

Control

Announcements

- Lecture tomorrow will be held in Soda 306
 - Unfortunately, we cannot accommodate everyone to attend live-lecture
 - A form will be released tomorrow morning to reserve a spot at 8 am
 - First 90 students
 - We'll also the option of joining live-lecture remotely
- Technical OH for lab are today
 - Warren 101B (section A)
 - Some hours are offered online as well
- Search for Teammates Ed post #111

Print and None

(Demo)

None Indicates that Nothing is Returned

The special value **None** represents nothing in Python

A function that does not explicitly return a value will return **None**

Careful: **None** is *not displayed* by the interpreter as the value of an expression

```
>>> def does_not_return_square(x):
```

```
...     x * x
... 
```

No return

```
>>> does_not_return_square(4)
```

None value is not displayed

```
>>> sixteen = does_not_return_square(4)
```

```
>>> sixteen + 4
```

The name **sixteen** is now bound to the value **None**

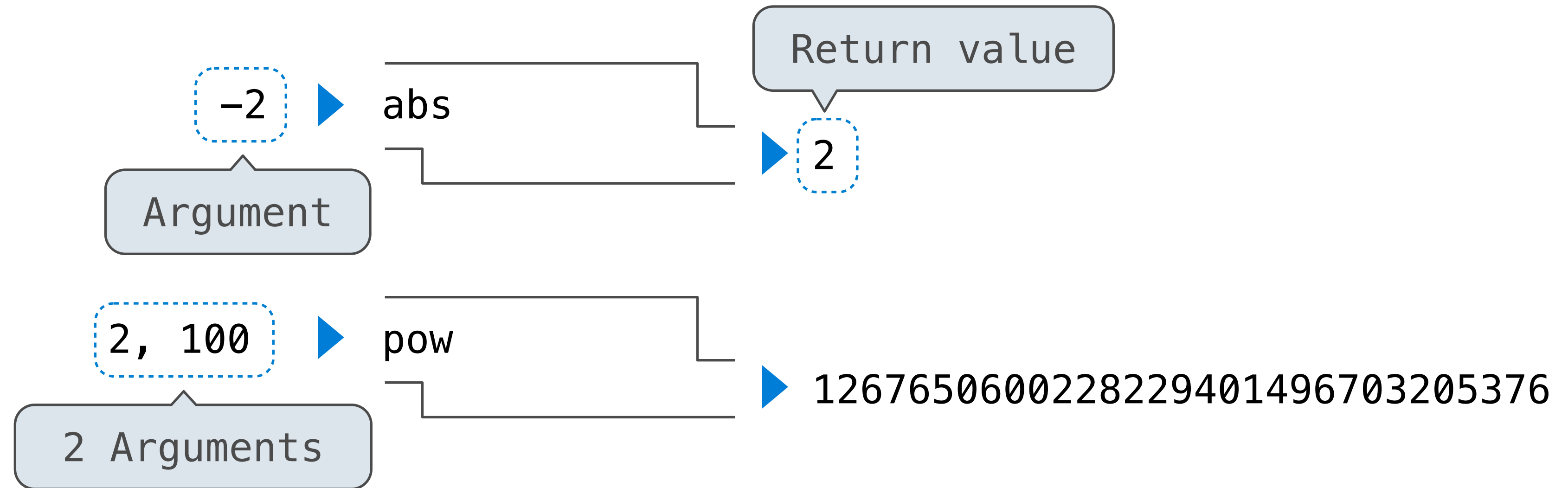
```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

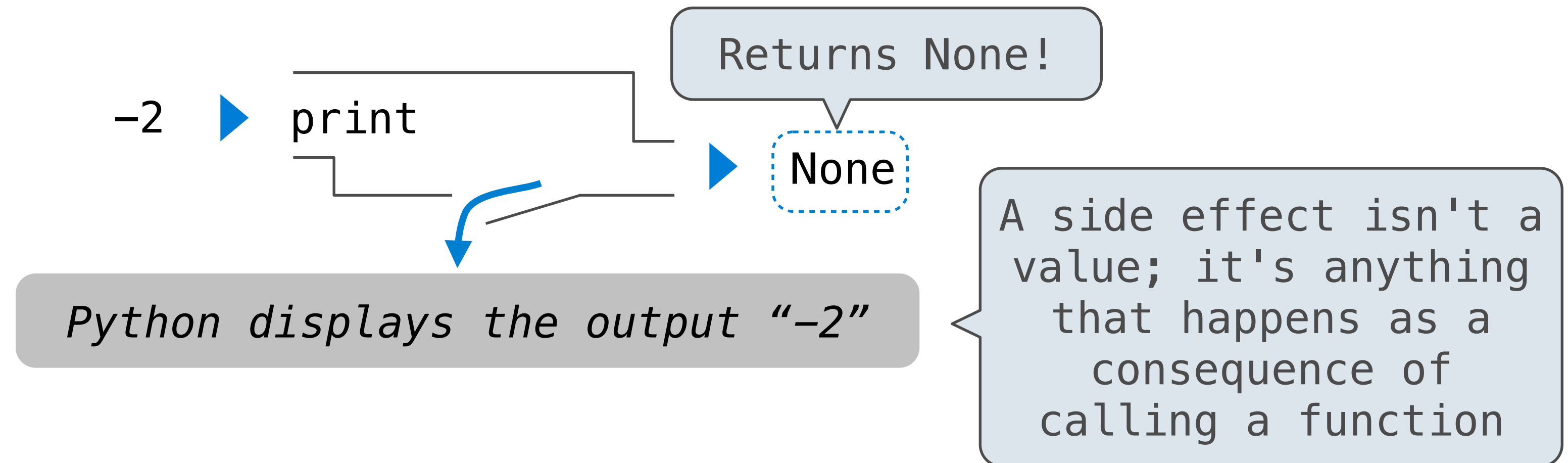
```
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Pure Functions & Non-Pure Functions

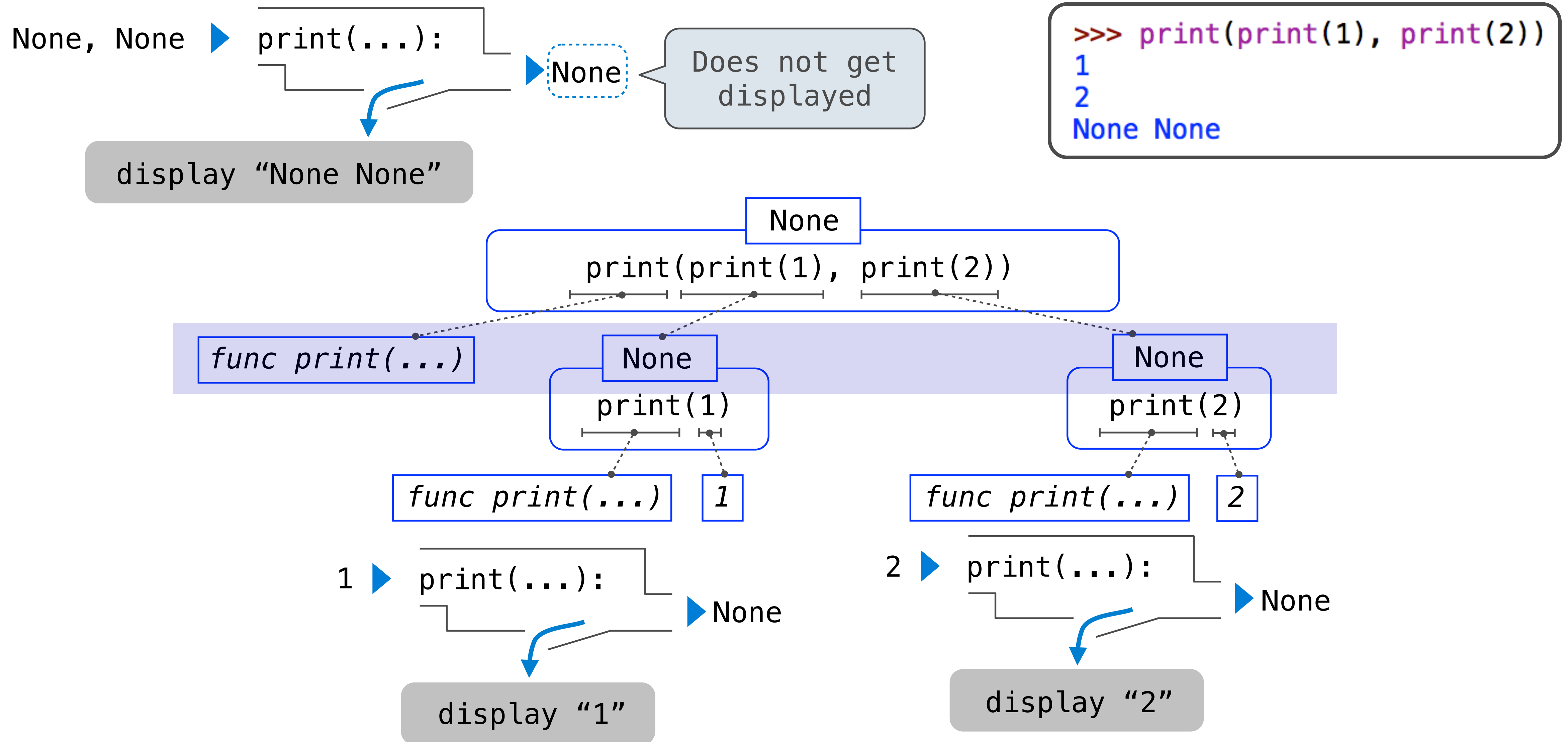
Pure Functions
just return values



Non-Pure Functions
have side effects

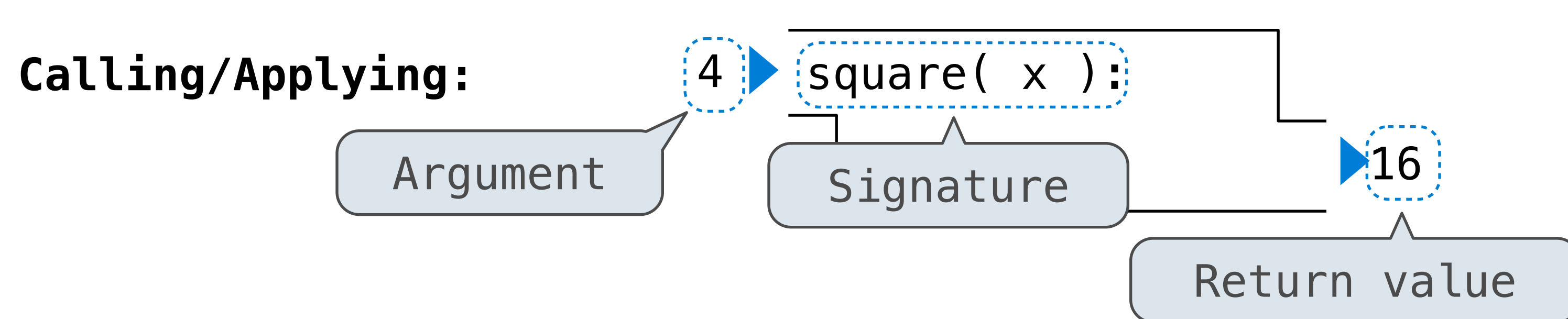
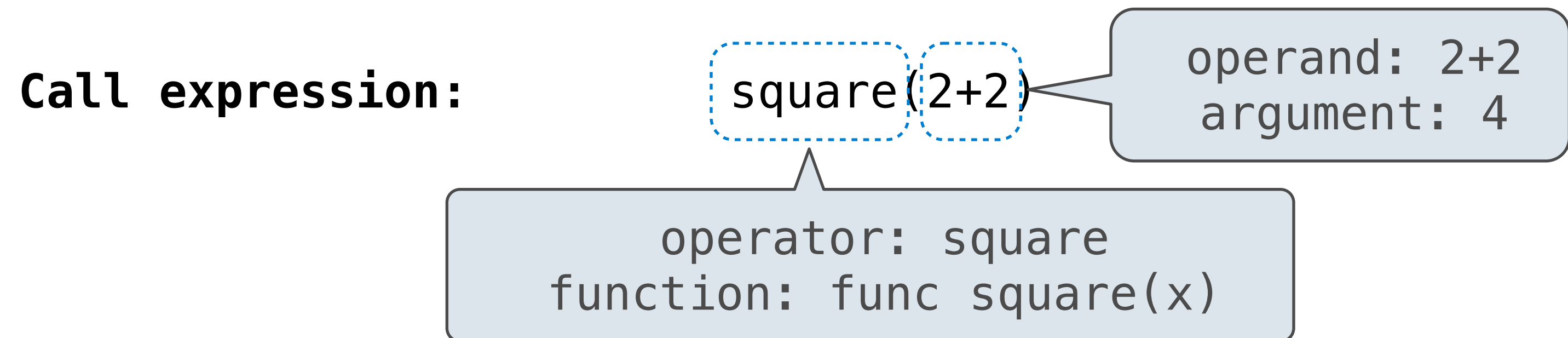
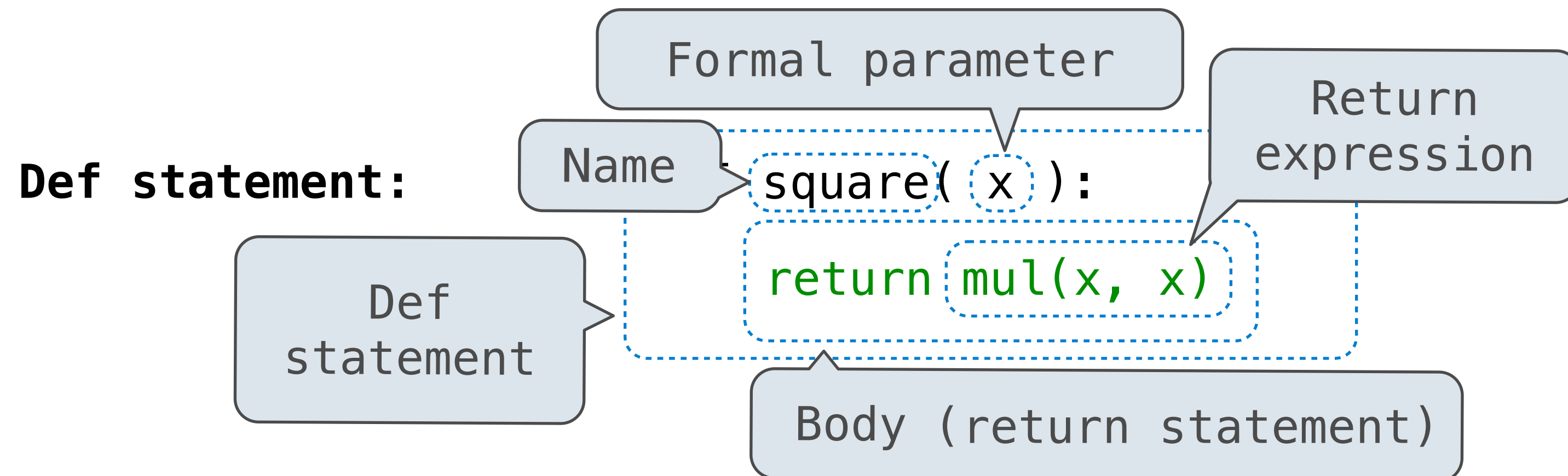


Nested Expressions with Print



Multiple Environments

Life Cycle of a User-Defined Function



What happens?

A new function is created!

Name bound to that function
in the current frame

Operator & operands evaluated

Function (value of operator)
called on arguments
(values of operands)

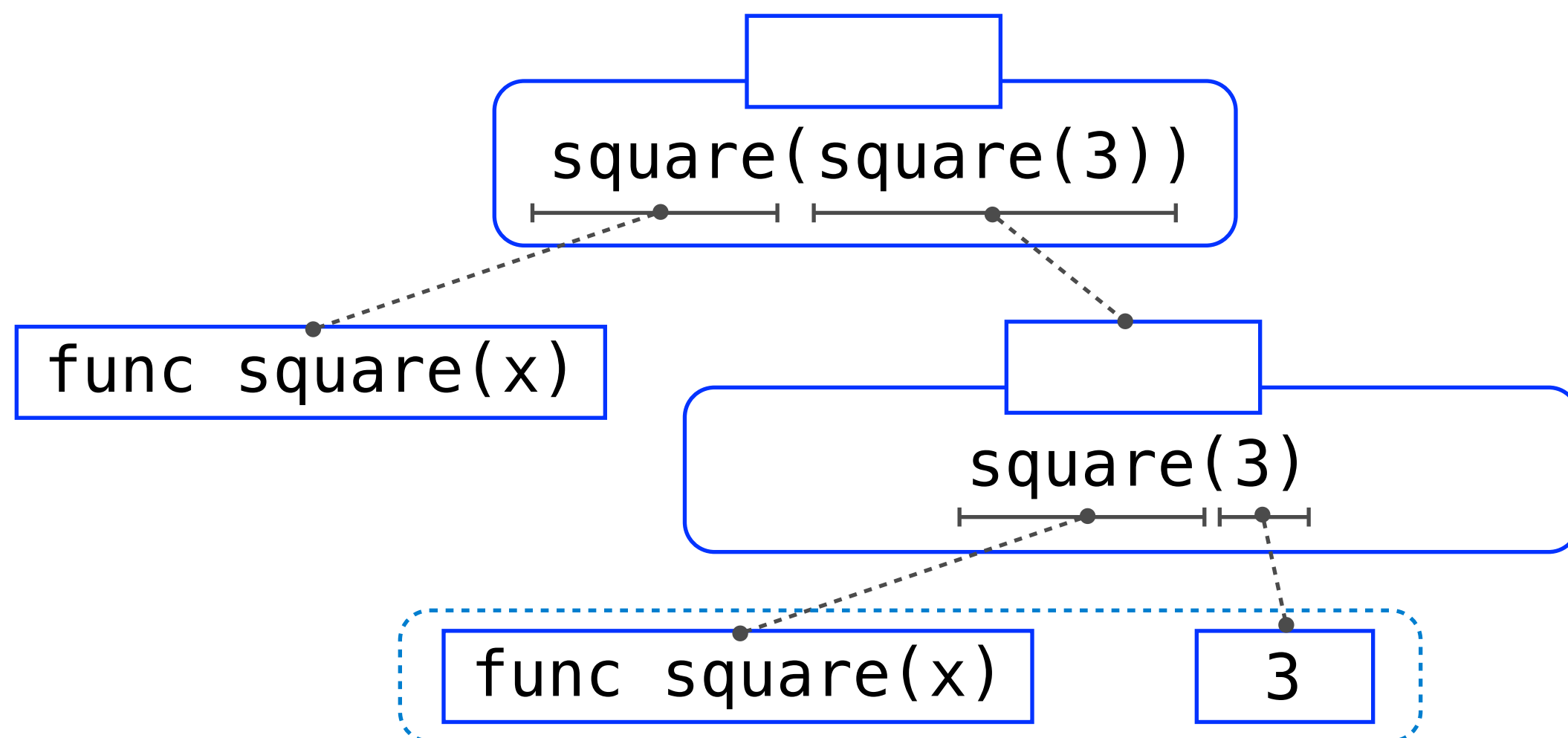
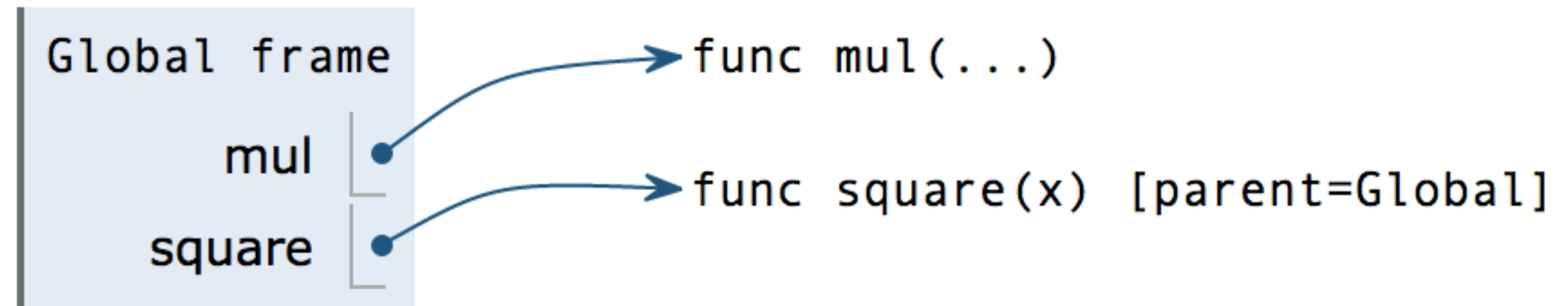
A new frame is created!

Parameters bound to arguments

Body is executed in that new
environment

Multiple Environments in One Diagram!

```
1 from operator import mul  
→ 2 def square(x):  
3     return mul(x, x)  
→ 4 square(square(3))
```

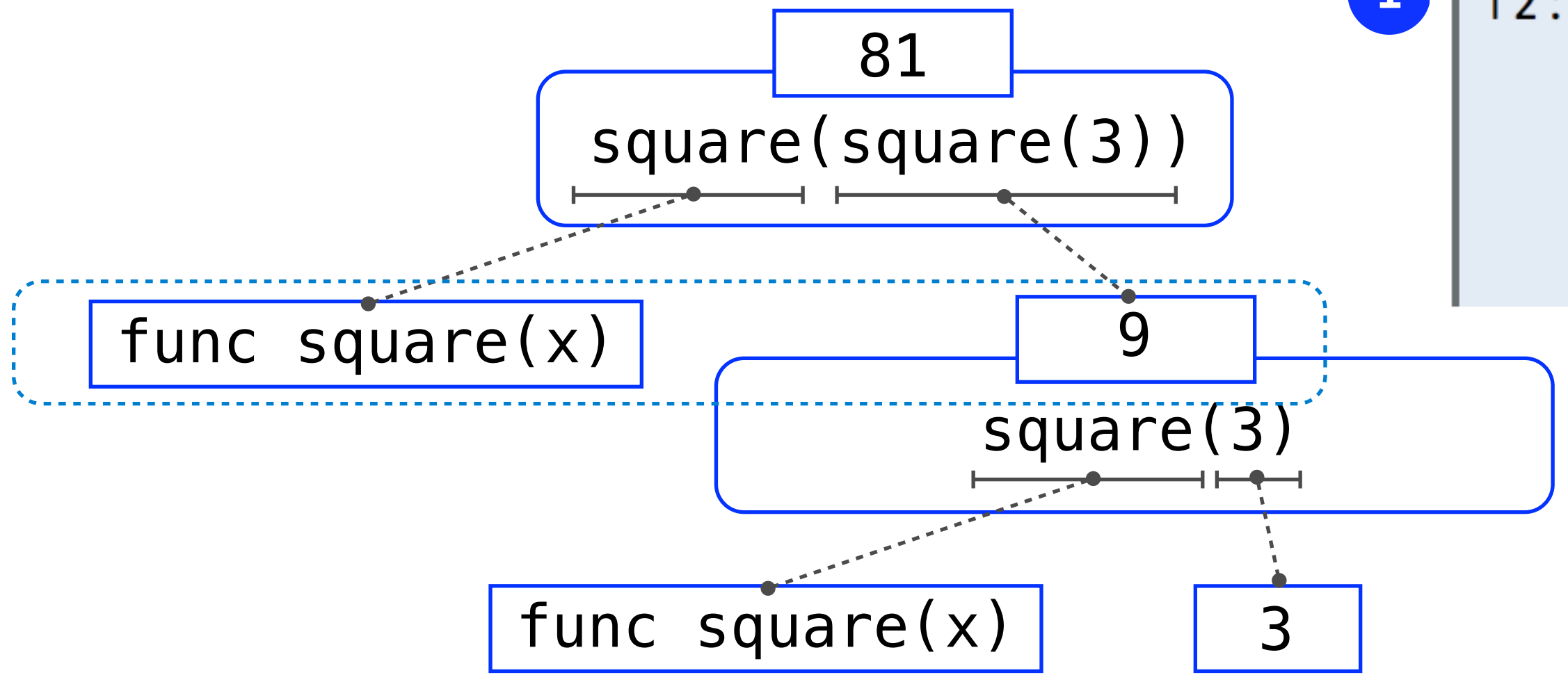
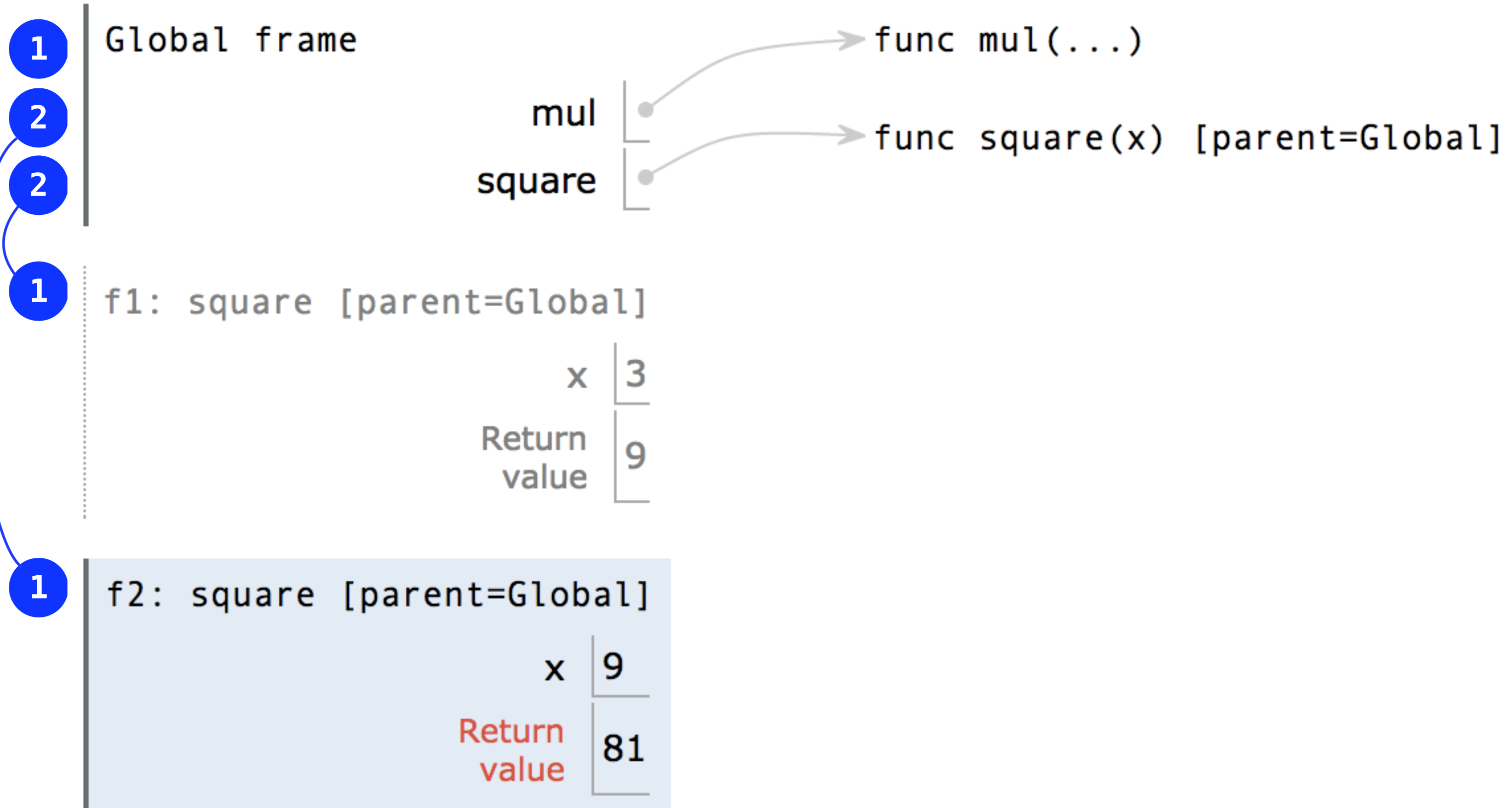


Multiple Environments in One Diagram!

```

1 from operator import mul
2 def square(x):
3     return mul(x, x)
4 square(square(3))

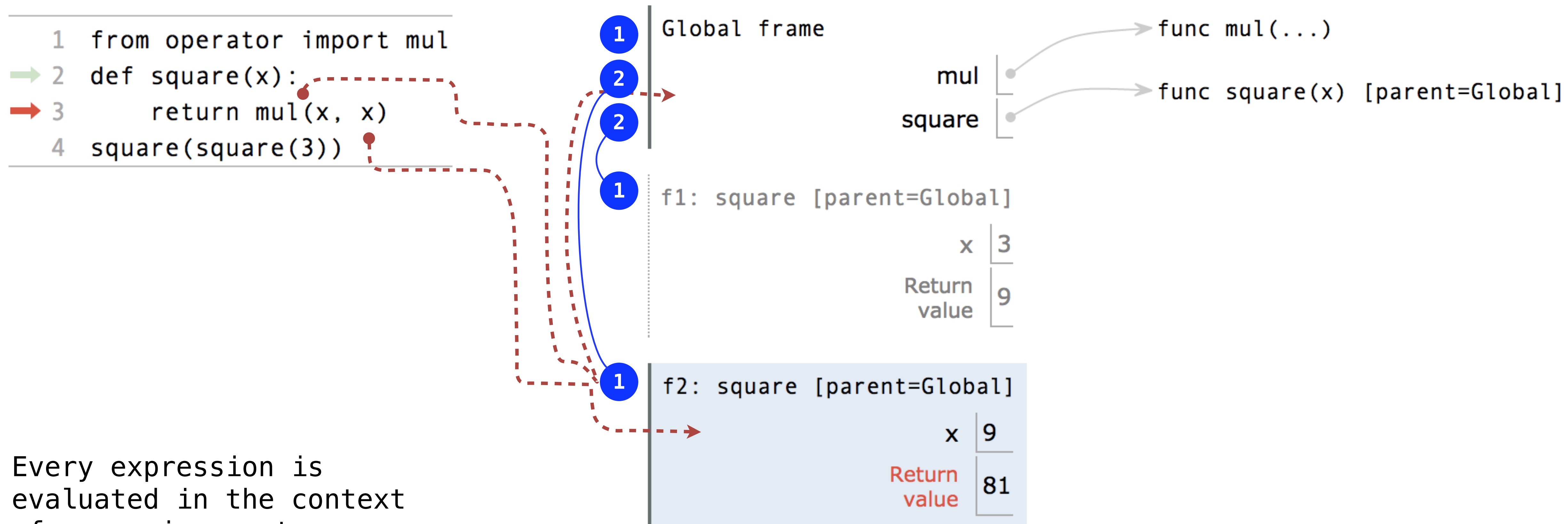
```



An environment is a sequence of frames.

- The global frame alone
- A local, then the global frame

Names Have No Meaning Without Environments



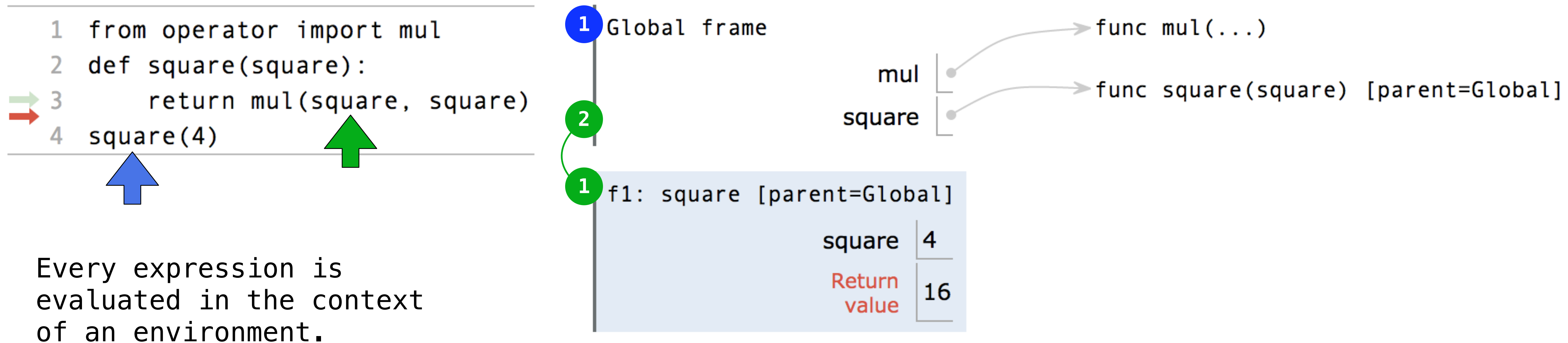
Every expression is evaluated in the context of an environment.

An environment is a sequence of frames.

- The global frame alone
- A local, then the global frame

Names Have Different Meanings in Different Environments

A call expression and the body of the function being called are evaluated in different environments



A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

Miscellaneous Python Features

Division

Multiple Return Values

Source Files

Doctests

Default Arguments

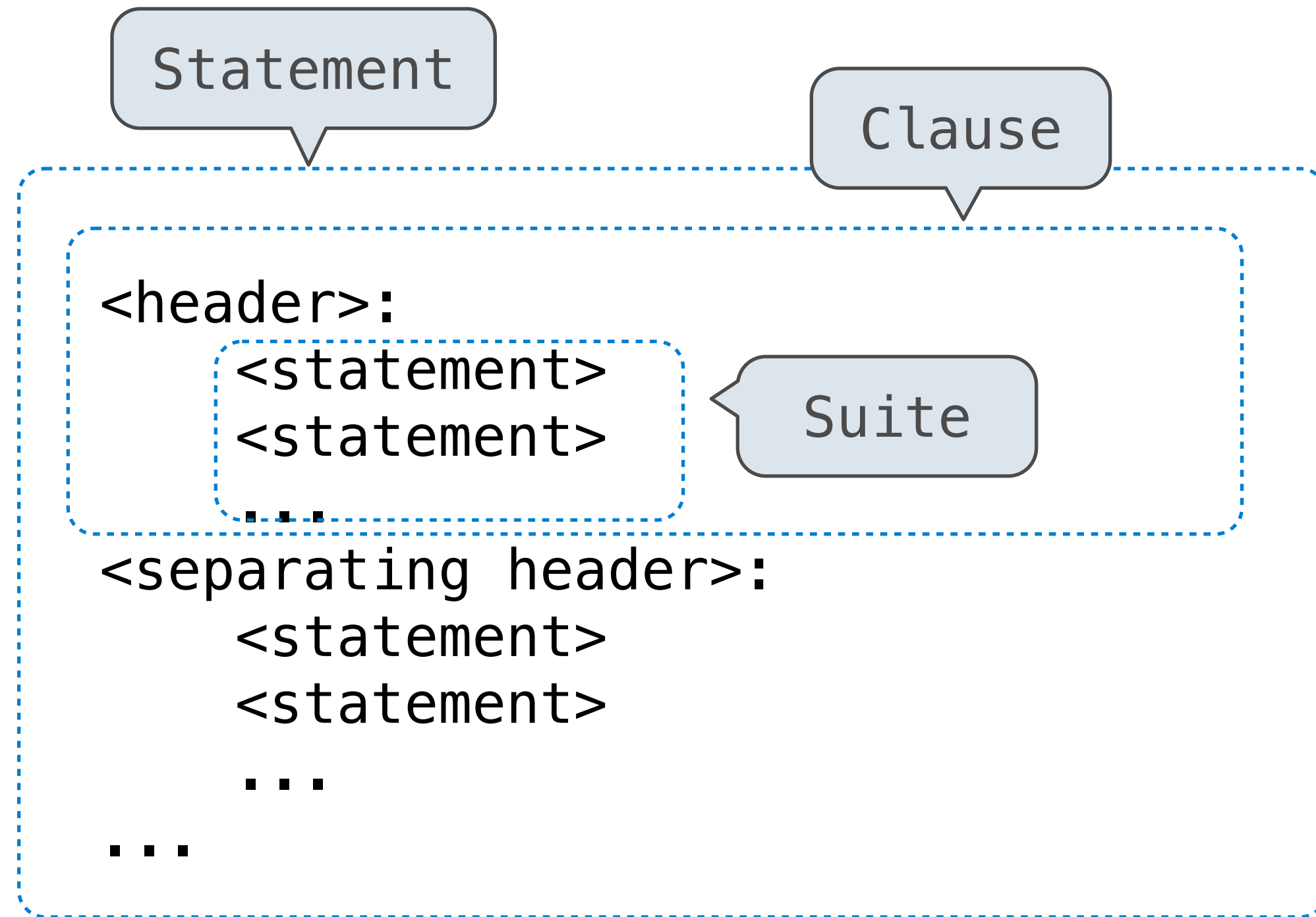
(Demo)

Conditional Statements

Statements

A *statement* is executed by the interpreter to perform an action

Compound statements:



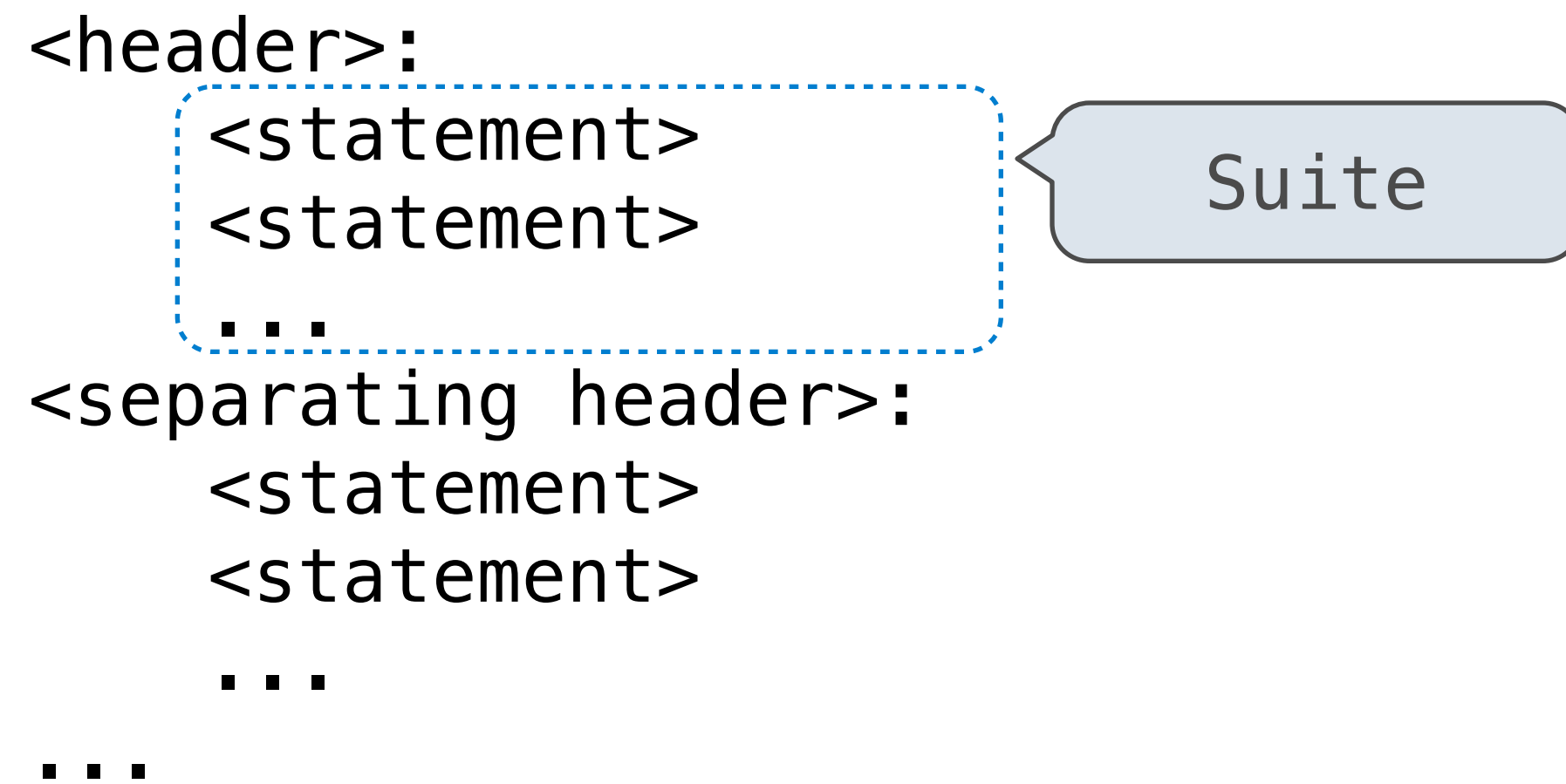
The first header determines a statement's type

The header of a clause "controls" the suite that follows

def statements are compound statements

Compound Statements

Compound statements:



A suite is a sequence of statements

To “execute” a suite means to execute its sequence of statements, in order

Execution Rule for a sequence of statements:

- Execute the first statement
- Unless directed otherwise, execute the rest

Conditional Statements

1 statement,
3 clauses,
3 headers,
3 suites

```
def absolute_value(x):  
    """Return the absolute value of x."""  
    if x < 0:  
        return -x  
    elif x == 0:  
        return 0  
    else:  
        return x
```

Execution Rule for Conditional Statements:

Each clause is considered in order.

1. Evaluate the header's expression.
2. If it is a true value, execute the suite & skip the remaining clauses.

Syntax Tips:

1. Always starts with "if" clause.
2. Zero or more "elif" clauses.
3. Zero or one "else" clause, always at the end.

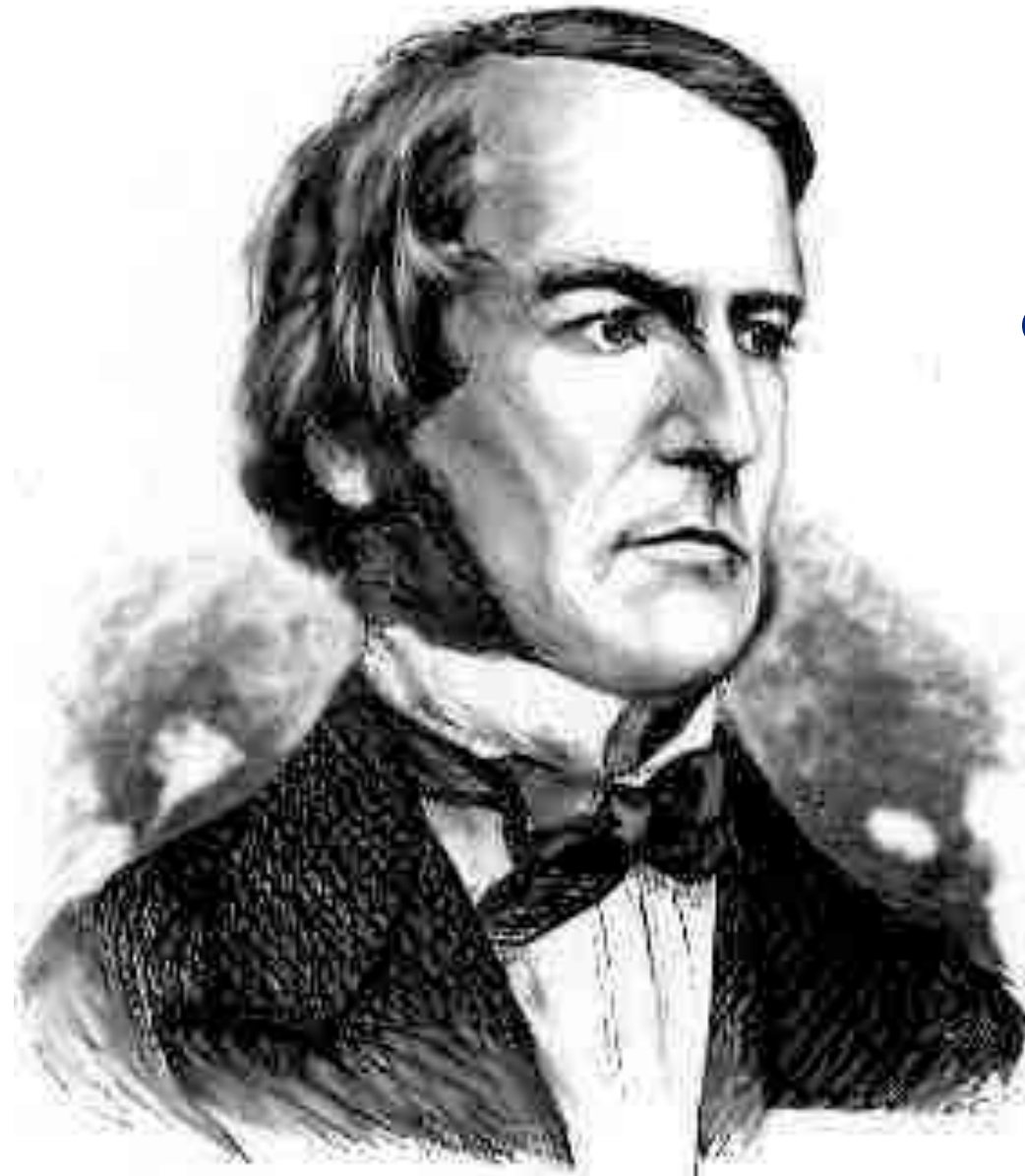
Boolean Contexts



George Boole

```
def absolute_value(x):  
    """Return the absolute value of x."""  
    if x < 0:  
        return -x  
    elif x == 0:  
        return 0  
    else:  
        return x
```

Boolean Contexts



George Boole

```
def absolute_value(x):  
    """Return the absolute value of x."""  
    if x < 0:  
        return -x  
    elif x == 0:  
        return 0  
    else:  
        return x
```

Two boolean contexts

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

True values in Python: Anything else (True)

Read Section 1.5.4!

(Demo)

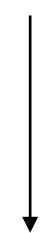
Control Expressions

Logical Operators

To evaluate the **expression** `<left>` and `<right>`:

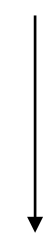
1. Evaluate the subexpression `<left>`
2. If it evaluates to a false-y value, `v`, the **expression** evaluates to `v`
3. Otherwise, the **expression** evaluates to `<right>`

True and 5



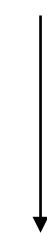
5

0 and None and 2



0

1 and 2 and None



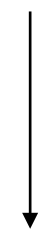
None

Logical Operators

To evaluate the **expression** `<left>` or `<right>`:

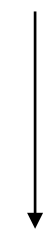
1. Evaluate the subexpression `<left>`
2. If it evaluates to a truth-y value, `v`, the **expression** evaluates to `v`
3. Otherwise, the **expression** evaluates to `<right>`

True or 5



True

0 or None or 2



2

1 and 2 or None



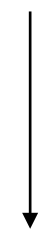
2

Logical Operators

To evaluate the **expression** `not <expression>`:

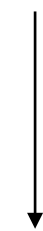
1. Evaluate the subexpression `<expression>`
2. If it evaluates to a truth-y value, the **expression** evaluates to **False**
3. Otherwise, the **expression** evaluates to **True**

`not True`



`False`

`not (None or 0)`



`True`

Iteration

While Statements

(Demo)



George Boole

```
▶ 1 i, total = 0, 0
▶ 2 while i < 3:
▶ 3     i = i + 1
▶ 4     total = total + i
```

Global frame

i	0	1	2	3
total	0	1	3	6

Execution Rule for While Statements:

1. Evaluate the header's expression.
2. If it is a true value, execute the (whole) suite, then return to step 1.

(Demo)

Summary

- There are **pure** and **non-pure** functions
 - Print is a non-pure function that return None, but displays something as a side effect
- **Multiple environments** can exists in a diagram
 - It is important to keep track an environment, tracing back to the parent of the earliest frame
 - Every expression is evaluated in the context of an environment
- How **floordiv**, **truediv**, and **mod** are used in a boolean context
 - later, we'll be seeing how they are used for digit manipulation
- A **conditional statement** is executed in order
 - only one suite is executed, and any following clauses are skipped
- Logical operators, **and** and **or**
- Using **while statements** for iteration