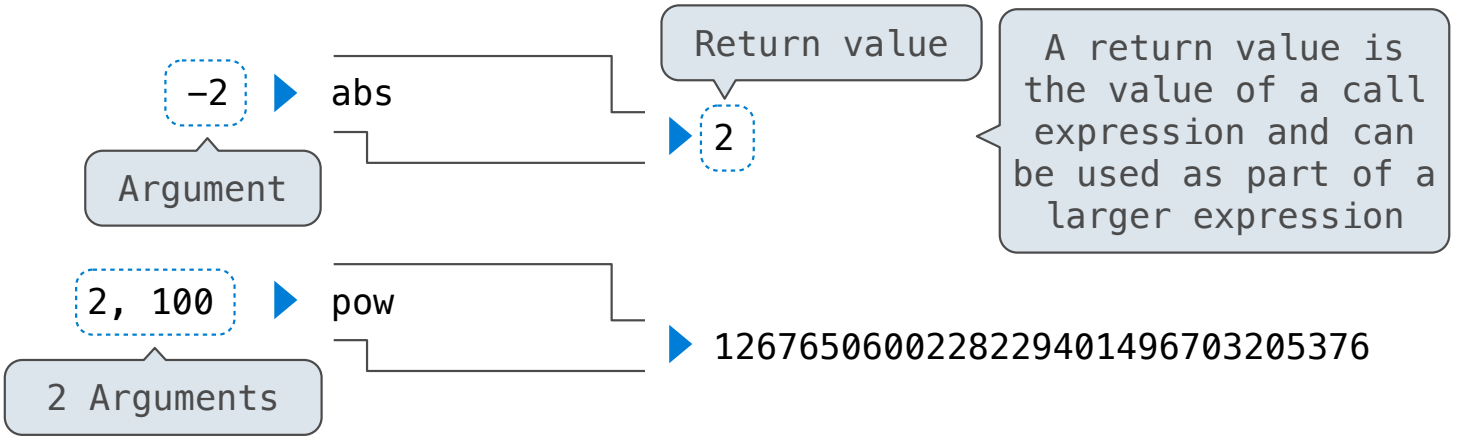
Control

Announcements

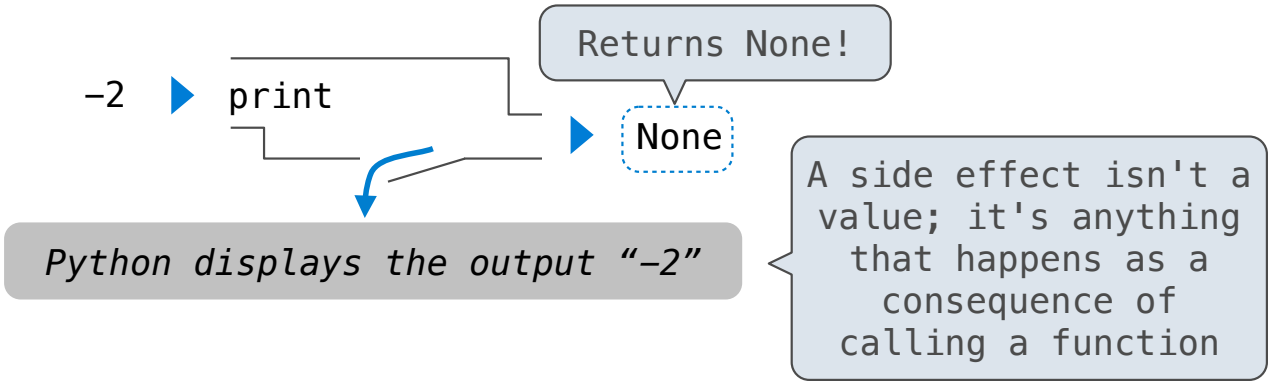
Print and None

Pure Functions & Non-Pure Functions

Pure Functions
just return values



Non-Pure Functions
have side effects



Example: Print Then Return

Implement a function h(x) that first prints, then returns, the value of f(x).

```
def h(x):
    return print(f(x))
```

(A)

```
def h(x):
    print(f(x))
    return f(x)
```

(B)

```
def h(x):
    y = f(x)
    print(y)
    return y
```

(C)

What's a function f for which implementations (B) and (C) would have different behavior?

```
>>> h(2)
...
```

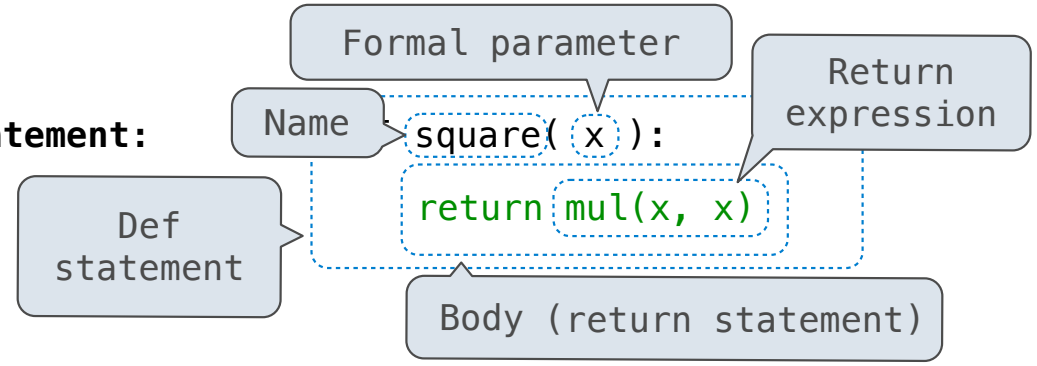
```
>>> h(2)
...
```

(Demo)

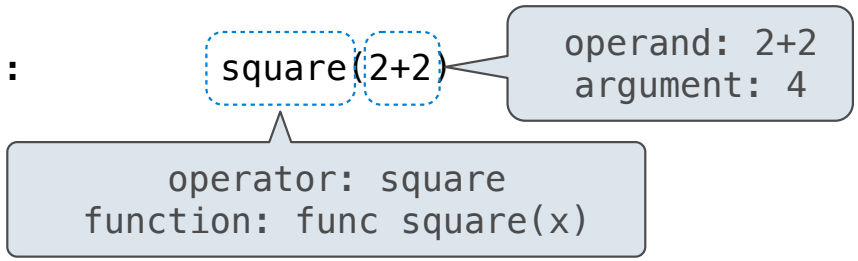
Multiple Environments

Life Cycle of a User-Defined Function

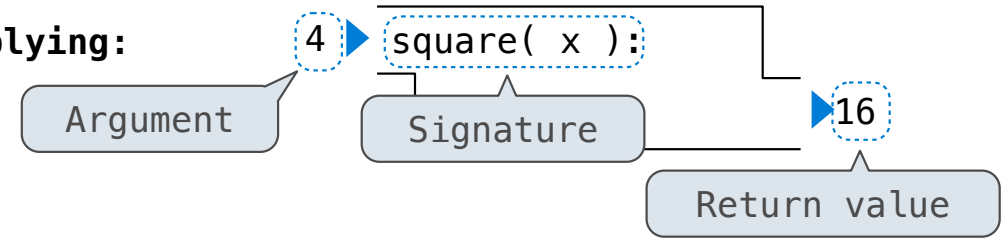
Def statement:



Call expression:



Calling/Applying:



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!

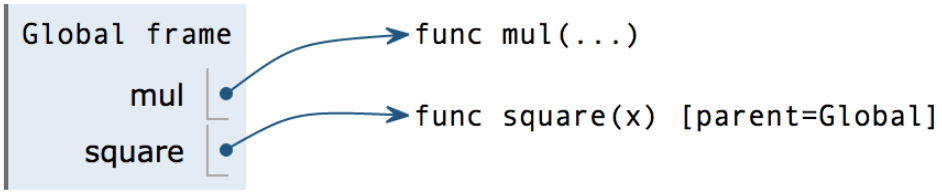
Parameters bound to arguments

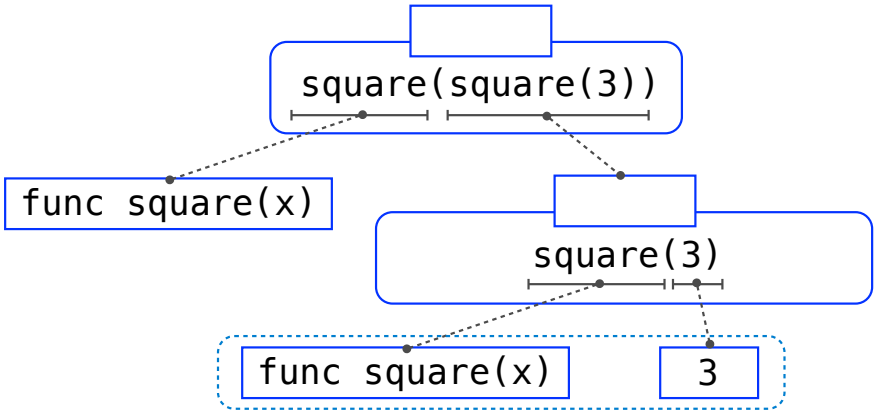
Body is executed in that new environment

(Demo)

Multiple Environments in One Diagram!

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

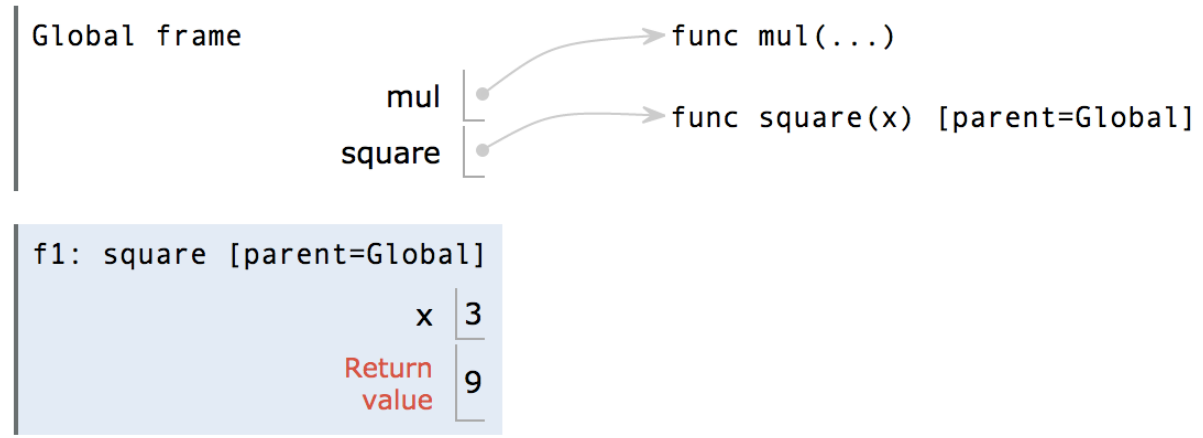


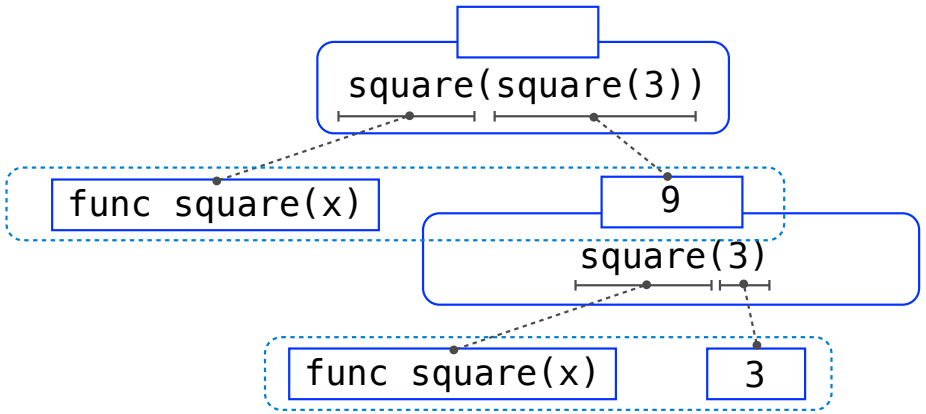


http://pythontutor.com/composingprograms.html#code=from%20operator%20import%20mul%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20mul%28x,%20x%29%0Asquare%28square%283%29%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Multiple Environments in One Diagram!

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

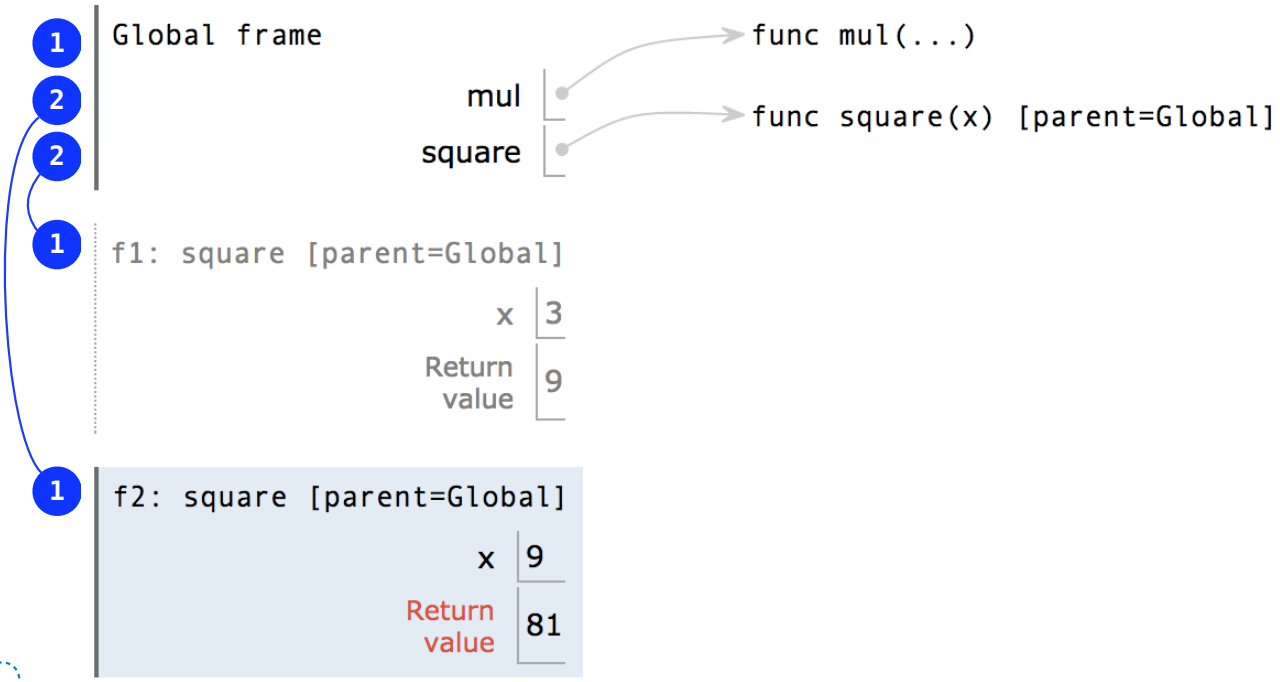


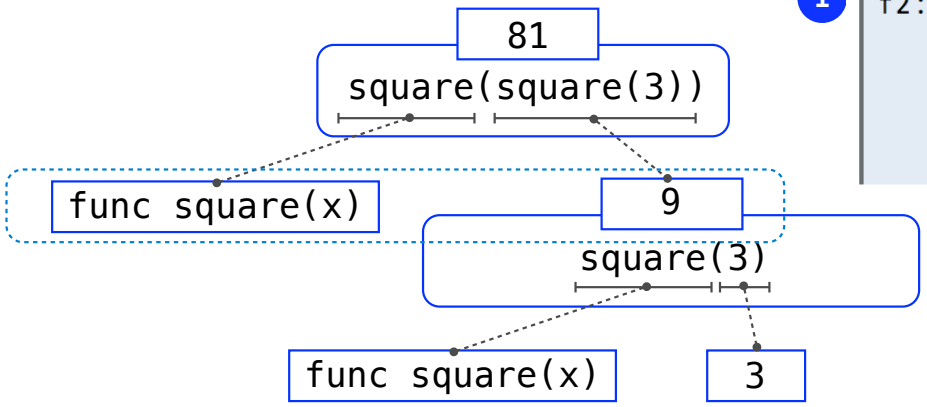


http://pythontutor.com/composingprograms.html#code=from%20operator%20import%20mul%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20mul%28x,%20x%29%0Asquare%28square%283%29%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Multiple Environments in One Diagram!

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



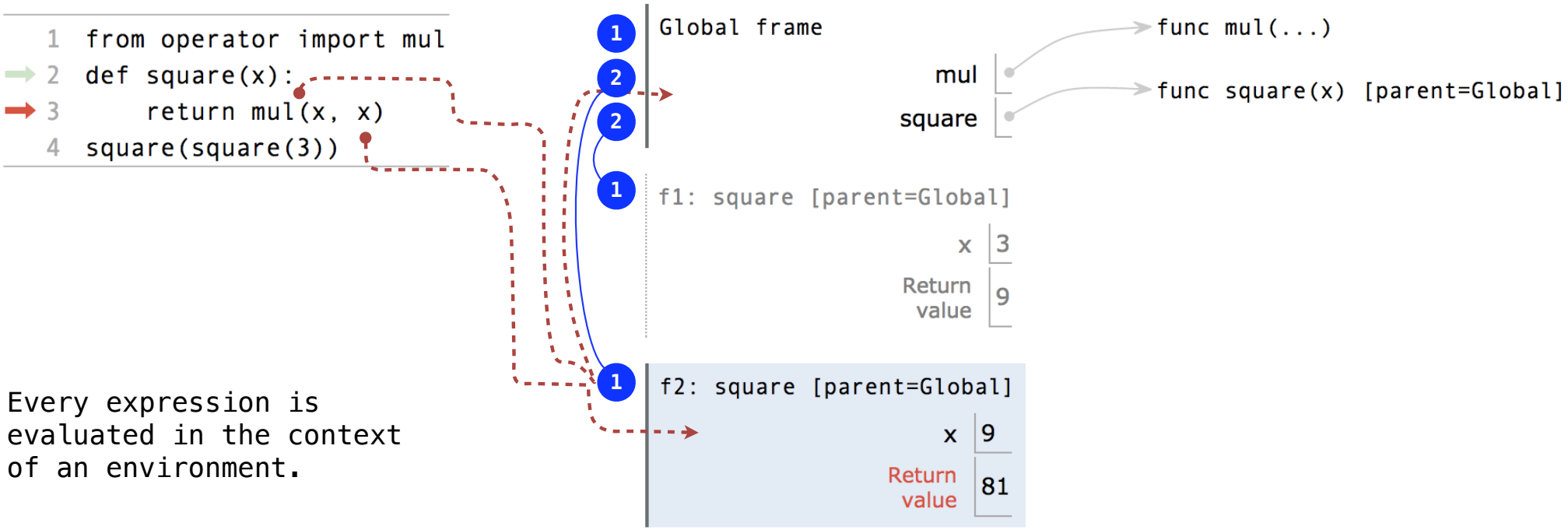


An environment is a sequence of frames.

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

http://pythontutor.com/composingprograms.html#code=from%20operator%20import%20mul%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20mul%28x,%20x%29%0Asquare%28square%283%29%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Names Have No Meaning Without Environments



Every expression is evaluated in the context of an environment.

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

An environment is a sequence of frames.

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

http://pythontutor.com/composingprograms.html#code=from%20operator%20import%20mul%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20mul%28x,%20x%29%0Asquare%28square%283%29%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Control

Conditional Statements

Conditional statements (often called "If" Statements) contain statements that may or may not be evaluated.

		x=10	x=1	x=-1

`if x > 2:`
`  print('big')`
`if x > 0:`
`  print('positive')`
	Two separate (unrelated) conditional statements	big
		positive
			positive	

`if x > 2:`
`  print('big')`
`elif x > 0:`
`  print('positive')`
	One statement with two clauses: if and elif
	Only one body can ever be executed
		big	positive	

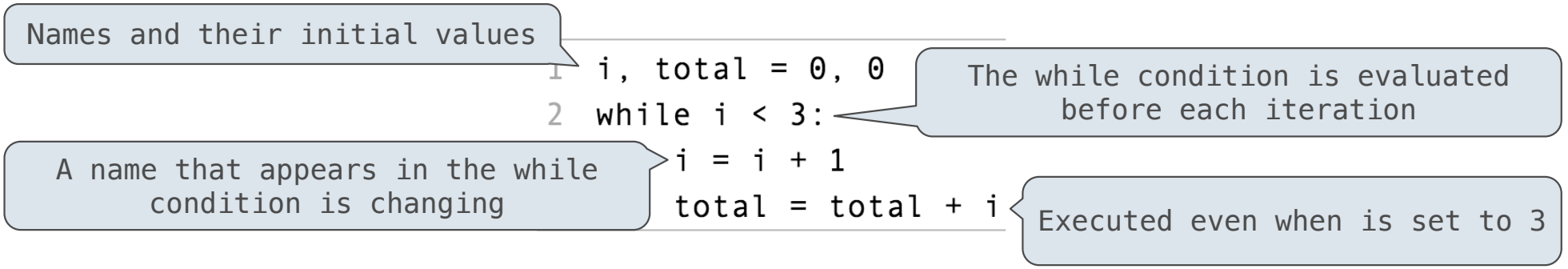
`if x > 2:`
`  print('big')`
`elif x > 0:`
`  print('positive')`
`else:`
`  print('not pos')`
	One statement with three clauses: if, elif, else
	Only one body can ever be executed
		big	positive	not pos

While Statements

While statements contain statements that are repeated as long as some condition is true.

Important considerations:

- How many separate names are needed and what do they mean?
- The while condition must eventually become a false value for the statement to end (unless there is a return statement inside the while body).
- Once the while condition is evaluated, the entire body is executed.



Example: Prime Factorization

Prime Factorization

Each positive integer n has a set of prime factors: primes whose product is n

```
...
8  = 2 * 2 * 2
9  = 3 * 3
10 = 2 * 5
11 = 11
12 = 2 * 2 * 3
...
```

One approach: Find the smallest prime factor of n, then divide by it

```
858  = 2 * 429  = 2 * 3 * 143  = 2 * 3 * 11 * 13
```

(Demo)