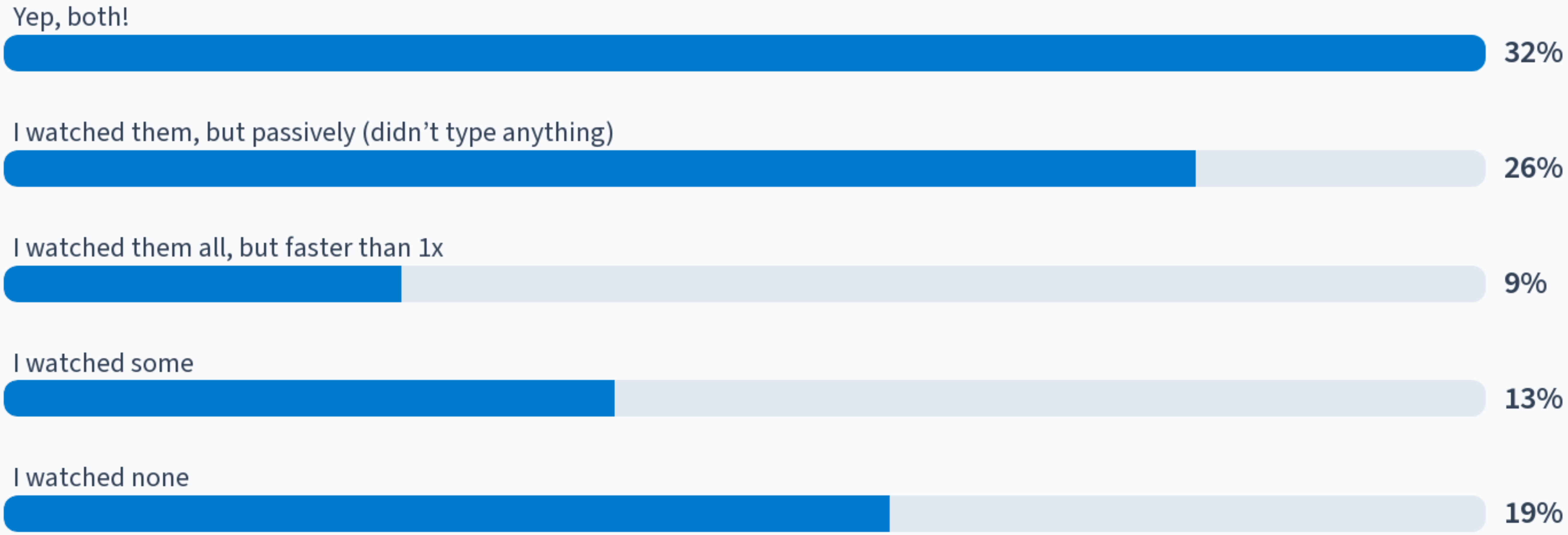
Control

Announcements

L2.1: Did you watch the lecture videos for today and follow along by typing the Python?



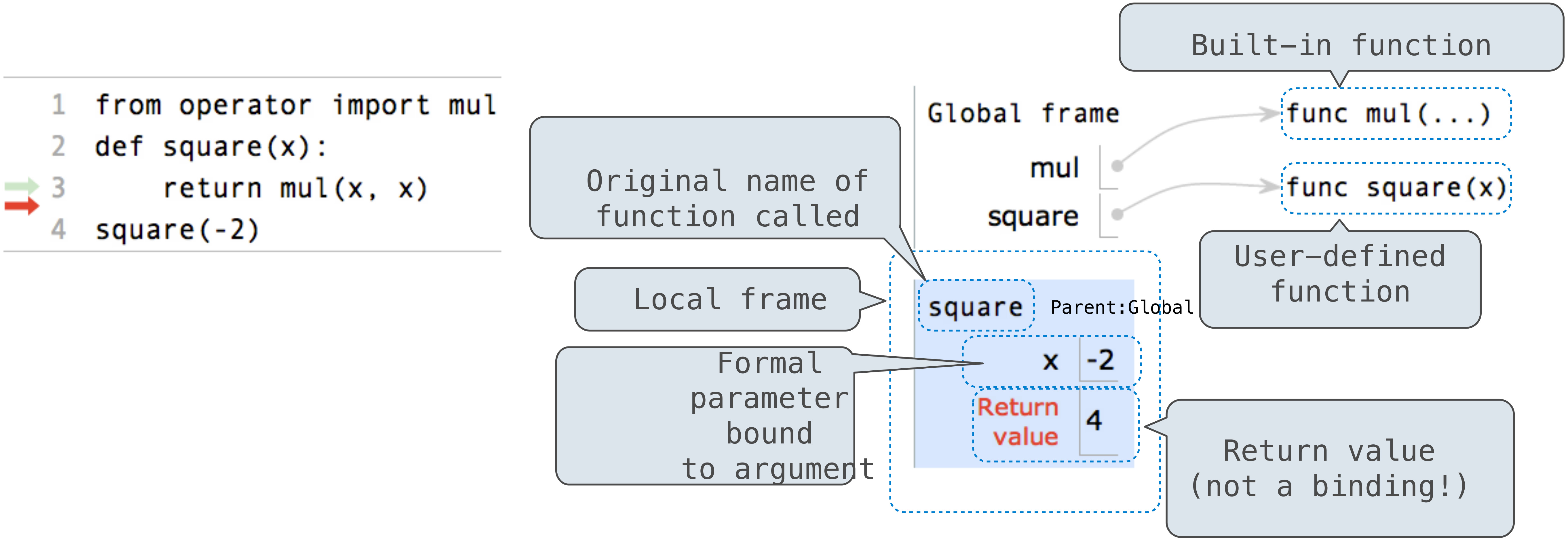
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Environment Diagrams

Calling User-Defined Functions

Procedure for calling/applying user-defined functions (version 1):

1. Add a local frame, forming a new environment
2. Bind the function's formal parameters to its arguments in that frame
3. Execute the body of the function in that new environment



```
from operator import mul
def square(x):
    return mul(x, x)
square(-2)
```

Built-in function

Original name of function called

Local frame

Formal parameter bound to argument

User-defined function

Return value (not a binding!)

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%28-2%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Frames & Environments

Frame: Holds name-value bindings; looks like a box; no repeated names allowed!

Global frame: The frame with built-in names (min, pow, etc.)

Environment: A sequence of frames that always ends with the global frame

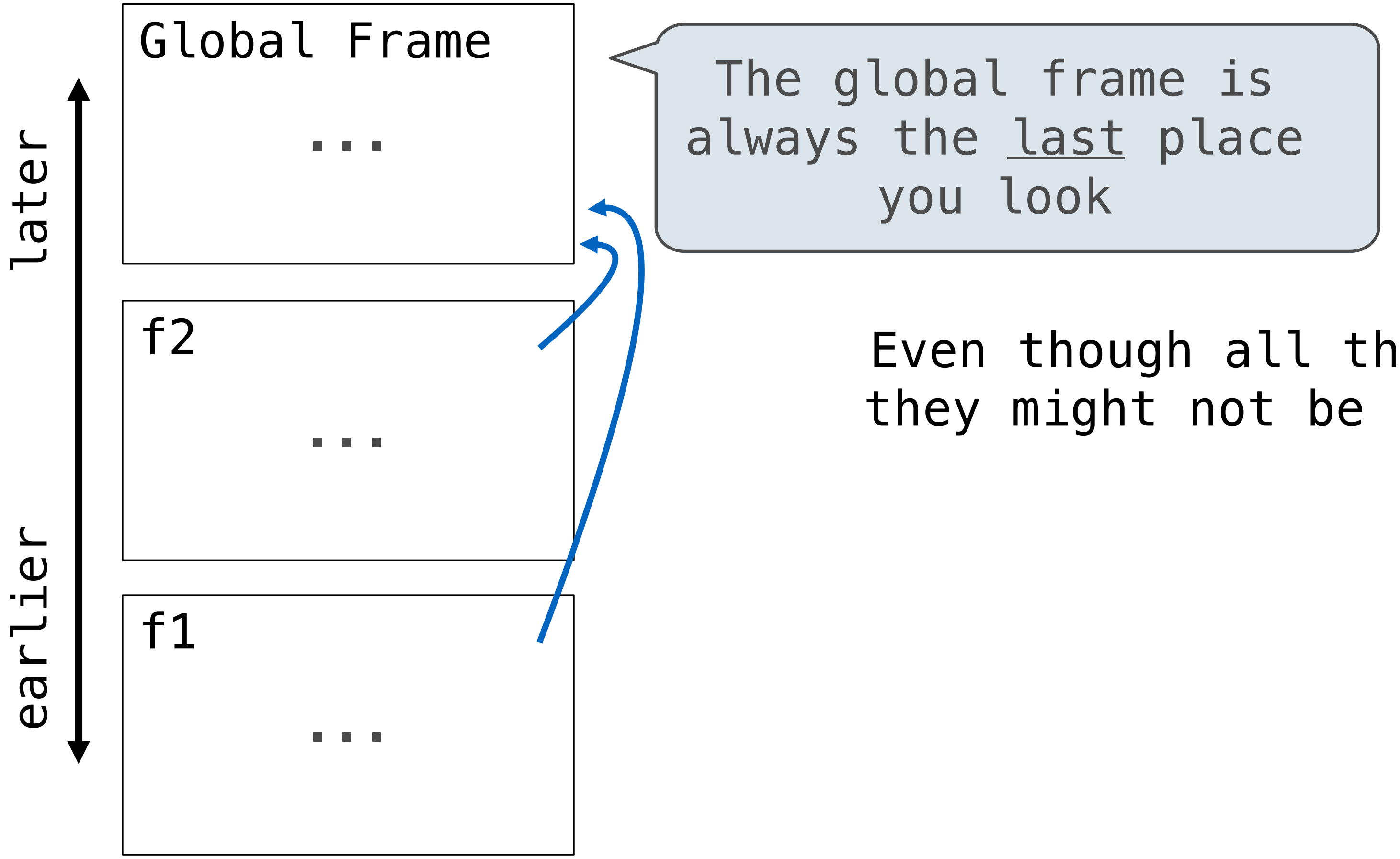
Lookup: Find the value for a name by looking in each frame of an environment

A name (which is a type of expression) such as **x** is evaluated by looking it up

A Sequence of Frames

An environment is a sequence of frames.

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



The global frame is always the last place you look

Even though all three frames are in the same **diagram,** they might not be in the same **environment**

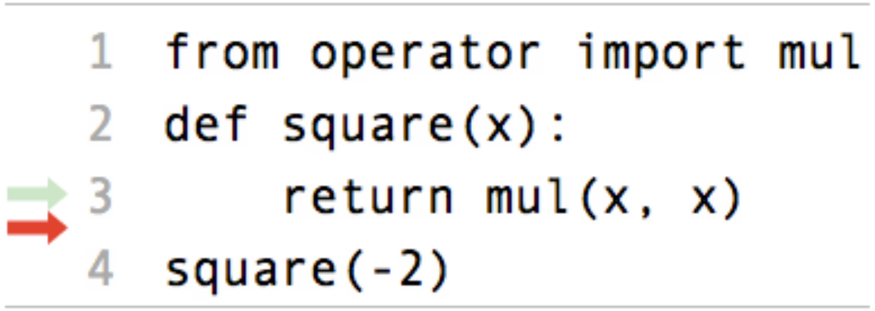
Frames & Environments

Why organize information this way?

- Local context before global context

- Calling or returning changes the local context

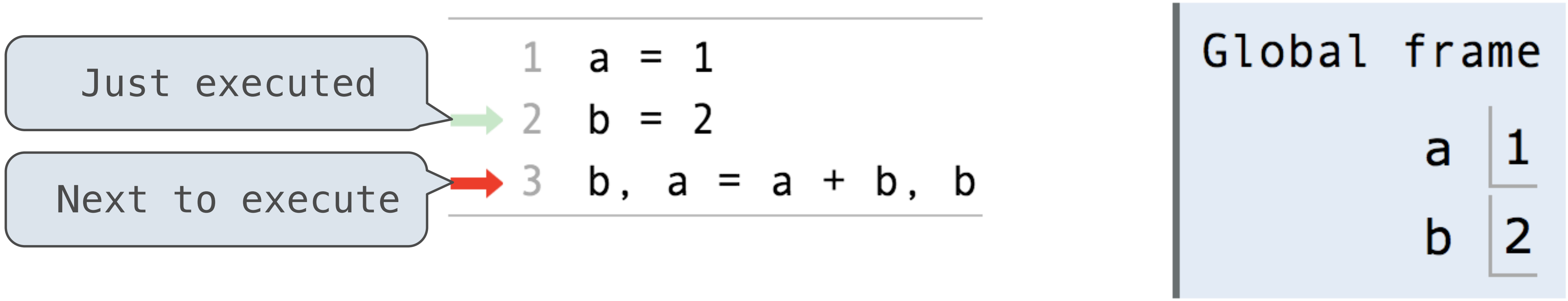
- Assignment within a function's local frame doesn't affect other frames



```
from operator import mul
def square(x):
    return mul(x, x)
square(-2)
```

Multiple Assignment

Multiple Assignment



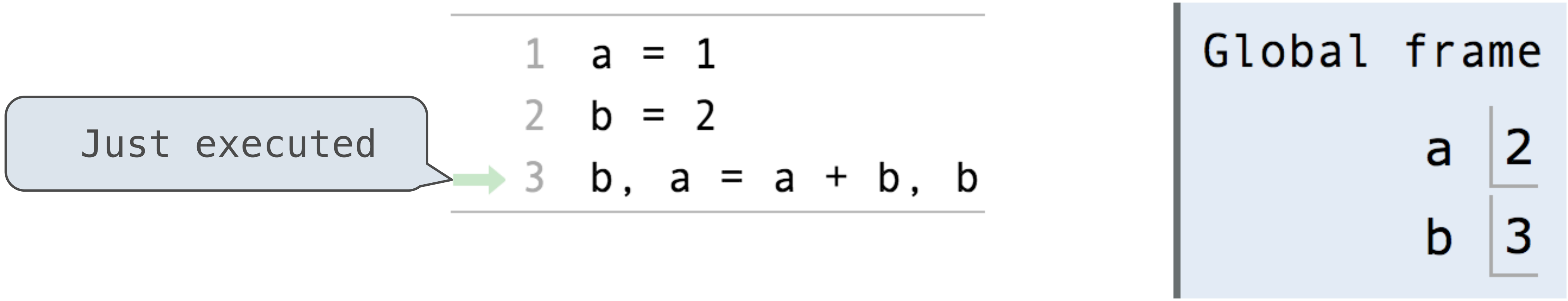
Just executed

Next to execute

```
a = 1
b = 2
b, a = a + b, b
```

Global frame

a	1
b	2



Just executed

```
a = 1
b = 2
b, a = a + b, b
```

Global frame

a	2
b	3

Execution rule for assignment statements:

1. Evaluate all expressions to the right of = from left to right.

2. Bind all names to the left of = to those resulting values in the current frame.

http://pythontutor.com/composingprograms.html#code=a%20%3D%201%0Ab%20%3D%202%0Ab,%20a%20%3D%20a%20%2B%20b,%20b&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Print and None

(Demo)

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(‘less big’)`
	One statement with two clauses: if and elif
	Only one body can ever be executed
		big	less big	

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

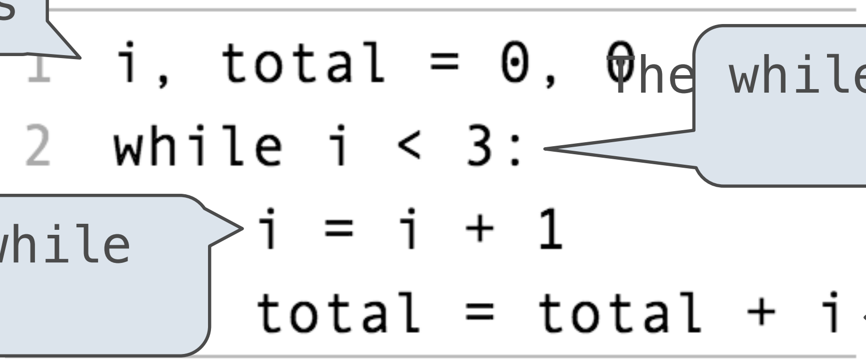
While Statements

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.

Names and their initial values



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

The while condition is evaluated before each iteration

A name that appears in the while condition is changing

Executed even when is set to 3

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
...
```

How can we determine whether a number is divisible by another?

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

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

(Demo)