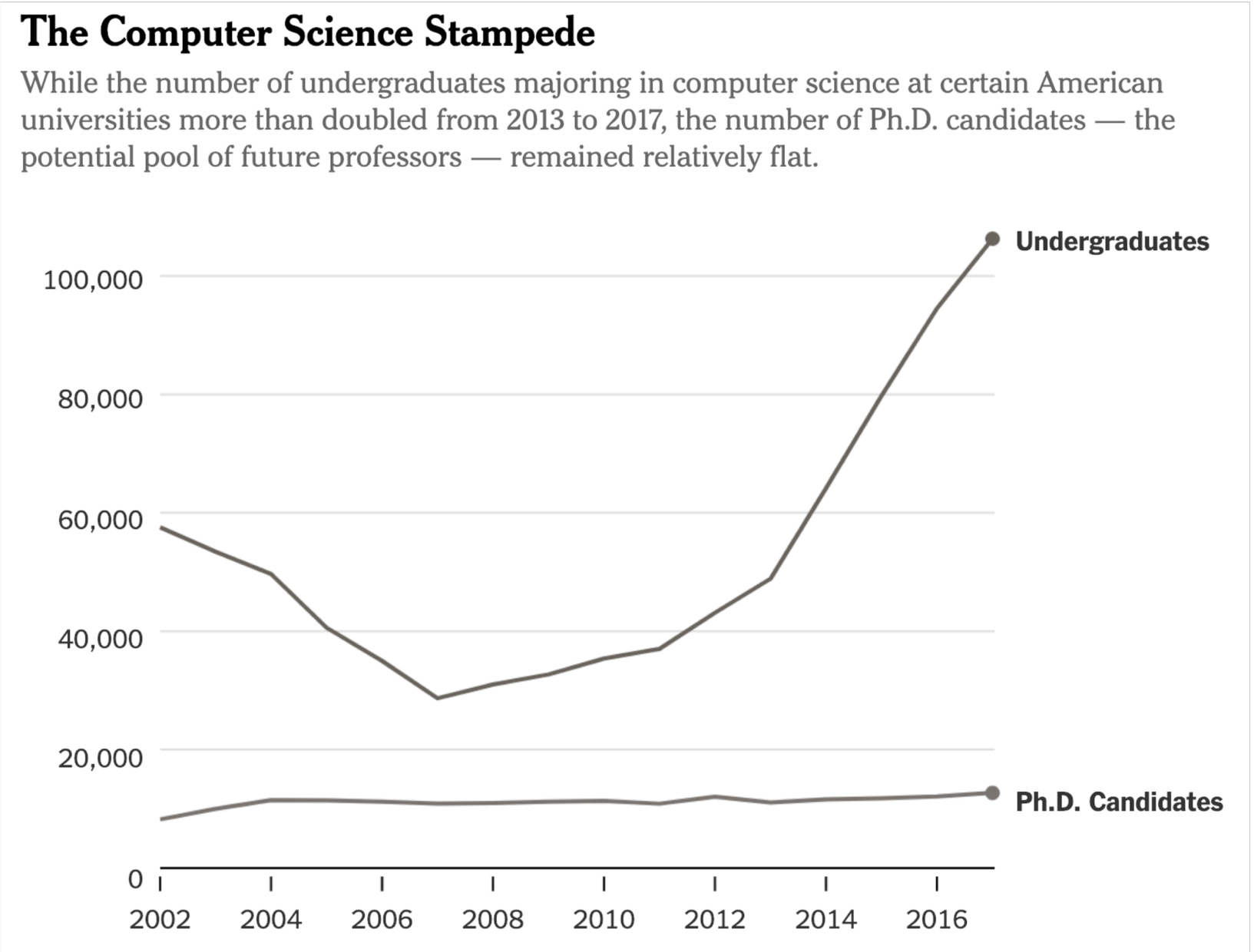
UC Berkeley's CS61A – Lecture 02 – Names

The Hard Part of Computer Science? Getting Into Class!

www.nytimes.com/2019/01/24/technology/computer-science-courses-college.html

"College students are rushing in record numbers to study CS. On campuses across the country, from major state universities to small private colleges, the surge in student demand for computer science courses is far outstripping the supply of professors, as the tech industry snaps up talent. … "demand is unbounded" … "It's going to get worse before it gets better,"



Announcements

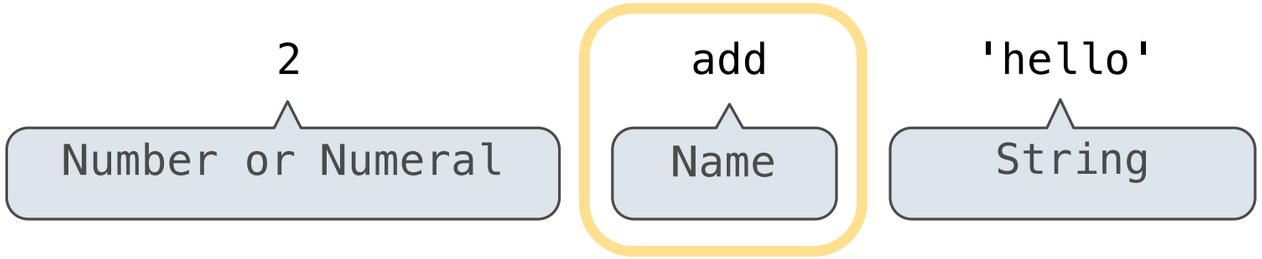
- There’s still room for lots of students to join the course! (so, I’ll sign all Concurrent Enrollment forms)…yay!
- Our website cs61a.org has all the information you need, link to Piazza, no need for bcourses for this course
- Lab and discussion next week
- You don’t have to come to lecture, today and onward there will only be one lecture 2–3pm, so 500/1250 of you have to decide to watch the webcast only. We can’t have anyone in the aisles per fire marshall rules. Videos will be available a few hours afterwards on CalCentral or bcourses, but we’ll also eventually link to them from the calendar.
- Don’t take a handout if you won’t use it, write on it, and keep it.
- Make sure you’re reading the textbook as well! (there are readings for every lecture)
- Take CS10 concurrently if you don’t have any programming experience. Follow me after class and we’ll walk together to CS10 lecture. There was also lab yesterday.

Names, Assignment, and User-Defined Functions

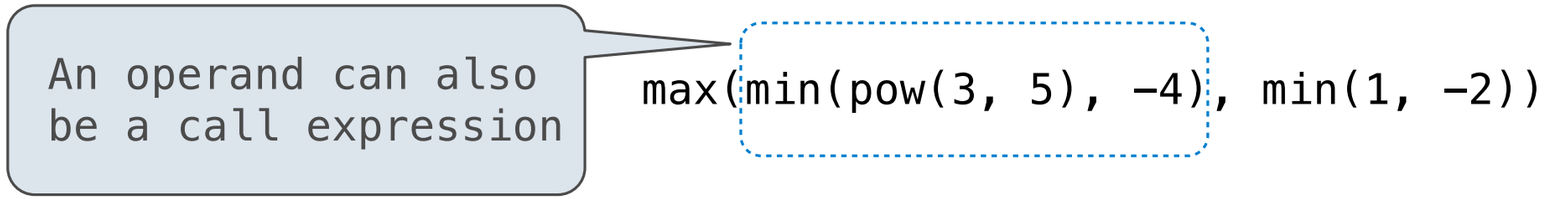
(Demo)

Types of Expressions

Primitive expressions:



Call expressions:

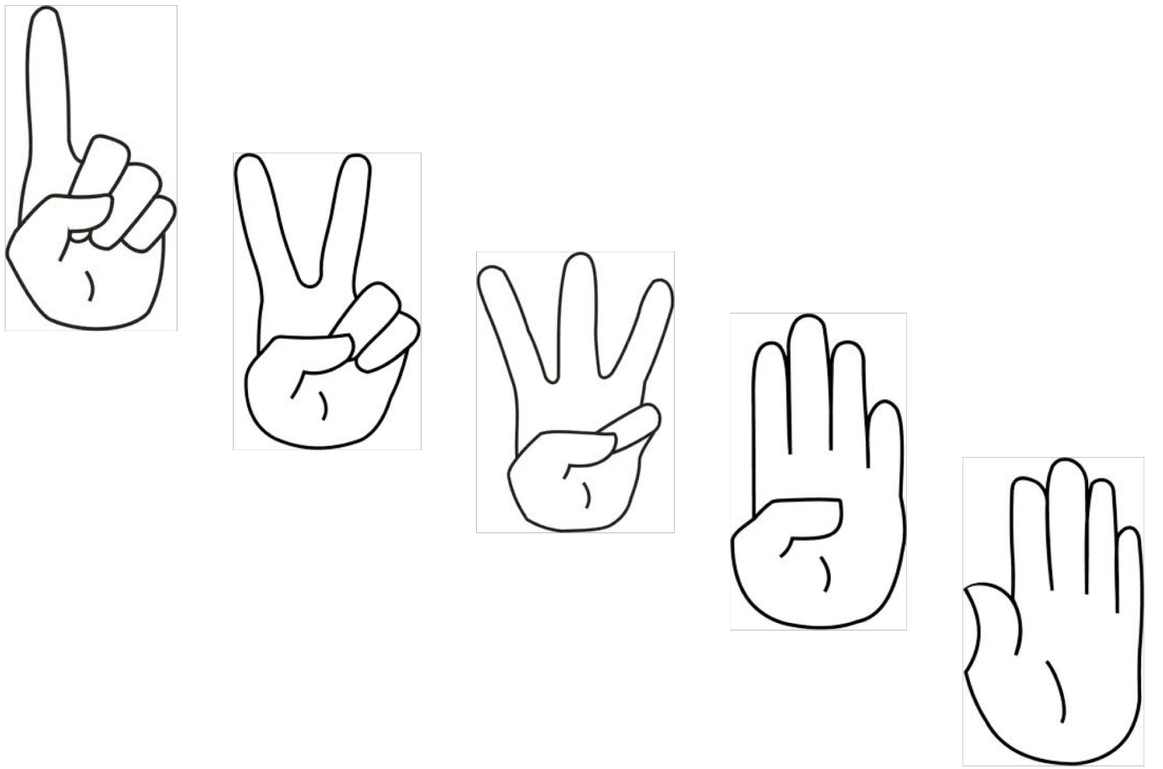


Discussion Question 1

What is the value of the final expression in this sequence?

```
>>> f = min
>>> f = max
>>> g, h = min, max
>>> max = g
>>> max(f(2, g(h(1, 5), 3)), 4)
```

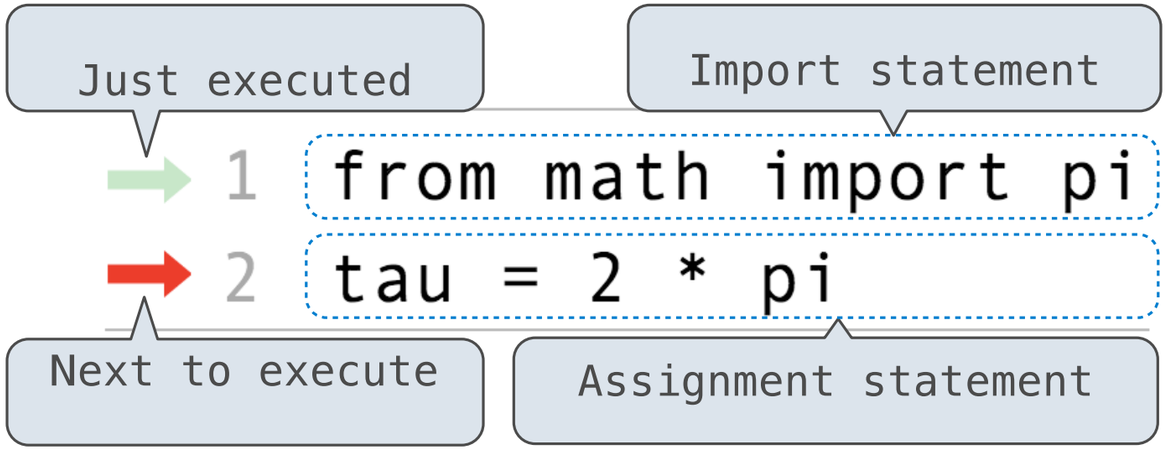
???

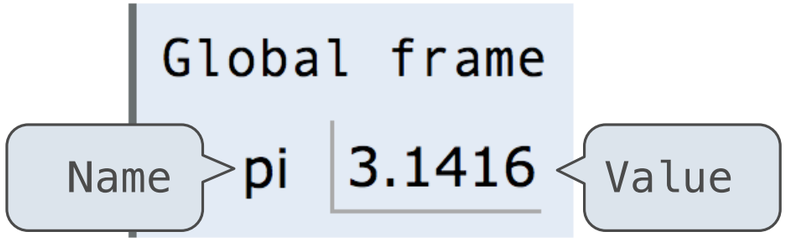


Environment Diagrams

Environment Diagrams

Environment diagrams visualize the interpreter's process.





Code (left):

Statements and expressions

Arrows indicate evaluation order

Frames (right):

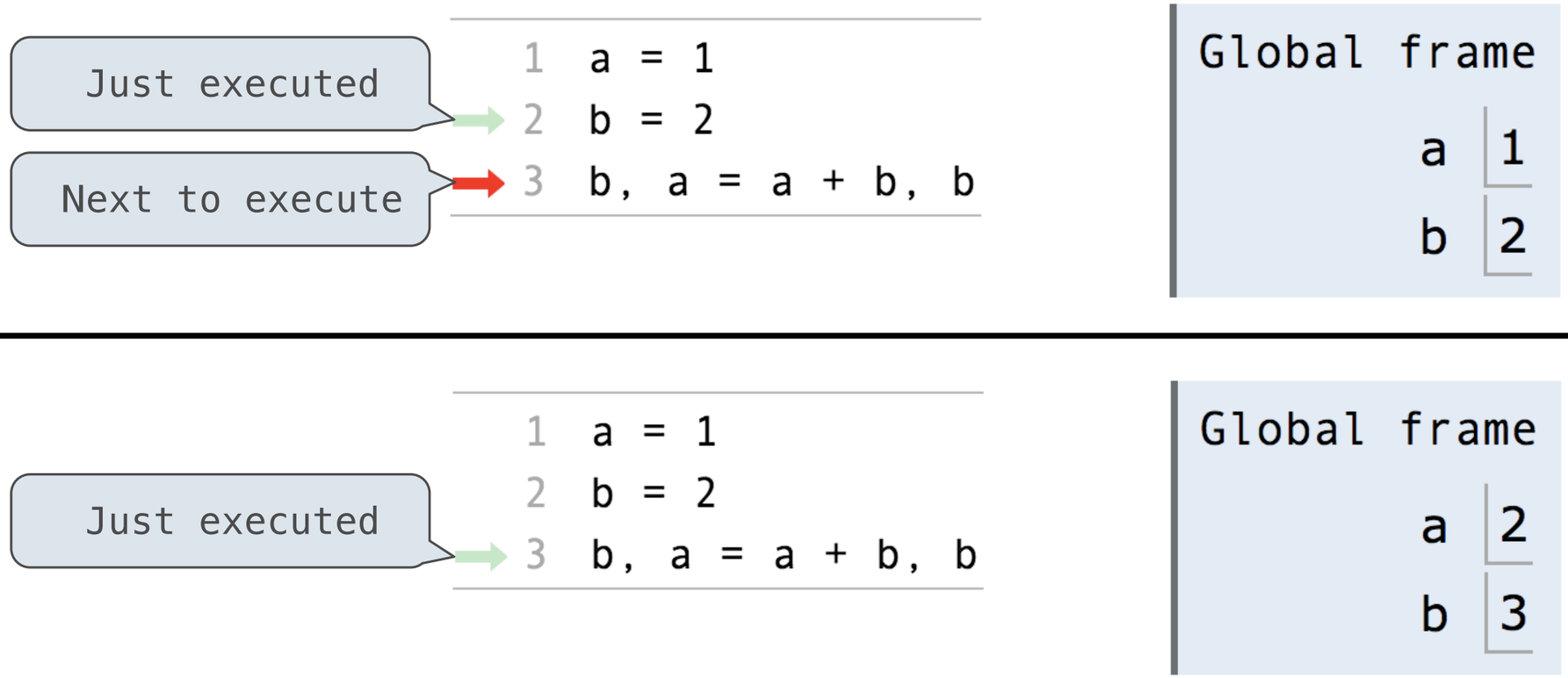
Each name is bound to a value

Within a frame, a name cannot be repeated

(Demo)

http://pythontutor.com/composingprograms.html#code=from%20math%20import%20pi%0Atau%20%3D%202%20*%20pi&cumulative=false&curInstr=1&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Assignment Statements



Execution rule for assignment statements:

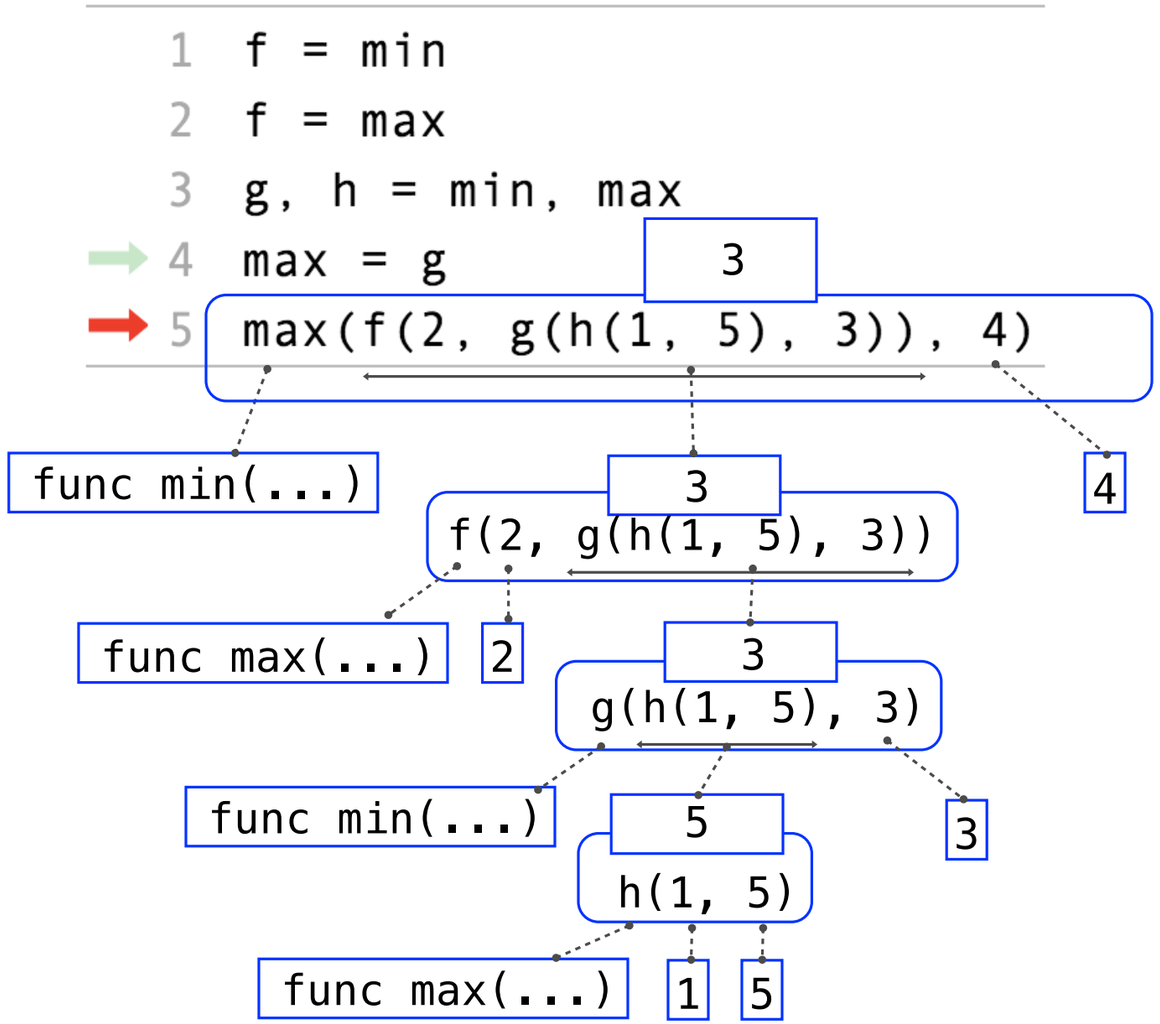
1. Evaluate all expressions to the right of = from left to right.

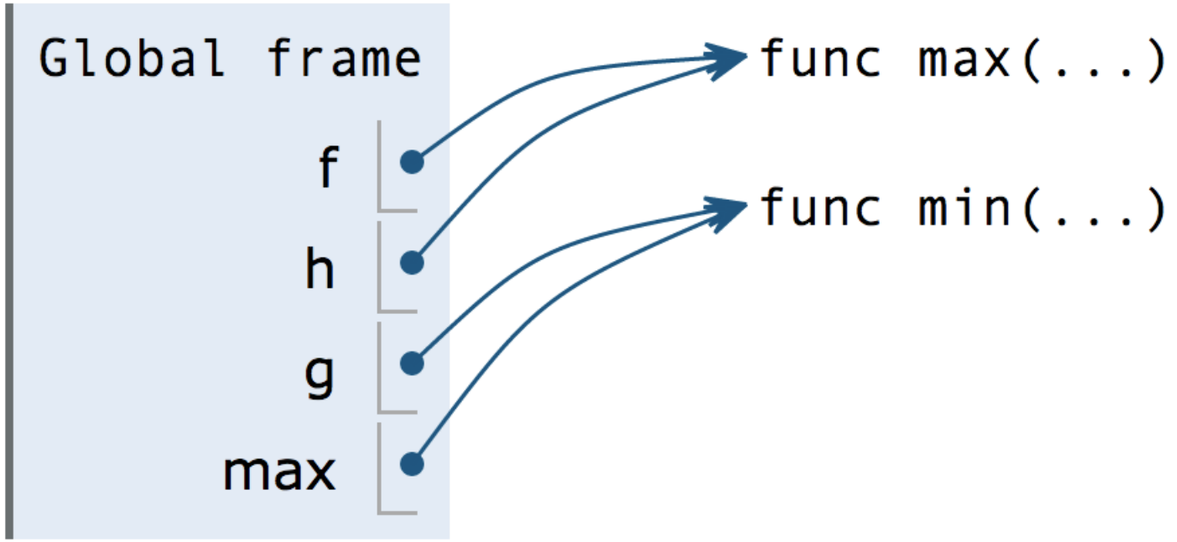
2. Bind all names to the left of = to those resulting values in the current frame.

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

Discussion Question 1 Solution

(Demo)





3

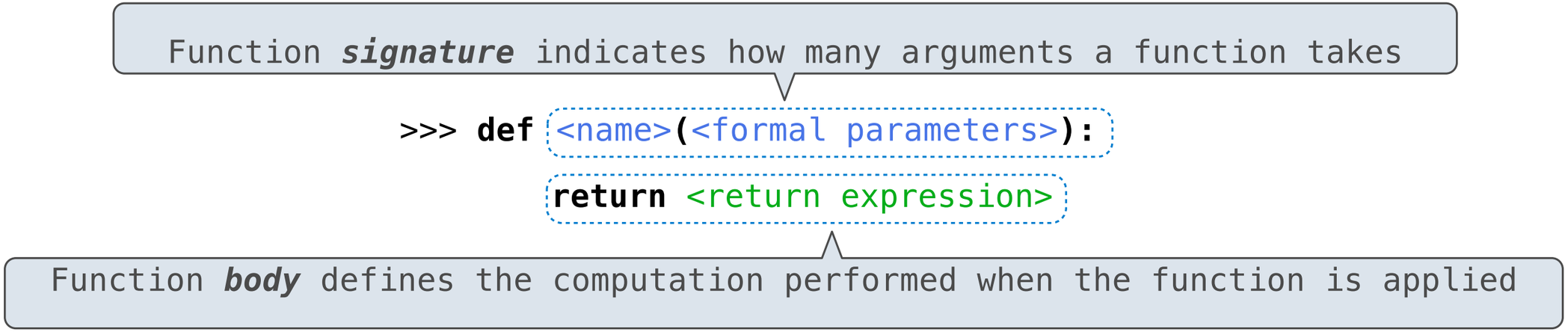
http://pythontutor.com/composingprograms.html#code=f%20%3D%20min%0Af%20%3D%20max%0Ag,%20h%20%3D%20min,%20max%0Amax%20%3D%20g%0Amax%28f%282,%20g%28h%281,%205%29,%203%29%29,%204%29&cumulative=false&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Defining Functions

Defining Functions

Assignment is a simple means of abstraction: binds names to values

Function definition is a more powerful means of abstraction: binds names to expressions



Execution procedure for def statements:

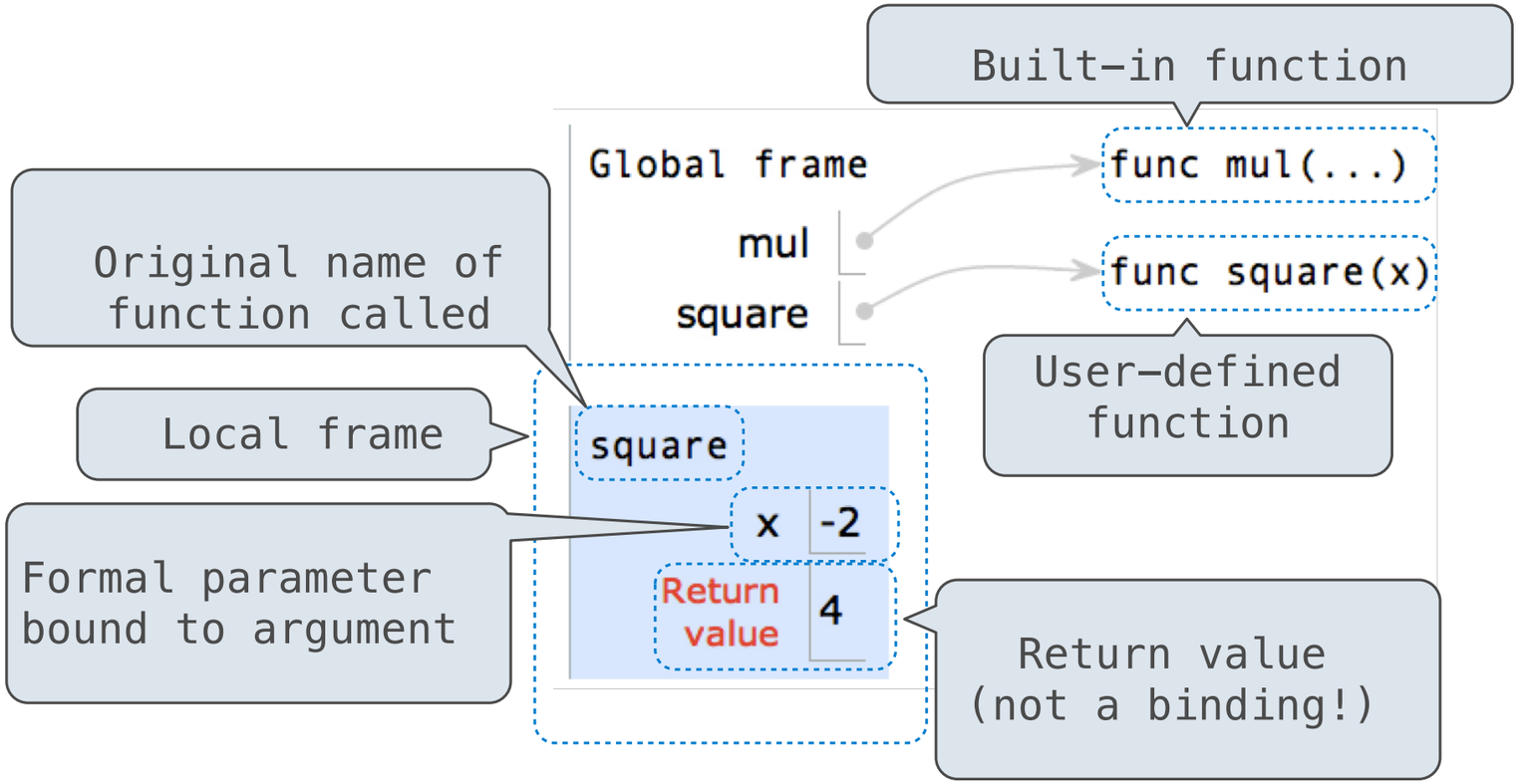
1. Create a function with signature <name>(<formal parameters>)
2. Set the body of that function to be everything indented after the first line
3. Bind <name> to that function in the current frame

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

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



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

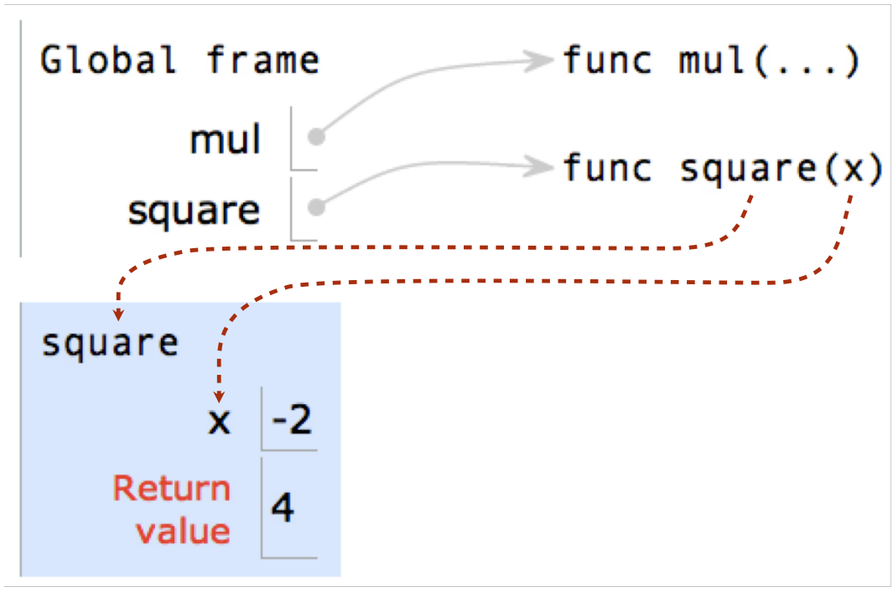
Calling User-Defined Functions

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

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

A function's signature has all the information needed to create a local frame



Looking Up Names In Environments

Every expression is evaluated in the context of an environment.

So far, the current environment is either:

- The global frame alone, or
- A local frame, followed by the global frame.

Most important two things I'll say all day:

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.

E.g., to look up some name in the body of the square function:

- Look for that name in the local frame.
- If not found, look for it in the global frame.
  (Built-in names like "max" are in the global frame too,
  but we don't draw them in environment diagrams.)

(Demo)