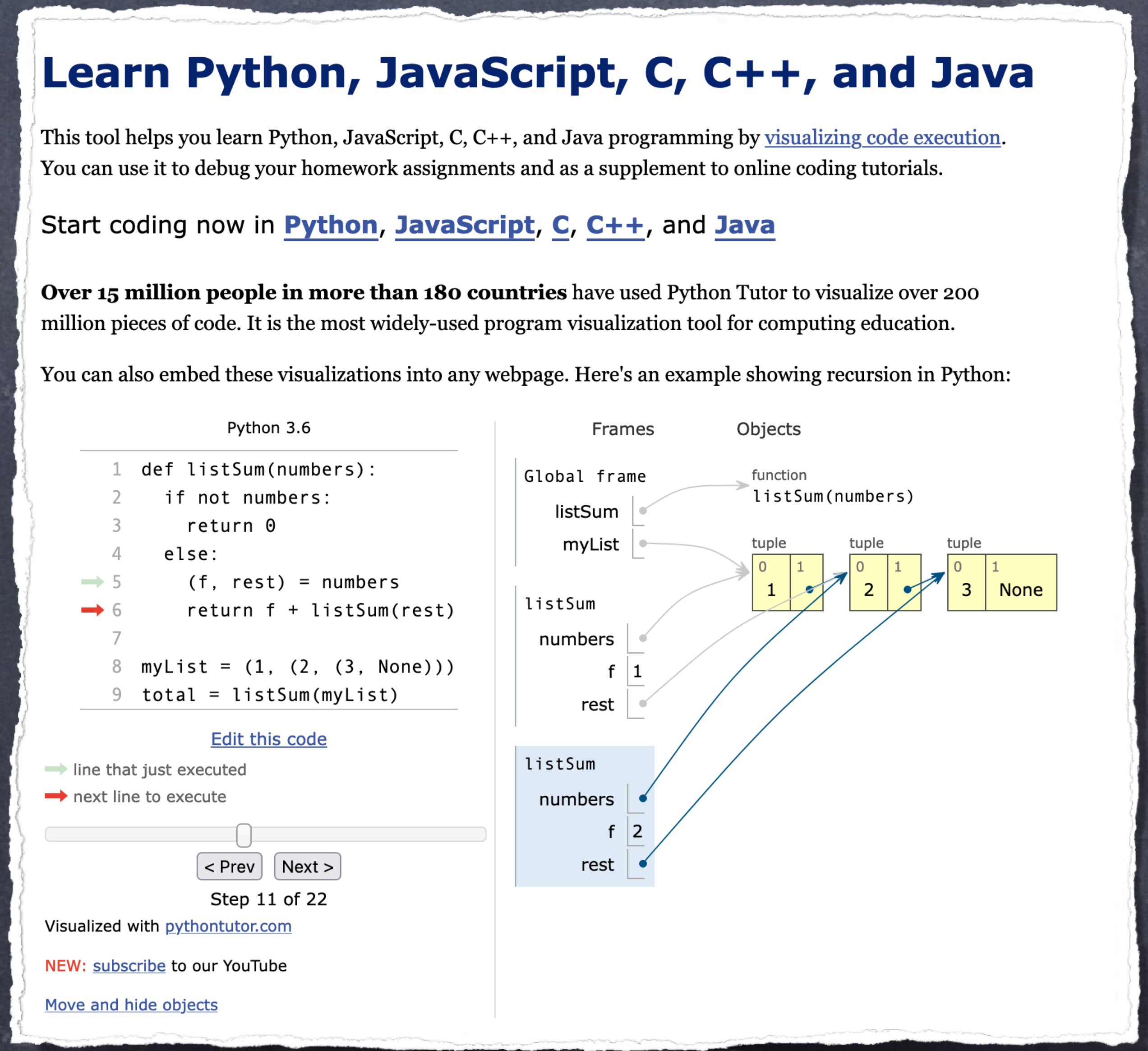
环境图



Learn Python, JavaScript, C, C++, and Java

This tool helps you learn Python, JavaScript, C, C++, and Java programming by visualizing code execution.
You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in Python, JavaScript, C, C++, and Java

Over 15 million people in more than 180 countries have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

Python 3.6

```
def listSum(numbers):
  if not numbers:
    return 0
  else:
    (f, rest) = numbers
    return f + listSum(rest)

myList = (1, (2, (3, None)))
total = listSum(myList)
```

Edit this code

line that just executed

next line to execute

< Prev   Next >

Step 11 of 22

Visualized with pythontutor.com

NEW: subscribe to our YouTube

Move and hide objects

https://pythontutor.com/cp/composingprograms.html

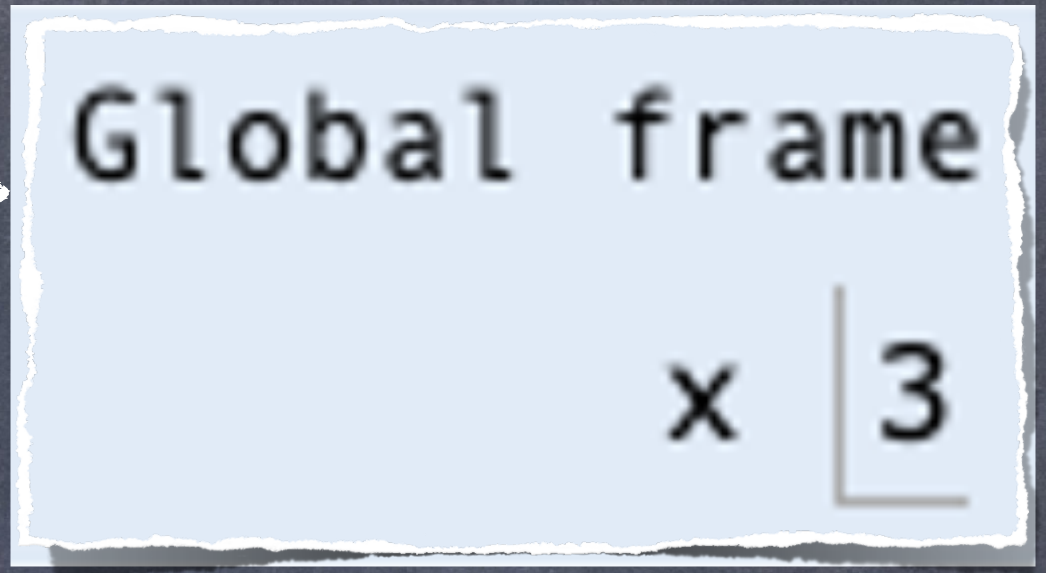
什么是环境图

- 一个可以追踪（Track）一个程序运行时的绑定和状态变化的可视化工具
  - 可以帮助我们理解程序的工作方式。
  - 有助于Debugging。
  - 由于其普适性，有助于学习后续课程（如编译原理）。

回顾

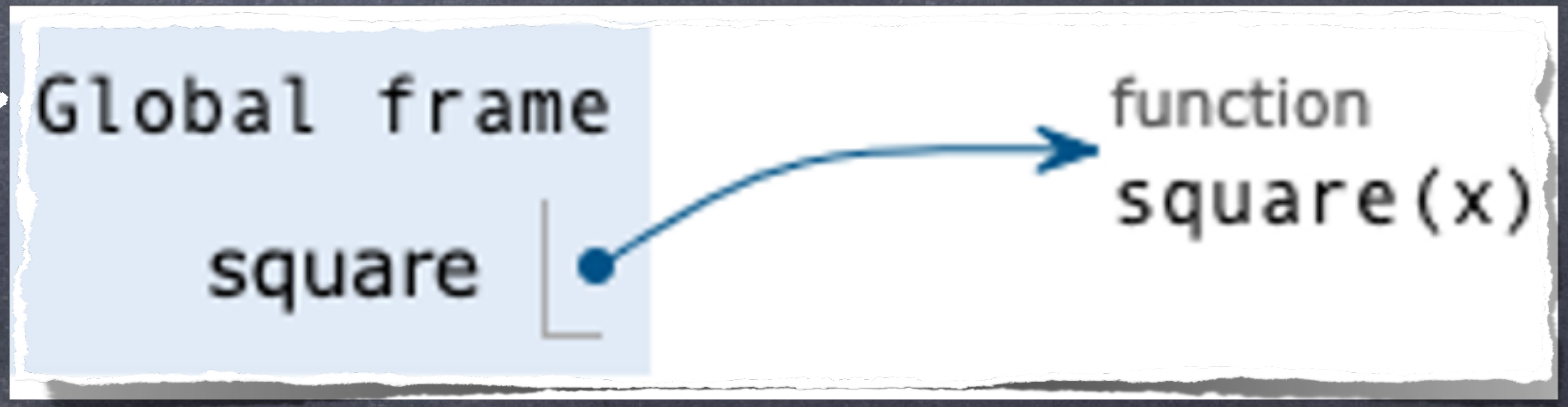
- 赋值语句

```
x = 3
```



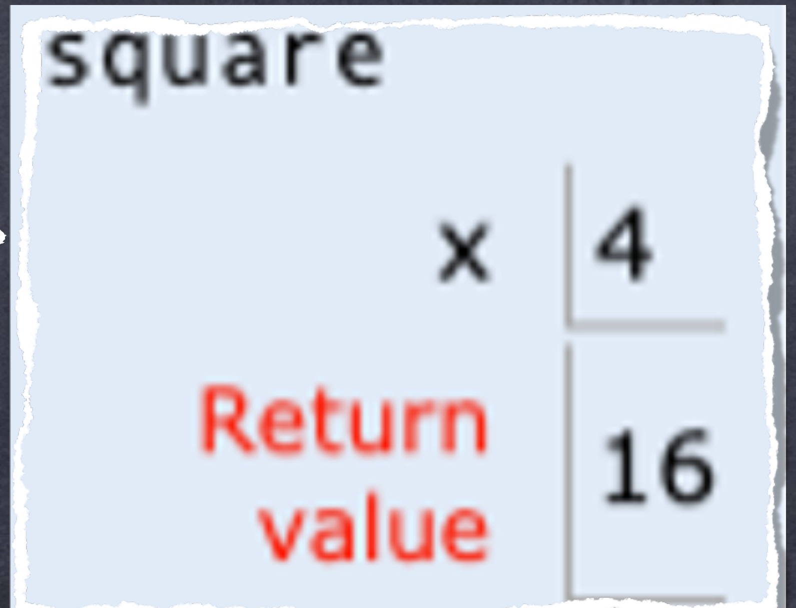
- Def 语句

```
def square(x):
        Return x * x
```



- 调用语句

```
square(3)
```



帧（Frames）

- 帧追踪着名字和值的绑定
  - 每一个函数调用都会有一个对应的帧
- 全局帧（Global Frame），就是最开始的帧
  - 它不对应函数调用
- 父帧（Parent Frame）
  - 某个函数的父帧就是定义这个函数语句的所在的帧
  - 如果你找不到一个变量名，你需要找到其父帧，如果还找不到，往其父帧的父帧查找，一直到全局帧。此时再找不到就是一个Name Error

变量查找

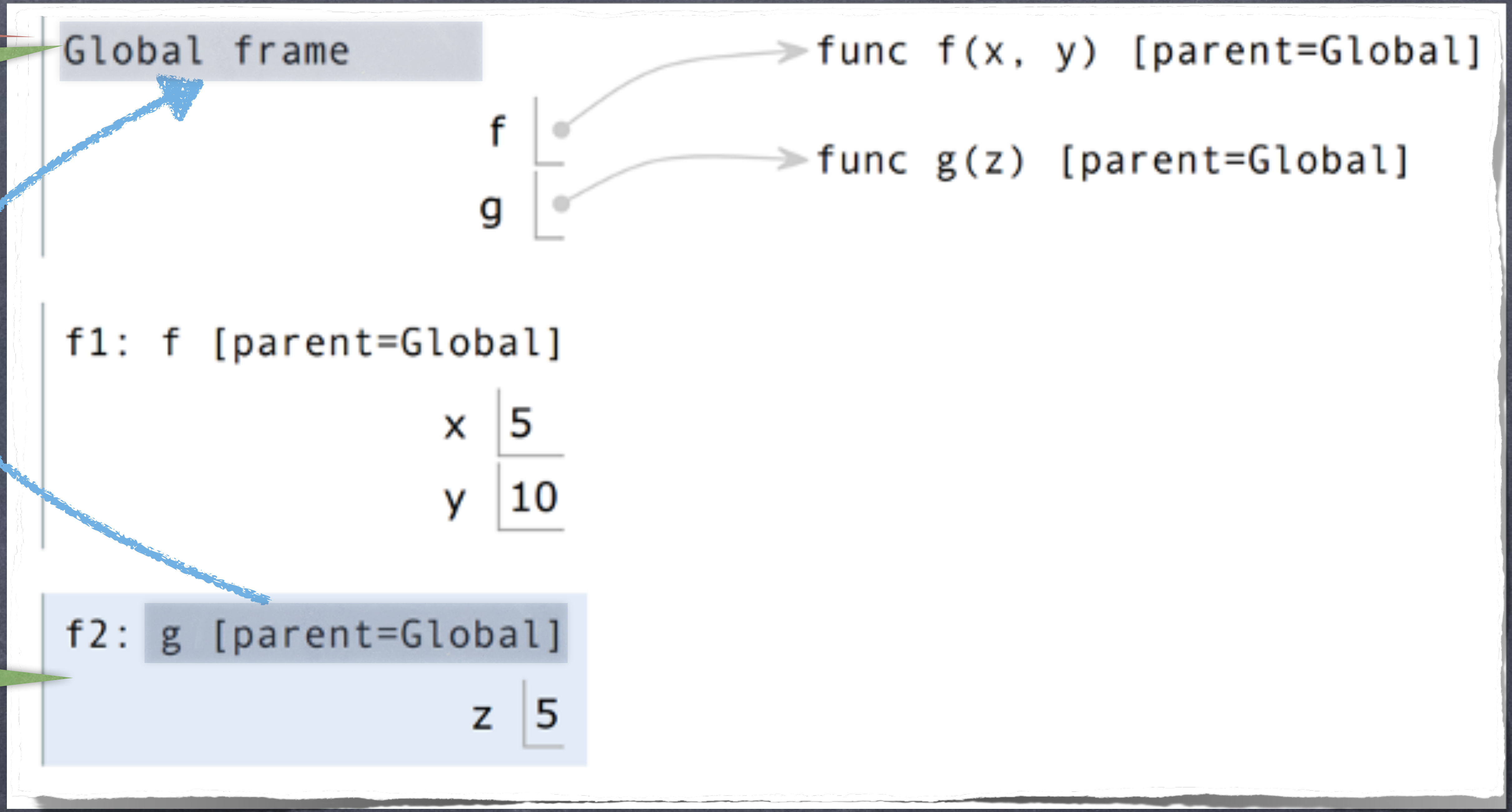
```
def f(x, y):
    return g(x)

def g(z):
    return z + x

result = f(5, 10)
```

An error is thrown

Name "x" is not found again

🤔下面我们需要在哪查找?

Name "x" is not found

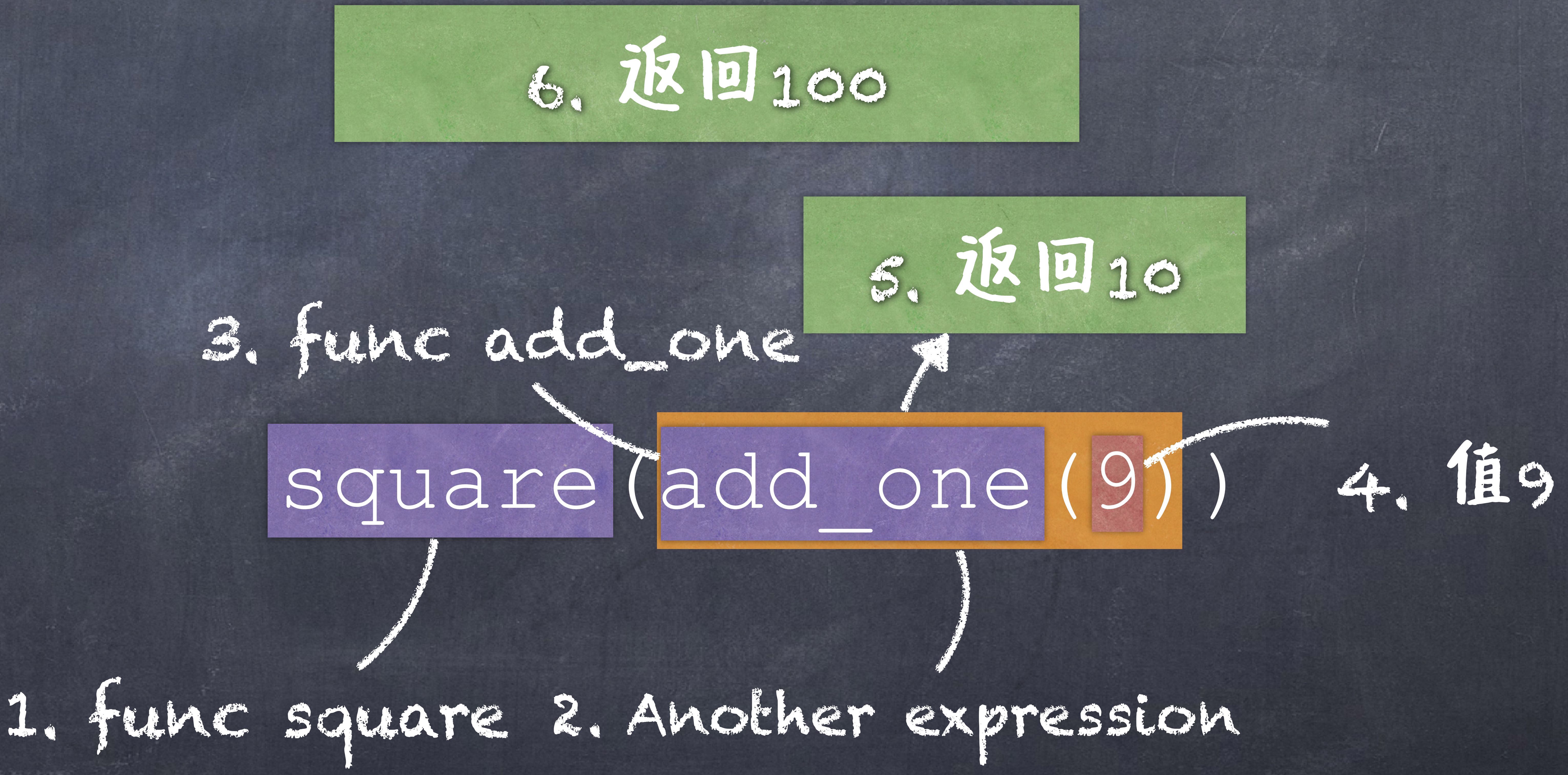


⚠️注意：我们没有在f1里查找x，这是因为f2的父帧是Global。

帧的创建和求值顺序

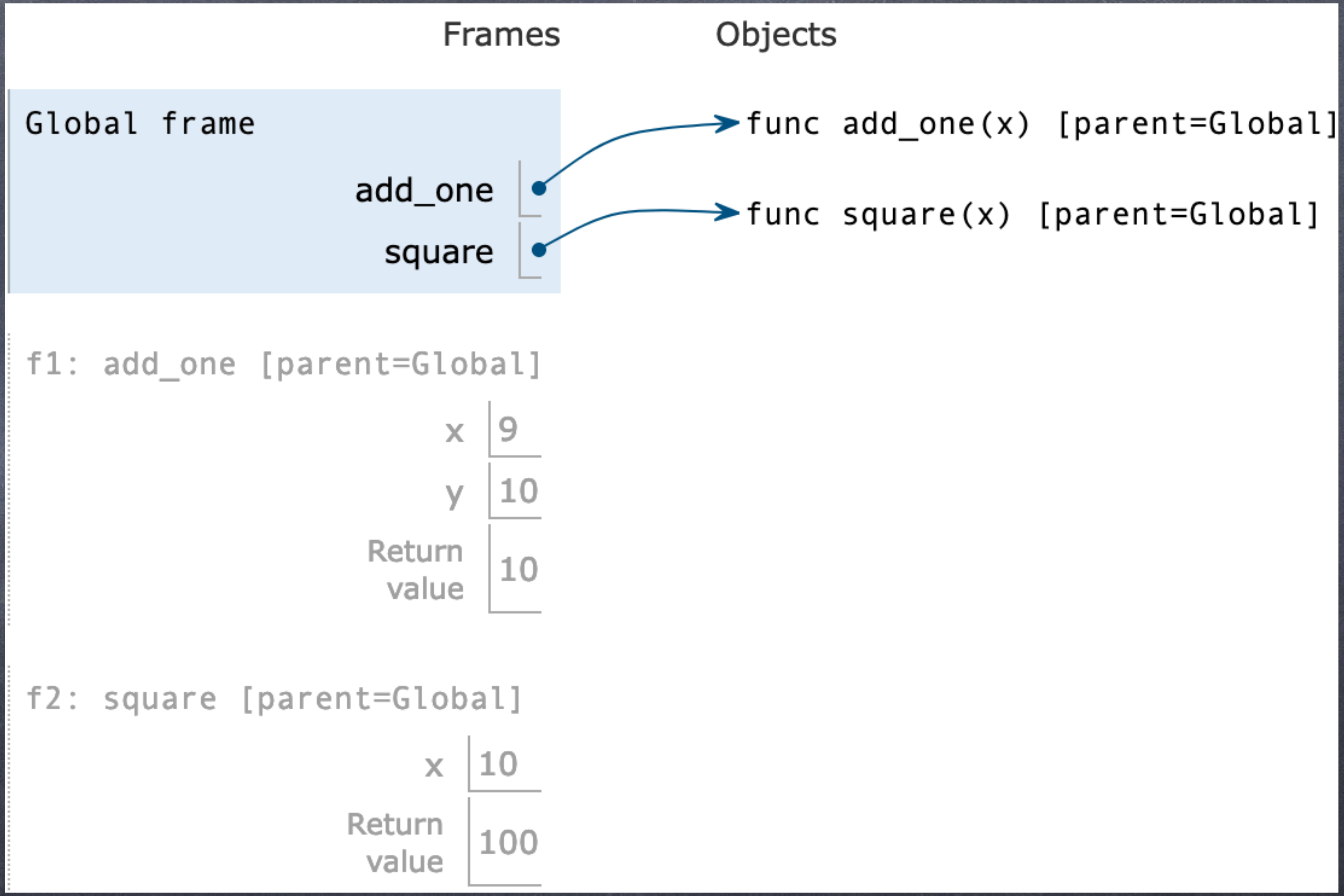
- 回顾求值规则：先对运算符求值、然后操作数，最后执行应用

```python
def add_one(x):
    y = x + 1
    return y

def square(x):
    return x*x

square(add_one(9))
```



帧的创建和求值顺序

```
def add_one(x):
    y = x + 1
    return y

def square(x):
    return x*x

square(add_one(9))
```


Frames
Objects
Global frame
add_one
square
func add_one(x) [parent=Global]
func square(x) [parent=Global]
f1: add_one [parent=Global]
x 9
y 10
Return value 10
f2: square [parent=Global]
x 10
Return value 100

Lambda表达式

Lambda表达式

- 求值为函数的表达式

一个以x为参数，返回x*x的函数

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

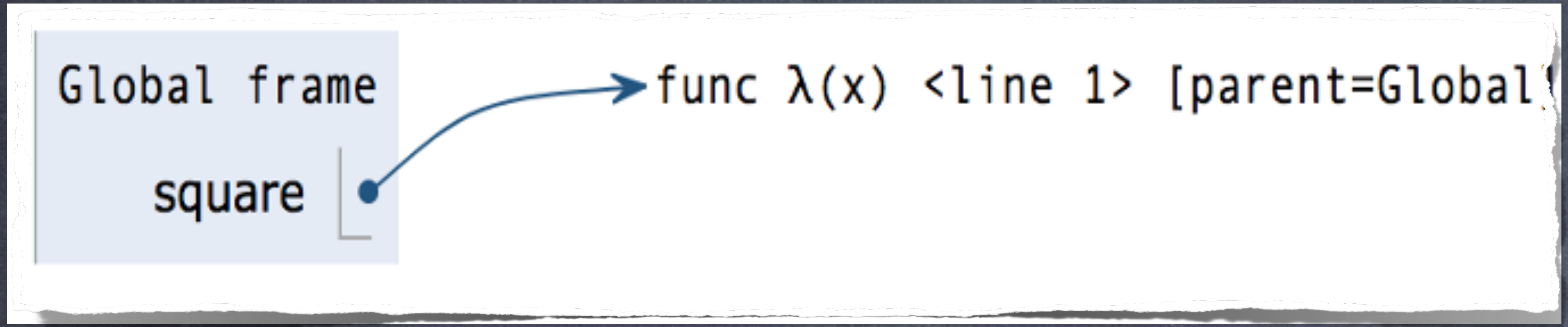
```
>>> square
<function <lambda> ... >
```

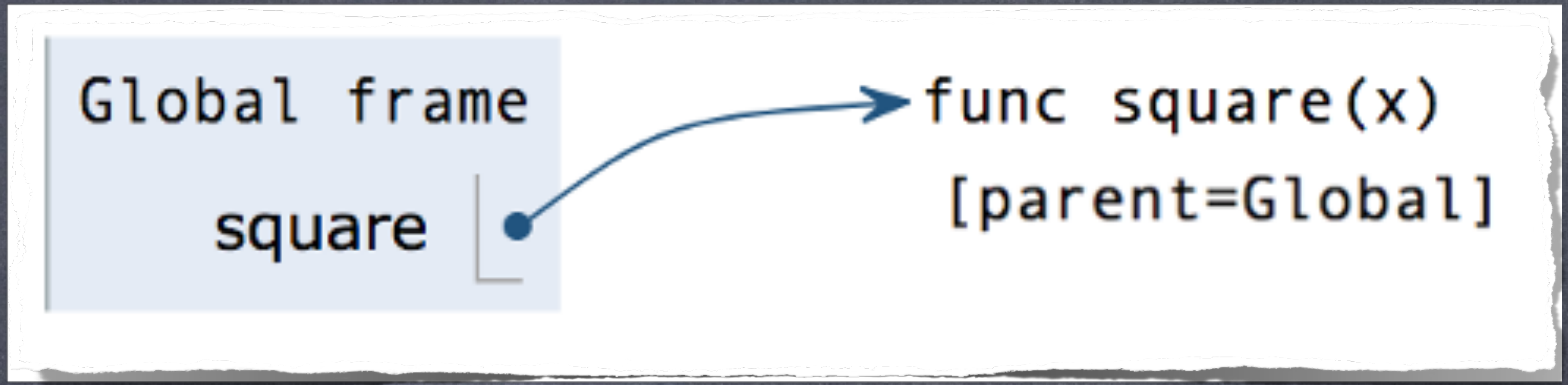
```
>>> square(5)
25
```

Lambda表达式 VS Def语句

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

```
def square(x):
    Return x * x
```





- 定义的函数行为一样
- 父帧都是定义处所在帧
- 都绑定了相同的名字

但只有def语句有intrinsic名，lambda表达式没有（匿名函数）

Lambda表达式 VS Def语句

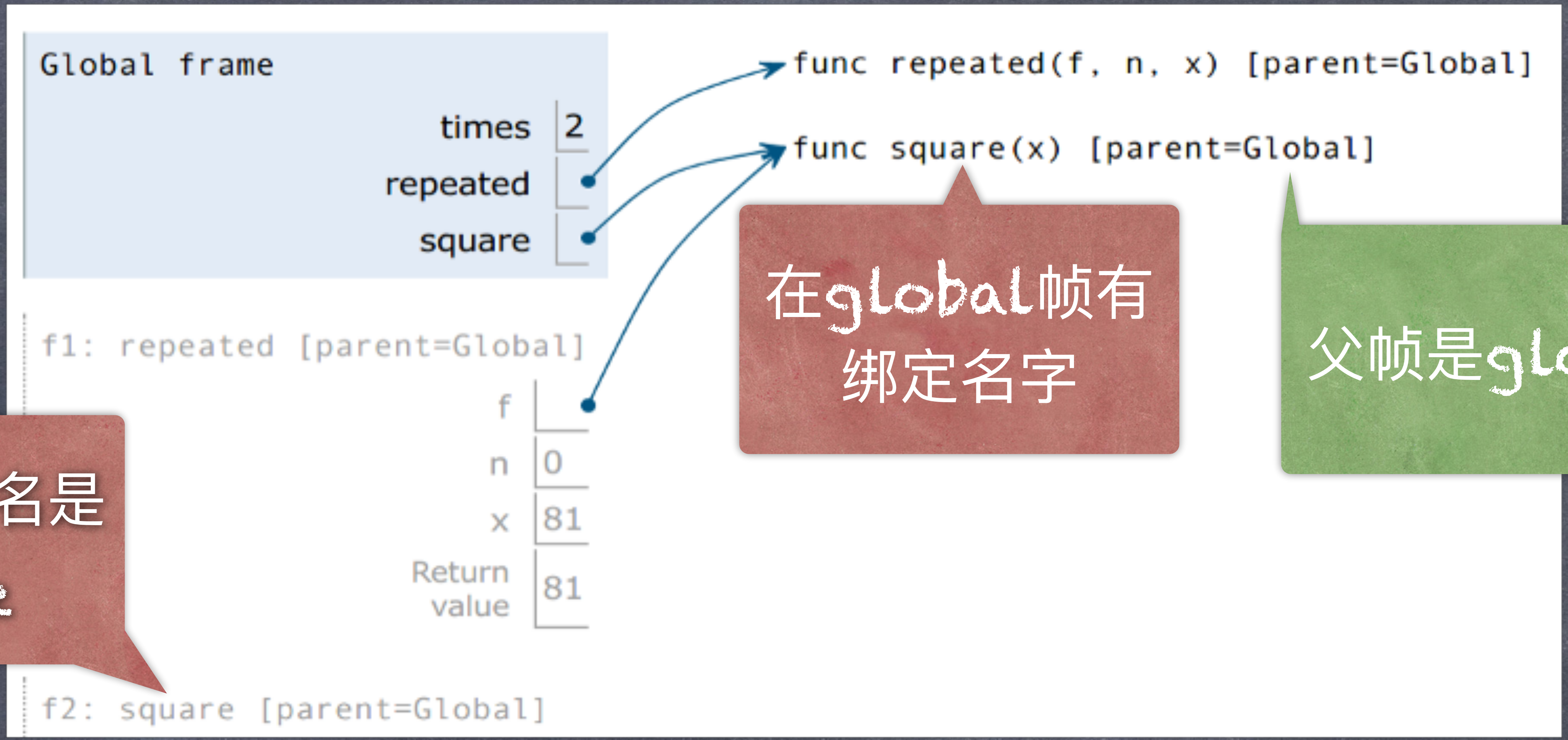
```python
times = 2

def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x

def square(x):
  return x * x

repeated(square, times, 3)
```

```python
times = 2

def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x

repeated(lambda x: x*x, times, 3)
```

Lambda表达式 VS Def语句

```
times = 2

def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x

def square(x):
  return x * x

repeated(square, times, 3)
```

intrinsic名是square

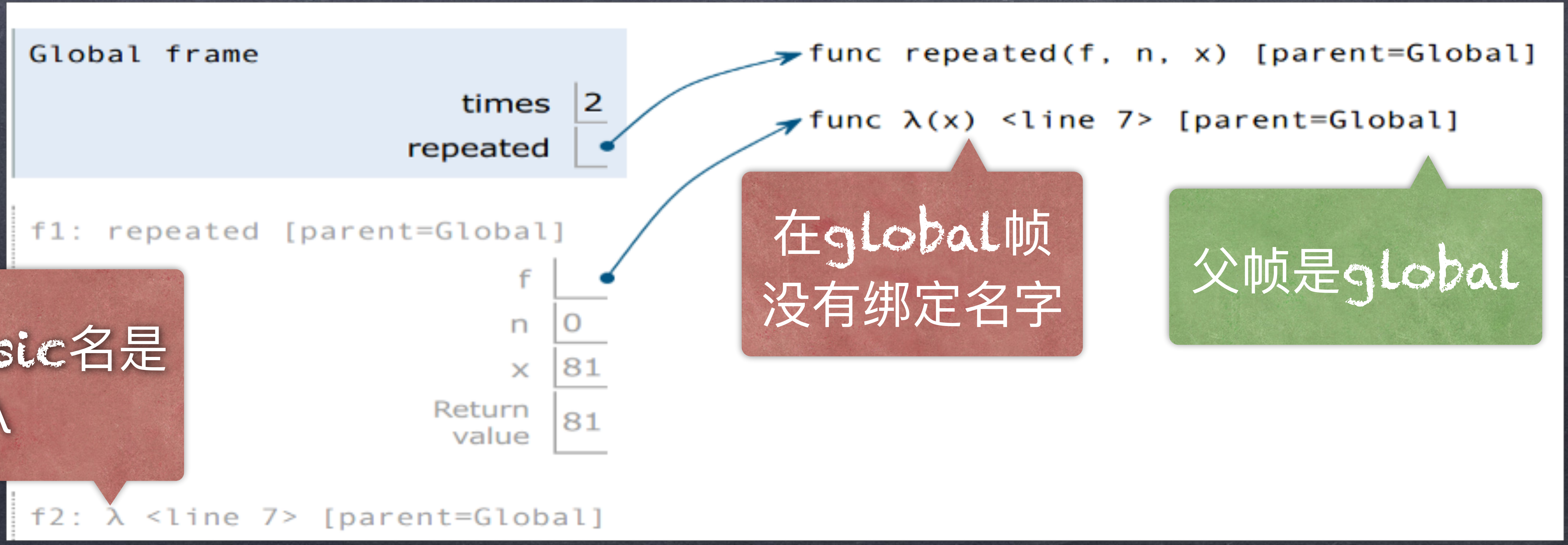

Global frame
times 2
repeated
square
func repeated(f, n, x) [parent=Global]
func square(x) [parent=Global]
f1: repeated [parent=Global]
f
n 0
x 81
Return value 81
f2: square [parent=Global]

在global帧有绑定名字

父帧是global

```
times = 2

def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x

repeated(lambda x: x*x, times, 3)
```

intrinsic名是λ


Global frame
times 2
repeated
func repeated(f, n, x) [parent=Global]
func λ(x) <line 7> [parent=Global]
f1: repeated [parent=Global]
f
n 0
x 81
Return value 81
f2: λ <line 7> [parent=Global]

在global帧没有绑定名字

父帧是global

Any questions ?