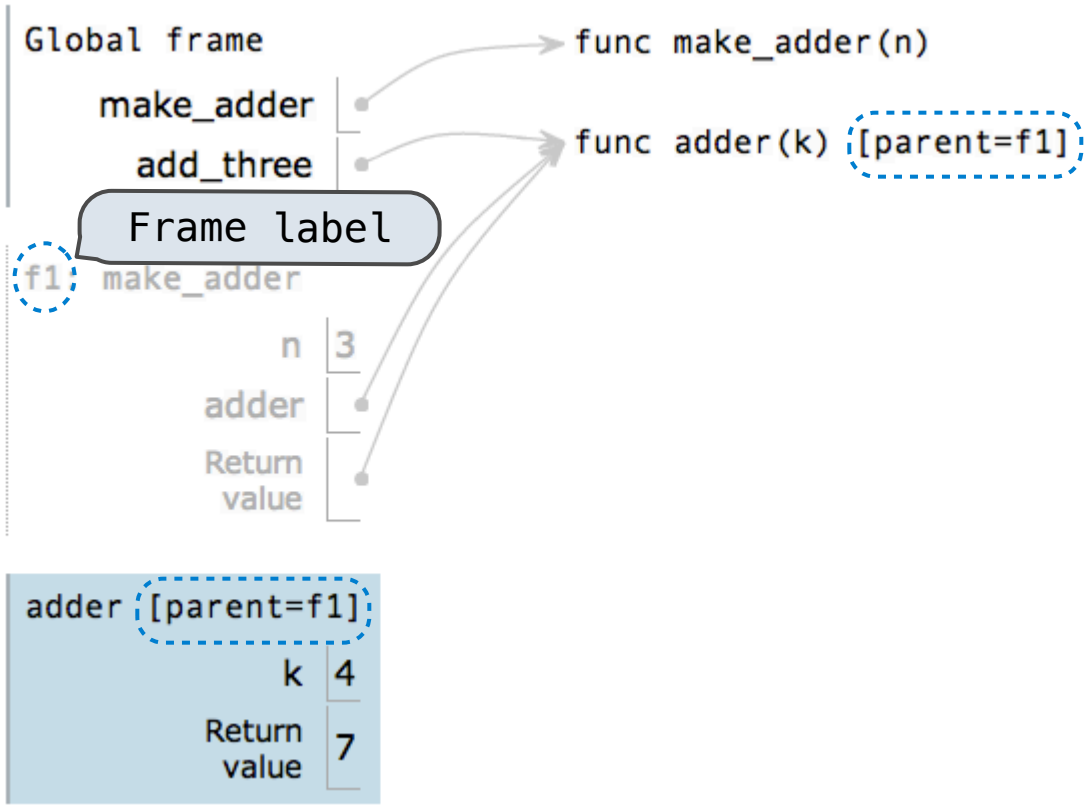


Example: http://goo.gl/L9G2q

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 result = add_three(4)
```

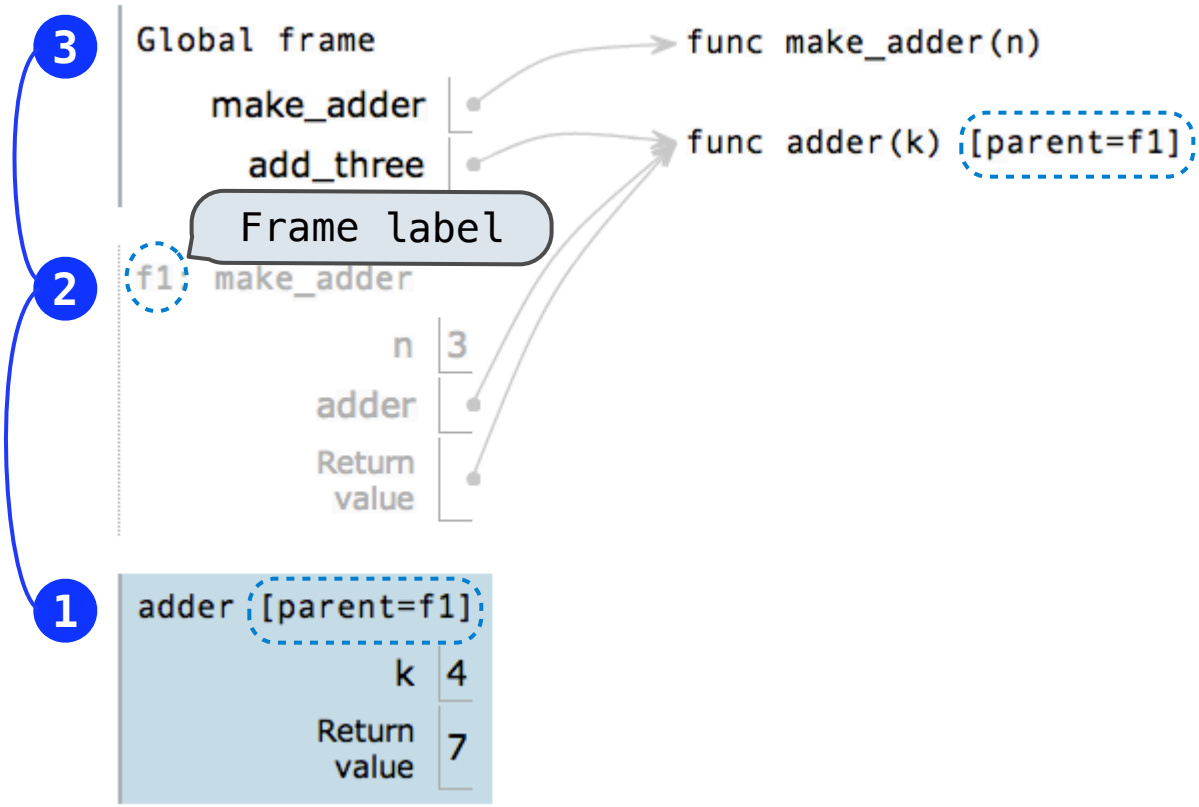


Example: http://goo.gl/L9G2q

Environment Diagrams for Nested Def Statements

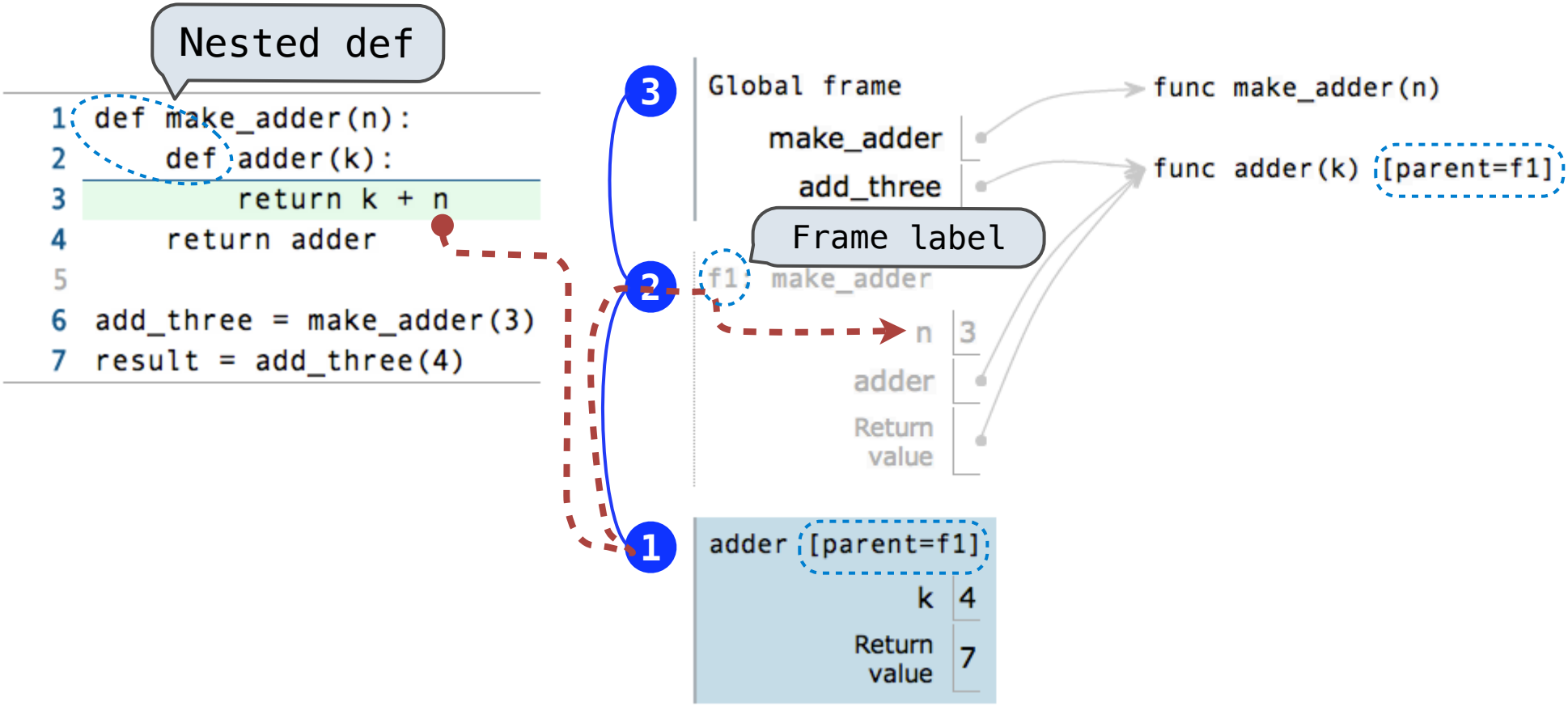
Nested def

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



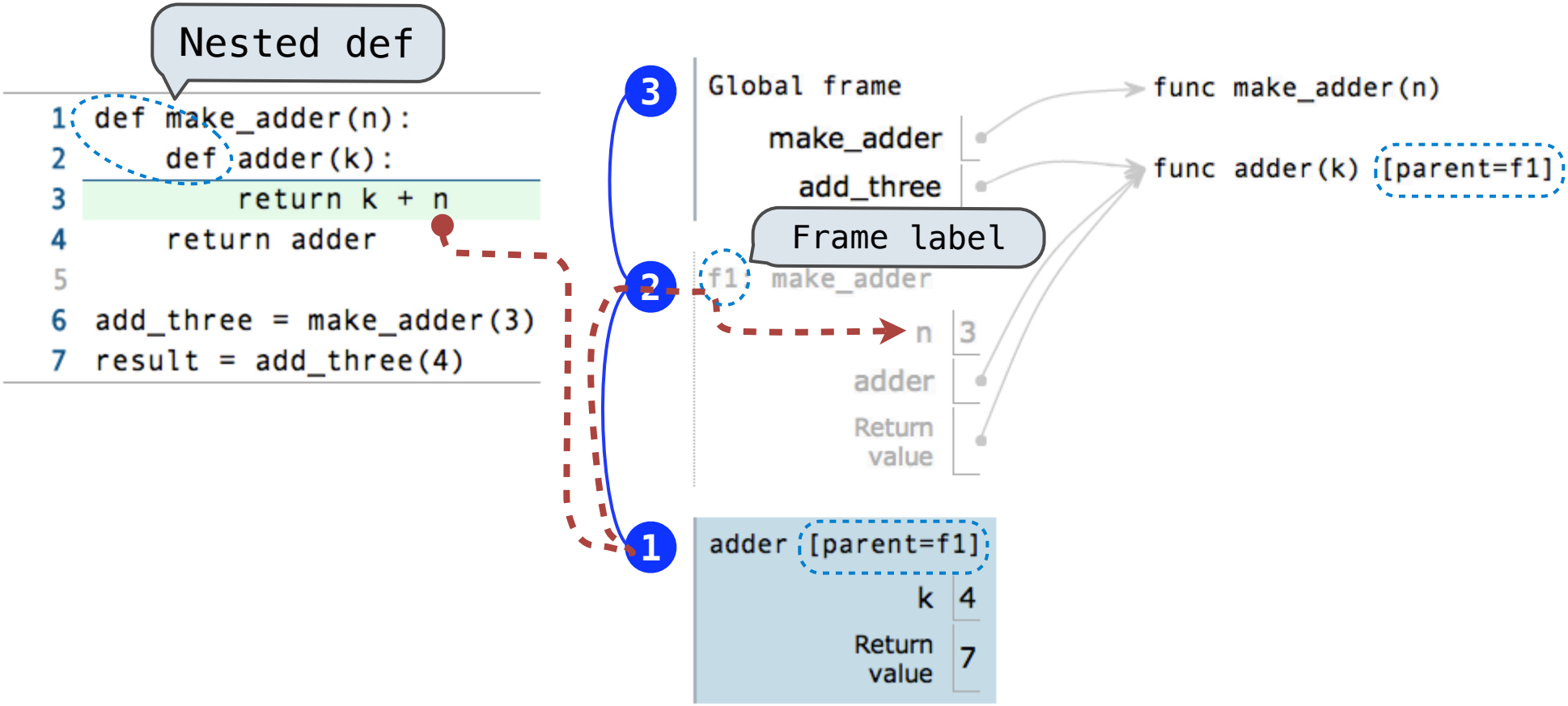
Example: http://goo.gl/L9G2q

Environment Diagrams for Nested Def Statements



Example: http://goo.gl/L9G2q

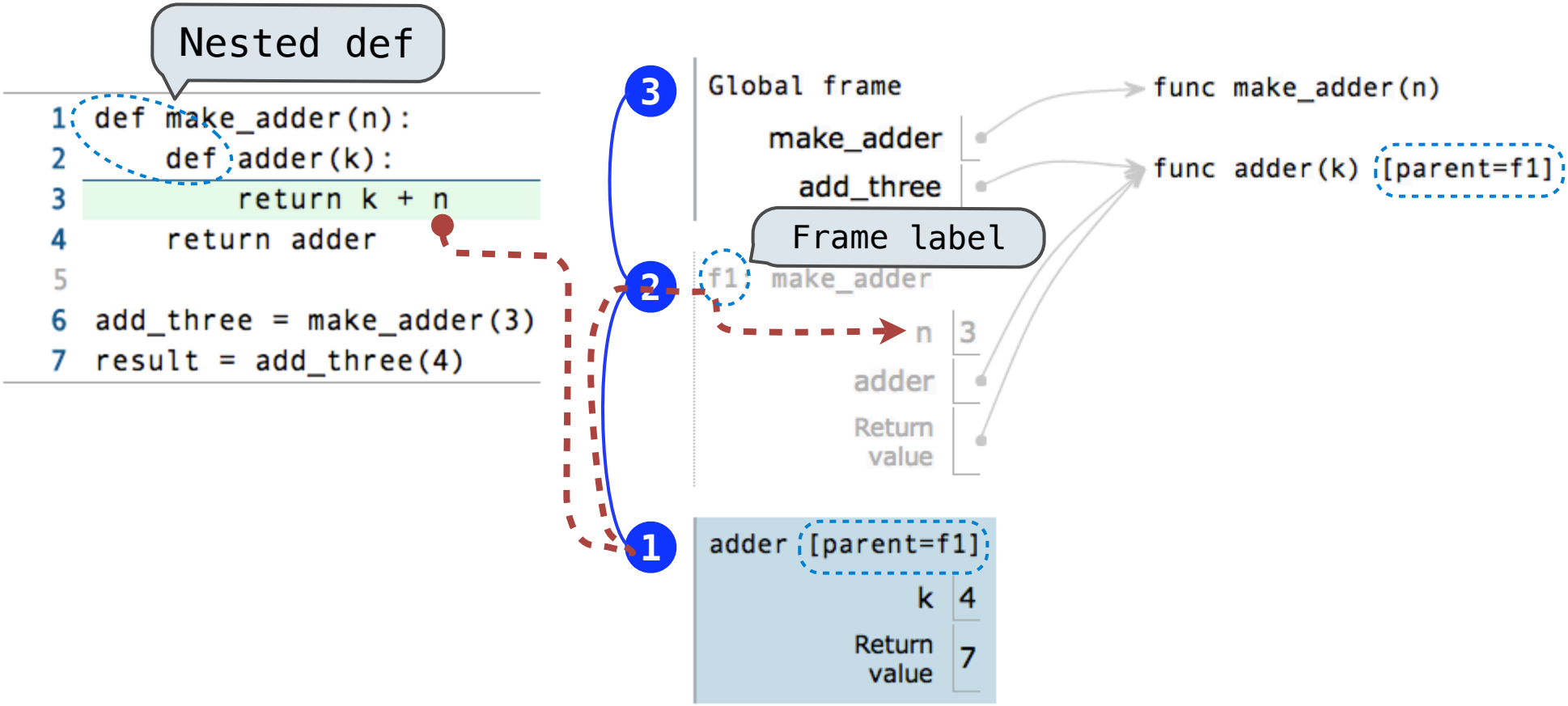
Environment Diagrams for Nested Def Statements



- Every user-defined **function** has a ***parent frame***

Example: http://goo.gl/L9G2q

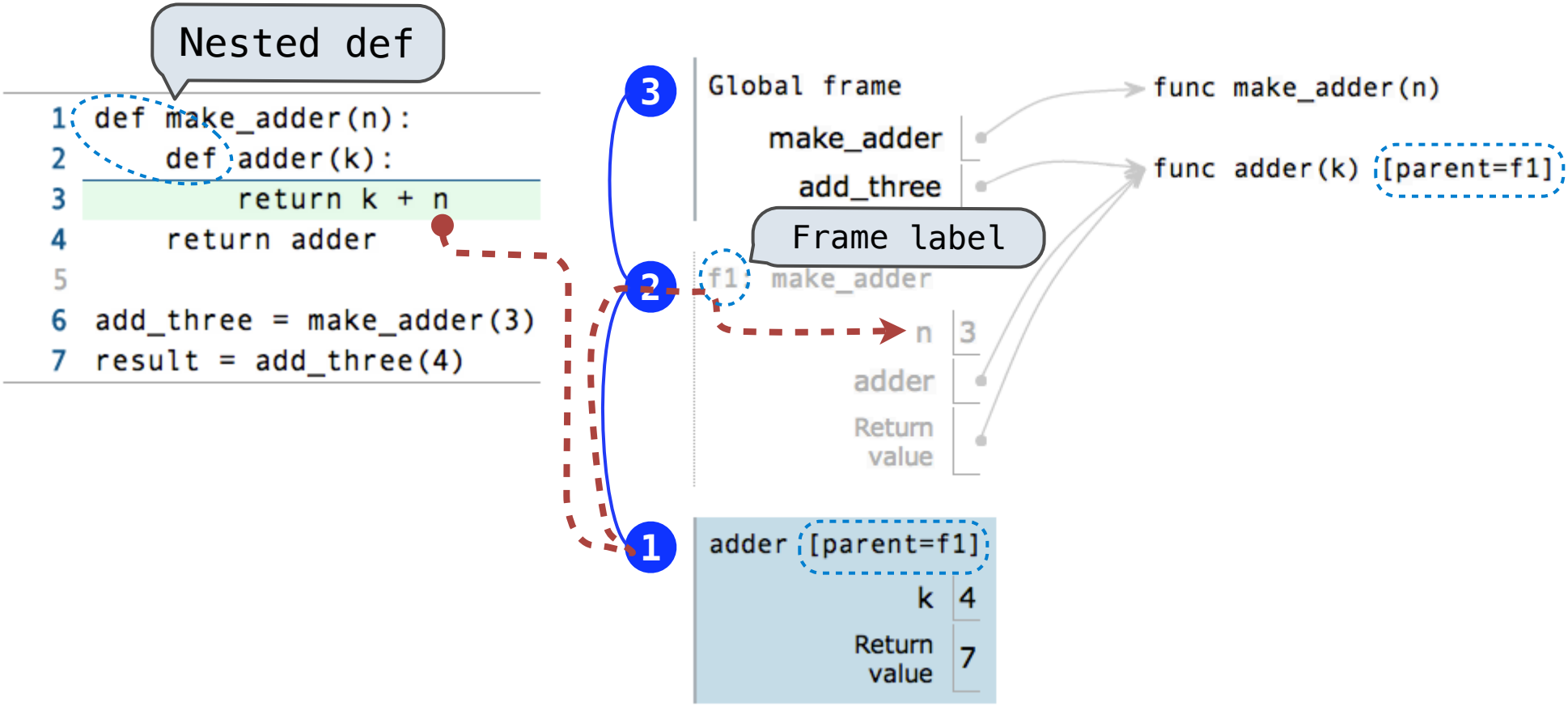
Environment Diagrams for Nested Def Statements



- Every user-defined **function** has a ***parent frame***
- The parent of a **function** is the frame in which it was *defined*

Example: http://goo.gl/L9G2q

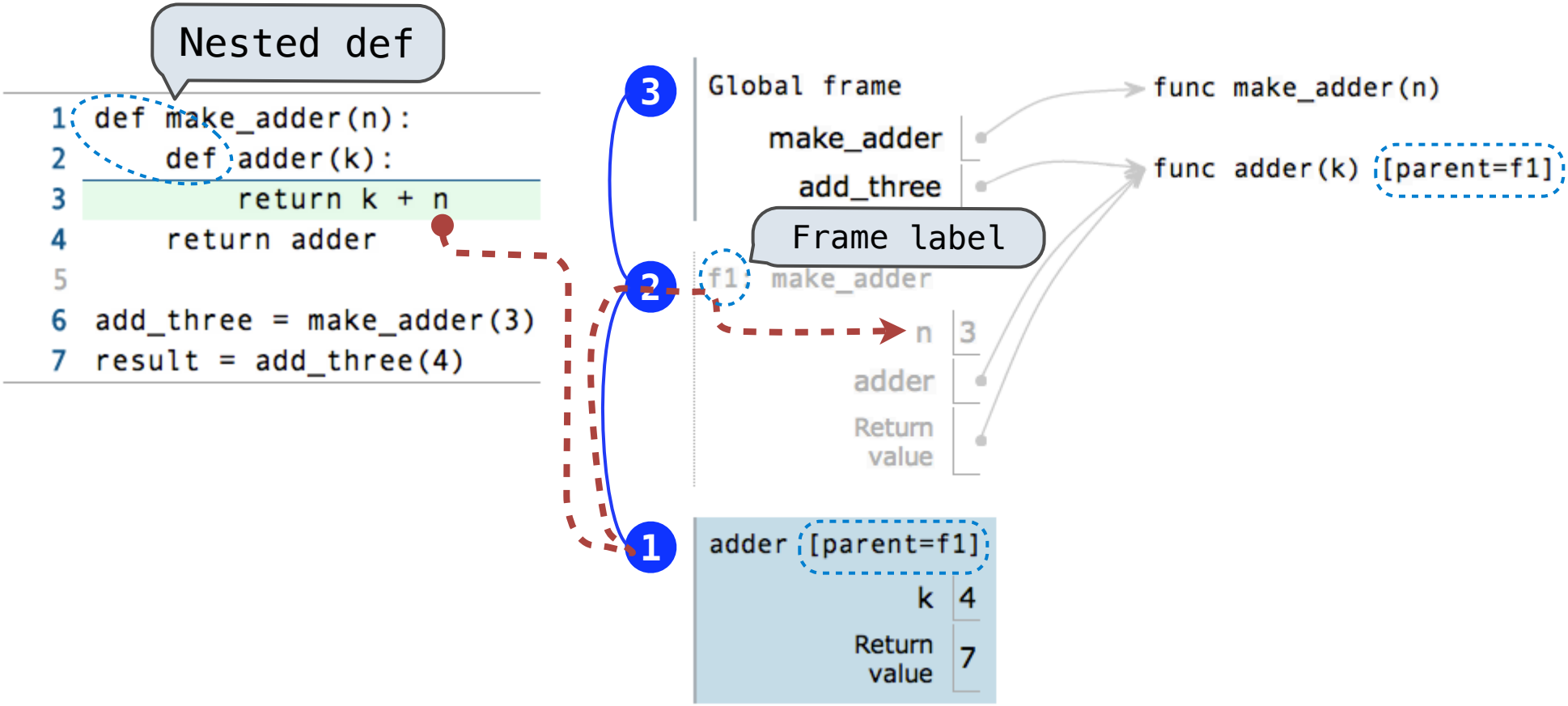
Environment Diagrams for Nested Def Statements



- Every user-defined **function** has a ***parent frame***
- The parent of a **function** is the frame in which it was *defined*
- Every local **frame** has a ***parent frame***

Example: http://goo.gl/L9G2q

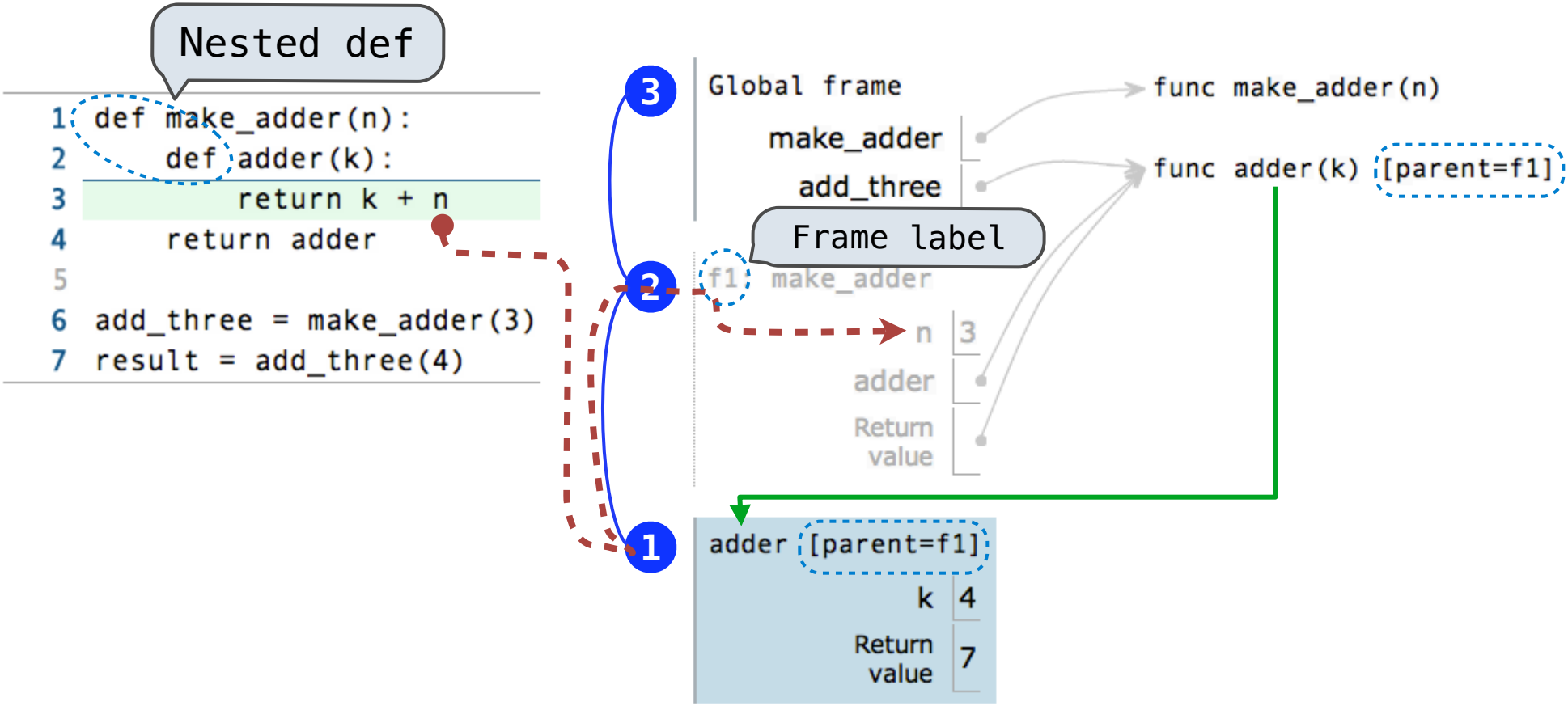
Environment Diagrams for Nested Def Statements



- Every user-defined **function** has a ***parent frame***
- The parent of a **function** is the frame in which it was *defined*
- Every local **frame** has a ***parent frame***
- The parent of a **frame** is the parent of the function *called*

Example: http://goo.gl/L9G2q

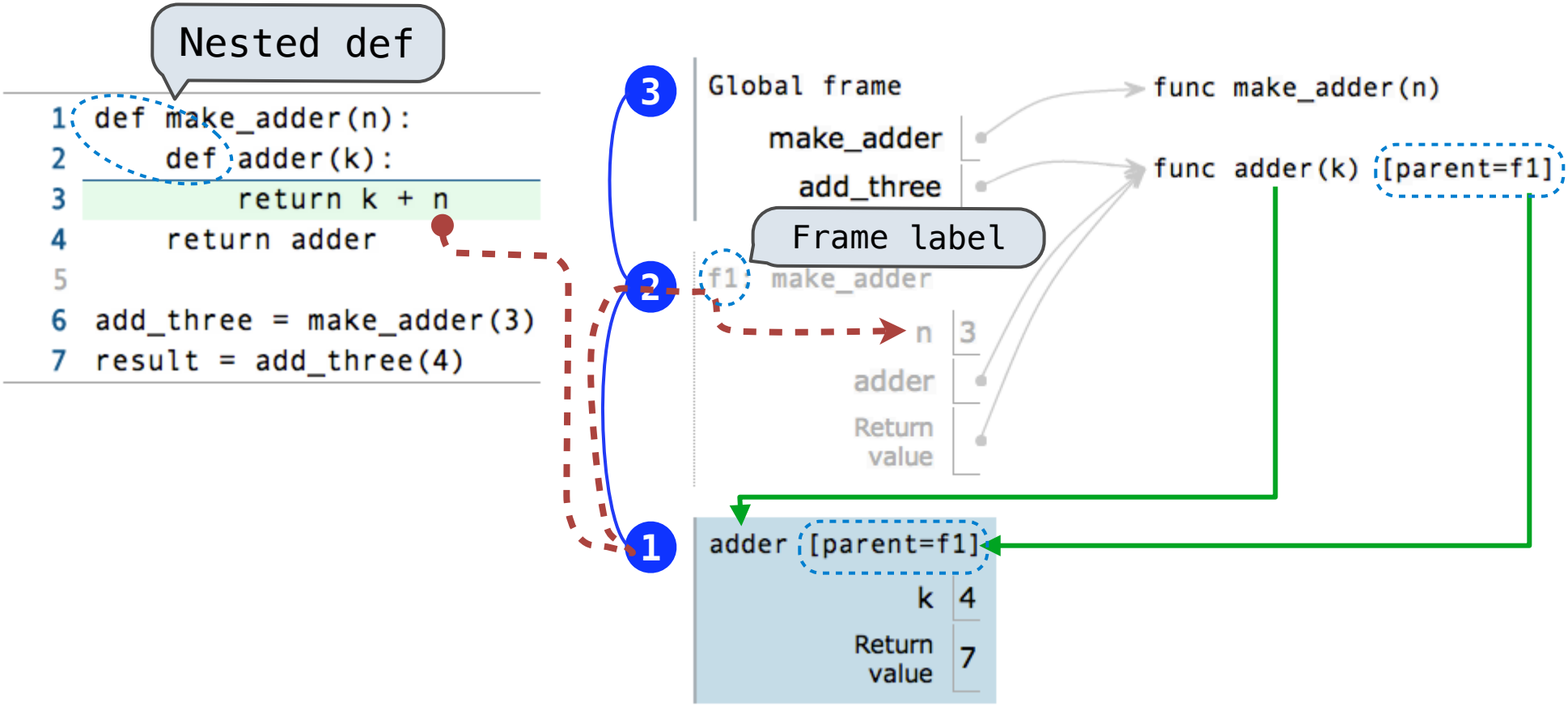
Environment Diagrams for Nested Def Statements



- Every user-defined **function** has a ***parent frame***
- The parent of a **function** is the frame in which it was *defined*
- Every local **frame** has a ***parent frame***
- The parent of a **frame** is the parent of the function *called*

Example: http://goo.gl/L9G2q

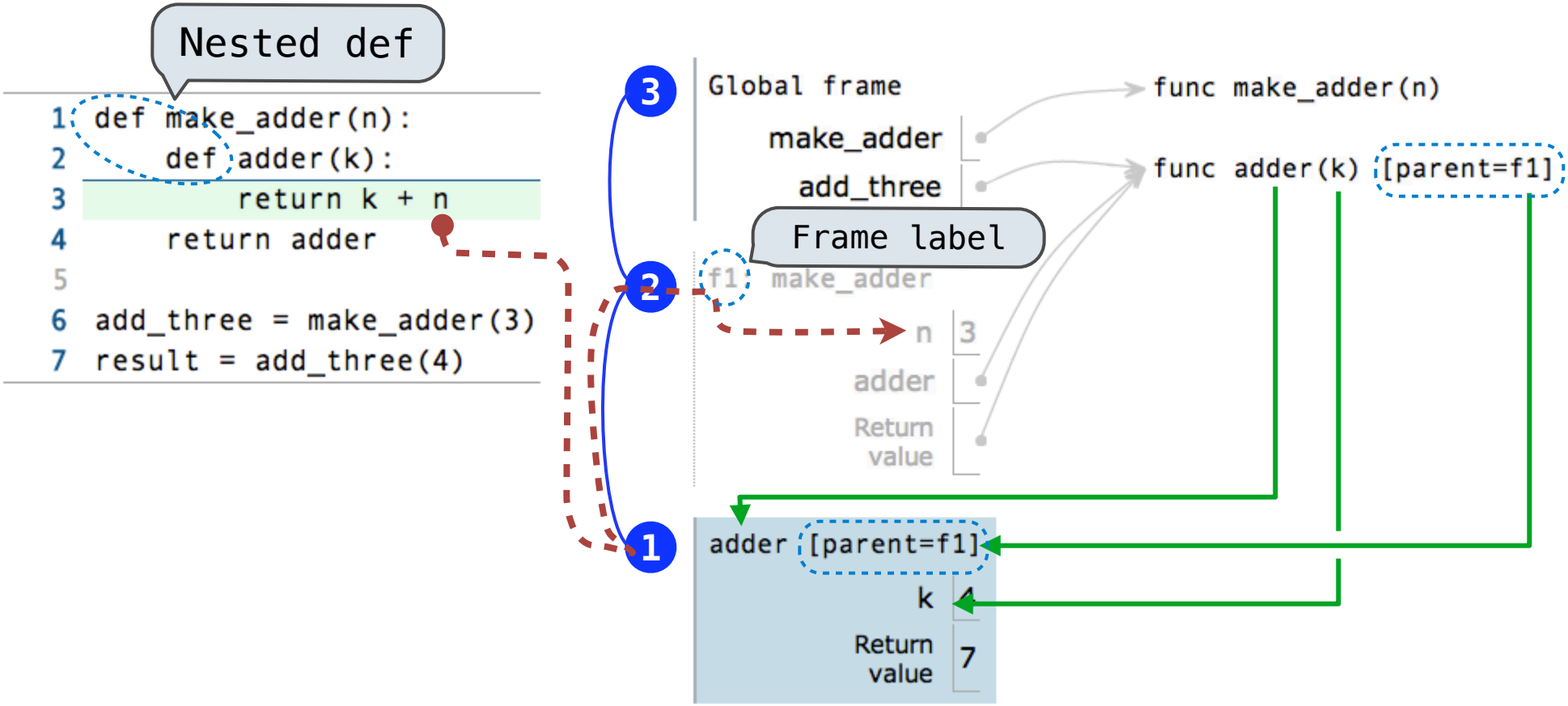
Environment Diagrams for Nested Def Statements



- Every user-defined **function** has a ***parent frame***
- The parent of a **function** is the frame in which it was *defined*
- Every local **frame** has a ***parent frame***
- The parent of a **frame** is the parent of the function *called*

Example: http://goo.gl/L9G2q

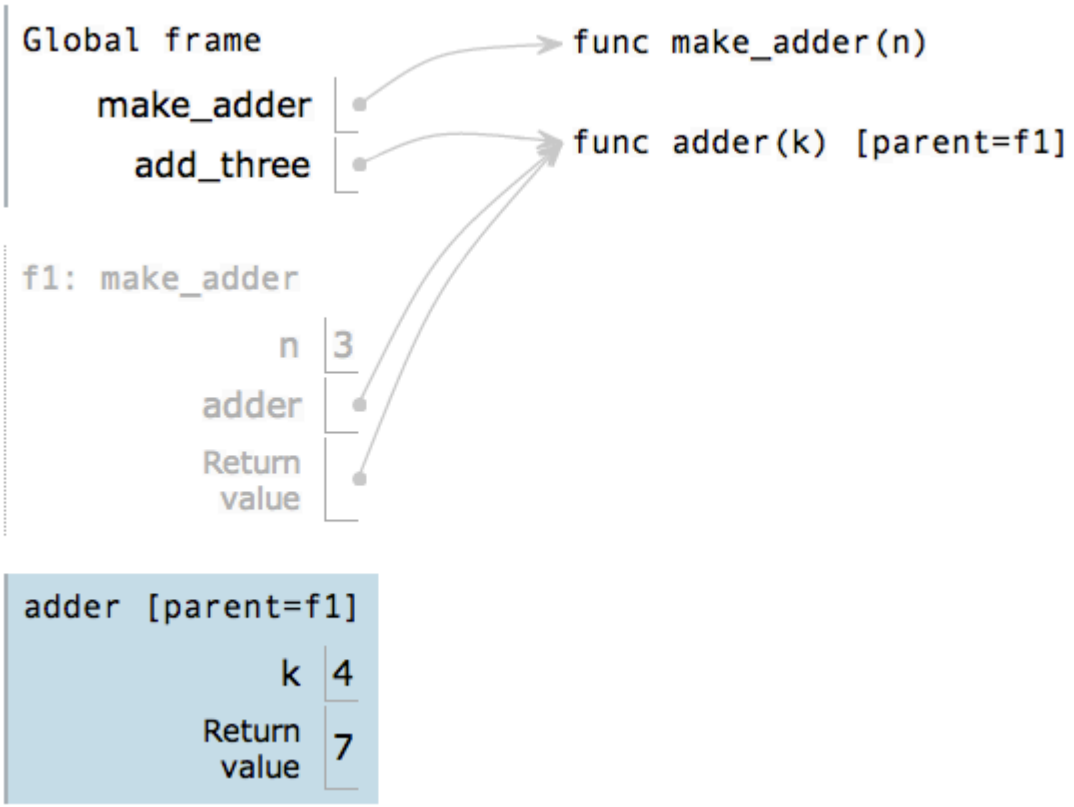
Environment Diagrams for Nested Def Statements



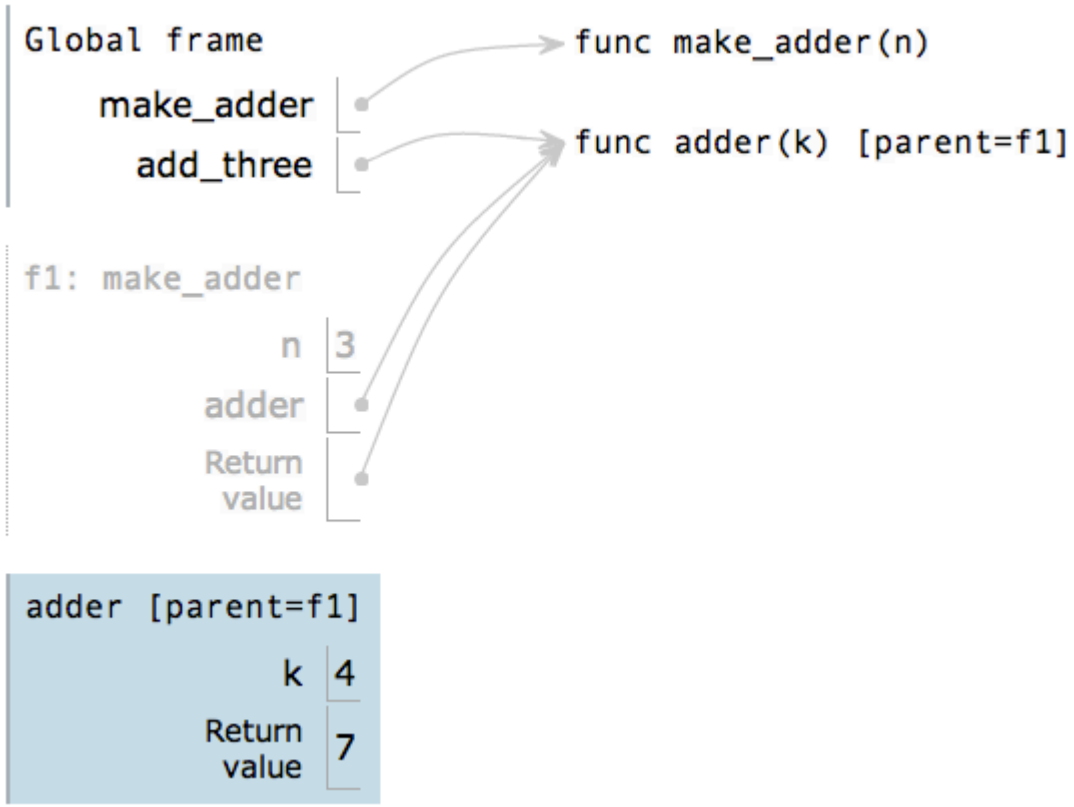
- Every user-defined **function** has a ***parent frame***
- The parent of a **function** is the frame in which it was *defined*
- Every local **frame** has a ***parent frame***
- The parent of a **frame** is the parent of the function *called*

Example: http://goo.gl/L9G2q

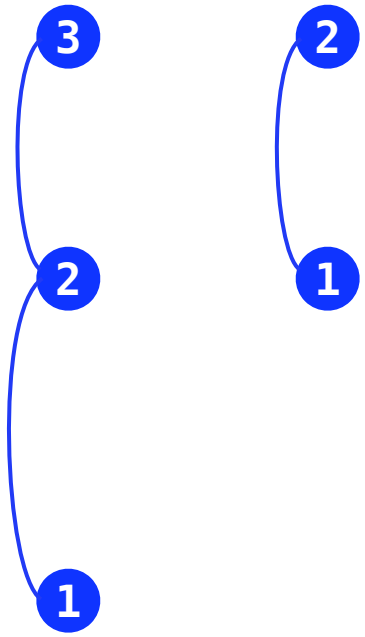
The Structure of Environments

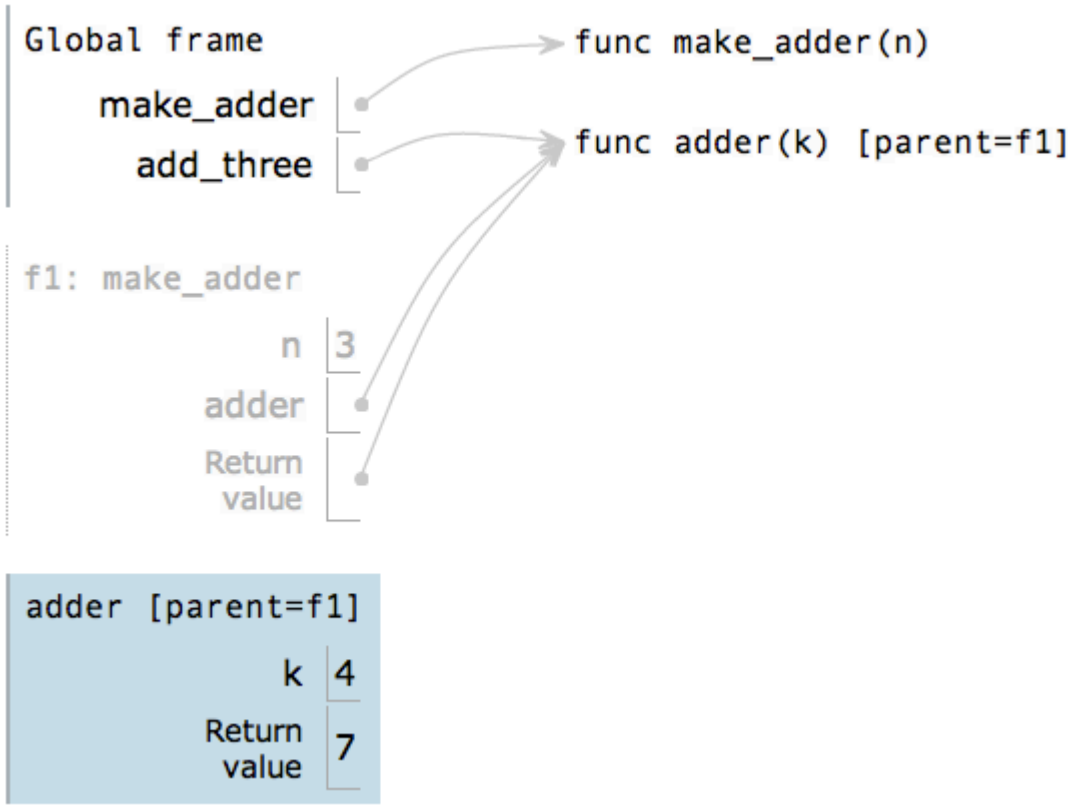


The Structure of Environments

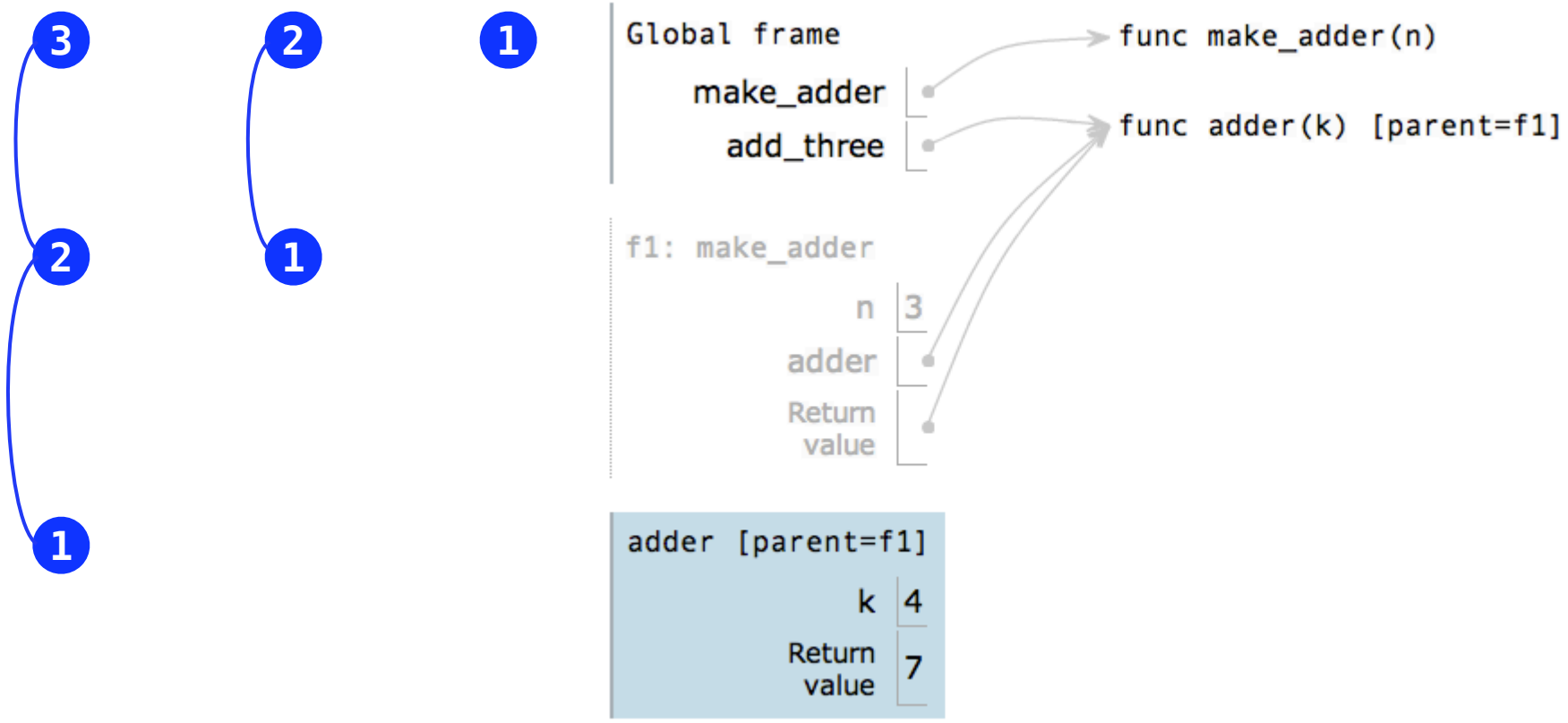


The Structure of Environments

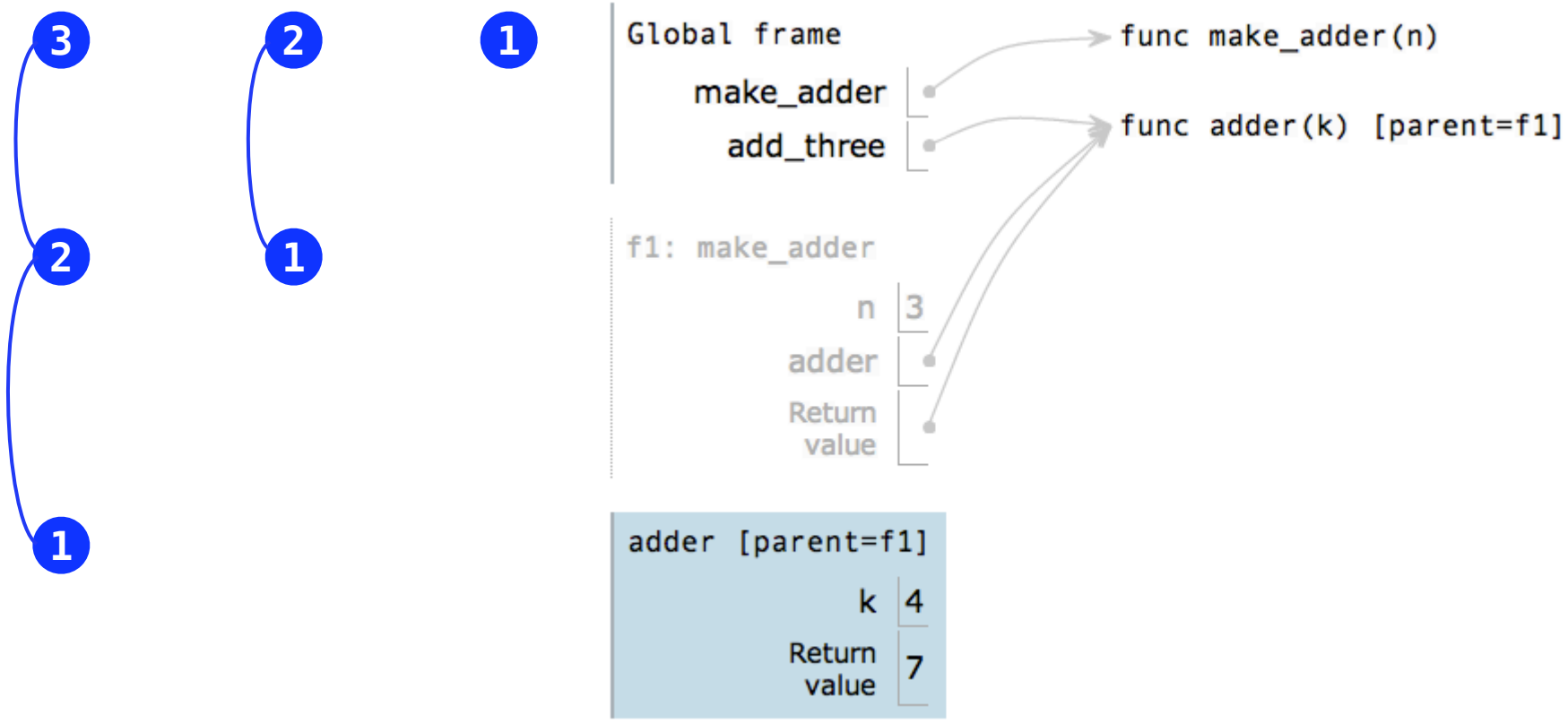




The Structure of Environments

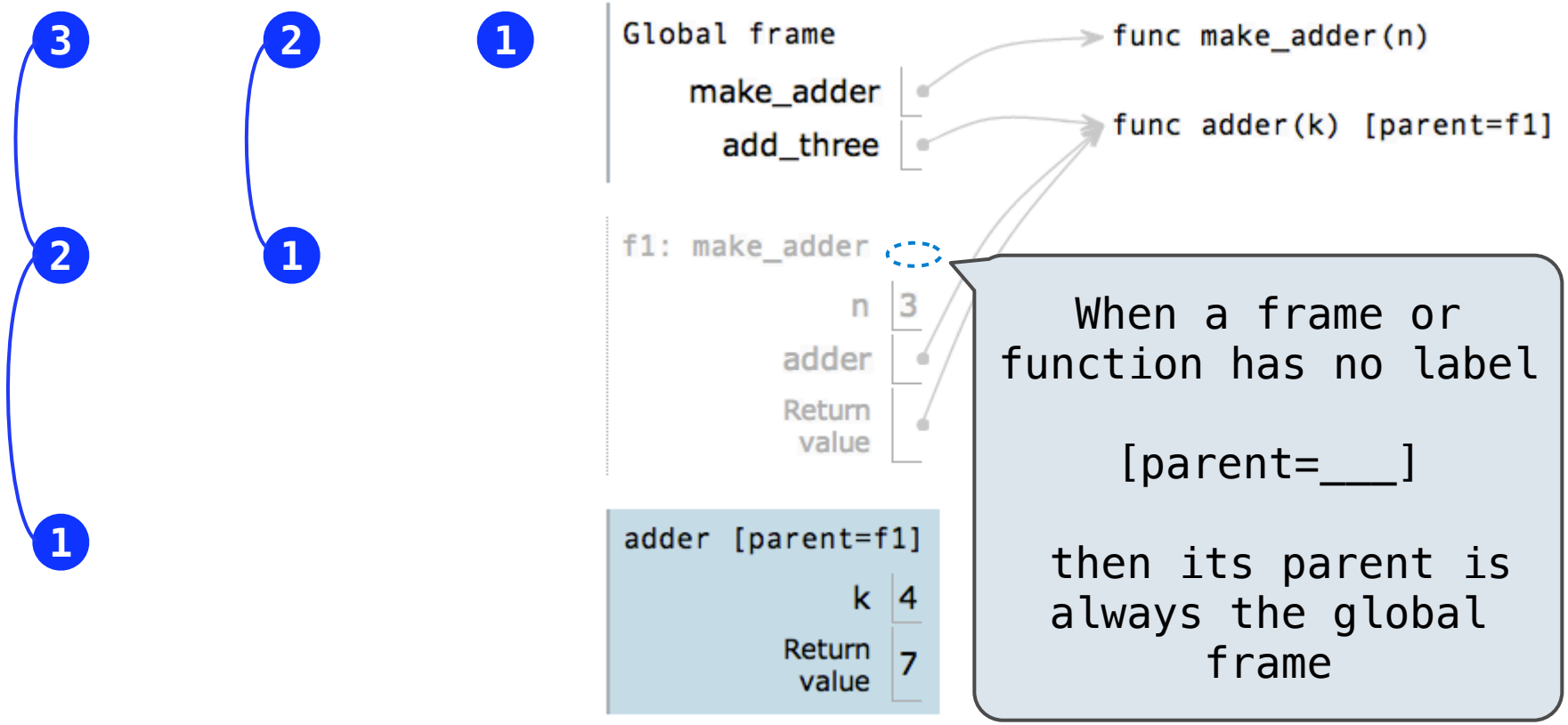


The Structure of Environments



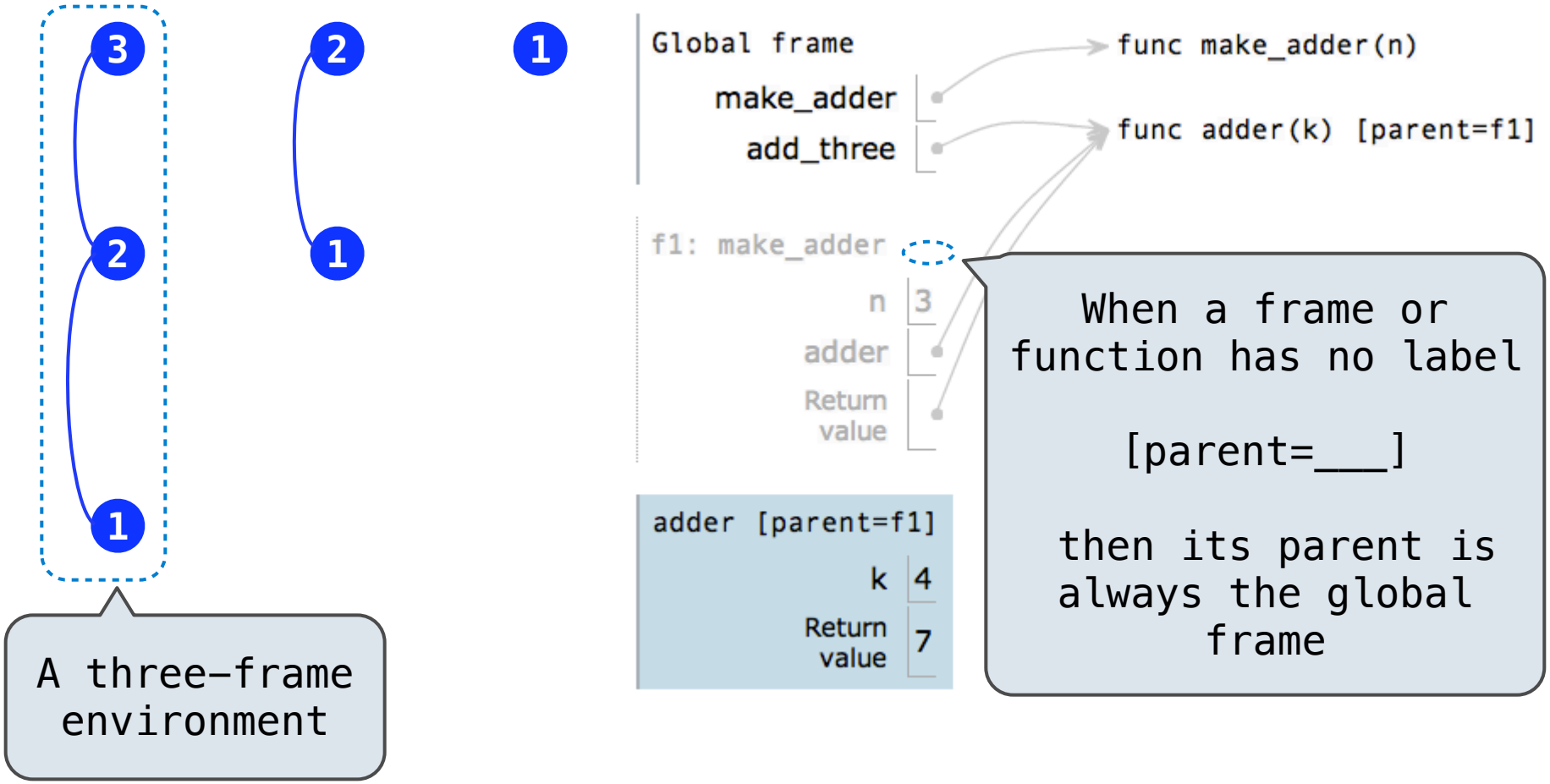
A frame *extends* the environment that begins with its parent

The Structure of Environments



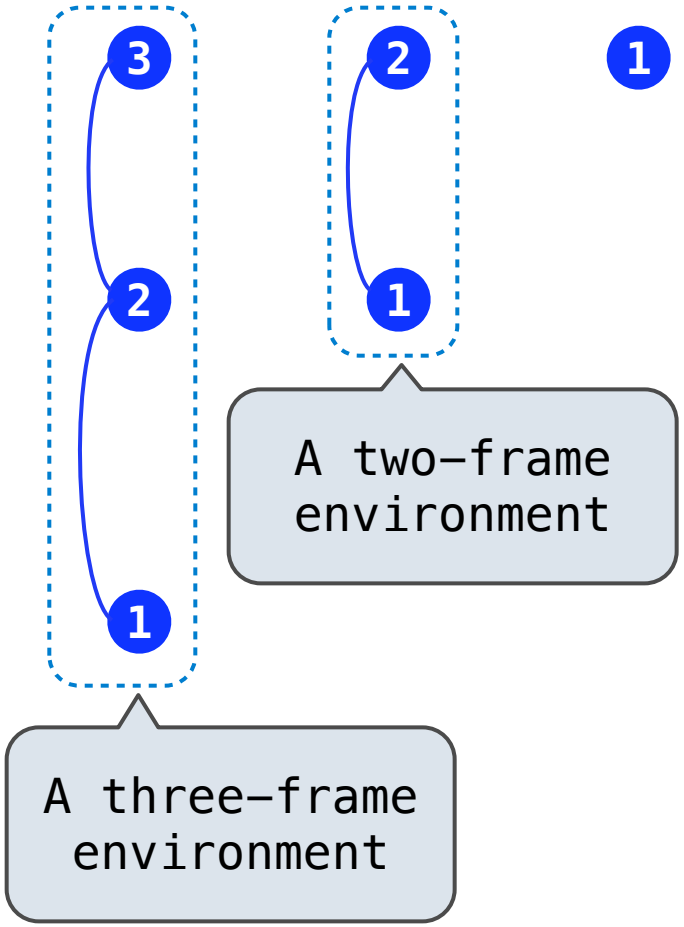
A frame *extends* the environment that begins with its parent

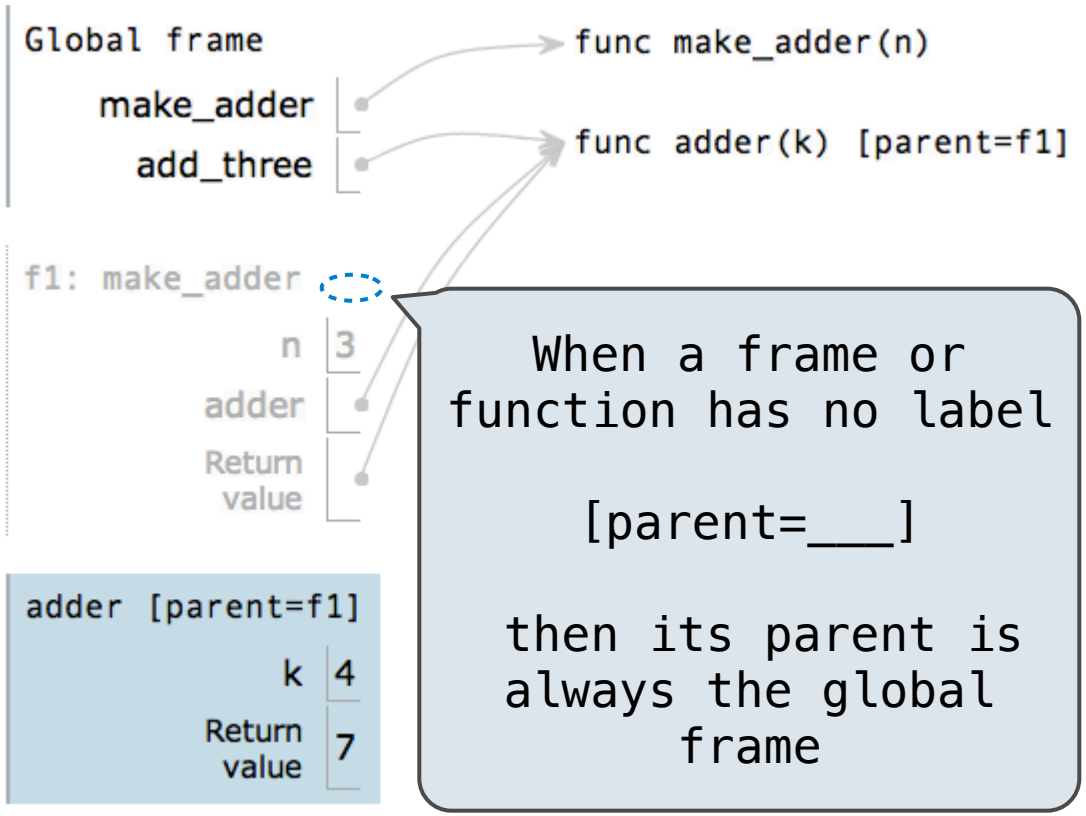
The Structure of Environments



A frame *extends* the environment that begins with its parent

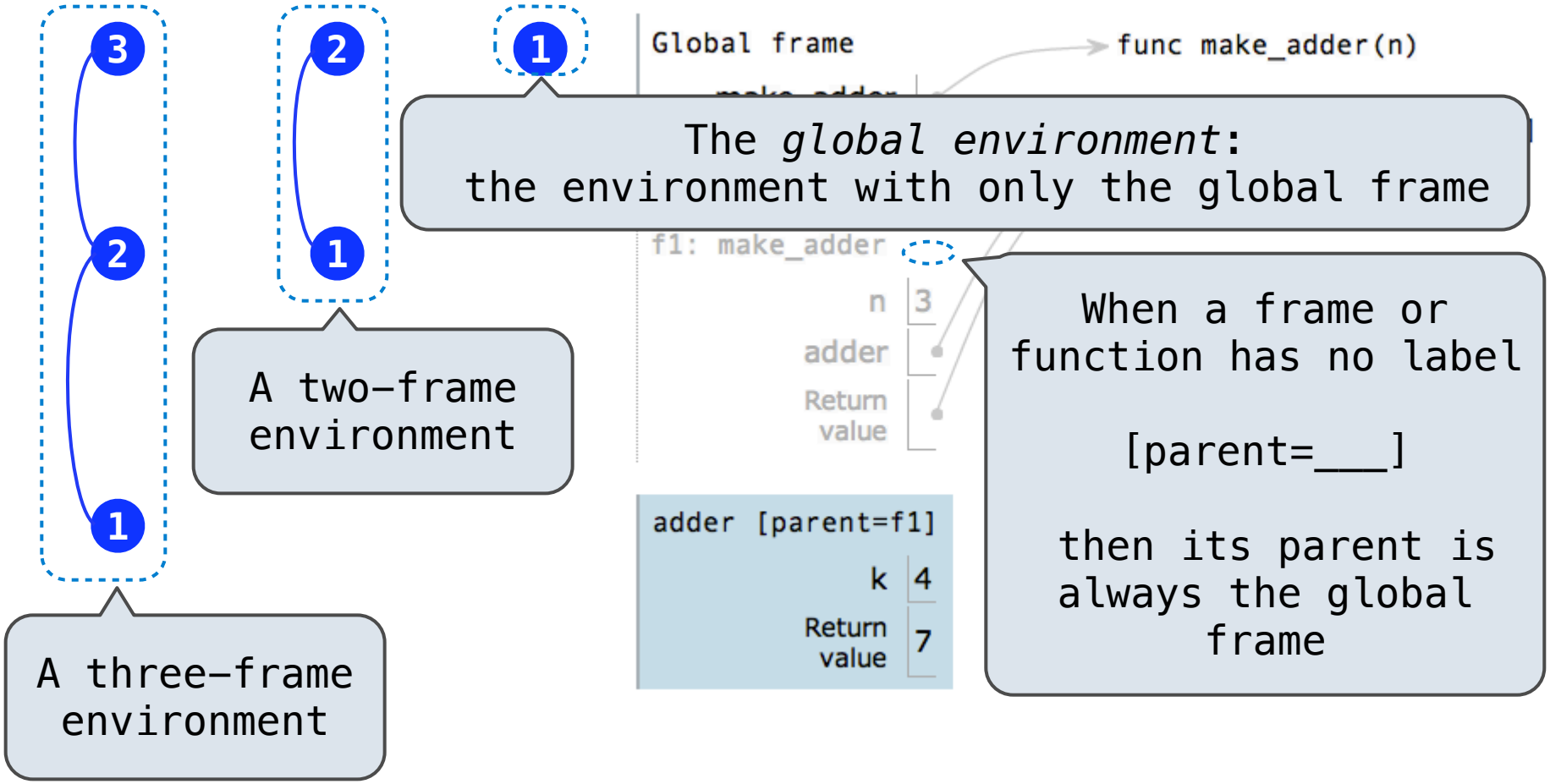
The Structure of Environments





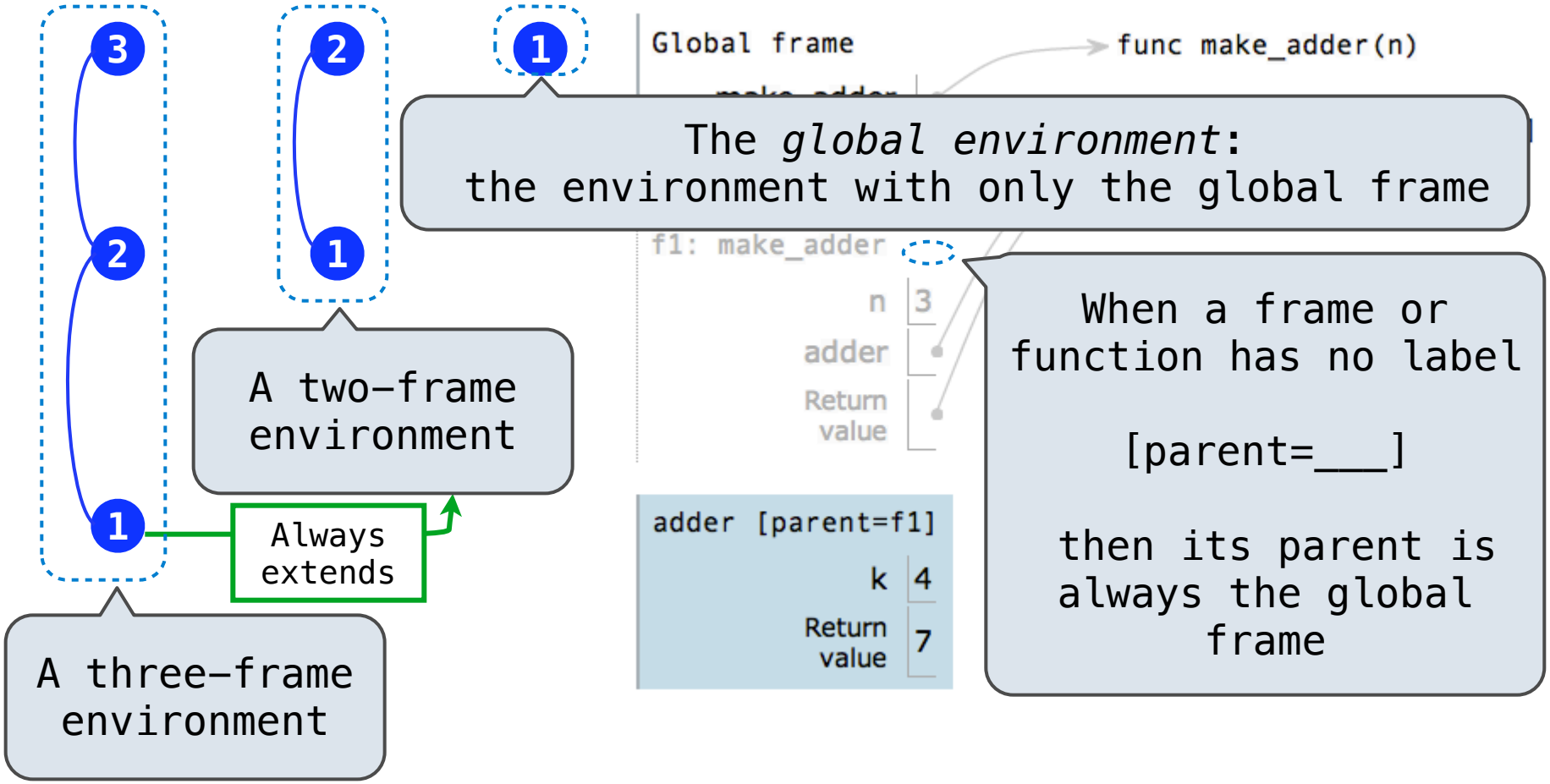
A frame *extends* the environment that begins with its parent

The Structure of Environments



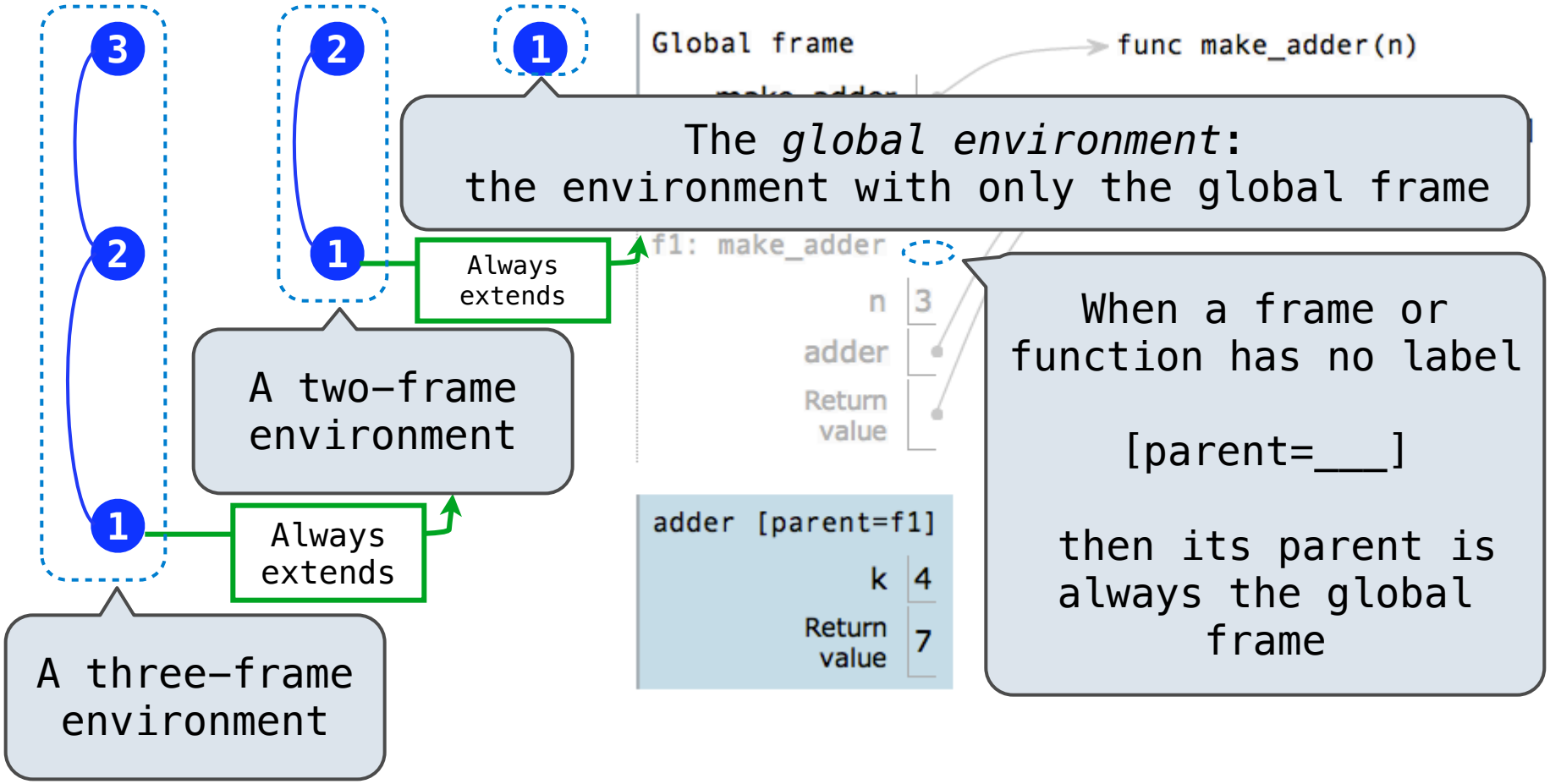
A frame *extends* the environment that begins with its parent

The Structure of Environments



A frame *extends* the environment that begins with its parent

The Structure of Environments



A frame *extends* the environment that begins with its parent

How to Draw an Environment Diagram

How to Draw an Environment Diagram

When defining a function:

How to Draw an Environment Diagram

When defining a function:

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

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

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

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:

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

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

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

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

The Environment for Function Composition

Example: http://goo.gl/2IuEO

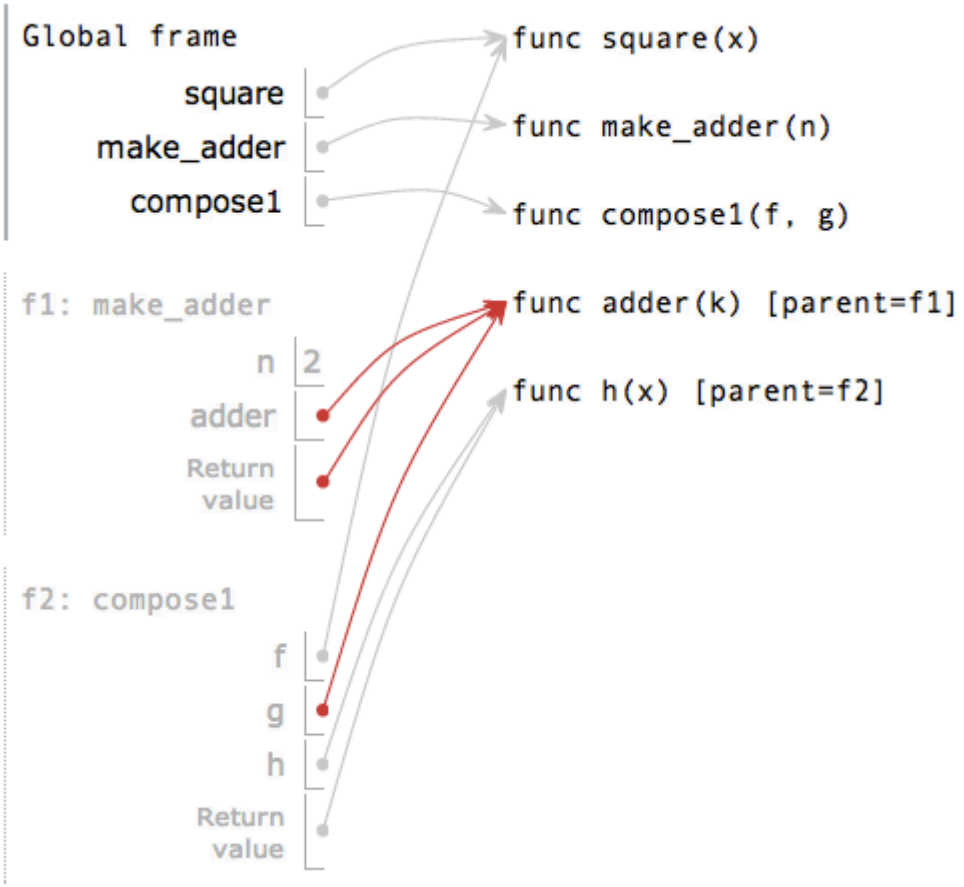
The Environment for Function Composition

```
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)
```

Example: http://goo.gl/2IuEO

The Environment for Function Composition

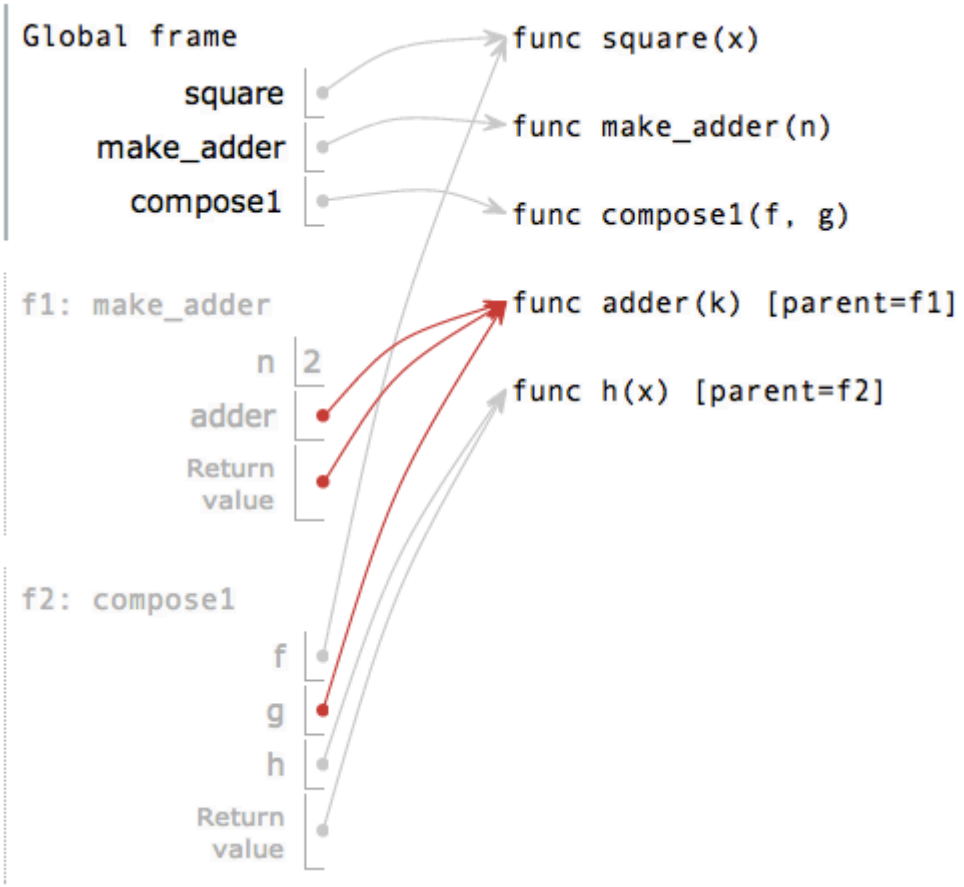
```
1  def square(x):
2      return x * x
3
4  def make_adder(n):
5      def adder(k):
6          return k + n
7      return adder
8
9  def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```



Example: http://goo.gl/2IuEO

The Environment for Function Composition

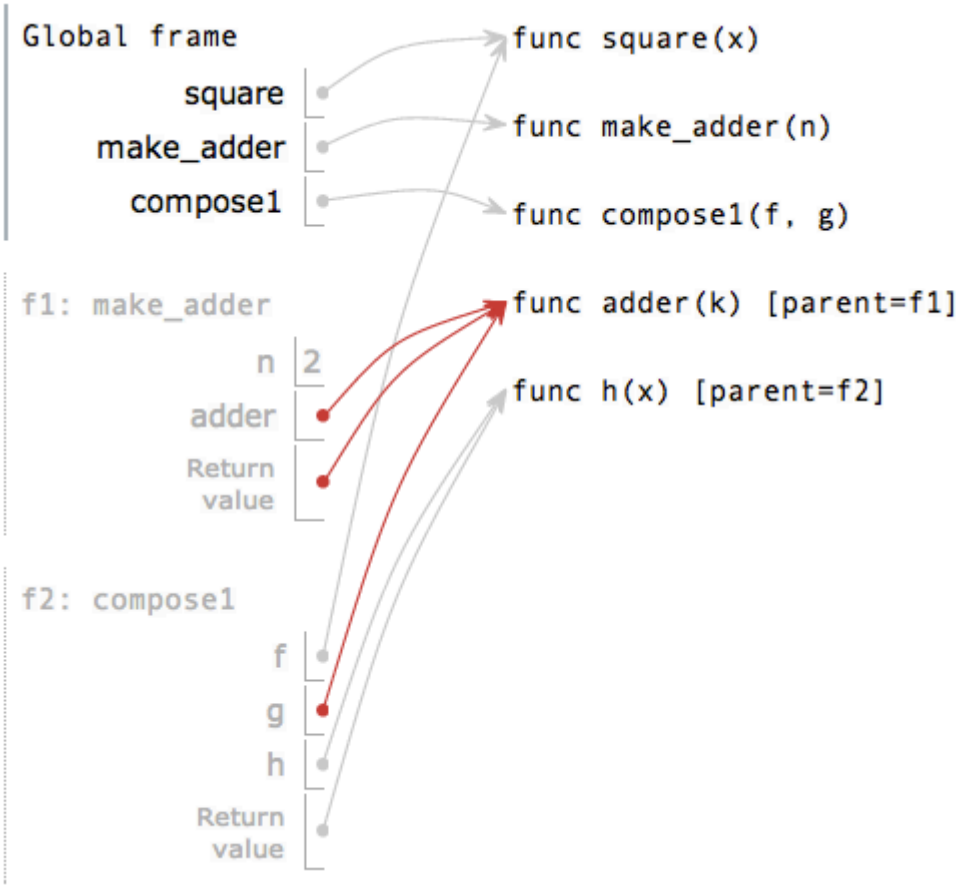
```
1  def square(x):
2      return x * x
3
4  def make_adder(n):
5      def adder(k):
6          return k + n
7      return adder
8
9  def compose1(f, g):
10     def h(x):
11         return f(g(x))
12     return h
13
14 compose1(square, make_adder(2))(3)
```



Example: http://goo.gl/2IuEO

The Environment 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)
```

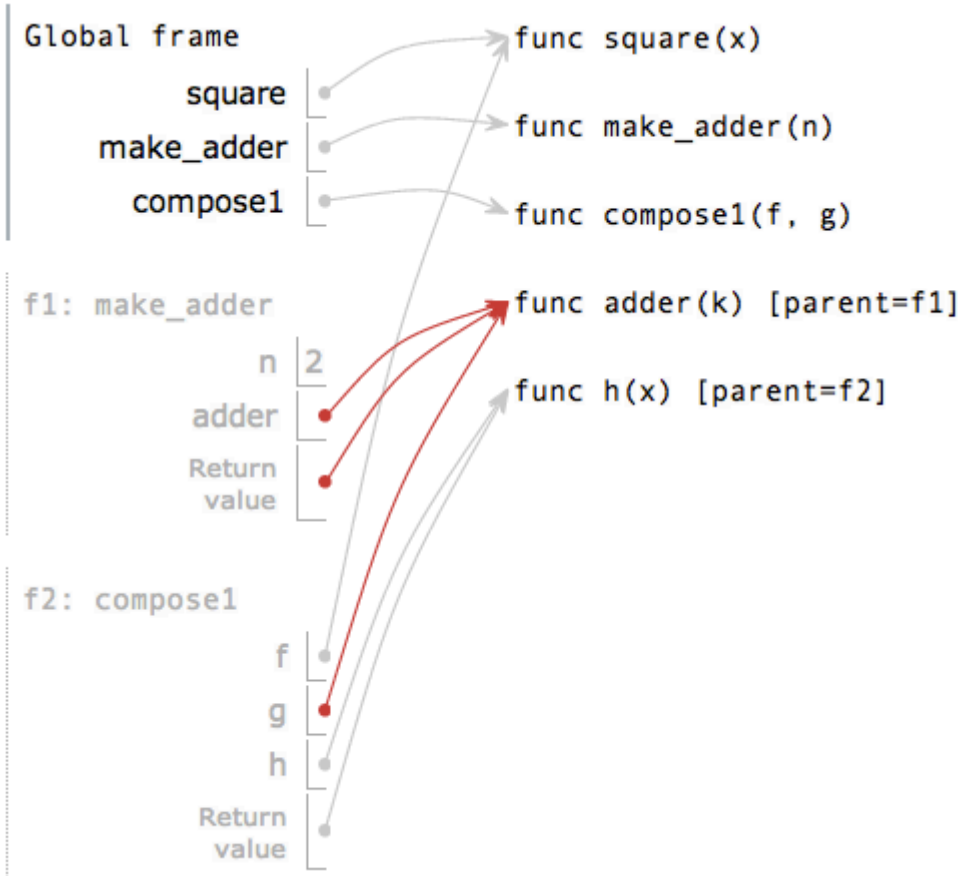


Example: http://goo.gl/2IuEO

The Environment 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

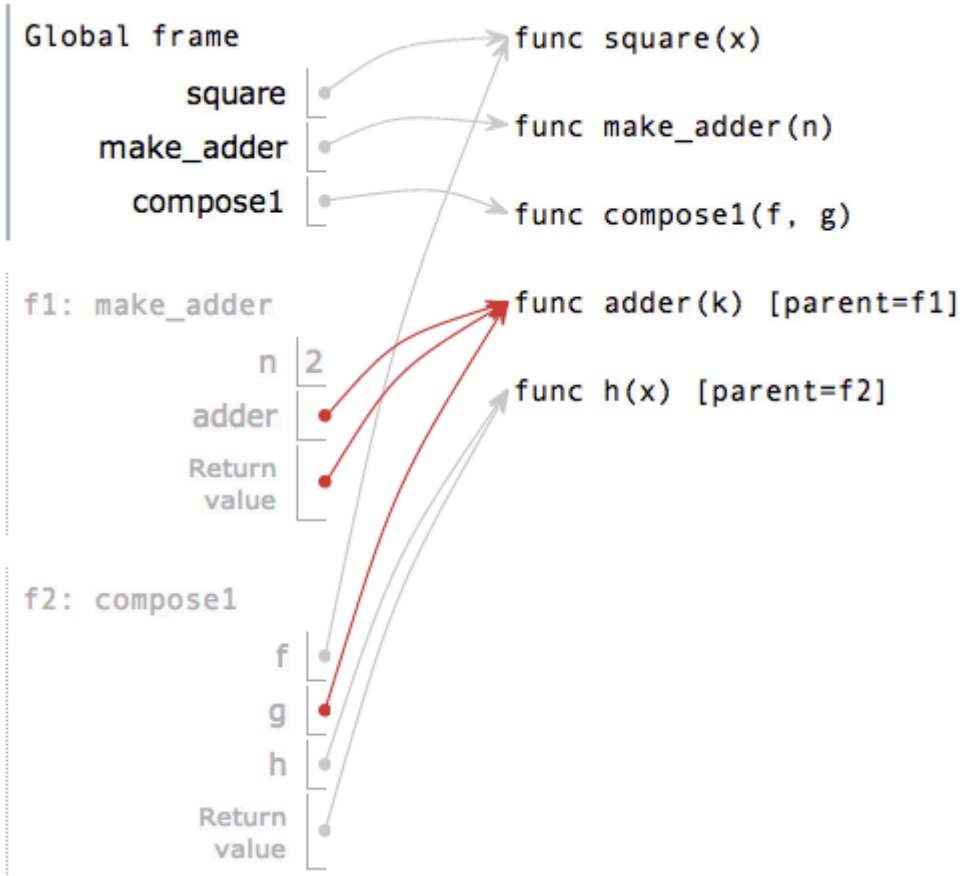


Example: http://goo.gl/2IuEO

The Environment 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

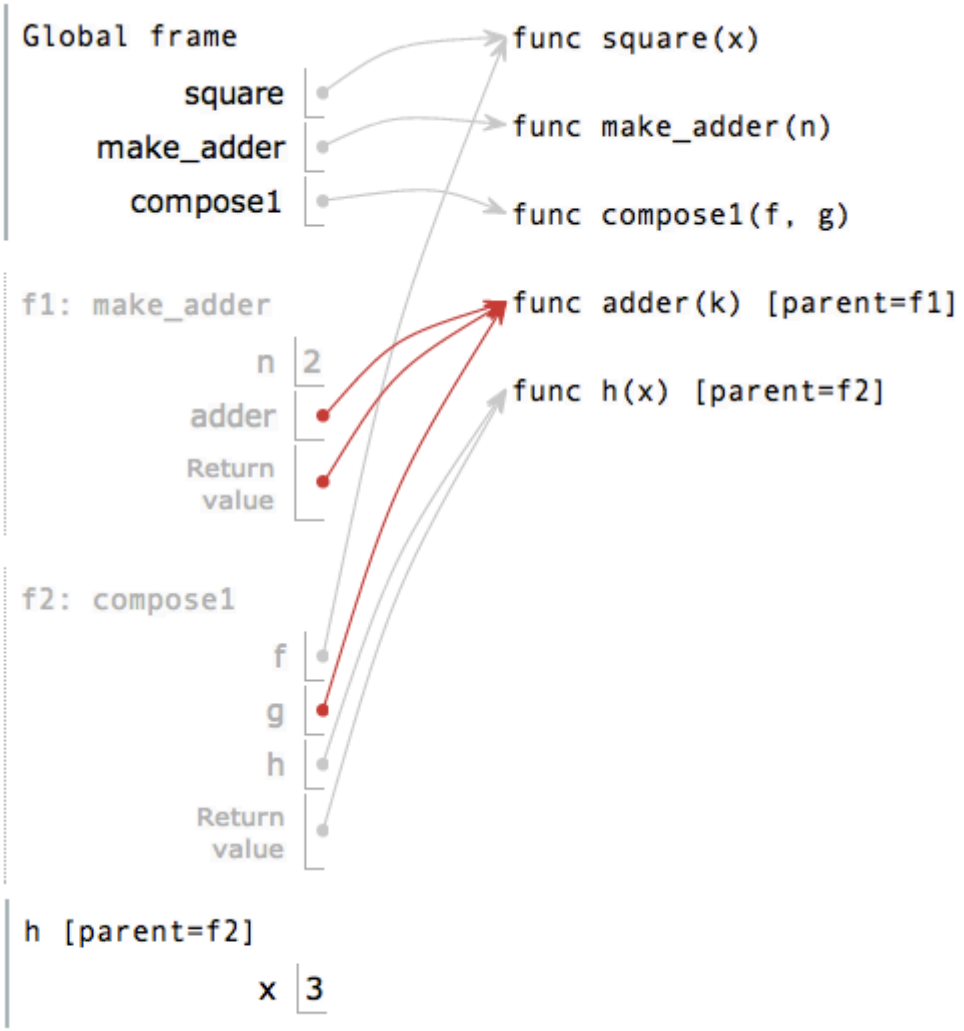


Example: http://goo.gl/2IuEO

The Environment 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

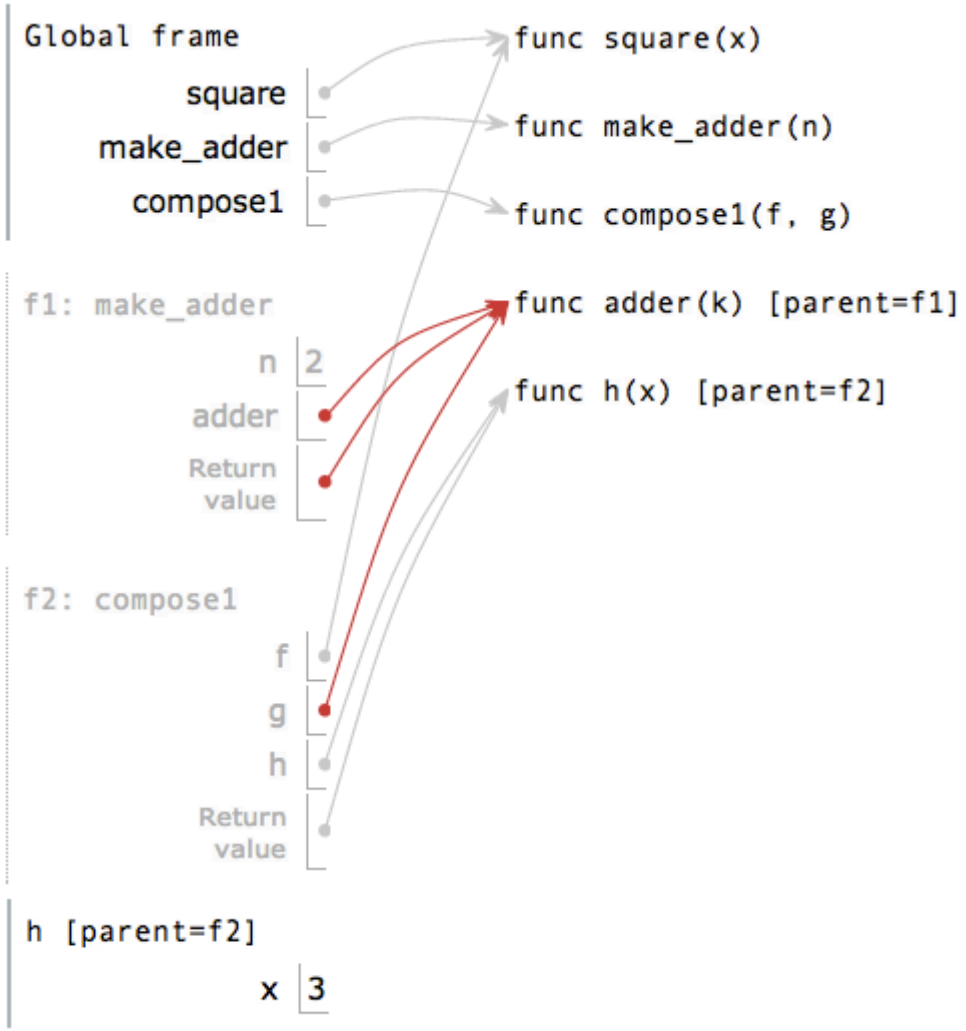


Example: http://goo.gl/2IuEO

The Environment 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

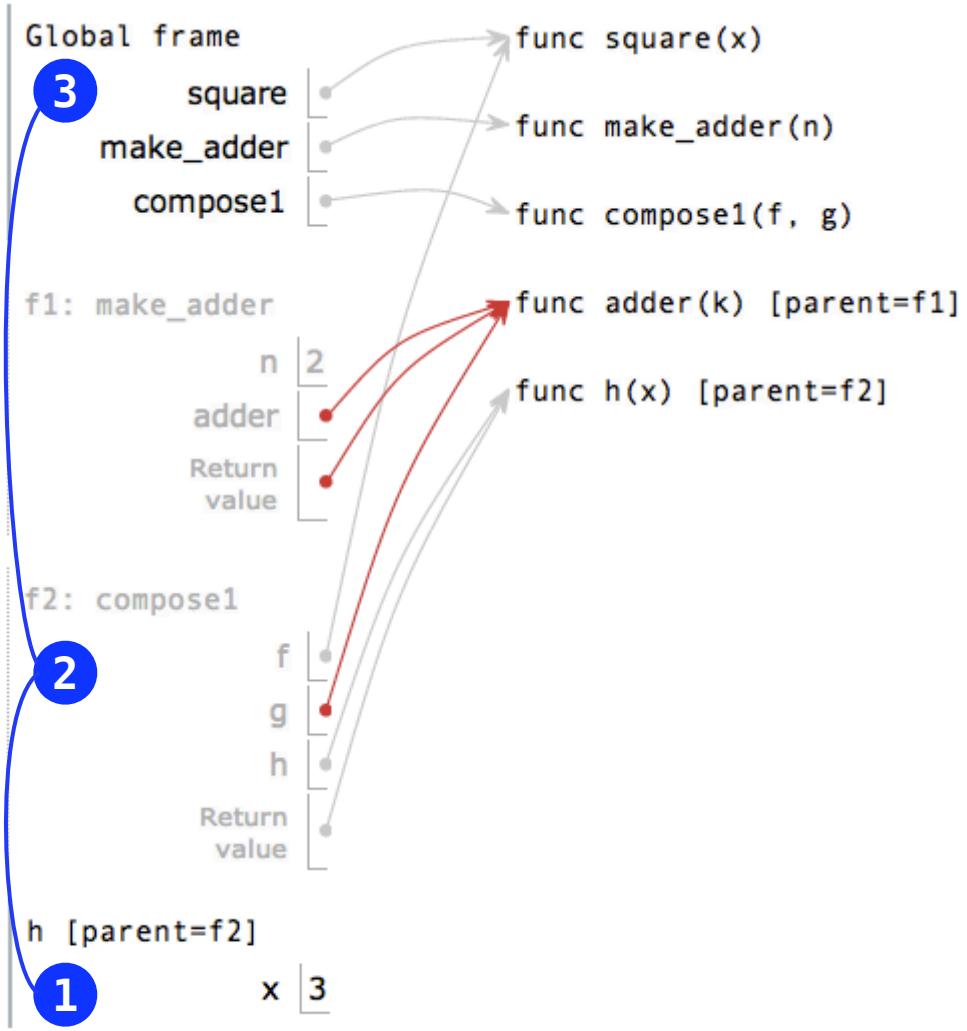


Example: http://goo.gl/2IuEO

The Environment 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

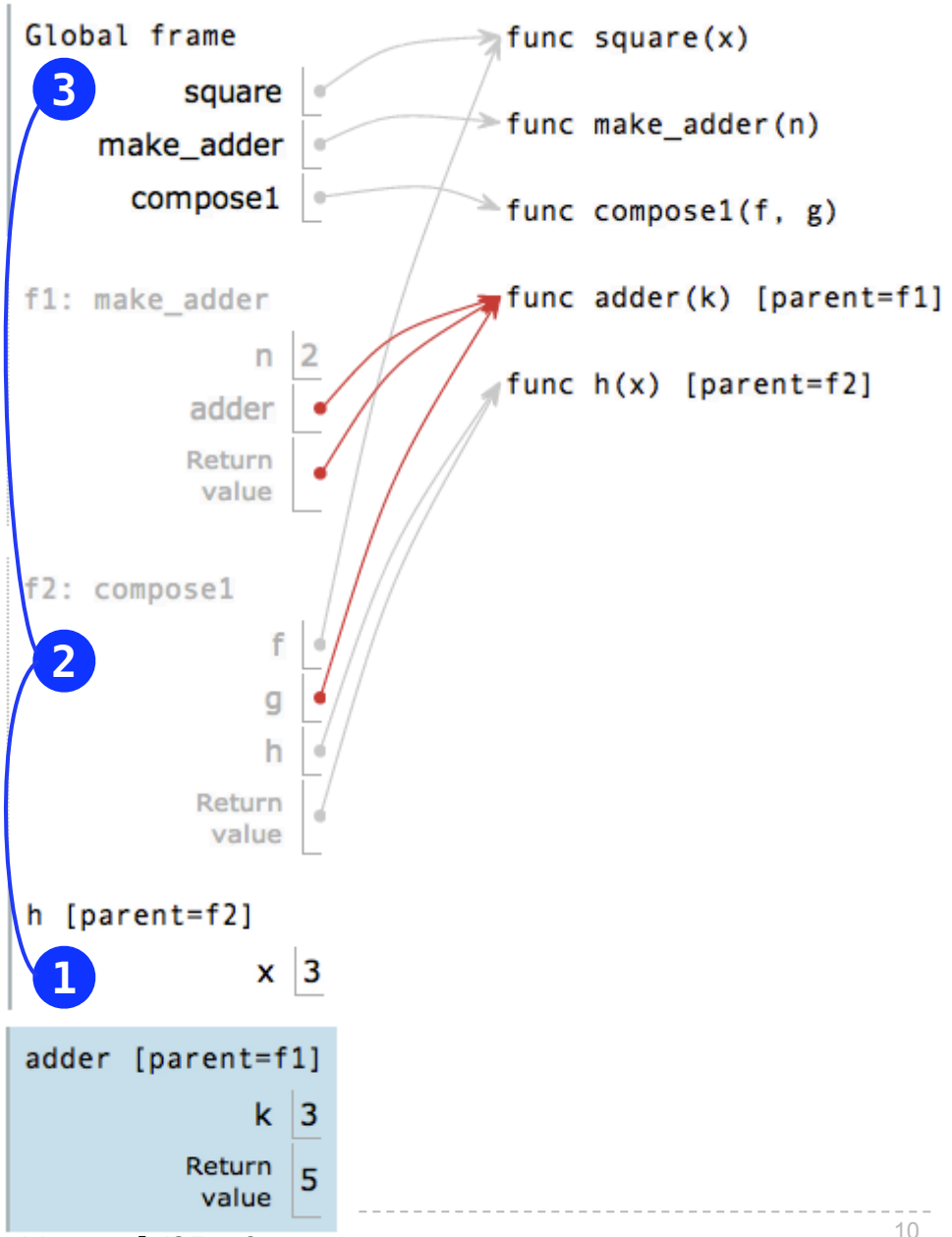


Example: http://goo.gl/2IuEO

The Environment 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

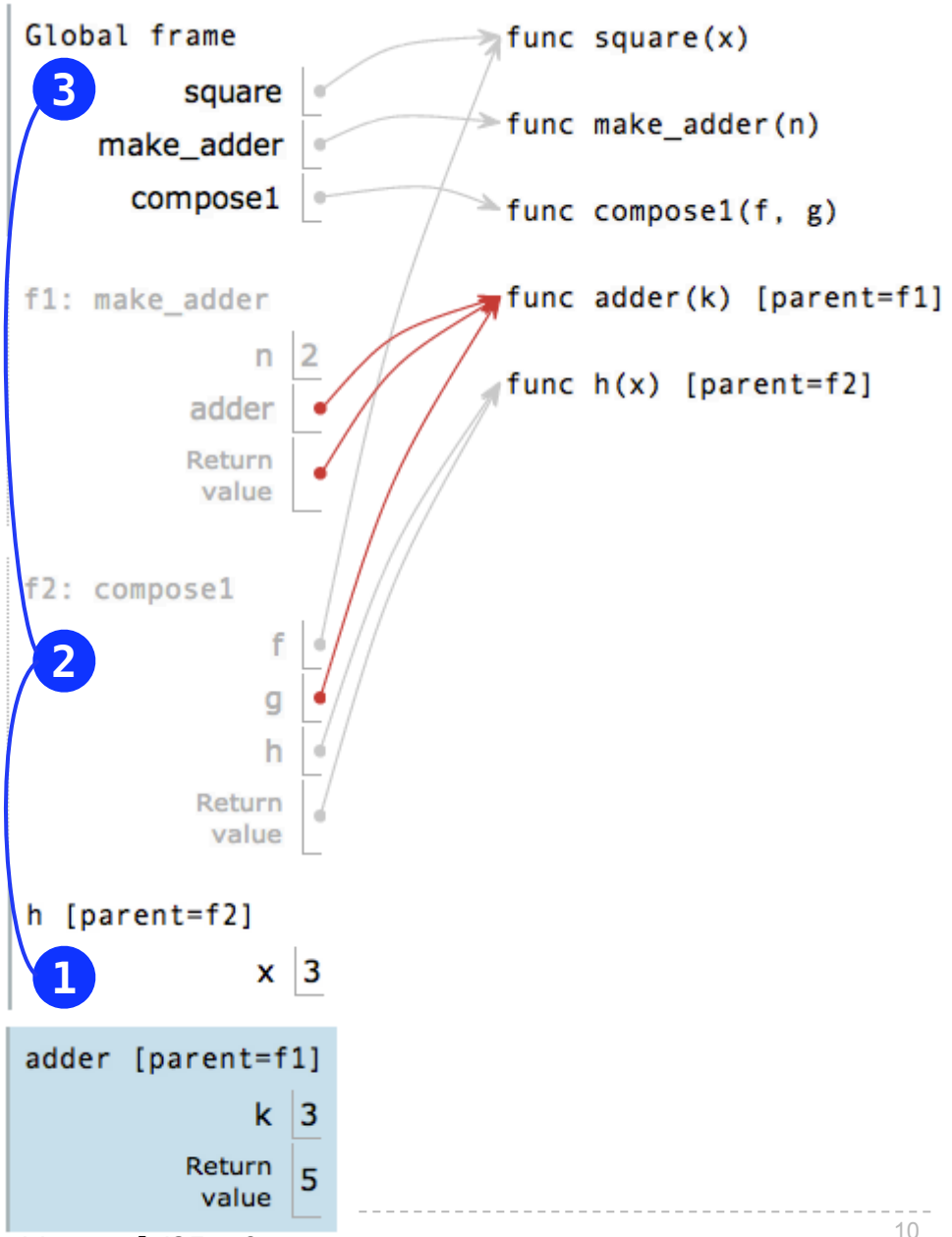


Example: http://goo.gl/2IuEO

The Environment 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

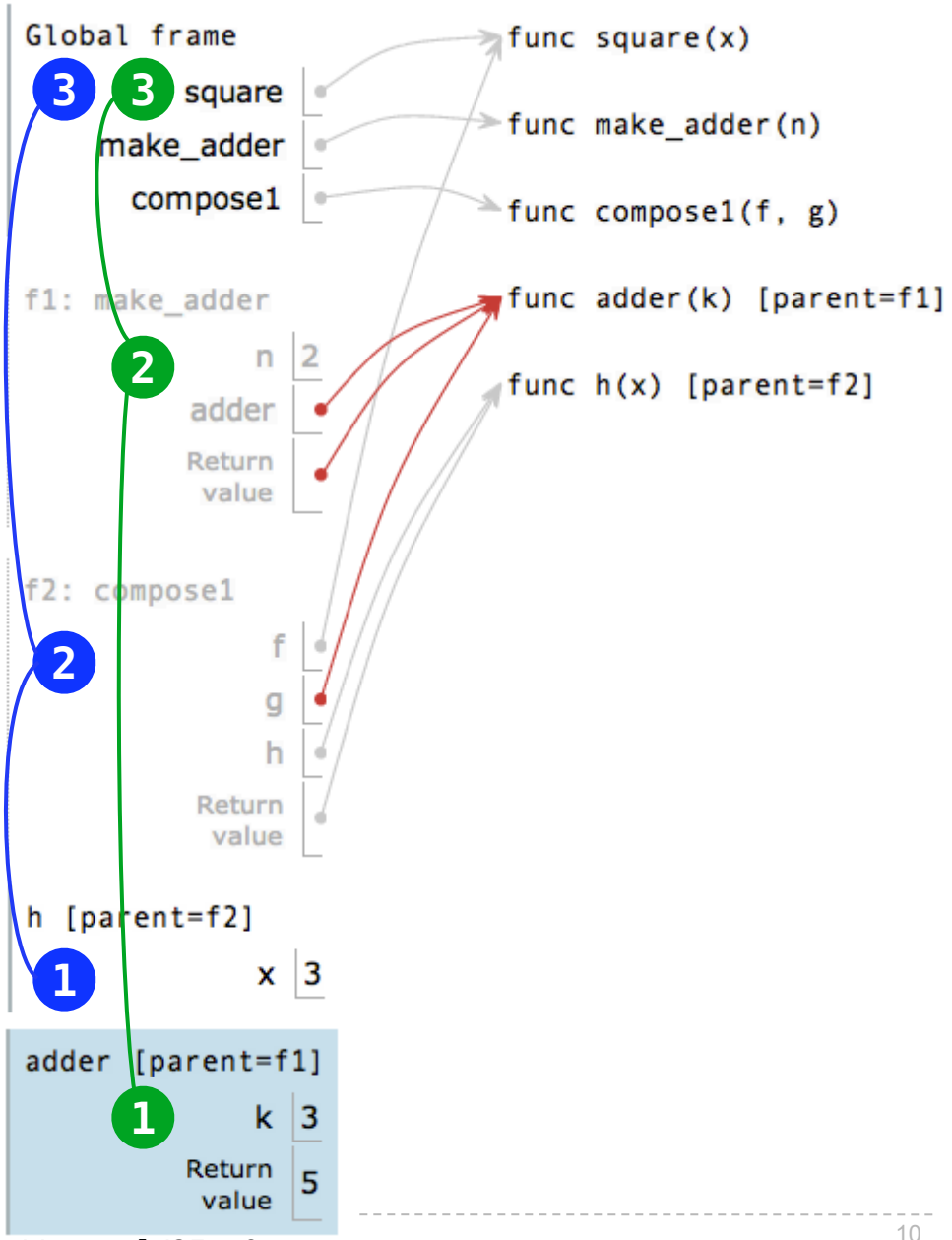


Example: http://goo.gl/2IuEO

The Environment 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



Example: http://goo.gl/2IuEO

Lambda Expressions

Lambda Expressions

```
>>> ten = 10
```

Lambda Expressions

```
>>> ten = 10
```

```
>>> square = x * x
```

Lambda Expressions

```
>>> ten = 10

>>> square = x * x
```

An expression: this one evaluates to a number

Lambda Expressions

```
>>> ten = 10
```

```
>>> square = x * x
```

An expression: this one evaluates to a number

```
>>> square = lambda x: x * x
```

Lambda Expressions

```
>>> ten = 10

>>> square = x * x

>>> square = lambda x: x * x
```

An expression: this one evaluates to a number

Also an expression: evaluates to a function

Lambda Expressions

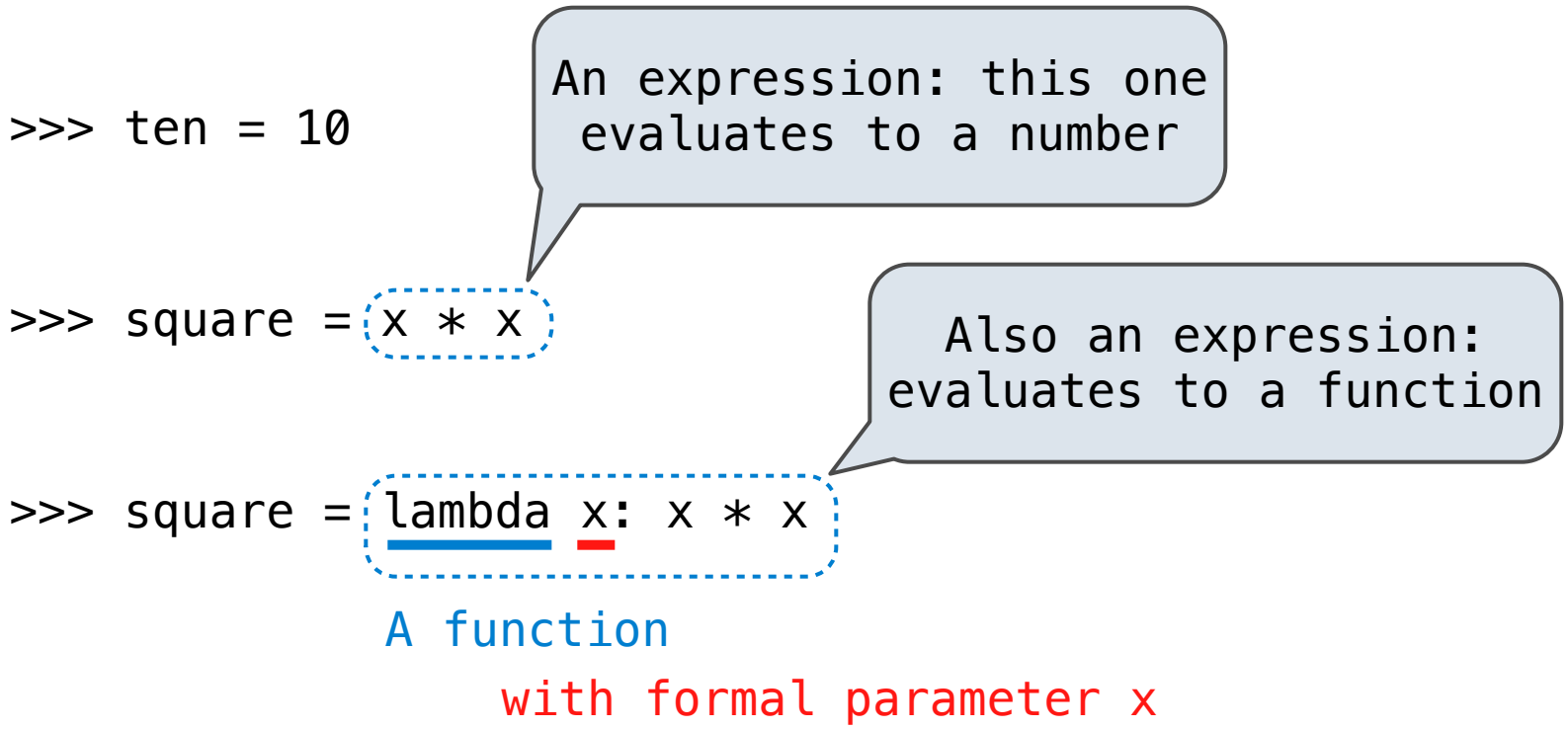
```
>>> ten = 10
>>> square = x * x
>>> square = lambda x: x * x
```

An expression: this one evaluates to a number

Also an expression: evaluates to a function

A function

Lambda Expressions



Lambda Expressions

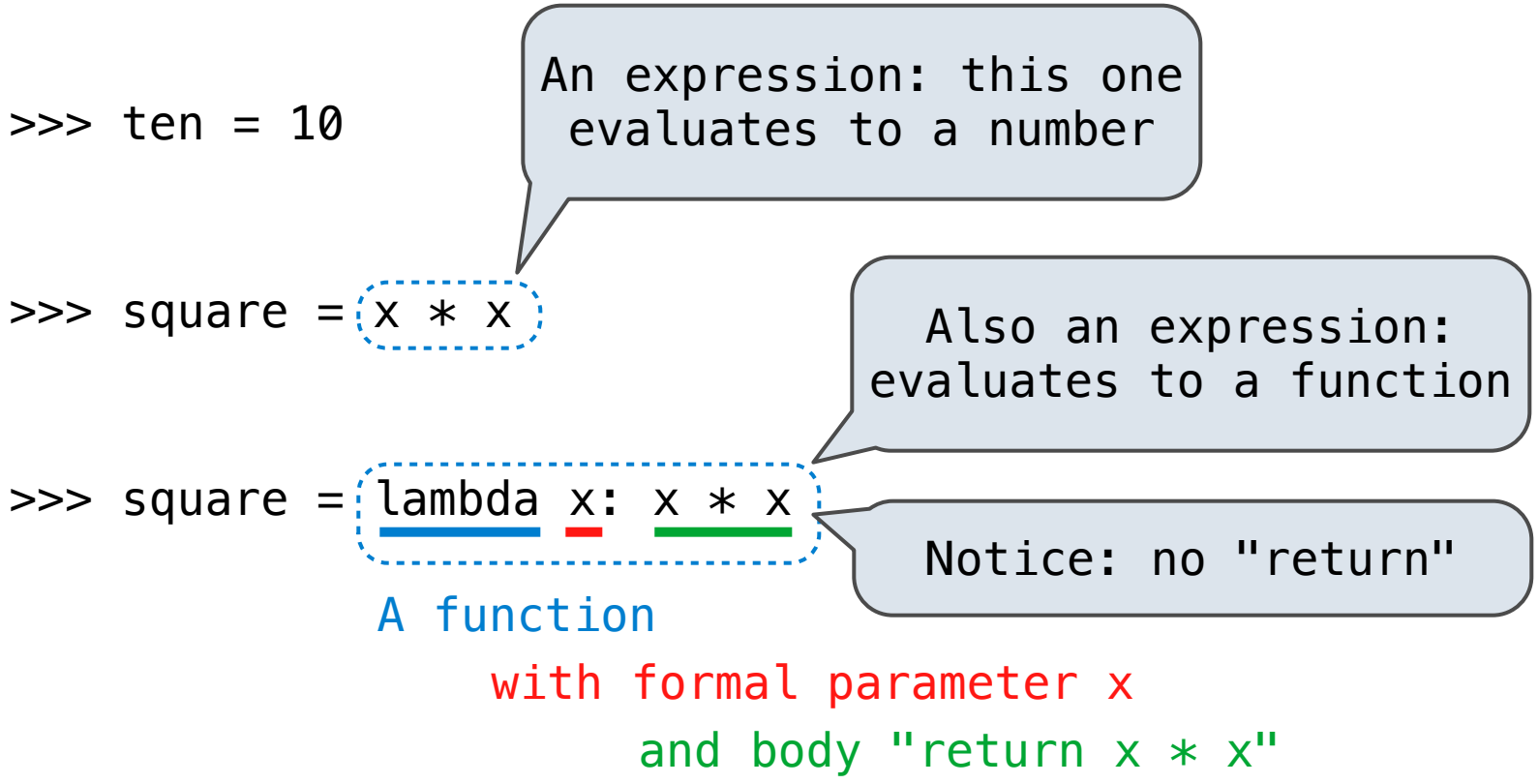
```
>>> ten = 10
>>> square = x * x
>>> square = lambda x: x * x
```

An expression: this one evaluates to a number

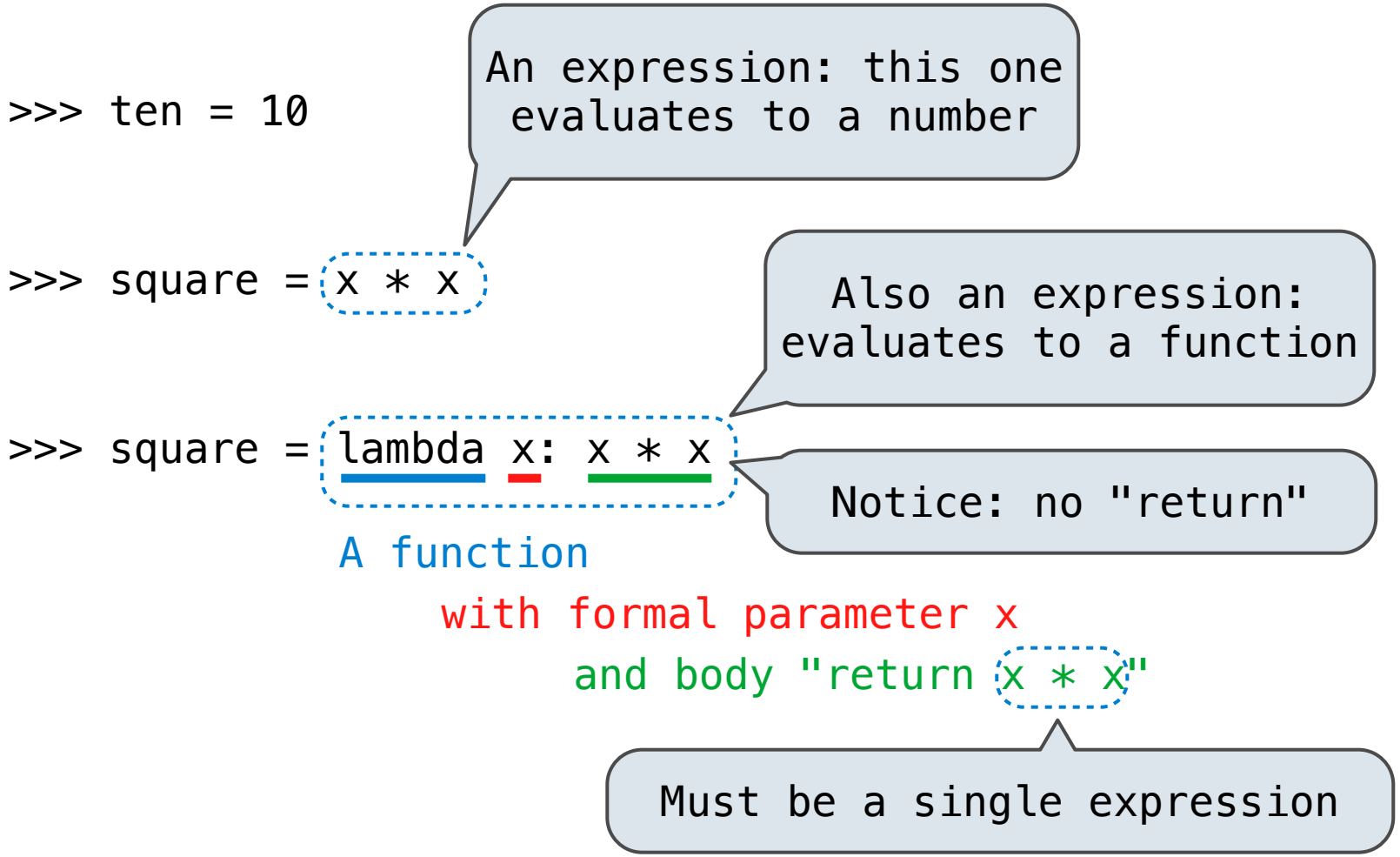
Also an expression: evaluates to a function

A function
with formal parameter x
and body "return x * x"

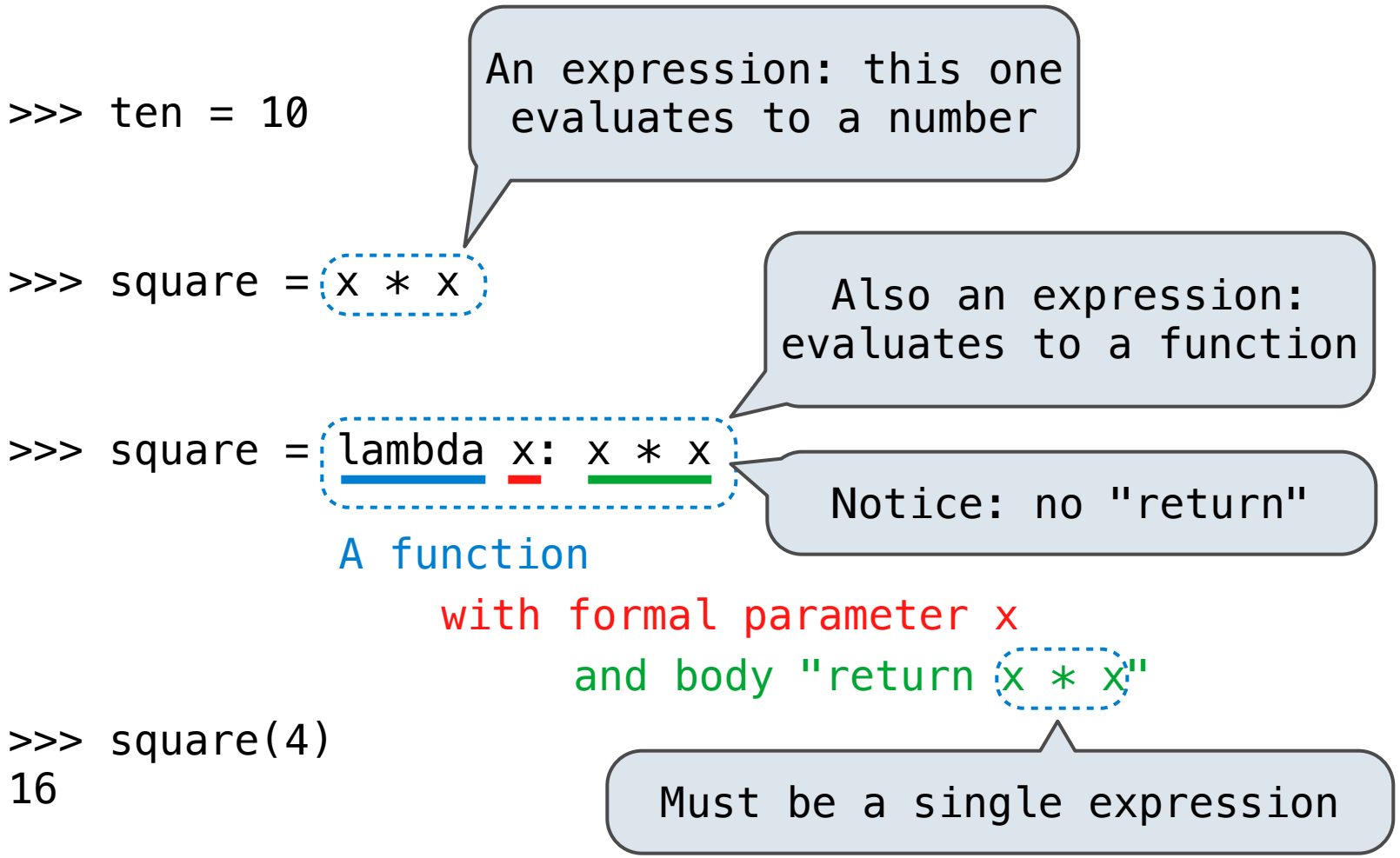
Lambda Expressions



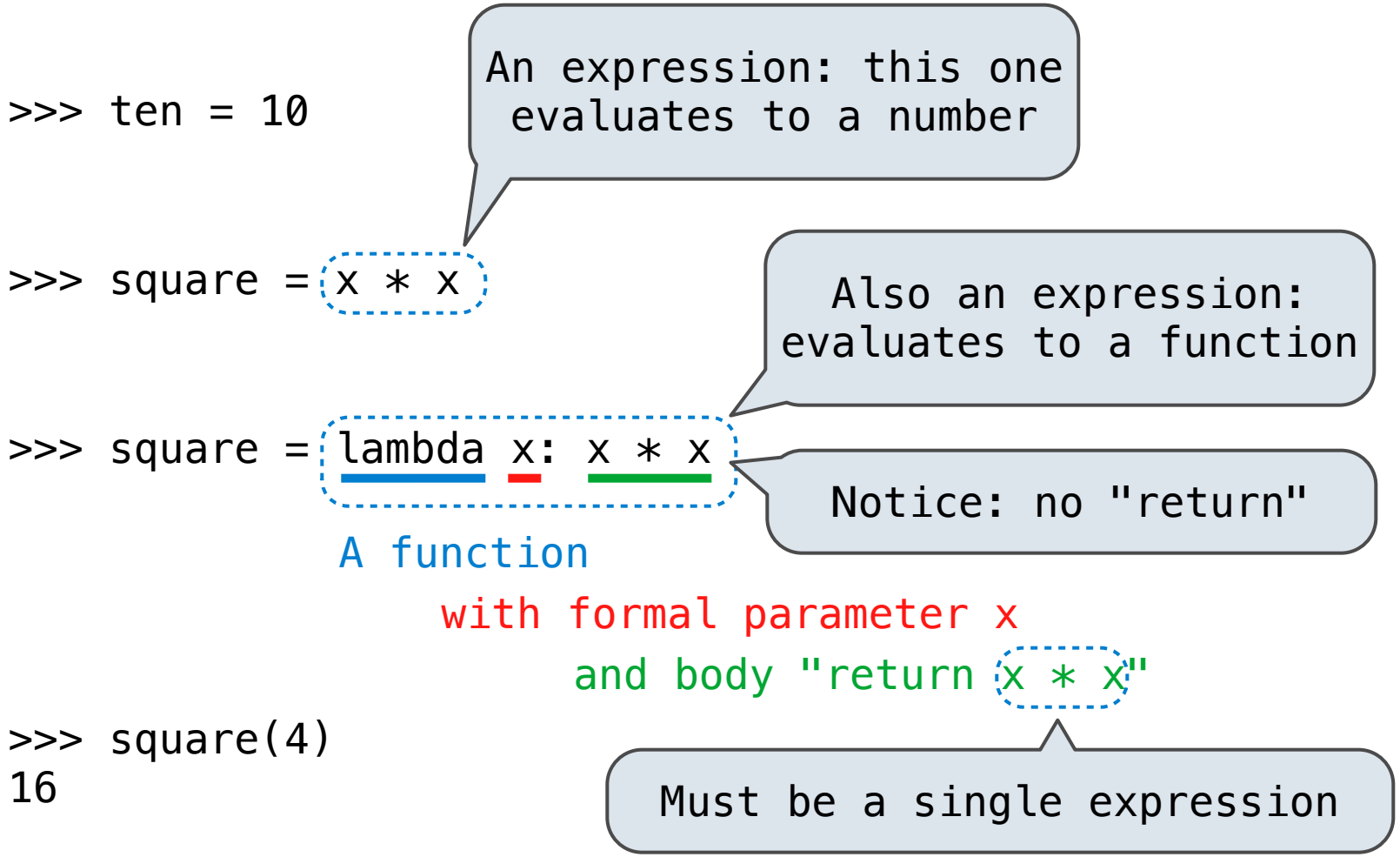
Lambda Expressions



Lambda Expressions



Lambda Expressions



Lambda expressions are rare in Python, but important in general

More Higher-Order Function Examples

(Demo)