Lecture 4

Defining Functions

Academic Integrity Quiz

- **Remember**: quiz about the course AI policy
  - Have posted grades for completed quizes
  - Right now, missing ~21 enrolled students
  - If did not receive 9/10 (~70 students), take it again
- If you are not aware of the quiz
  - Go to http://www.cs.cornell.edu/courses/cs1110/
  - Click **Academic Integrity** under **Assessments**
  - Read and take quiz in CMS

The ACCEL Drama Continues

- Now hear construction going into October
  - This is best case; who *knows* when done
- We are working on long term plans
  - **Section 211** permanently moved to Phillips 318
  - **Sunday Consulting Hours** in Upson 216
  - **Mon-Thur Consulting Hours** in Upson 255
- Will update website when finalized
  - Choose menu **INFO > OFFICE HOURS**

Recall: Modules

- Modules provide extra functions, variables
  - **Example**: `math` provides `math.cos()`, `math.pi`
  - Access them with the `import` command
- Python provides a lot of them for us
- **This Lecture**: How to make modules
  - Atom Editor to *make* a module
  - Python to *use* the module

Two different programs

We Write Programs to Do Things

- Functions are the **key doers**

Function Call

- Command to **do** the function

```
>>> plus(23)
24
>>>
```

Function Definition

- Defines what function **does**

```
def plus(n):
    return n+1
```

- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

We Write Programs to Do Things

- Functions are the **key doers**

Function Call

- Command to **do** the function

```
>>> plus(23)
24
>>>
```

Function Definition

- Defines what function **does**

Function **Header**

```
def plus(n):
    return n+1
```

- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

We Write Programs to Do Things

- Functions are the **key doers**

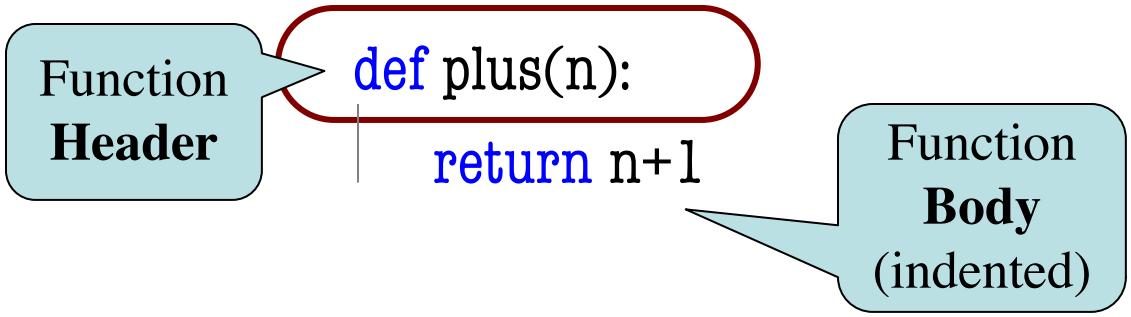
Function Call

- Command to **do** the function

```
>>> plus(23)
24
>>>
```

Function Definition

- Defines what function **does**



- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

We Write Programs to Do Things

- Functions are the **key doers**

Function Call

- Command to **do** the function

```
>>> plus(23)
24
```

argument to assign to n

Function Definition

- Defines what function **does**

```
def plus(n):
    return n+1
```

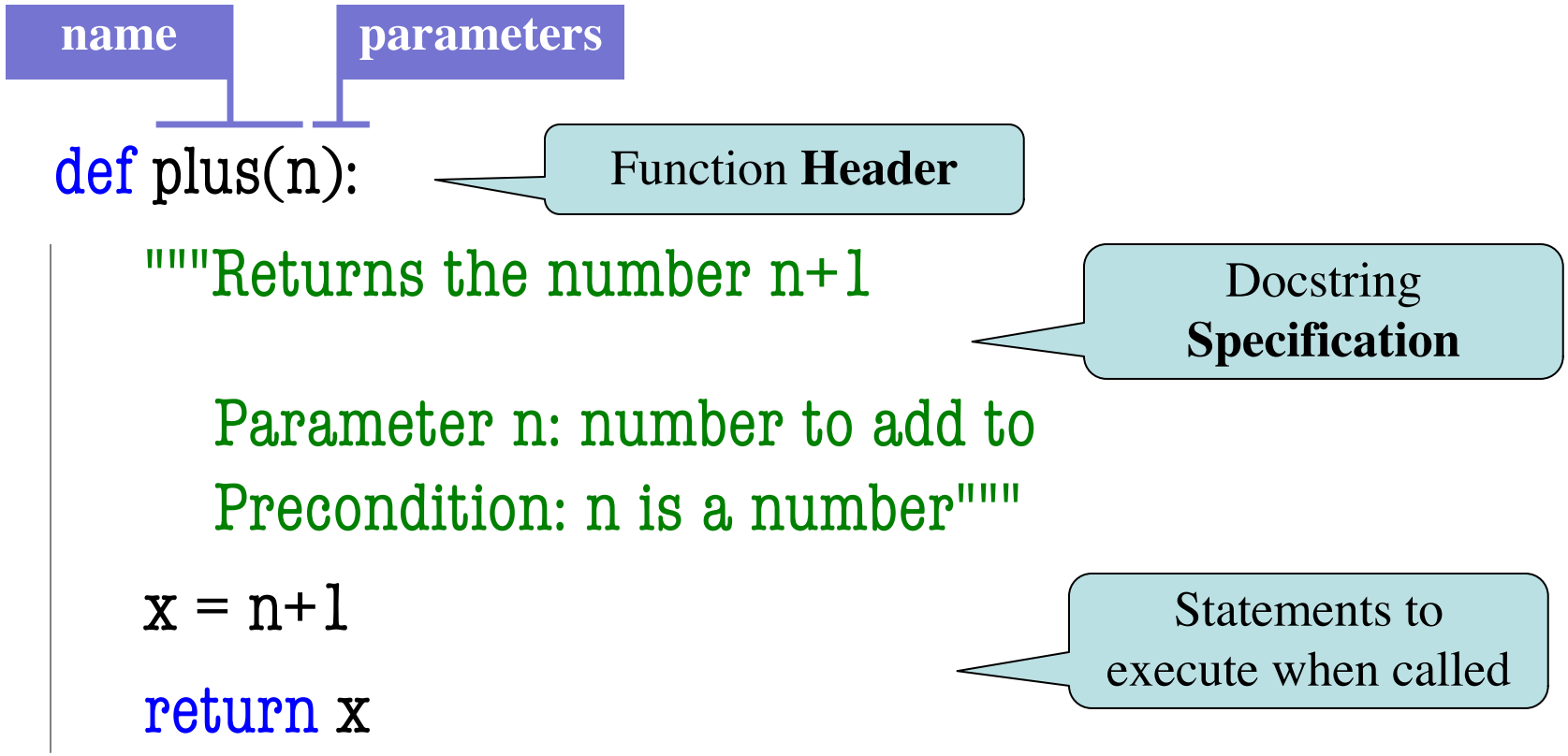
Function **Header**

declaration of **parameter** n

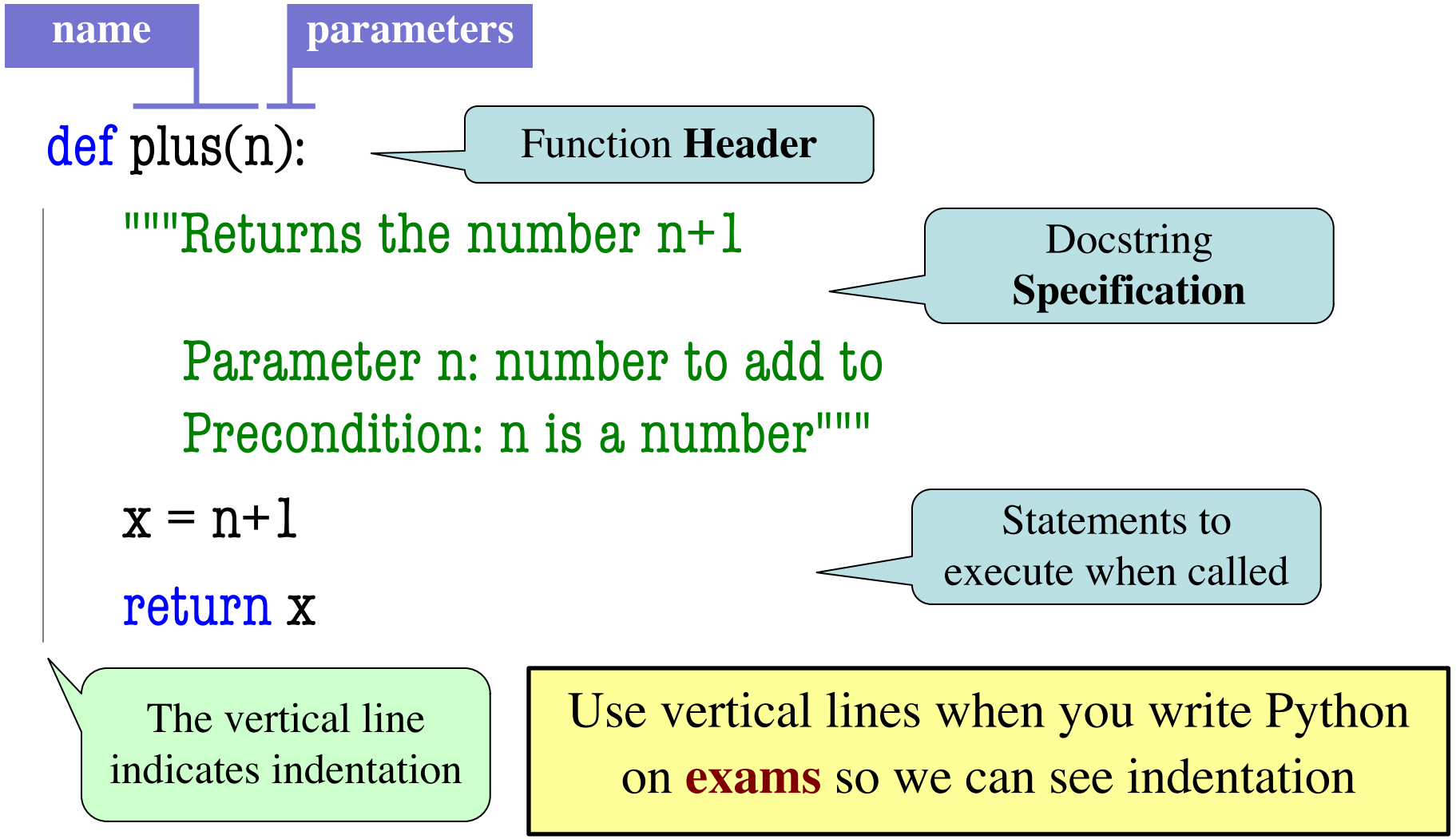
Function **Body** (indented)

- **Parameter**: variable that is listed within the parentheses of a method header.
- **Argument**: a value to assign to the method parameter when it is called

Anatomy of a Function Definition



Anatomy of a Function Definition



The **return** Statement

- **Format**: return <*expression*>
  - Used to evaluate ***function call*** (as an expression)
  - Also stops executing the function!
  - Any statements after a **return** are ignored
- **Example**: temperature converter function

```
def to_centigrade(x):
    """Returns: x converted to centigrade"""
    return 5*(x-32)/9.0
```

A More Complex Example

Function Definition

```
def foo(a,b):
    """Return something
       Param a: number
       Param b: number"""
    x = a
    y = b
    return x*y+y
```

Function Call

```
>>> x = 2
>>> foo(3,4)
```

x ?

What is in the box?

A More Complex Example

Function Definition

```
def foo(a,b):
    """Return something
       Param a: number
       Param b: number"""
    x = a
    y = b
    return x*y+y
```

Function Call

```
>>> x = 2
>>> foo(3,4)
```

x ?

What is in the box?

A: 2
B: 3
C: 16
D: Nothing!
E: I do not know

A More Complex Example

Function Definition

```
def foo(a,b):
    """Return something
       Param a: number
       Param b: number"""
    x = a
    y = b
    return x*y+y
```

Function Call

```
>>> x = 2
>>> foo(3,4)
```

x ?

What is in the box?

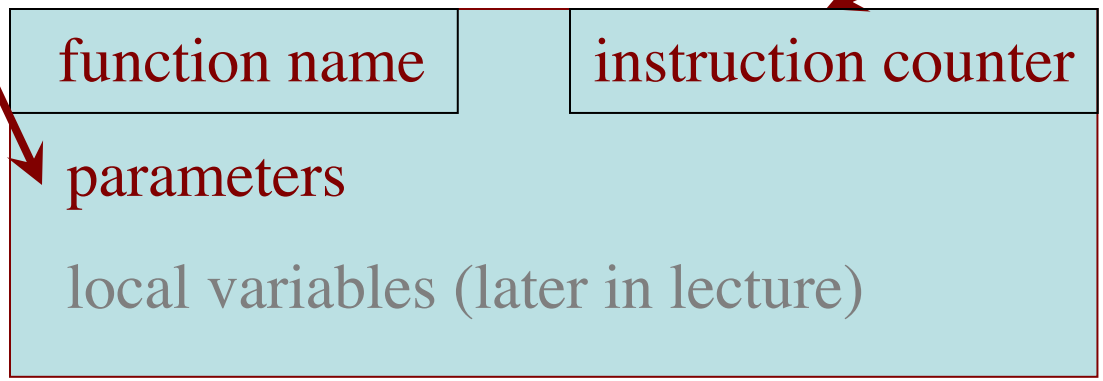
A: 2 **CORRECT**
B: 3
C: 16
D: Nothing!
E: I do not know

Understanding How Functions Work

- **Function Frame**: Representation of function call
- A **conceptual model** of Python

Draw parameters as variables (named boxes)

- Number of statement in the function body to execute next
- **Starts with line no. of body**



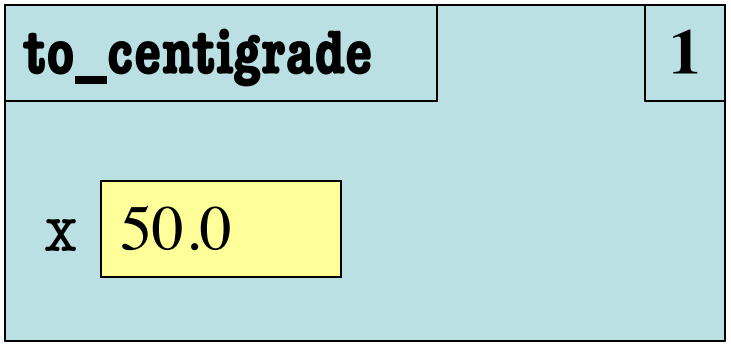
Text (Section 3.10) vs. Class

Textbook

to_centigrade

x –> 50.0

This Class



Definition:

```
1  def to_centigrade(x):
       return 5*(x-32)/9.0
```

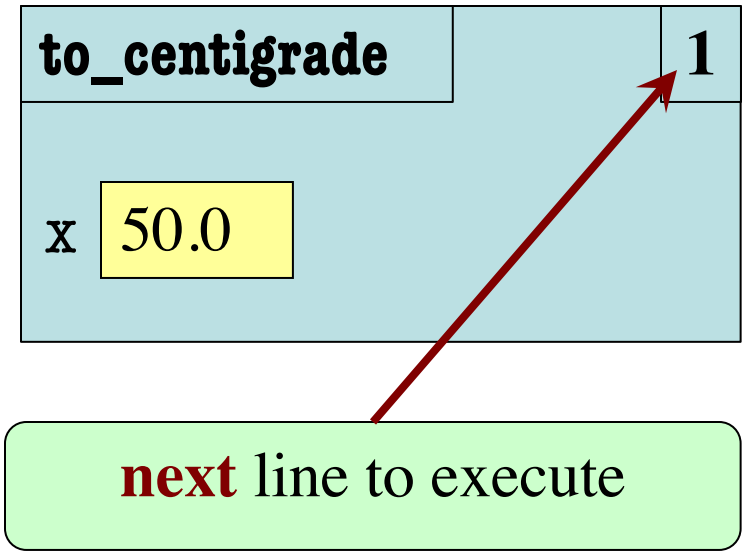
Call: to_centigrade(50.0)

Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
def to_centigrade(x):
1   |   return 5*(x-32)/9.0
```

Initial call frame (before exec body)

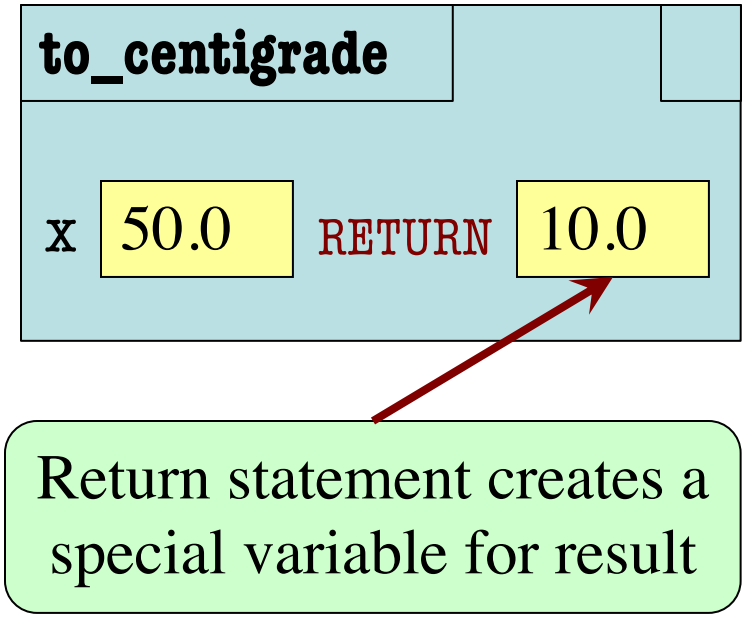


Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
def to_centigrade(x):
    return 5*(x-32)/9.0
```
1

Executing the return statement

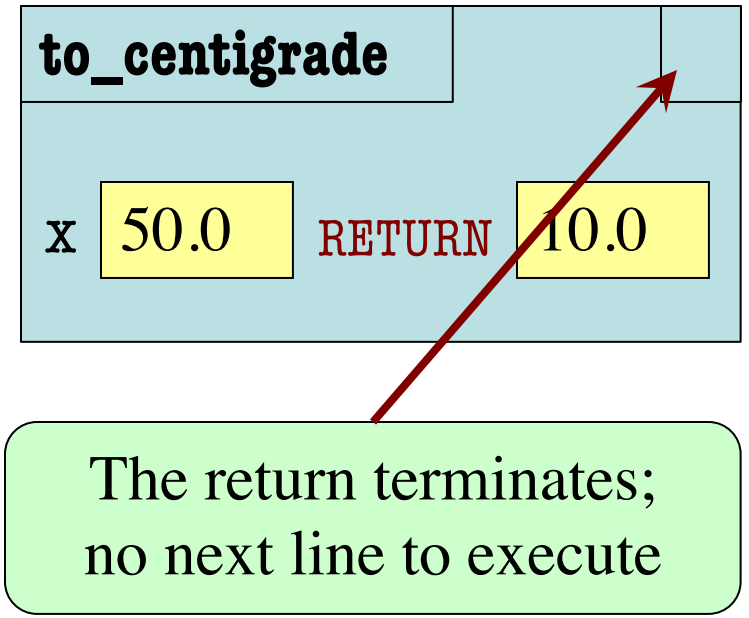


Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
def to_centigrade(x):
1   |   return 5*(x-32)/9.0
```

Executing the return statement



Example: to_centigrade(50.0)

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

```
1  def to_centigrade(x):
       return 5*(x-32)/9.0
```

ERASE WHOLE FRAME

But don't actually erase on an exam

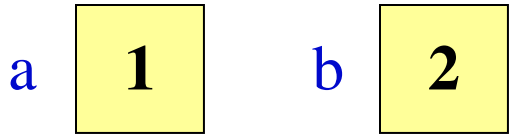
Call Frames vs. Global Variables

The specification is a **lie**:

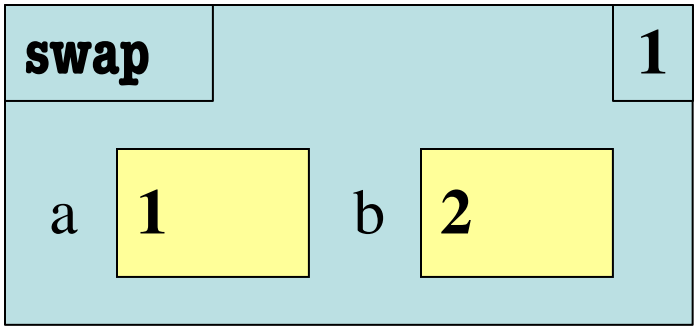
```
def swap(a,b):
    """Swap global a & b"""
1   tmp = a
2   a = b
3   b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

Global Variables



Call Frame



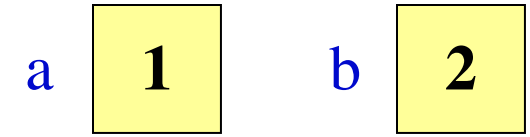
Call Frames vs. Global Variables

The specification is a **lie**:

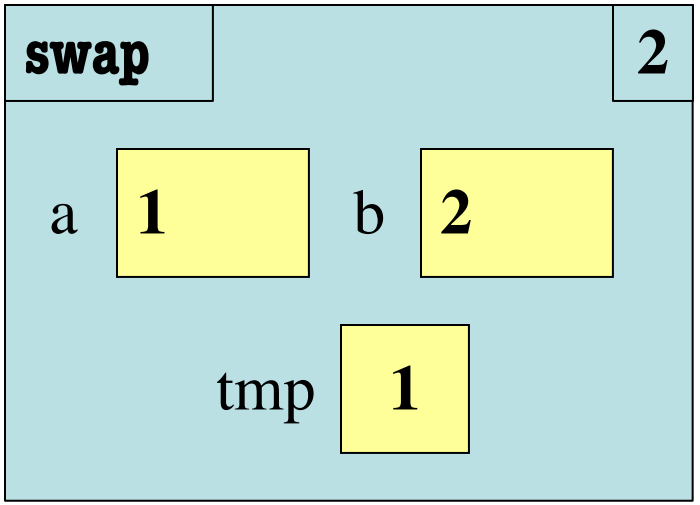
```
def swap(a,b):
    """Swap global a & b"""
1   tmp = a
2   a = b
3   b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

Global Variables



Call Frame



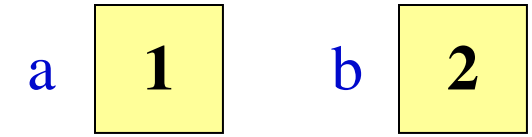
Call Frames vs. Global Variables

The specification is a **lie**:

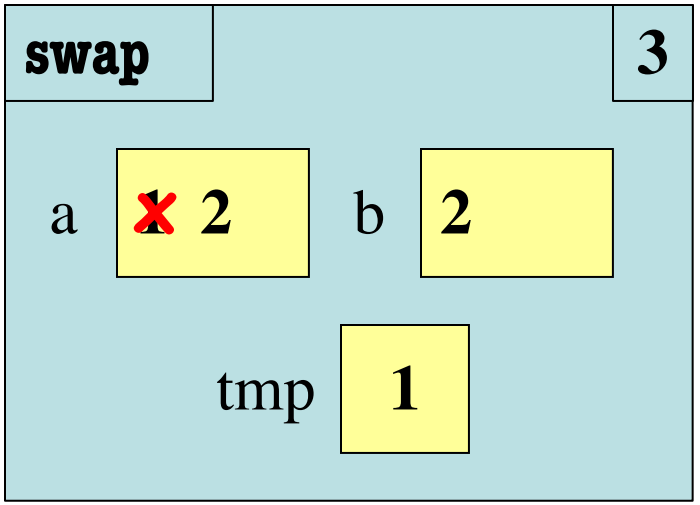
```
def swap(a,b):
    """Swap global a & b"""
1   tmp = a
2   a = b
3   b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

Global Variables



Call Frame



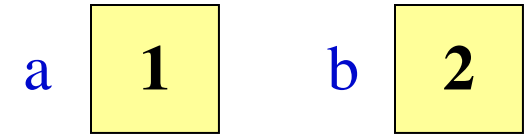
Call Frames vs. Global Variables

The specification is a **lie**:

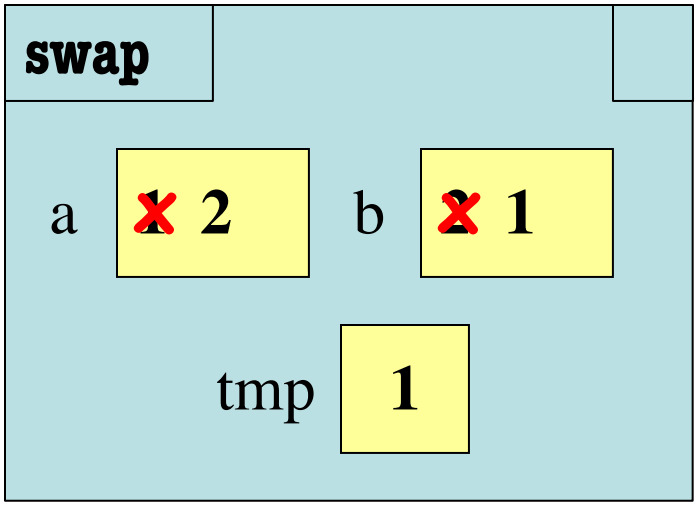
```
def swap(a,b):
    """Swap global a & b"""
1   tmp = a
2   a = b
3   b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

Global Variables



Call Frame



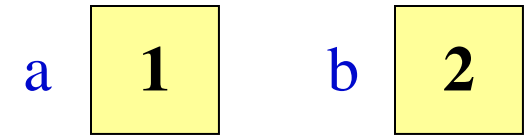
Call Frames vs. Global Variables

The specification is a **lie**:

```
def swap(a,b):
    """Swap global a & b"""
    tmp = a
    a = b
    b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

Global Variables



Call Frame

ERASE THE FRAME

Exercise Time

Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

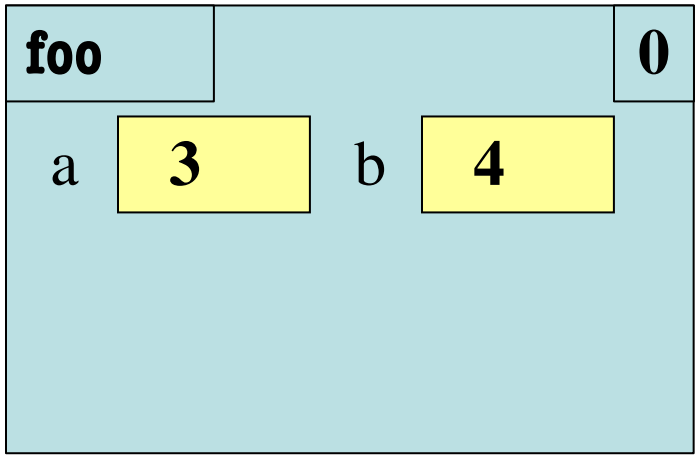
Function Call

```
>>> x = foo(3,4)
```

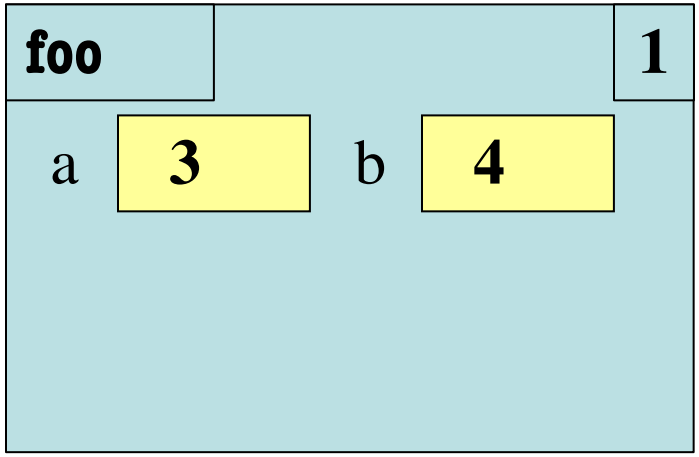
What does the frame look like at the **start**?

Which One is Closest to Your Answer?

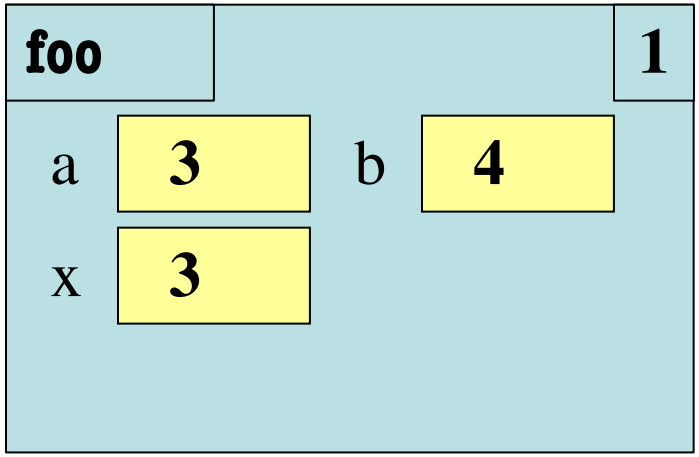
A:



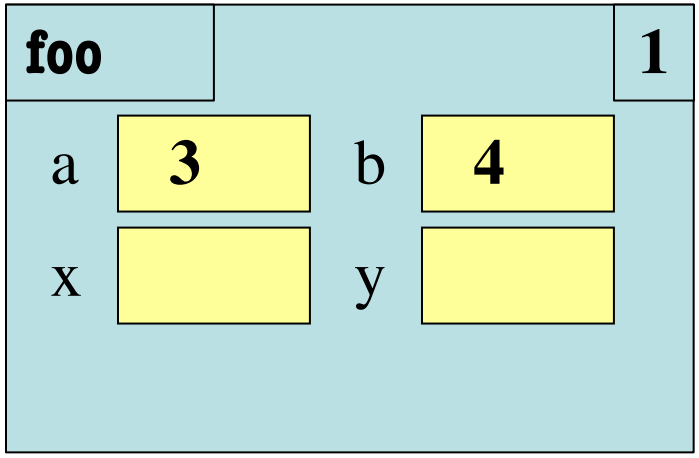
B:



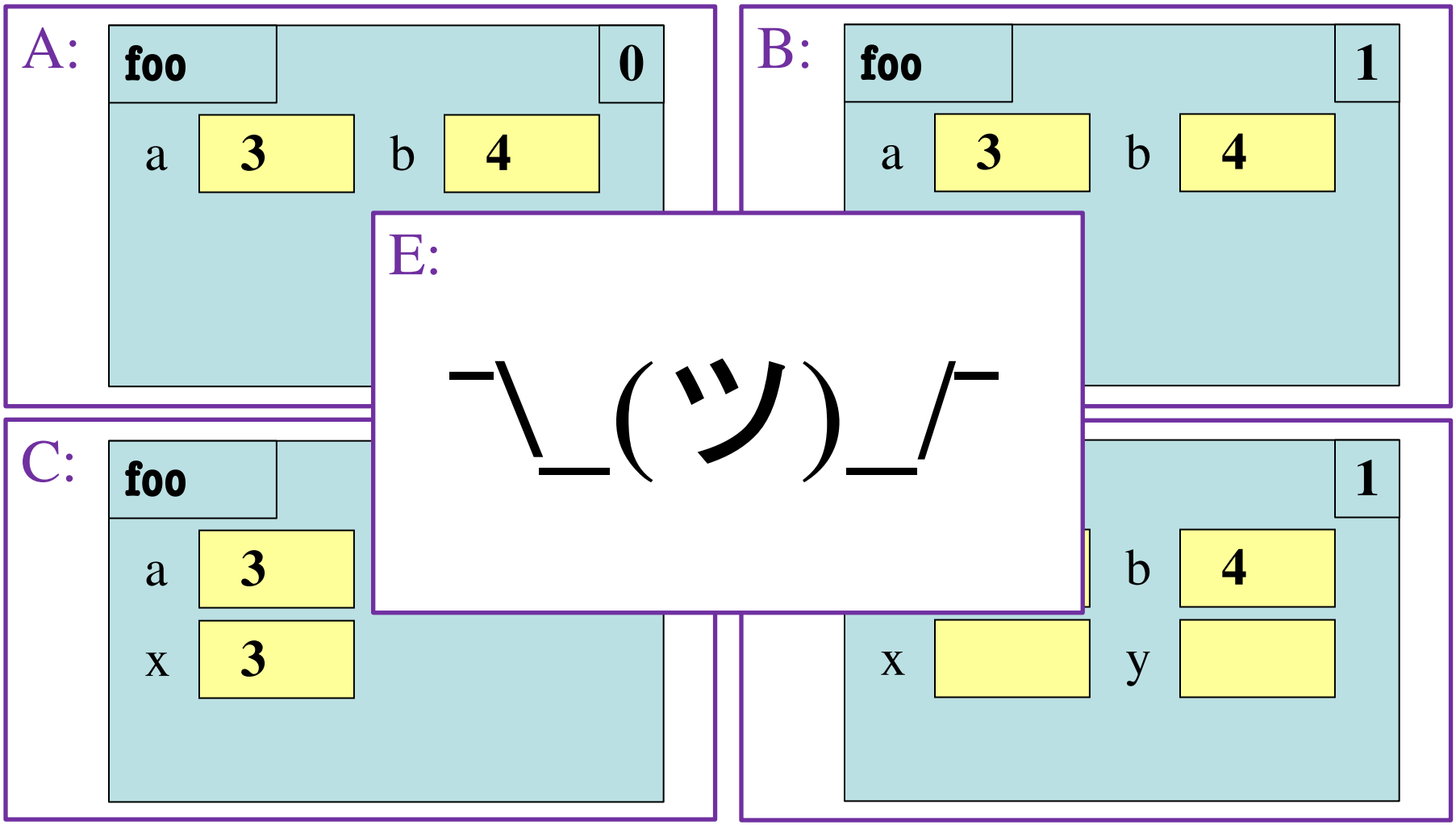
C:



D:



Which One is Closest to Your Answer?



Exercise Time

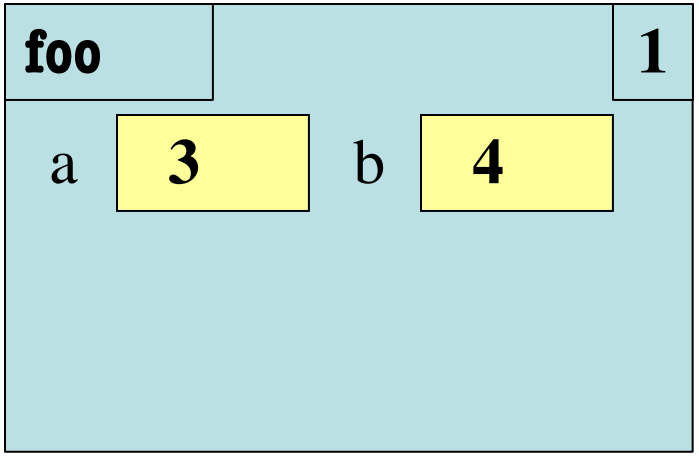
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

B:



Exercise Time

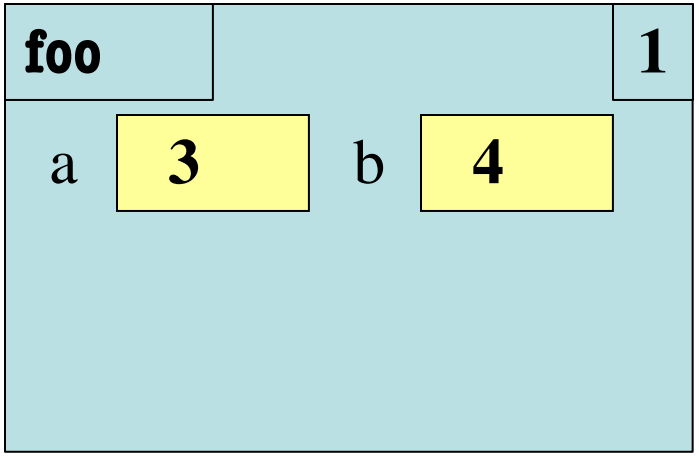
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

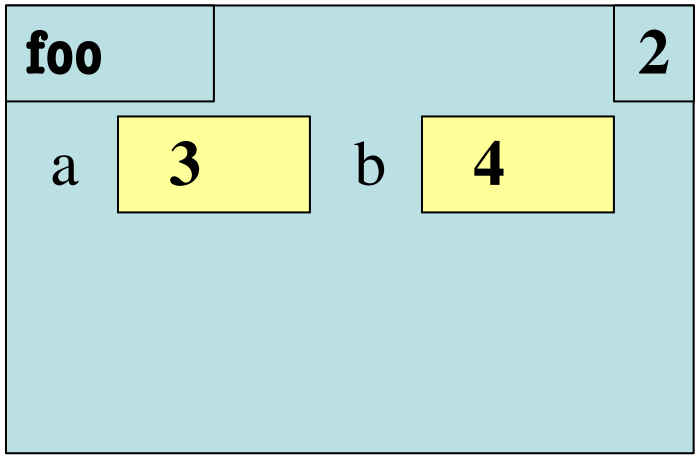
B:



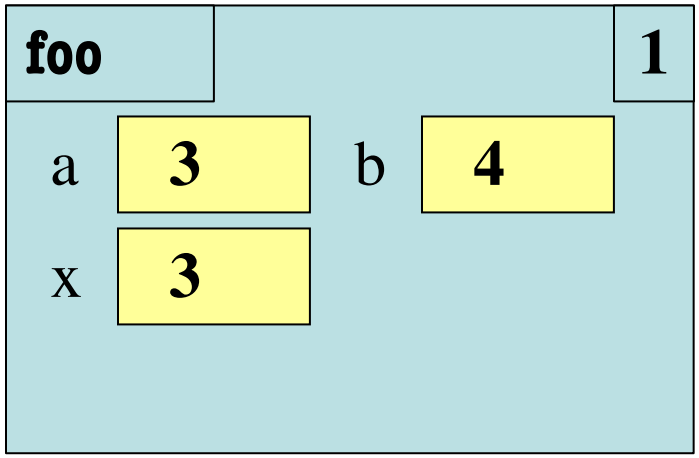
What is the **next step**?

Which One is Closest to Your Answer?

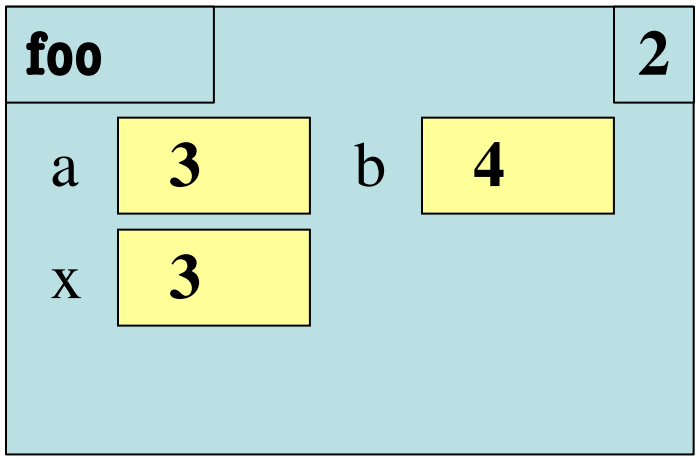
A:



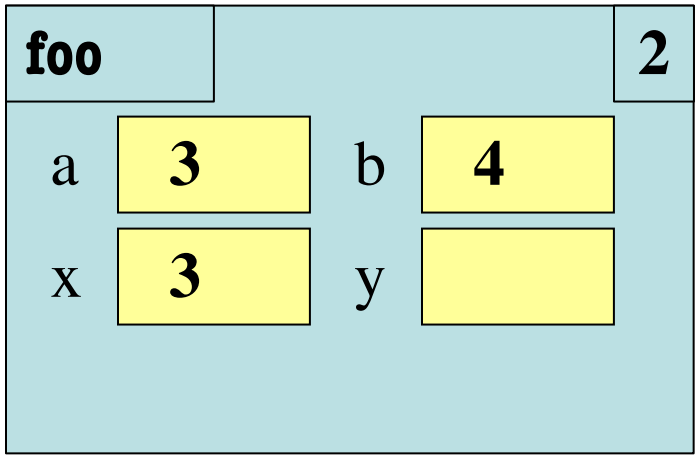
B:



C:



D:



Exercise Time

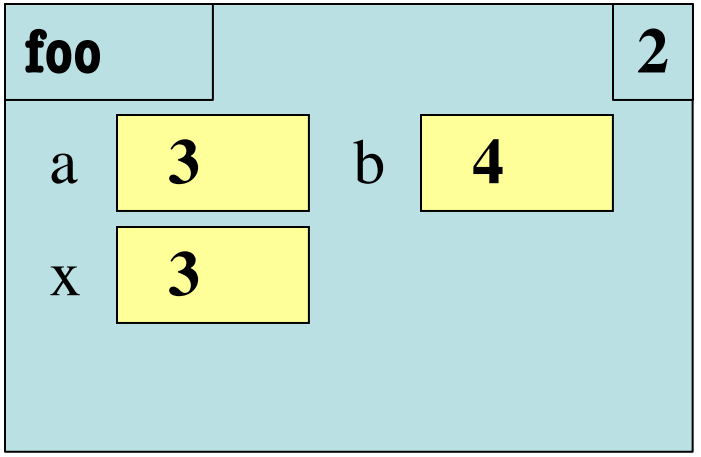
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

C:



Exercise Time

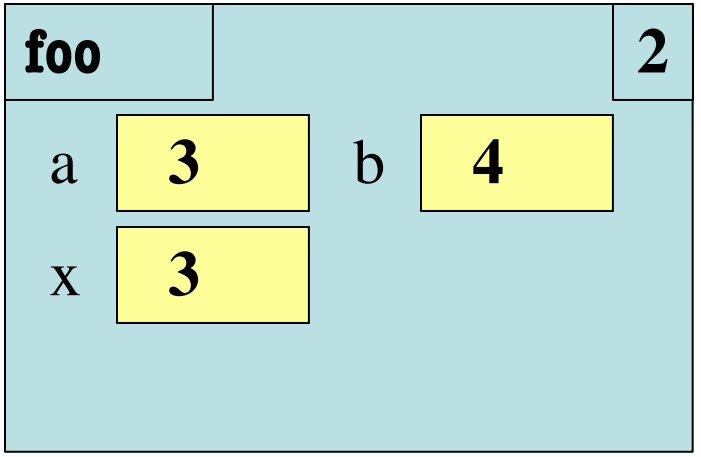
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

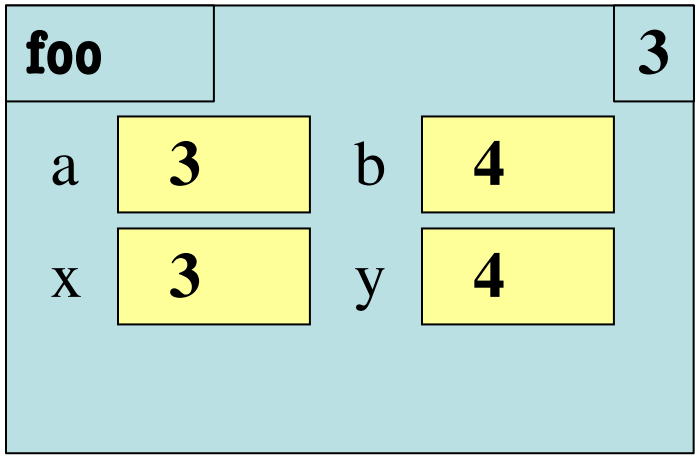
C:



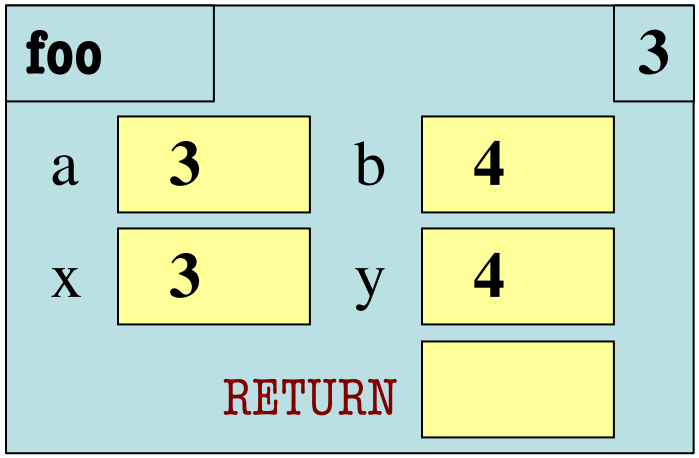
What is the **next step**?

Which One is Closest to Your Answer?

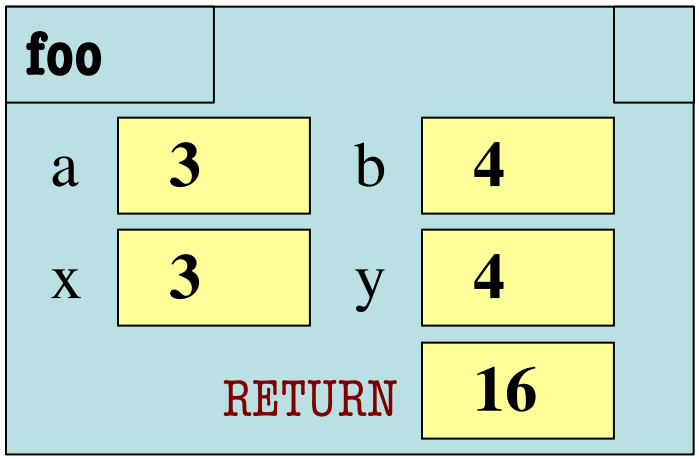
A:



B:



C:



D:

ERASE THE FRAME

Exercise Time

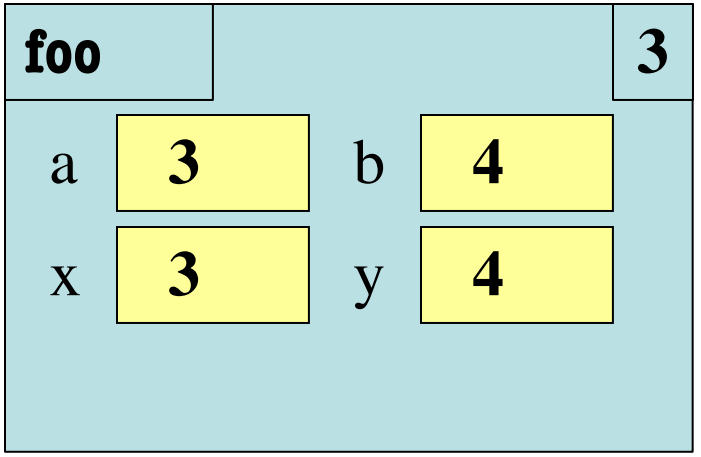
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

A:



Exercise Time

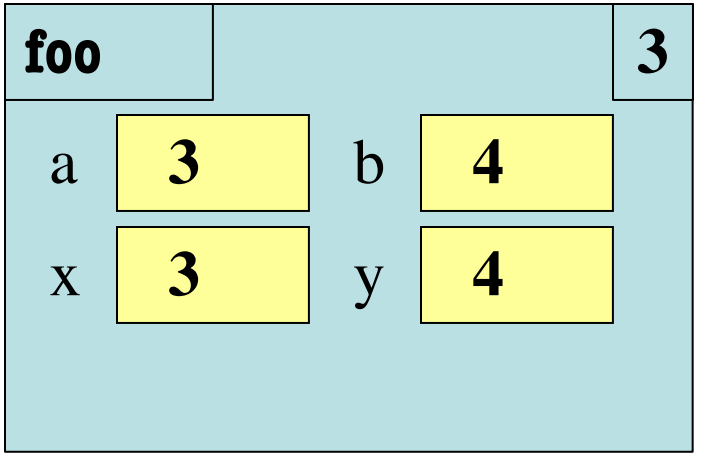
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

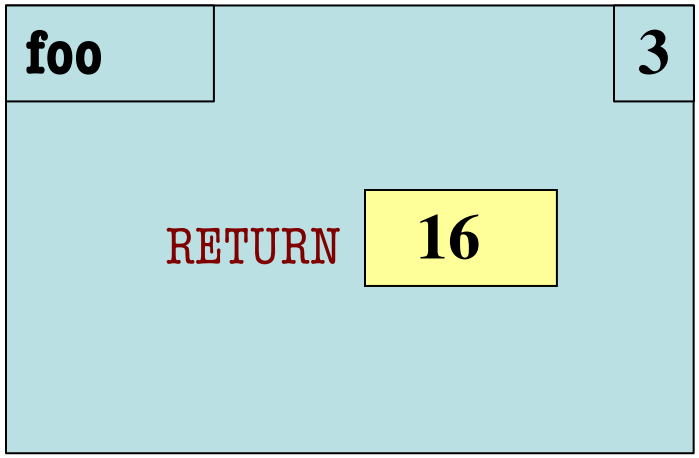
A:



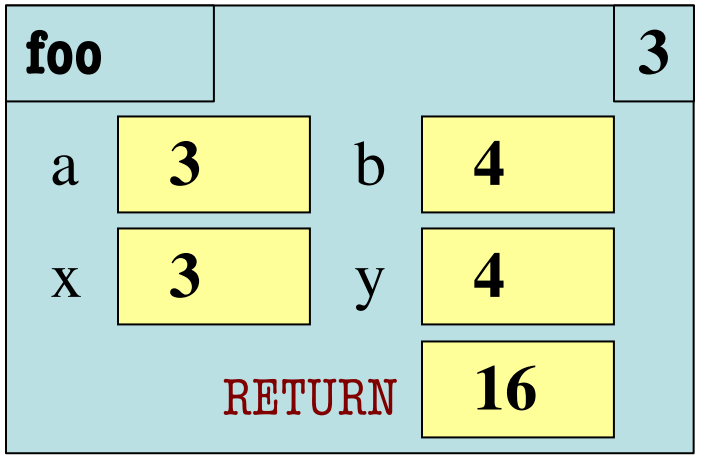
What is the **next step**?

Which One is Closest to Your Answer?

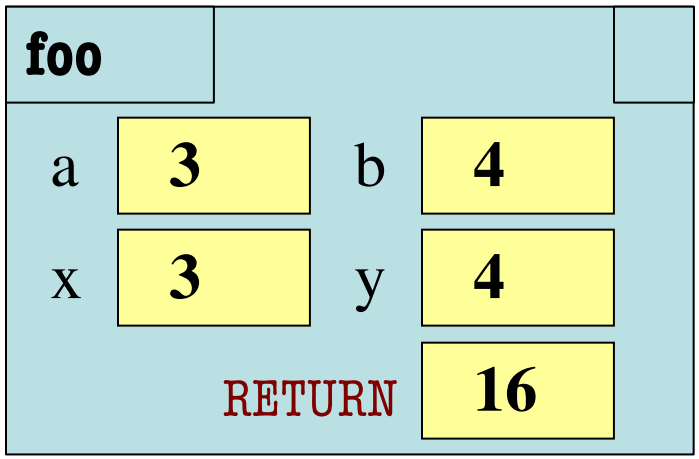
A:



B:



C:



D:

ERASE THE FRAME

Exercise Time

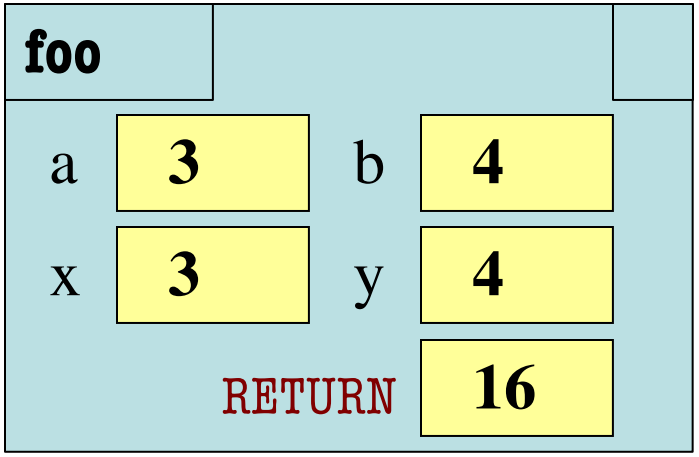
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

C:



Exercise Time

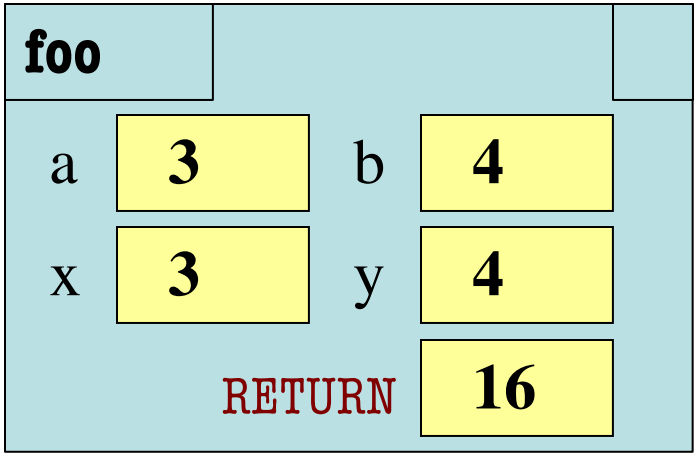
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

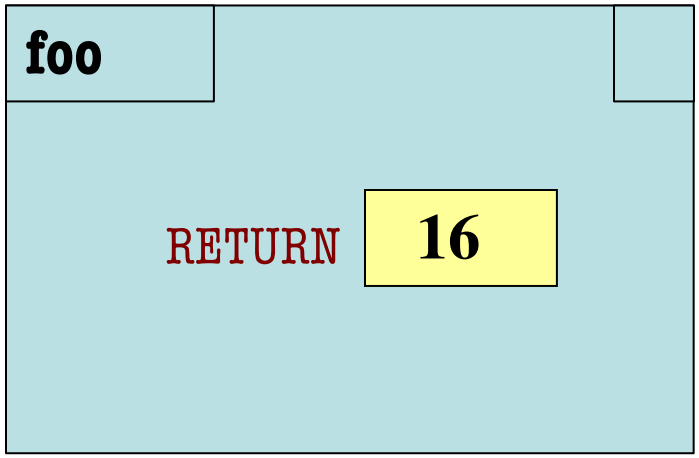
C:



What is the **next step**?

Which One is Closest to Your Answer?

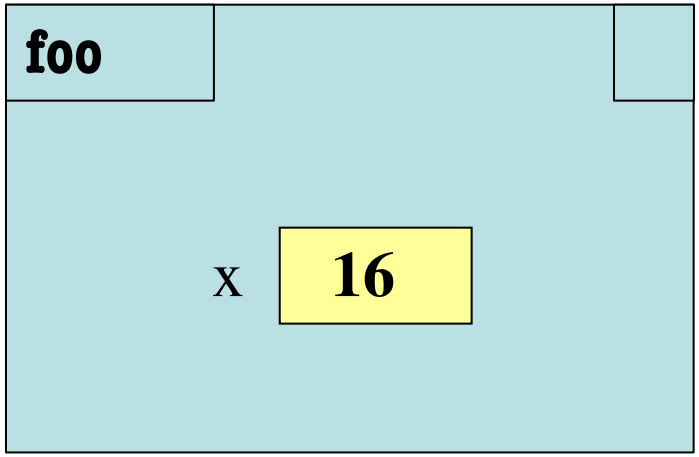
A:



B:

ERASE THE FRAME

C:



D:

x 16

ERASE THE FRAME

Exercise Time

Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

```
>>> x = foo(3,4)
```

D:

x **16**

ERASE THE FRAME

Exercise Time

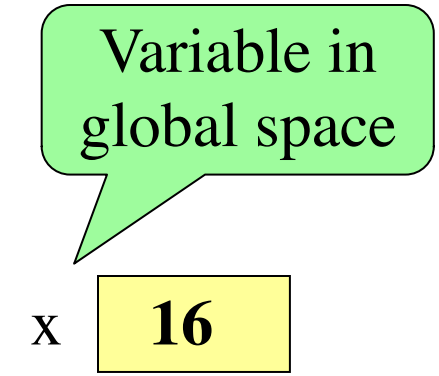
Function Definition

```
def foo(a,b):
    """Return something
       Param x: a number
       Param y: a number"""
1   x = a
2   y = b
3   return x*y+y
```

Function Call

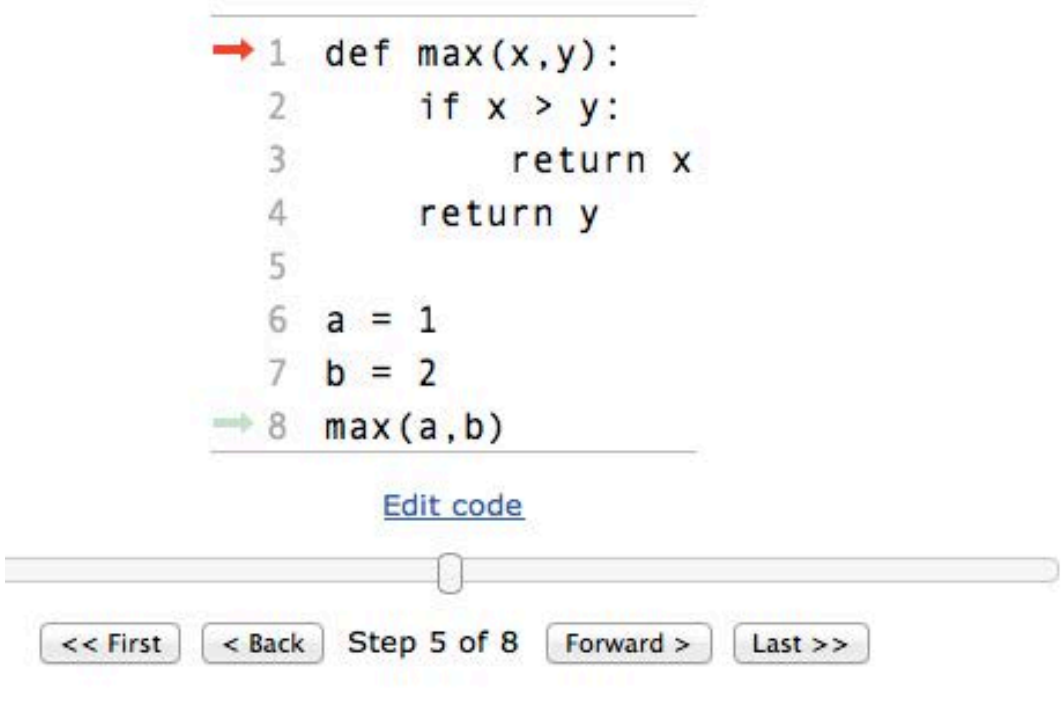
```
>>> x = foo(3,4)
```

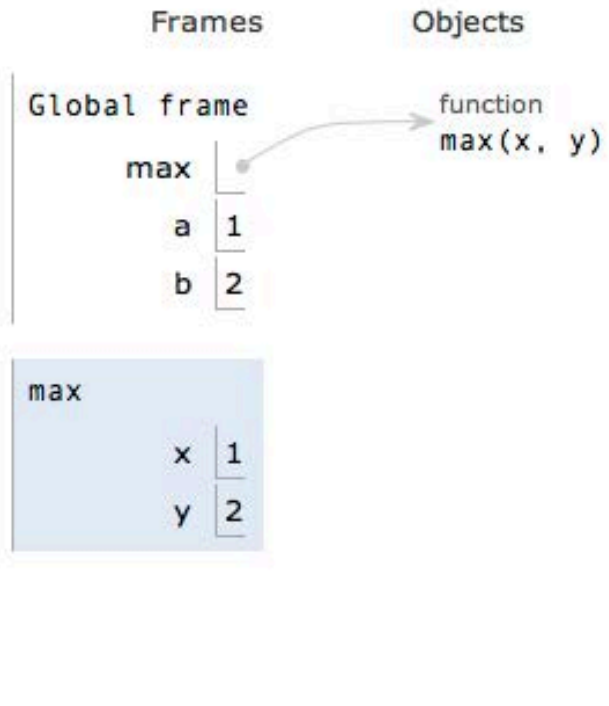
D:



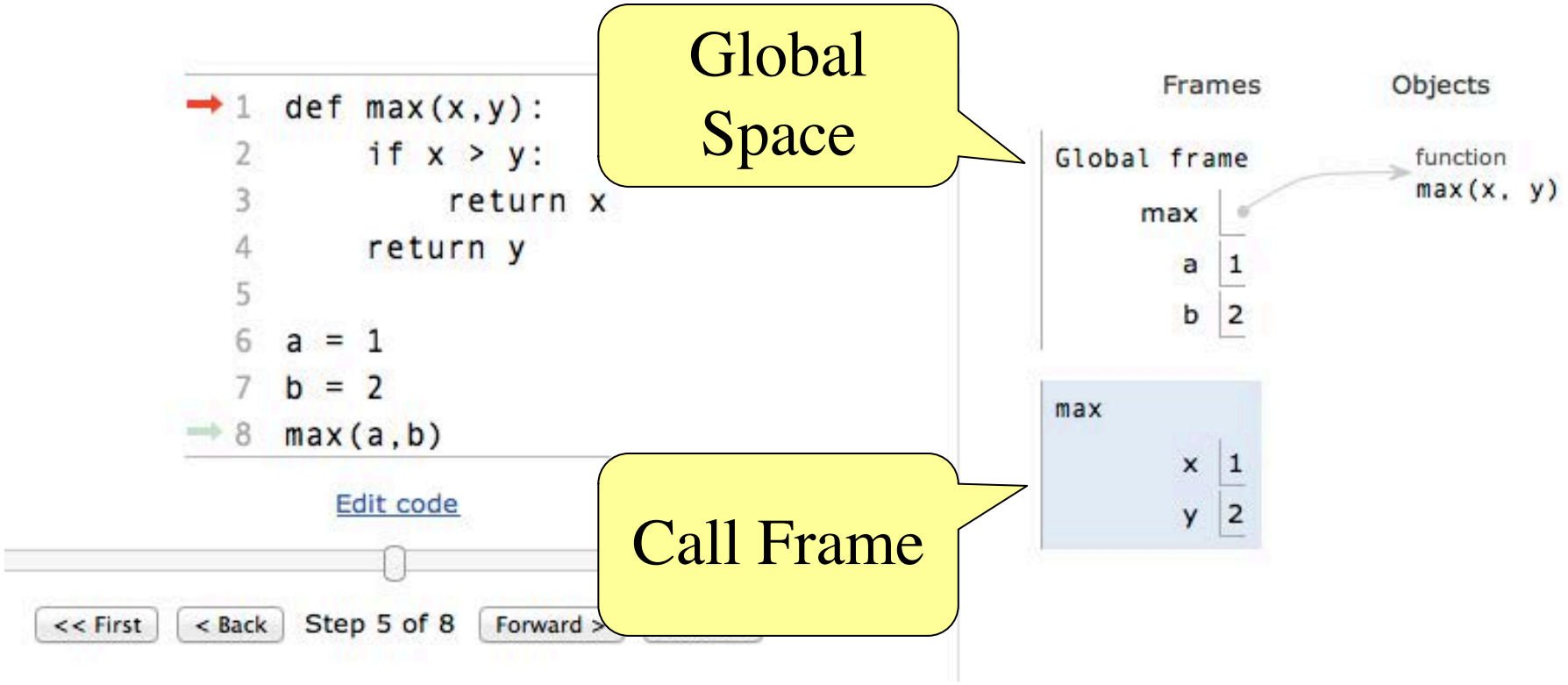
ERASE THE FRAME

Visualizing Frames: The Python Tutor

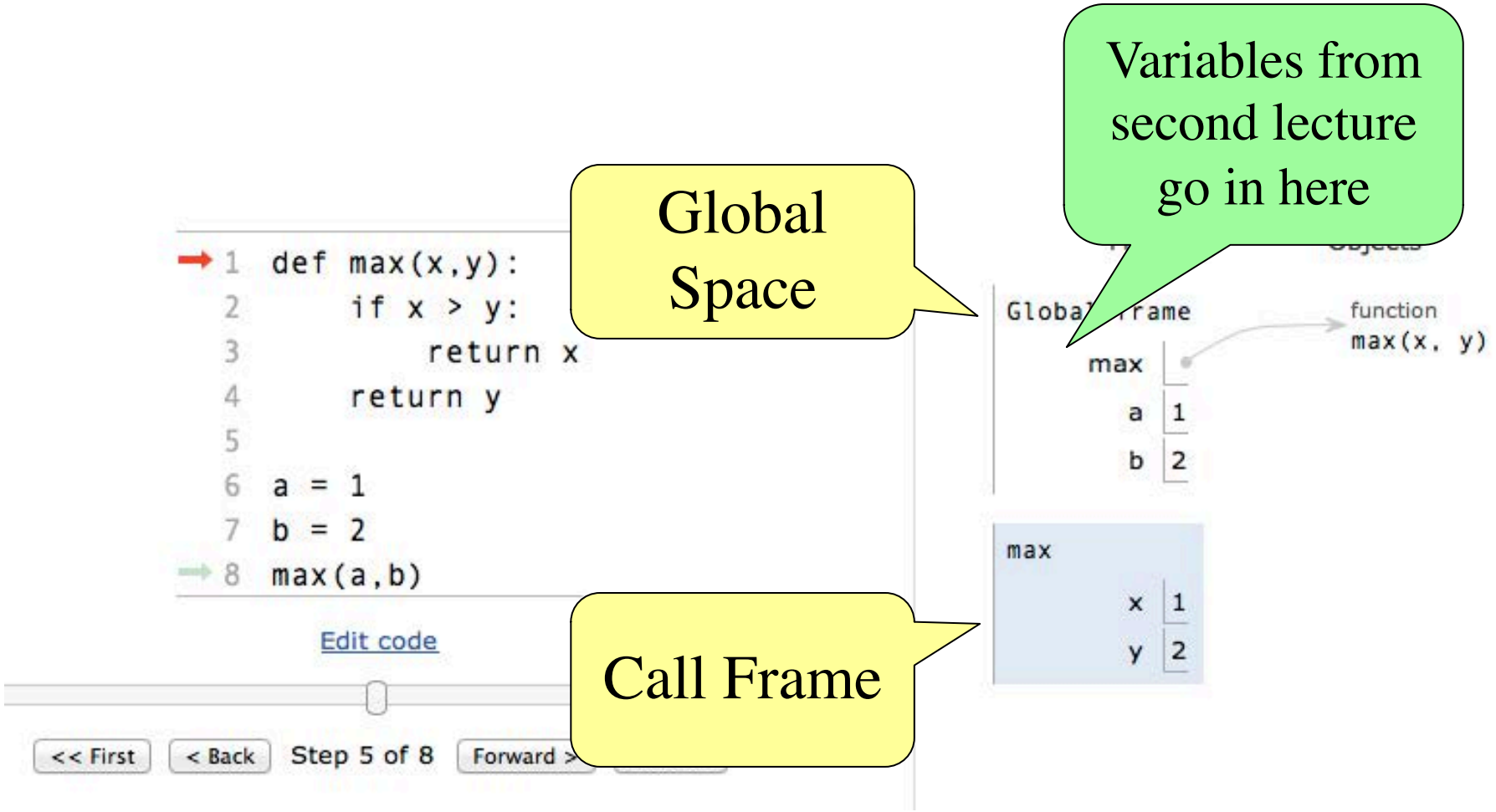




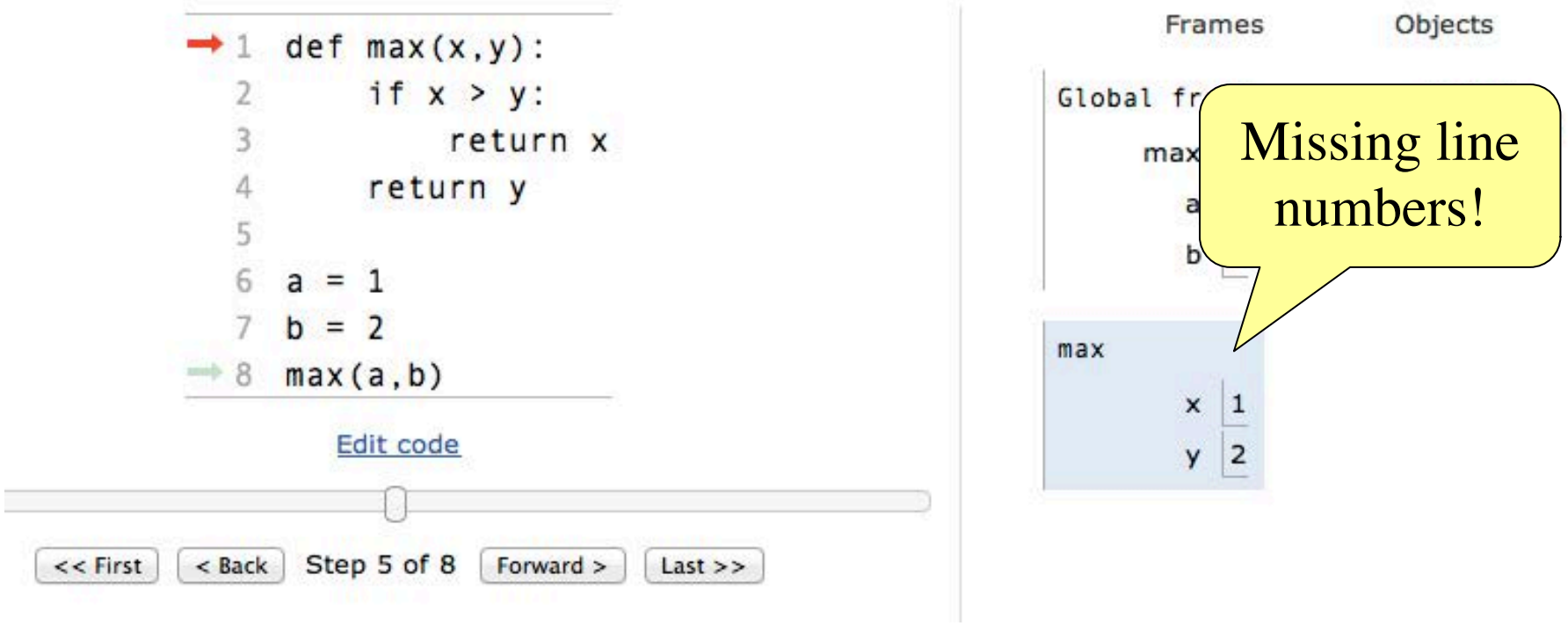
Visualizing Frames: The Python Tutor



Visualizing Frames: The Python Tutor

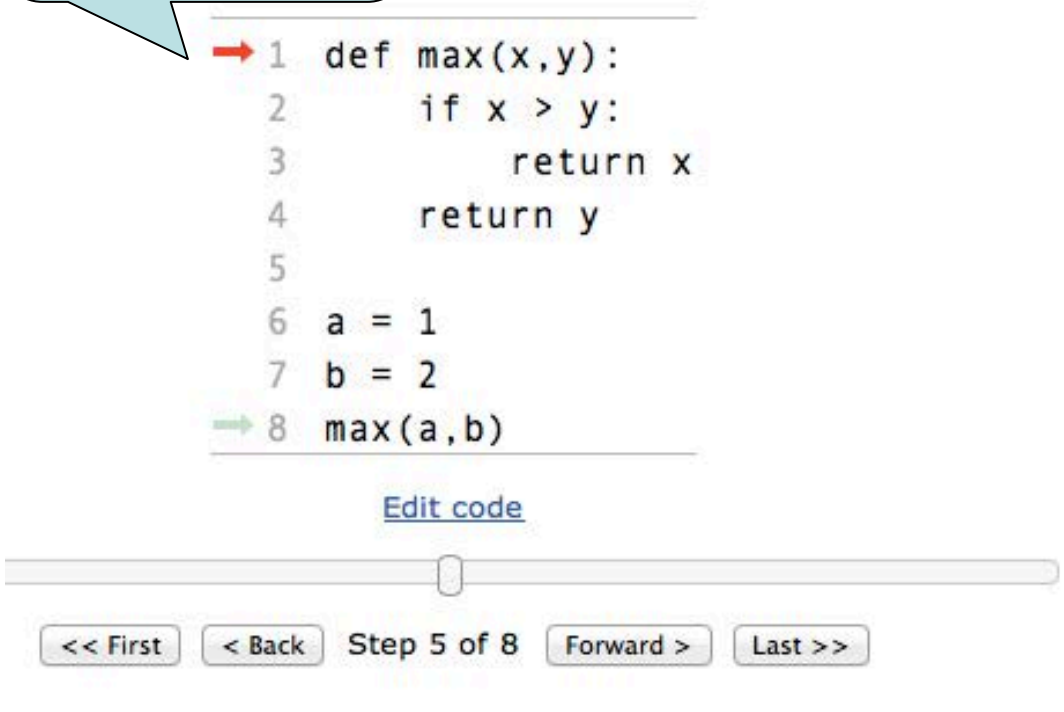


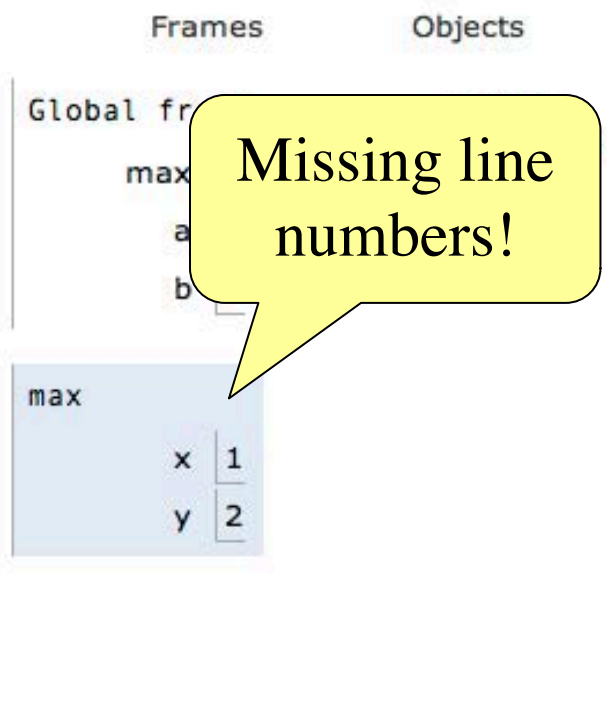
Visualizing Frames: The Python Tutor



Visualizing Frames: The Python Tutor

Line number marked here (sort-of)





Missing line numbers!

Next Time: Text Processing