global

f

CSCI 141

Scott Wehrwein

Function Parameters are Local Variables
Executing Function Calls

Goals

- Know the meaning of local variables and variable scope and how it relates to function parameters.
- Be able to execute functions the way Python does, and understand the implications for local variables and scope

Parameters vs Arguments

Parameters: variable names that will refer to the input arguments.

Parameters (these are new):
variables that take on the value of the arguments

```
def add2(a, b):
    """ Print the sum of a and b """
    print(a + b)


add2(4, 10)
```

Arguments (we've seen these before):
values passed into a function.

Parameters vs Arguments

Parameters: variable names that will refer to the input arguments.

Parameters (these are new):
variables that take on the value of the arguments

```
def add2(a, b):
    """ Print the sum of a and b """
    print(a + b)
```

Inside the function, the parameters are **local variables** that refer to the arguments passed into the function.

```
add2(4, 10)
```

Arguments (we've seen these before):
values passed into a function.

Parameters: Digging Deeper

```
def name(parameters):
    statements
```

Parameters: Digging Deeper

```
def name(parameters):
    statements
```

inputs

comma-separated list of parameters

Parameters: Digging Deeper

```
def name(parameters):
    statements
```

inputs

comma-separated list of parameters

Inside the function, the parameters are **local variables** that refer to the arguments passed into the function.

Parameters are Local Variables

- They **only** exist inside the function.
- Any other variables declared inside a function are also local variables.
- This is an example of a broader concept called scope: a variable's scope is the set of statements in which it is visible/usable.
- A local variable's scope is limited to the function inside which it's defined.

Variable Scope

```
1  def print_rectangle_area(width, height):
2      """ Print the area of a width-by-height
3          rectangle """
4
5      area = width * height
6      print(area)
7
8  w = 4
9  h = 3
10 a = w * h
11 print_rectangle_area(w, h)
12
```

Variable Scope

```
1  def print_rectangle_area(width, height):
2      """ Print the area of a width-by-height
3          rectangle """
4
5      area = width * height
6      print(area)
7
8  w = 4
9  h = 3
10 a = w * h
11 print_rectangle_area(w, h)
12
```

In which line is `area` in scope?

A. 2
B. 6
C. 8
D. 12

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

What happens when we run this program?

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result
```

Which of the variables in this program are **local** variables?

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

What happens when we run this program?

```
10
2
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a
```

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

What happens when we run this program?

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a
```

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

What happens when we run this program?

```
10
2
```

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1
```

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

What happens when we run this program?

Local Variables: Example

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1
```

```
a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

What happens when we run this program?

```
10
2
```

How did I know all that?

I know how to execute functions like Python does.

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

Technically, global variables can be *read* but not *modified* from an inner scope.

In this class we'll always avoid referring to global variables from inside functions entirely.

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

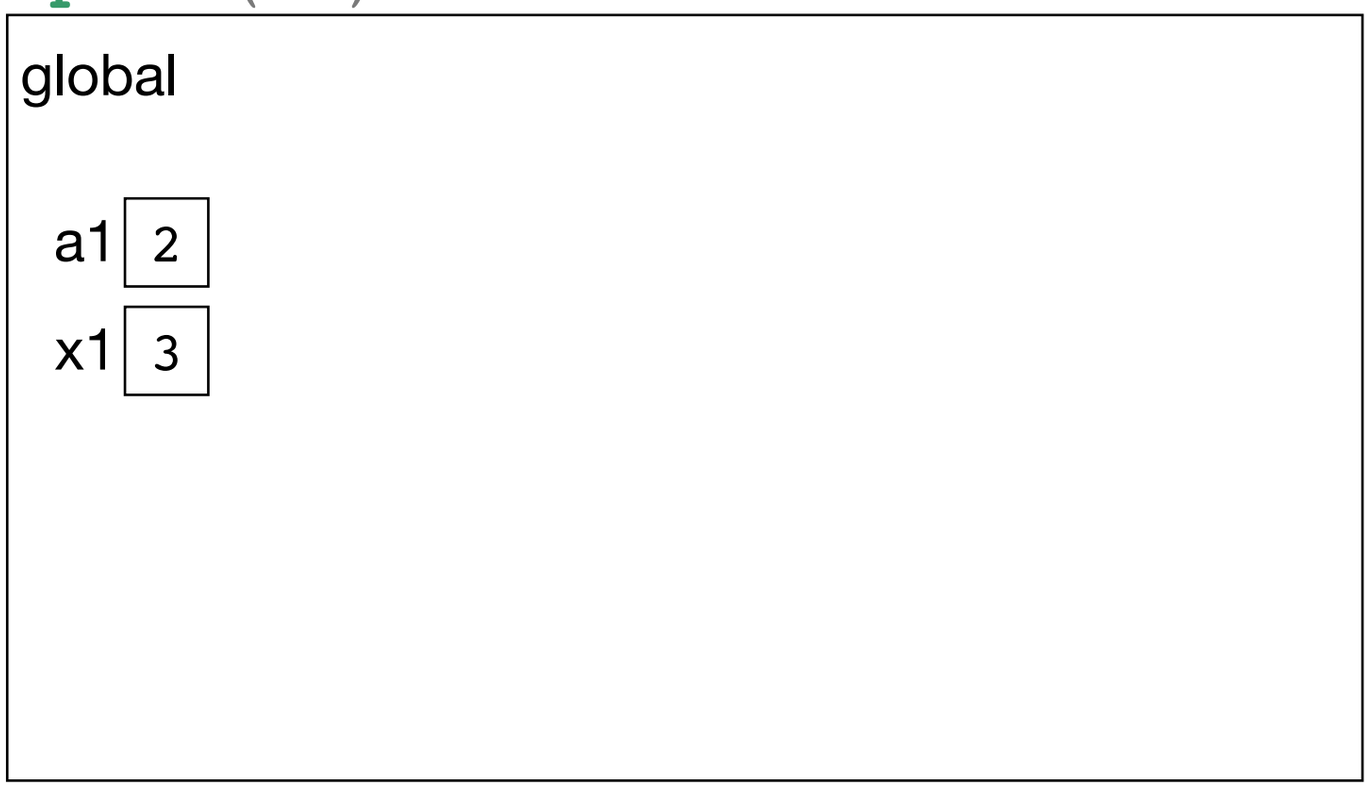
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



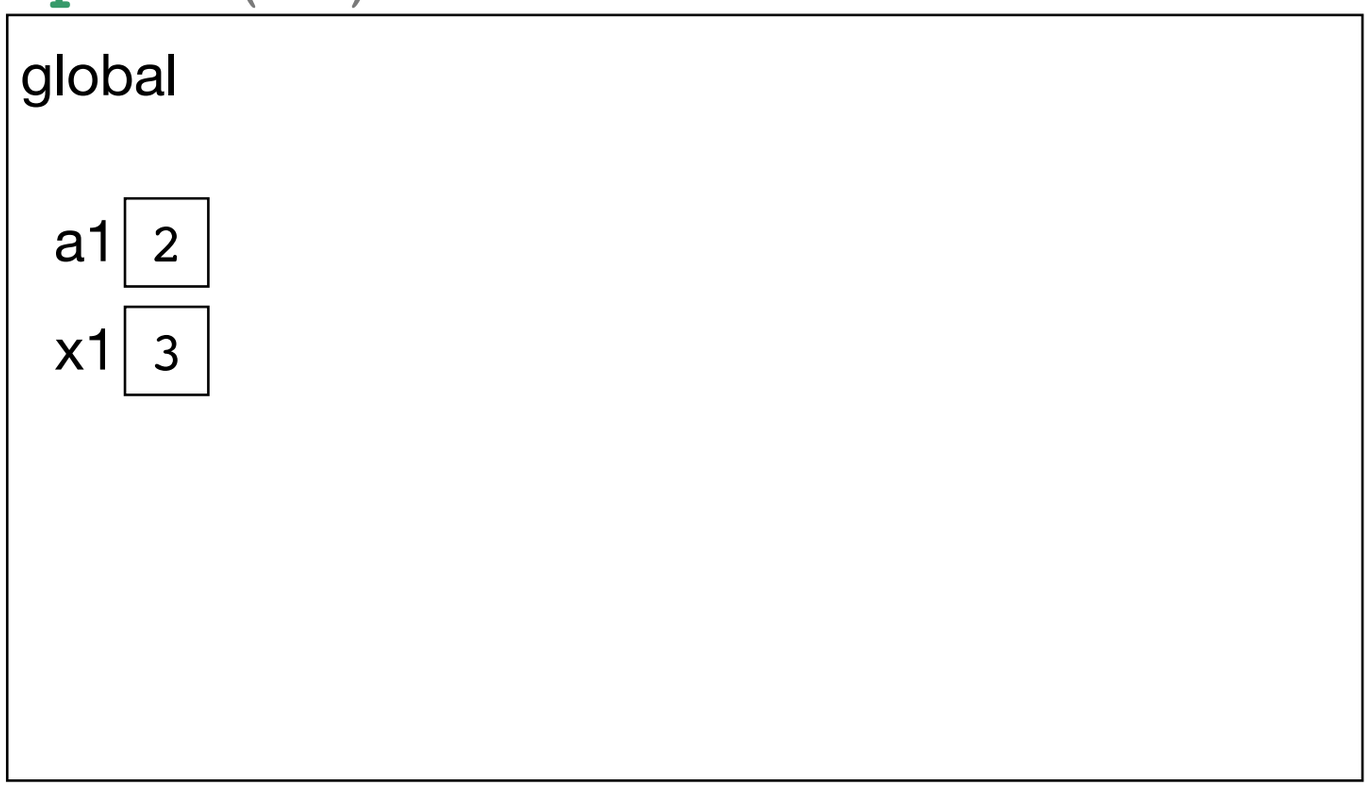
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```

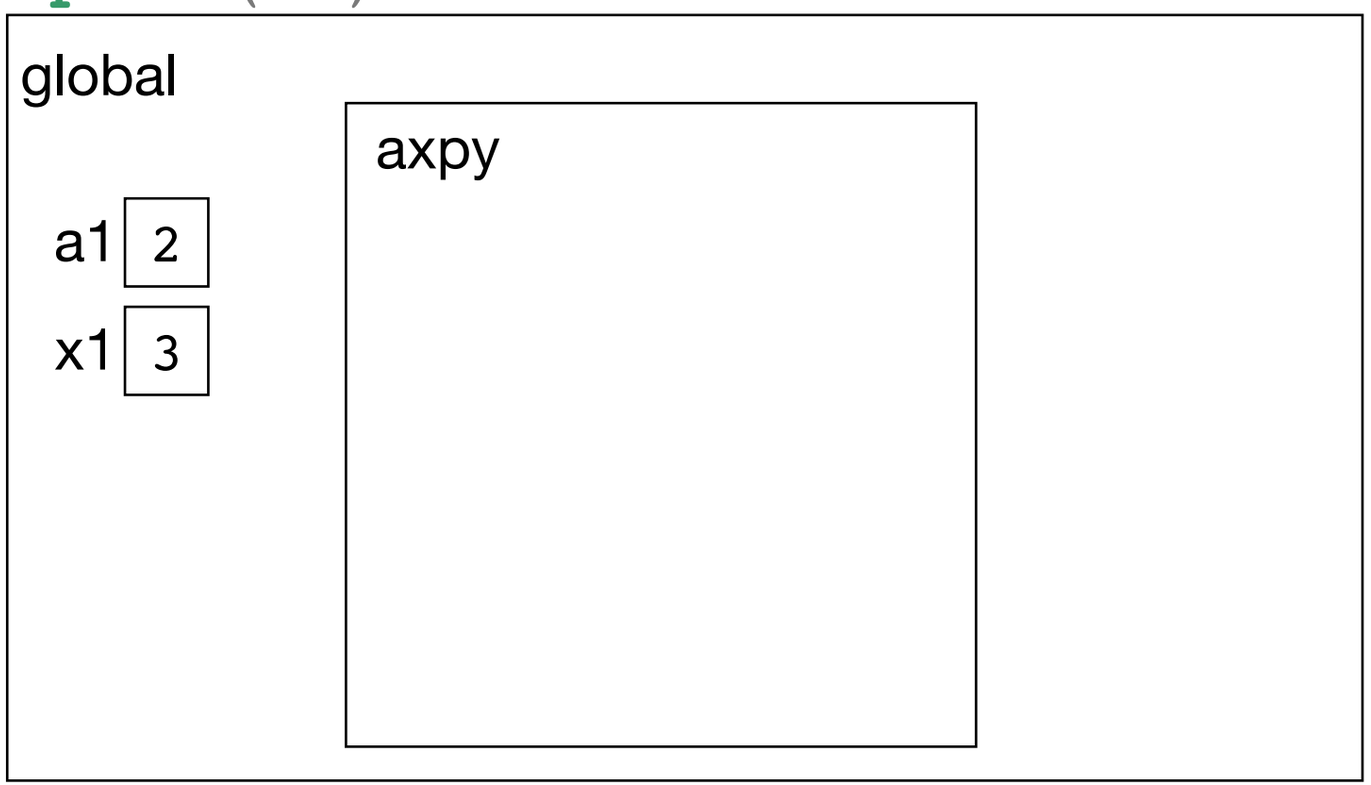
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



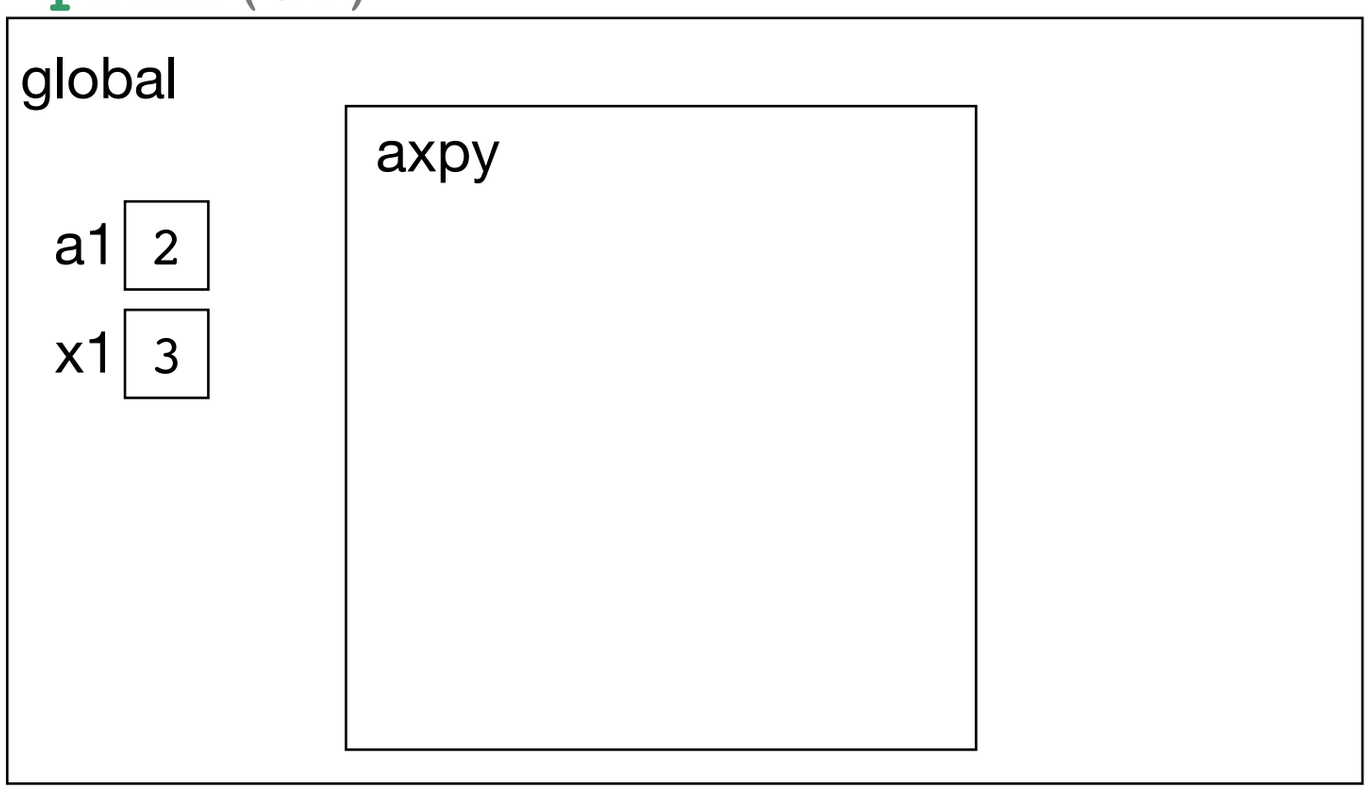
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



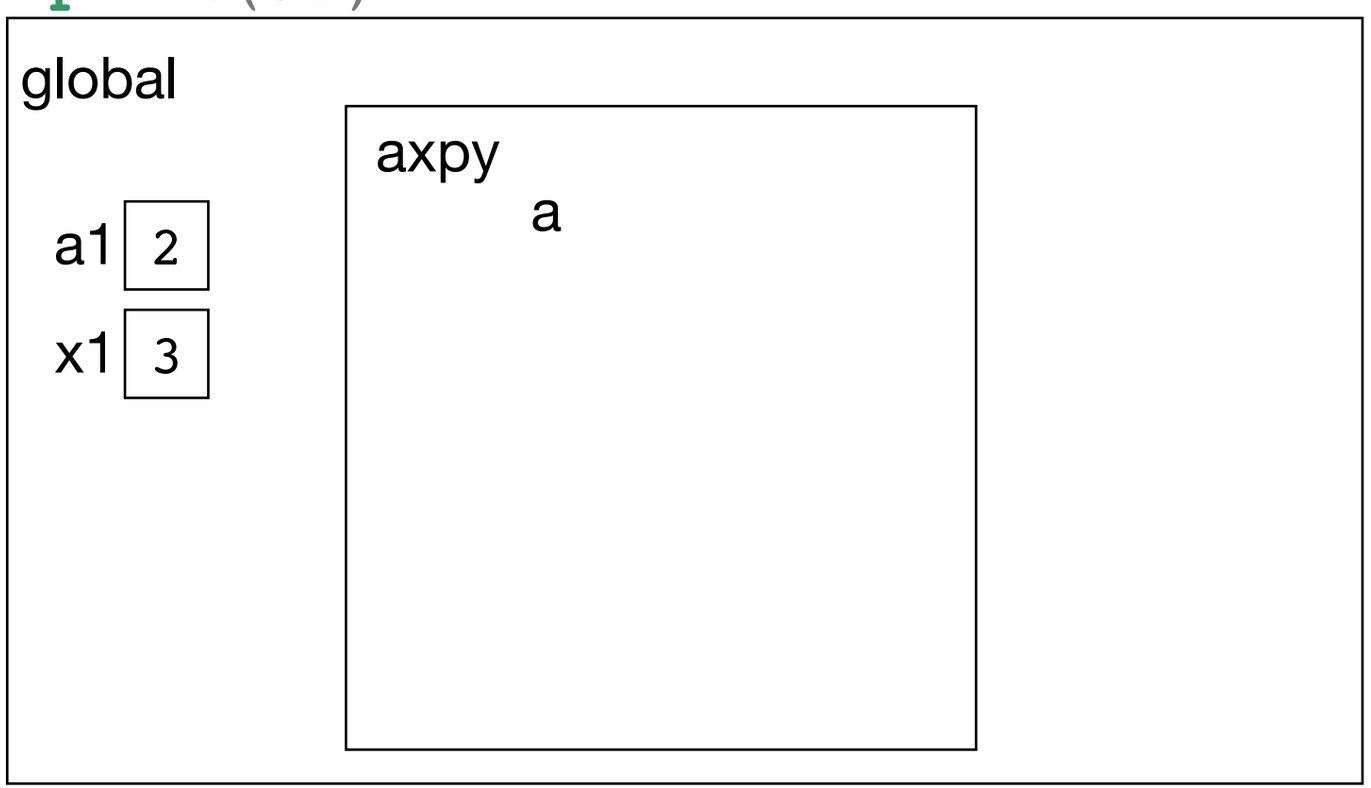
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



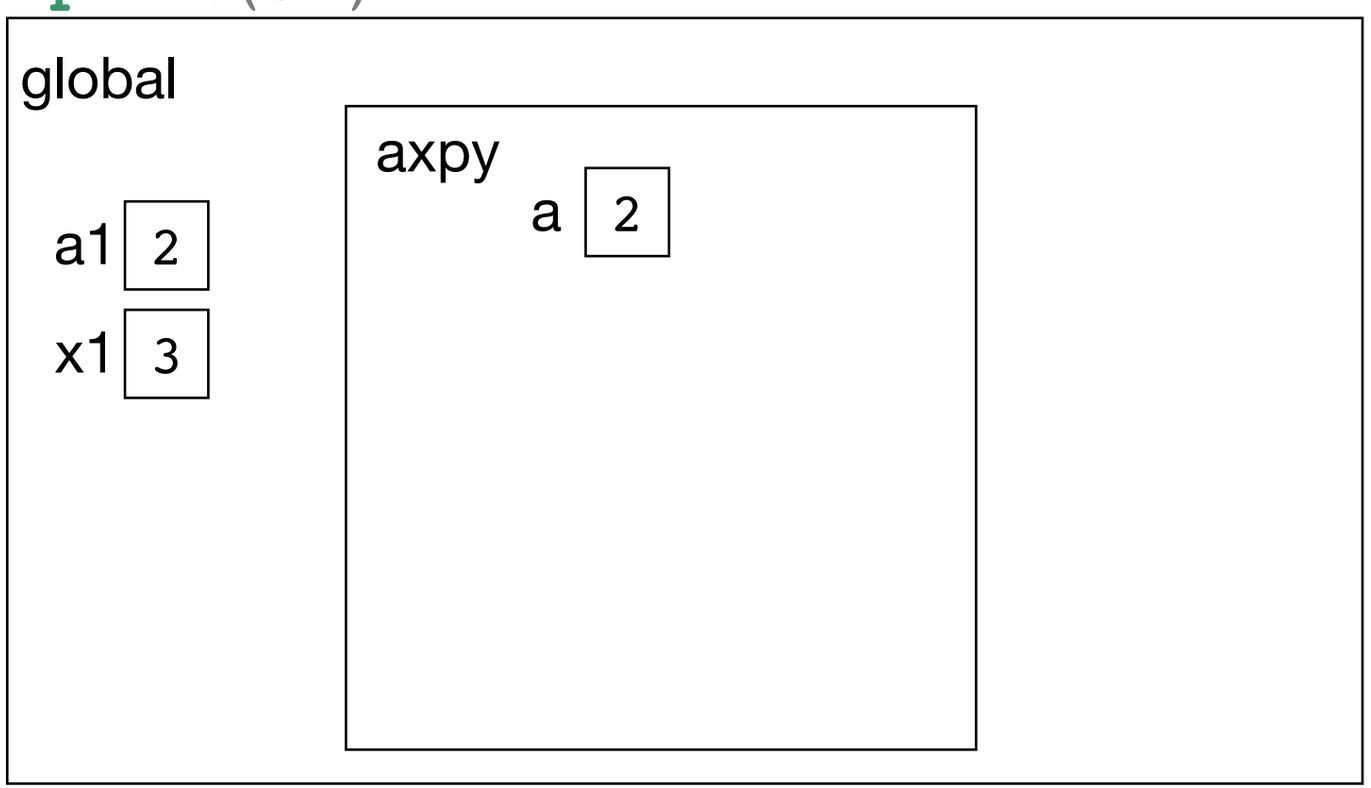
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



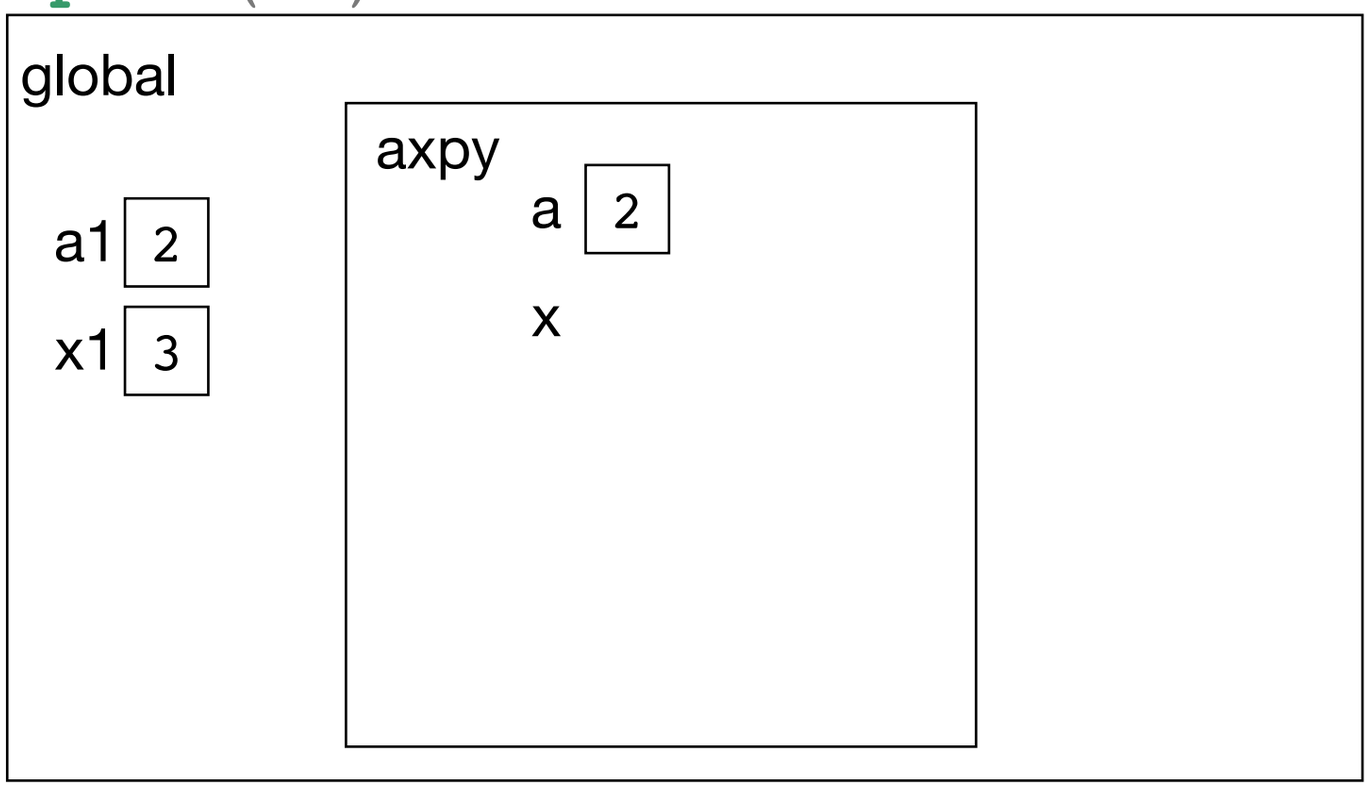
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



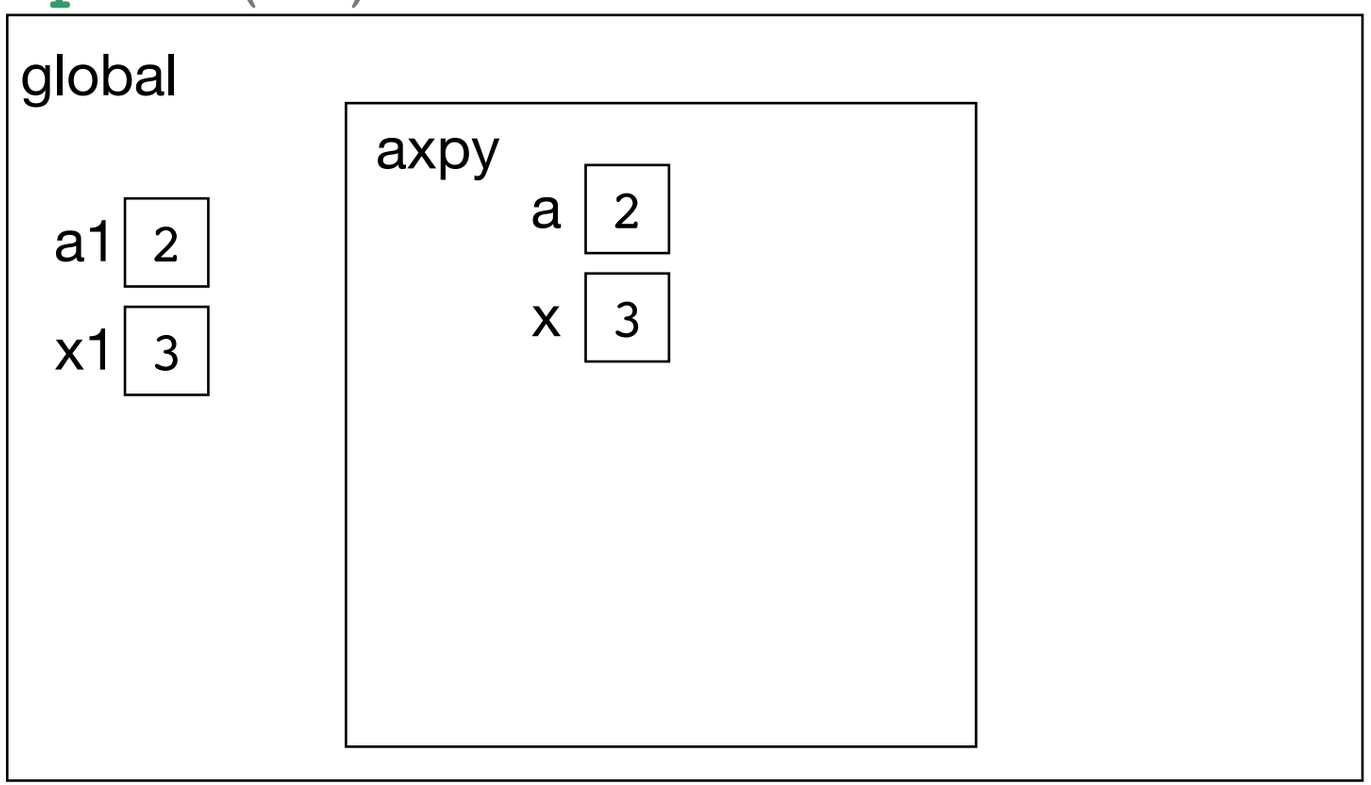
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



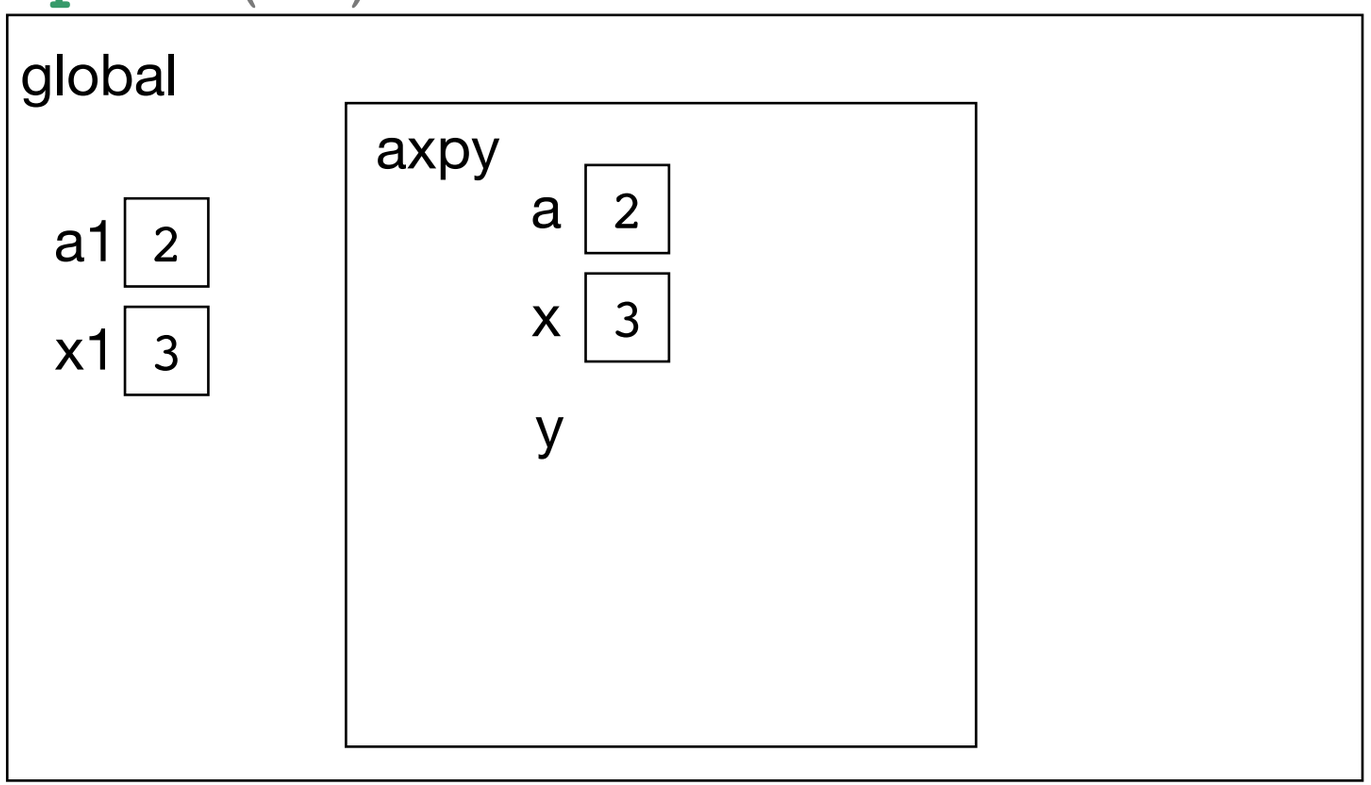
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



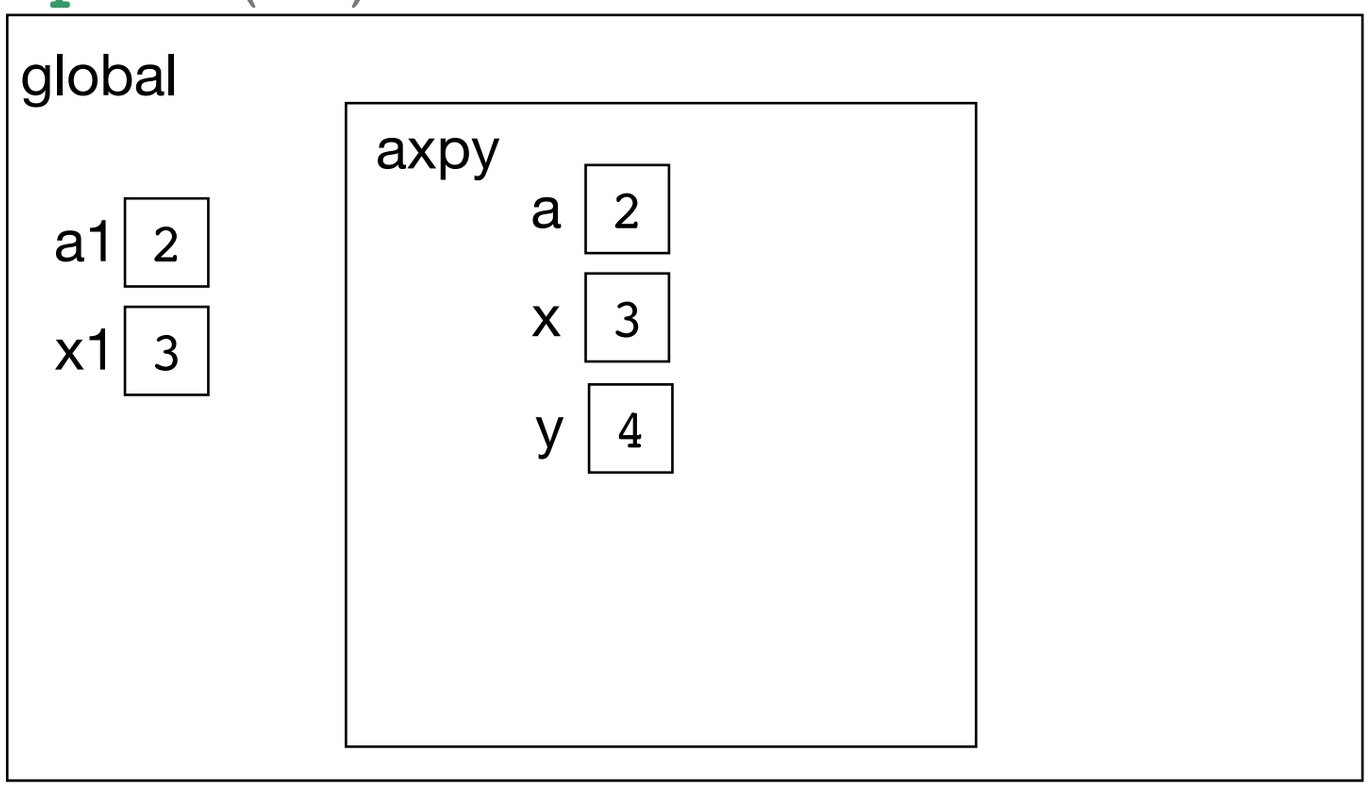
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



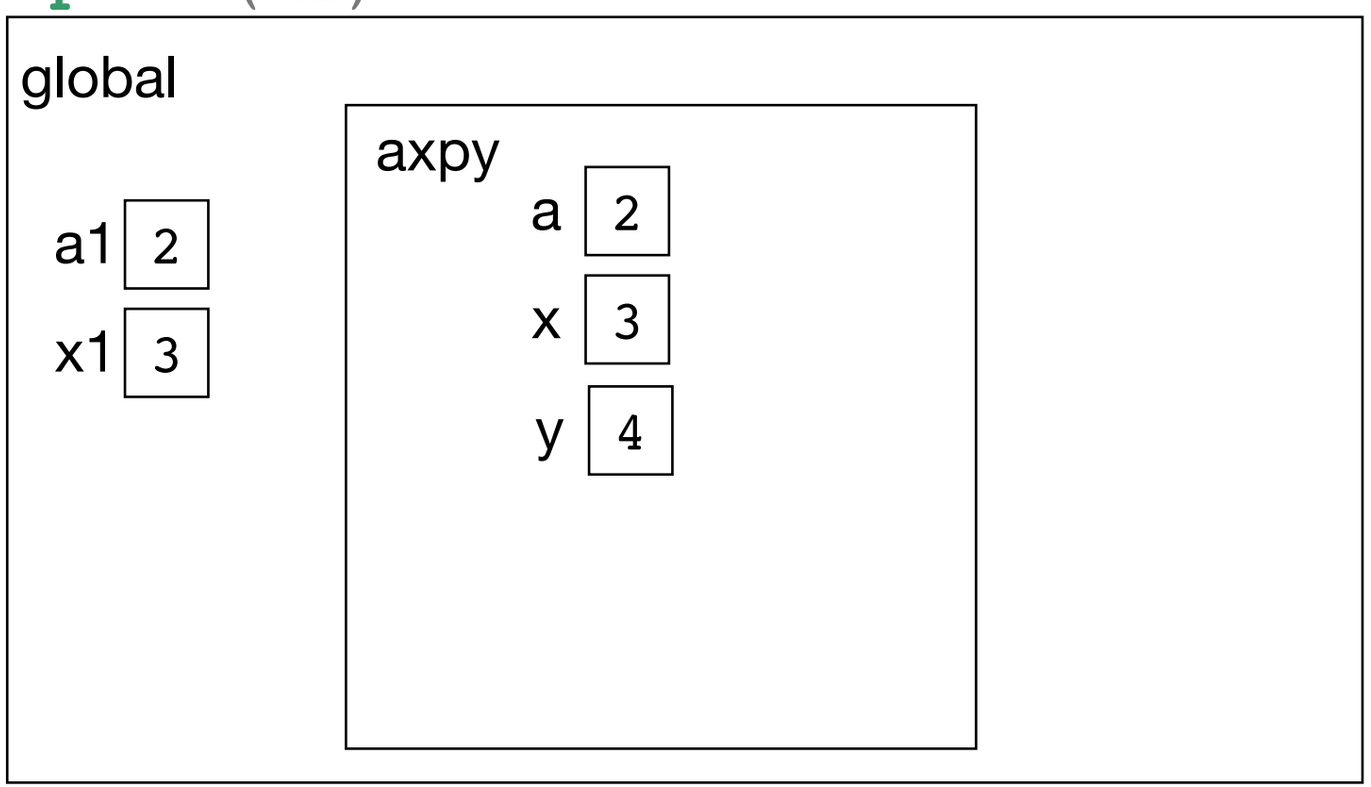
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



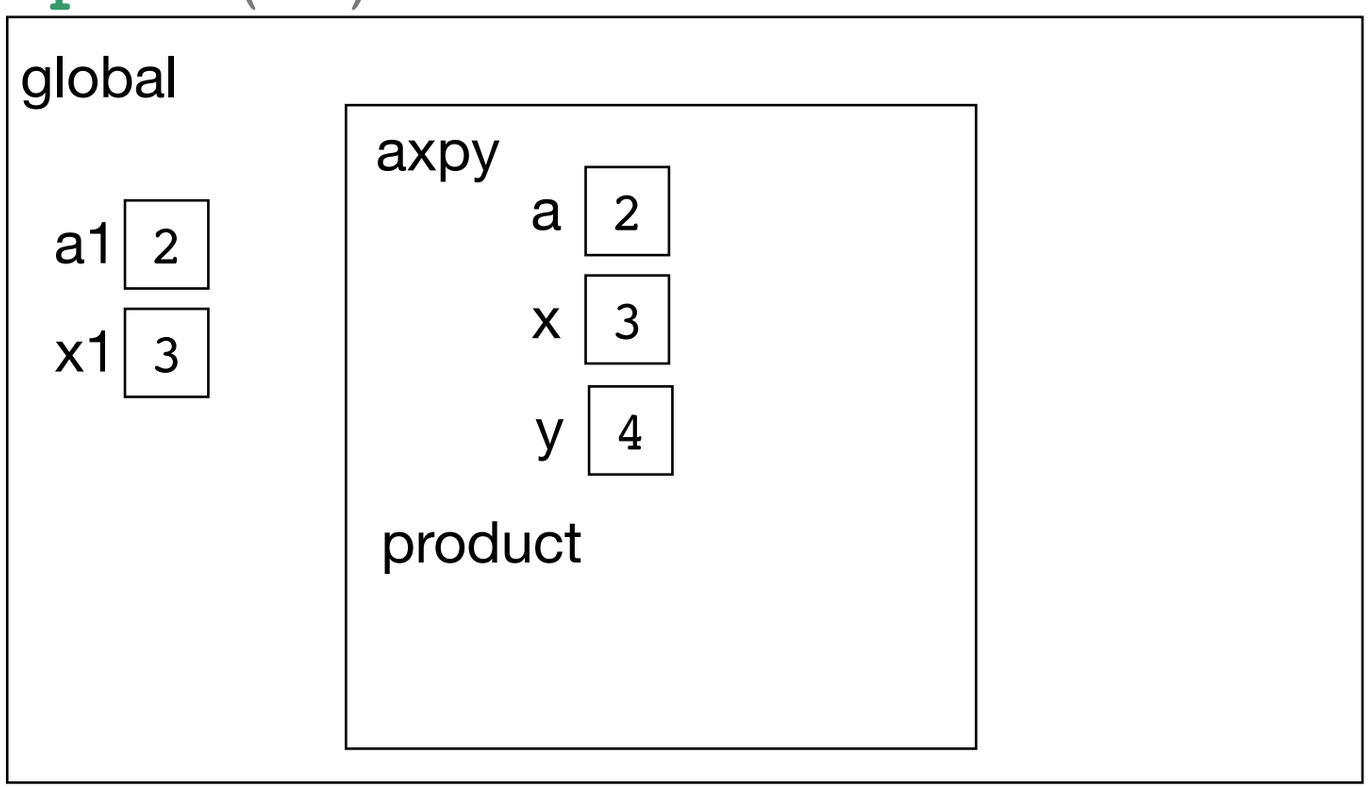
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



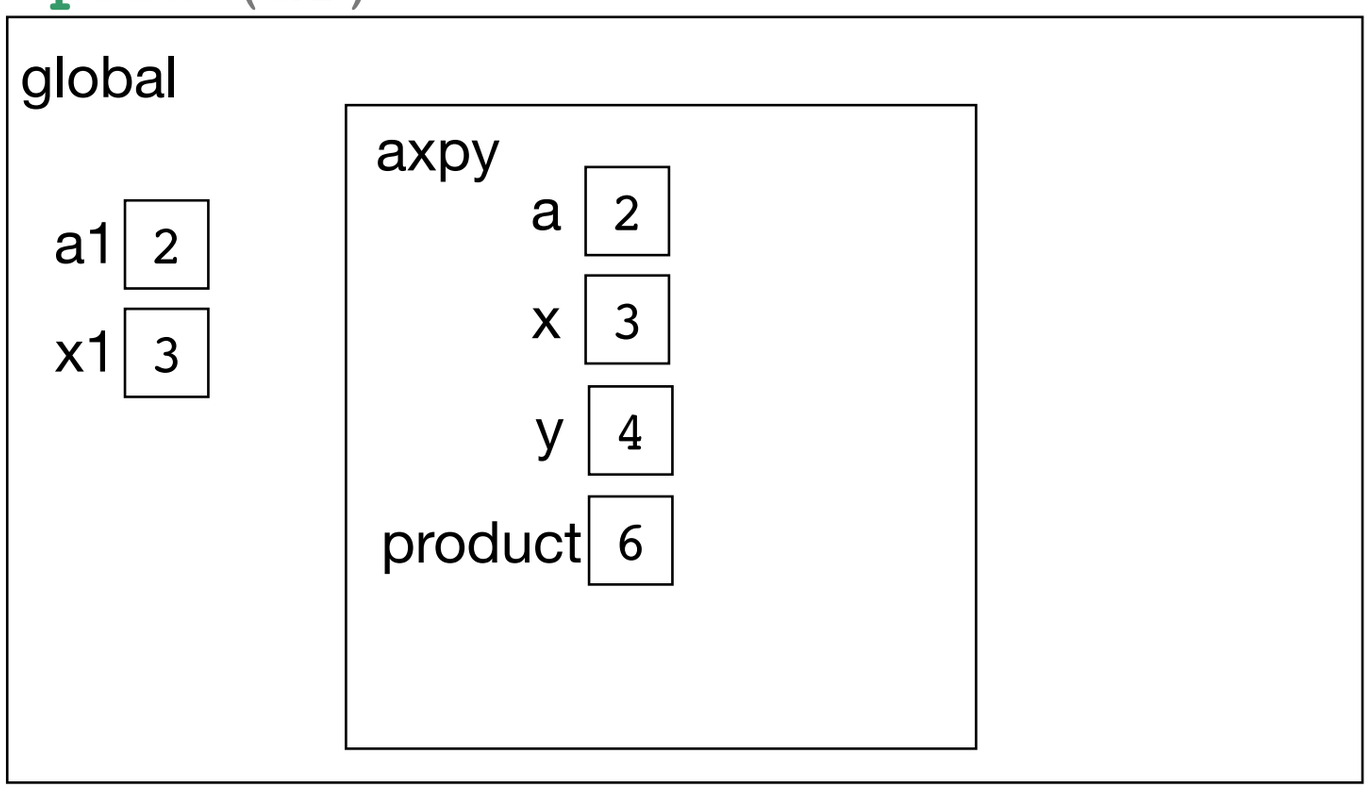
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



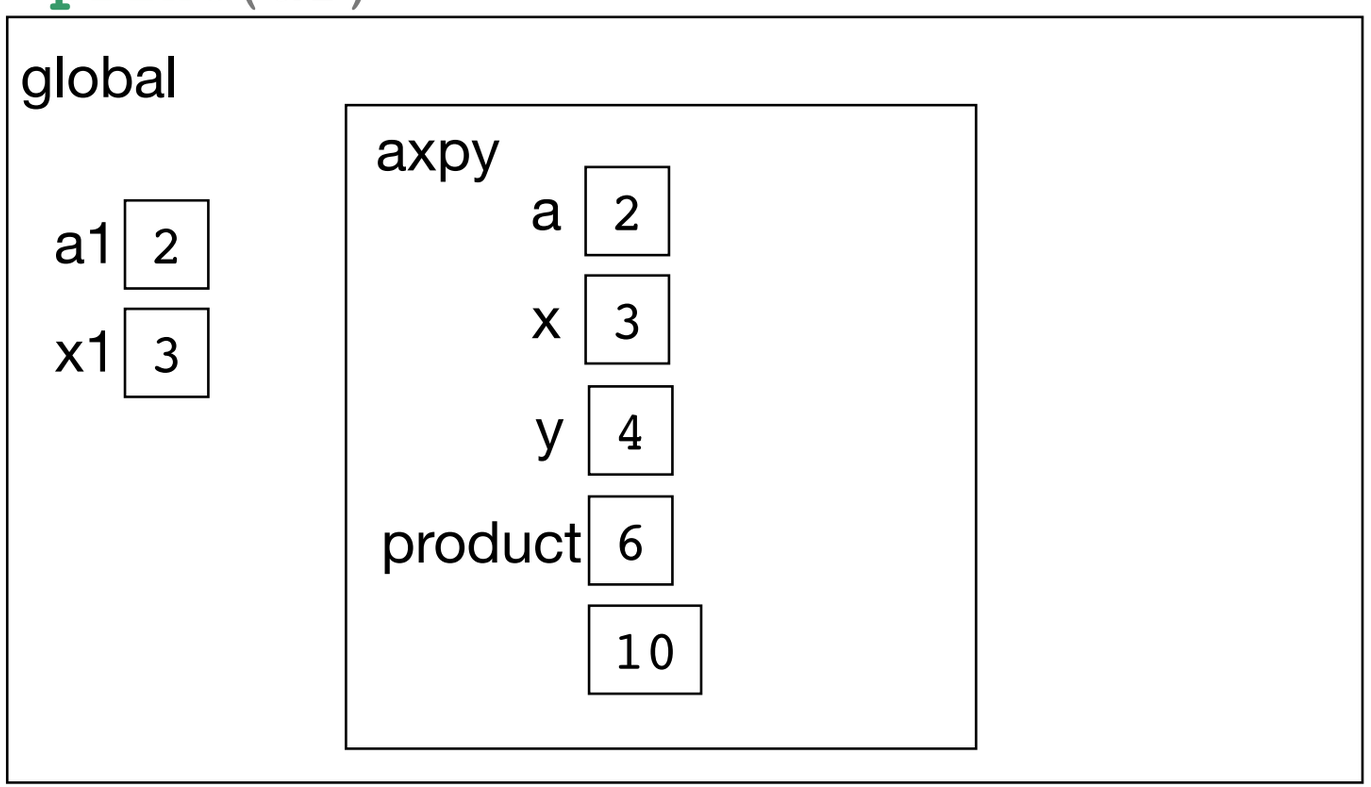
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



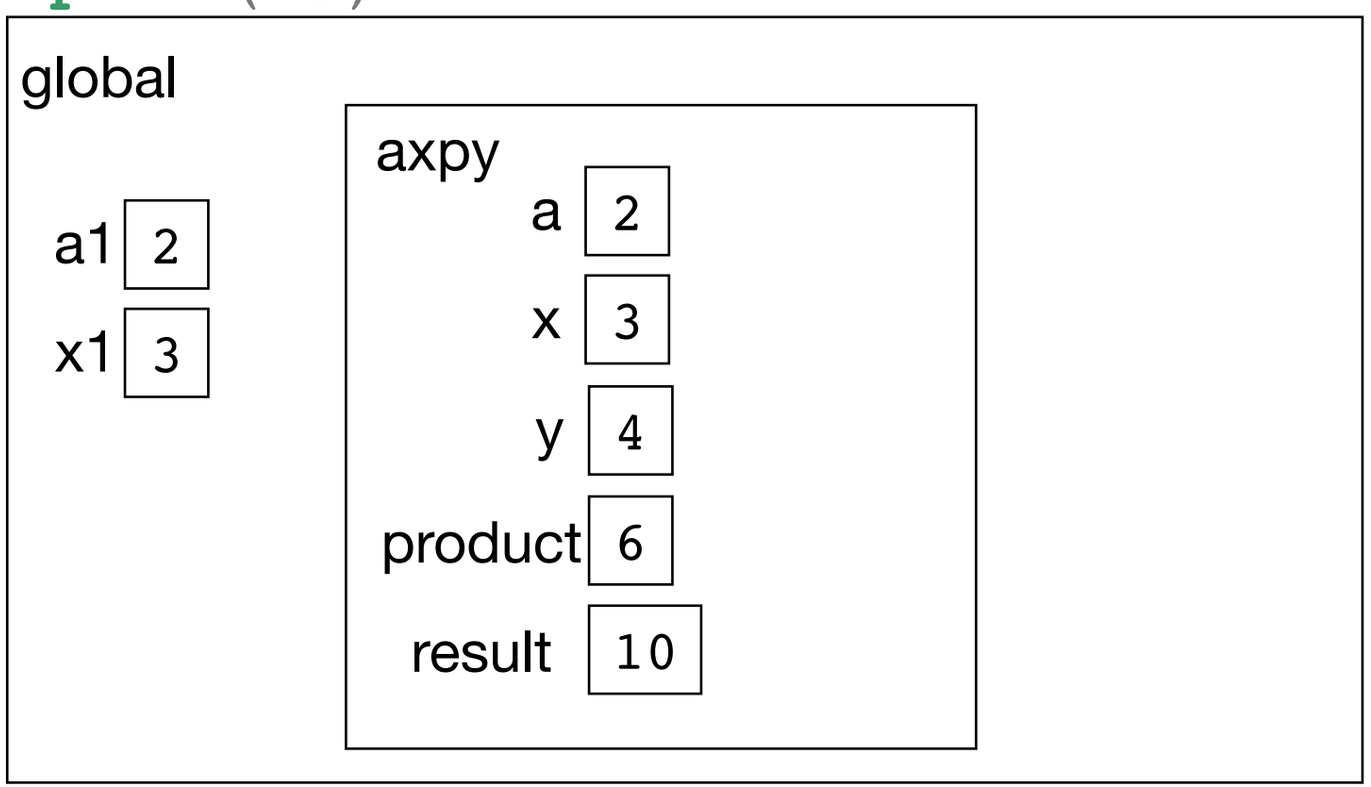
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



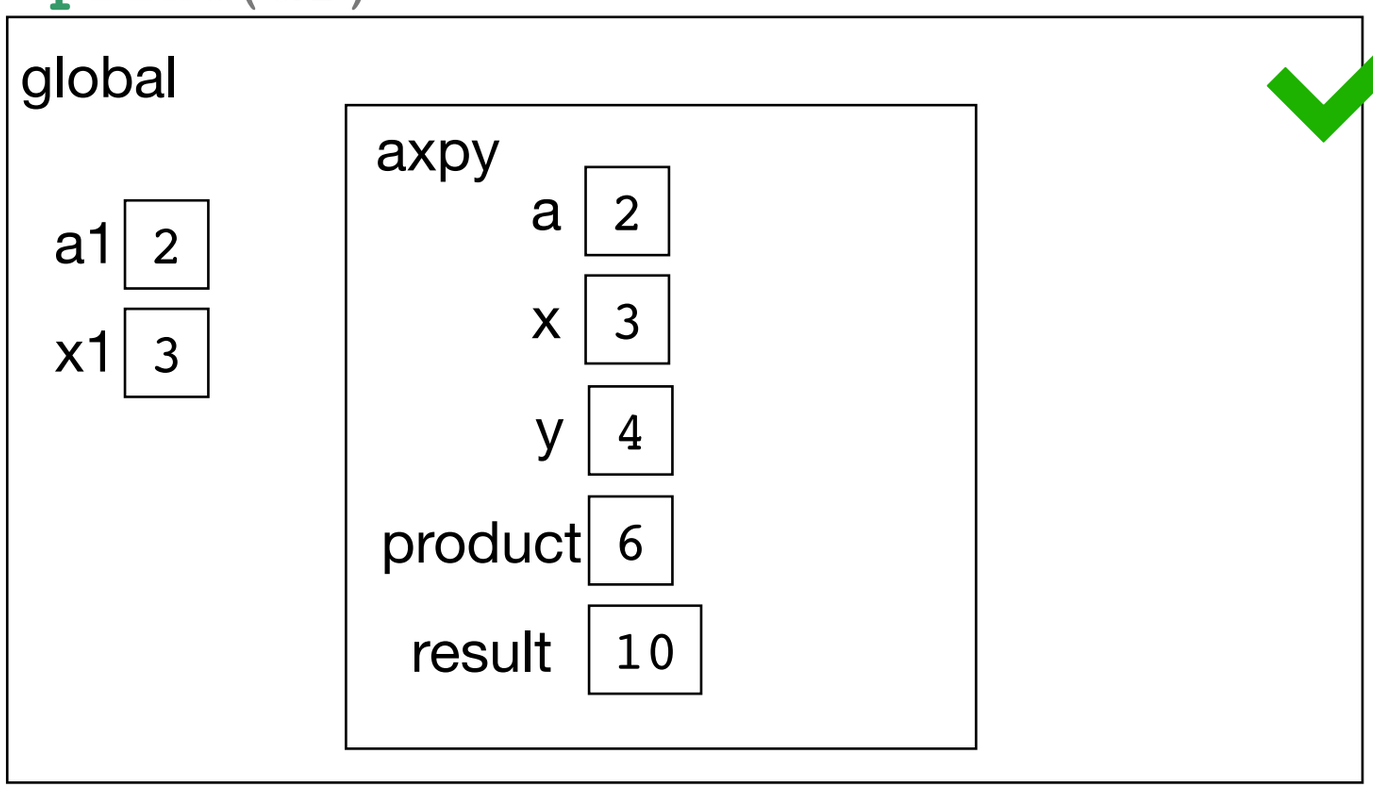
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, 3, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    product = a * x
    result = product + y
    return result

a1 = 2
x1 = 3
print(axpy(a1, 3, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

global

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

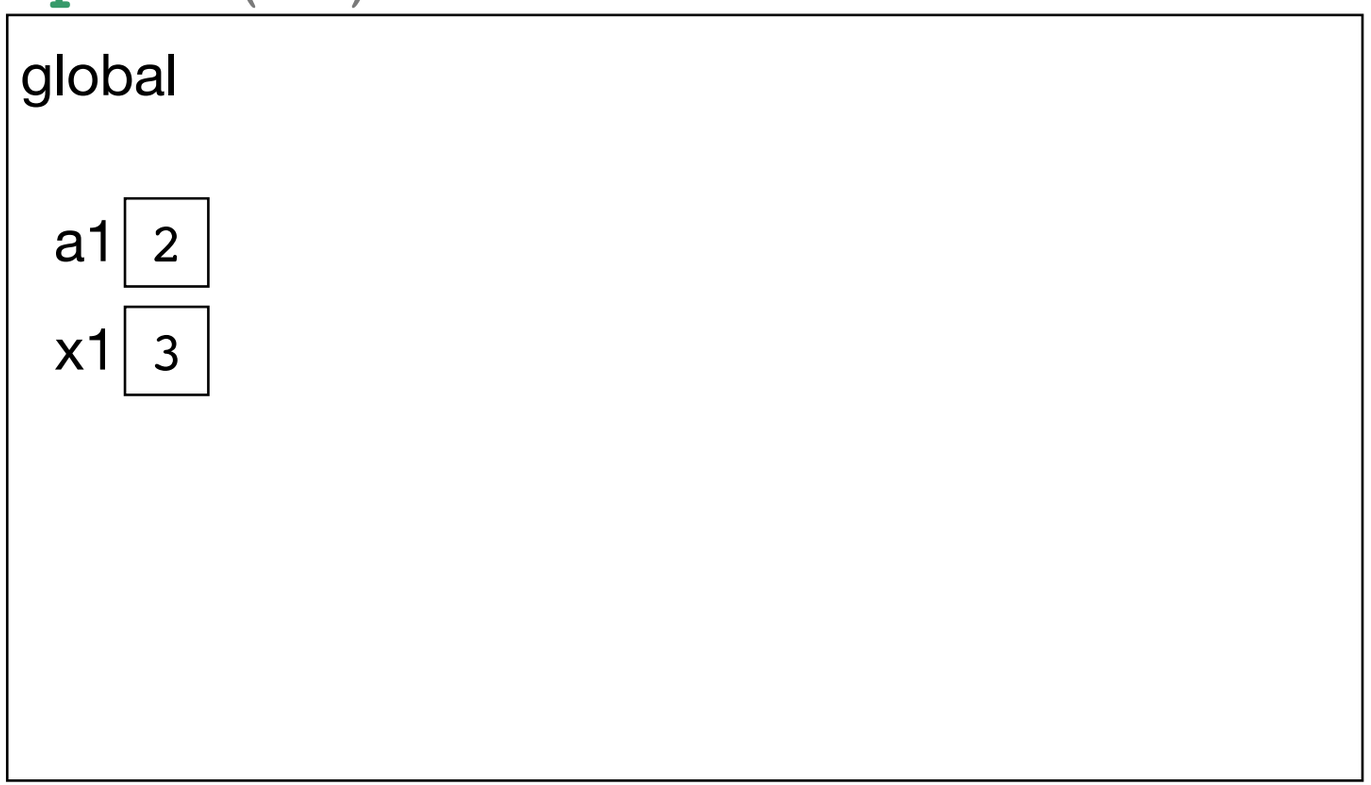
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```

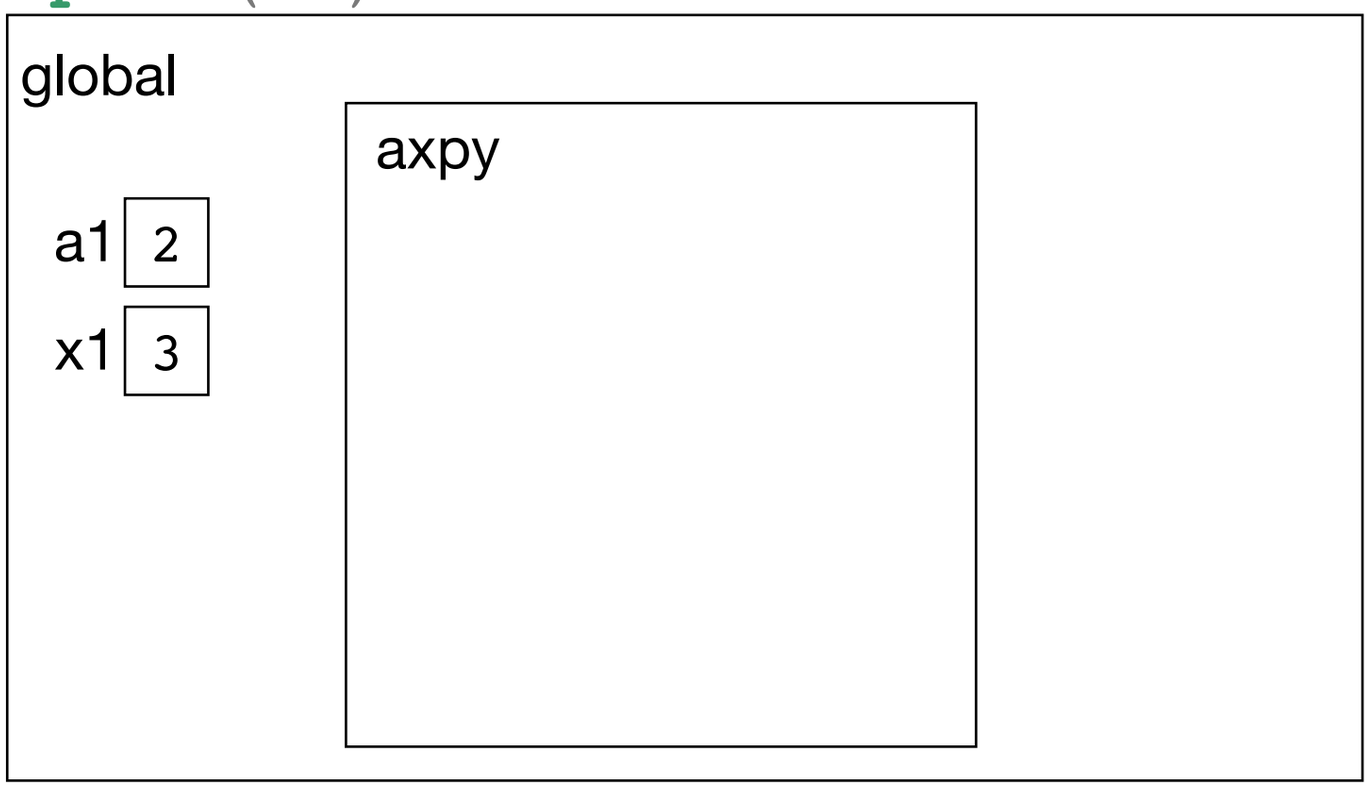
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



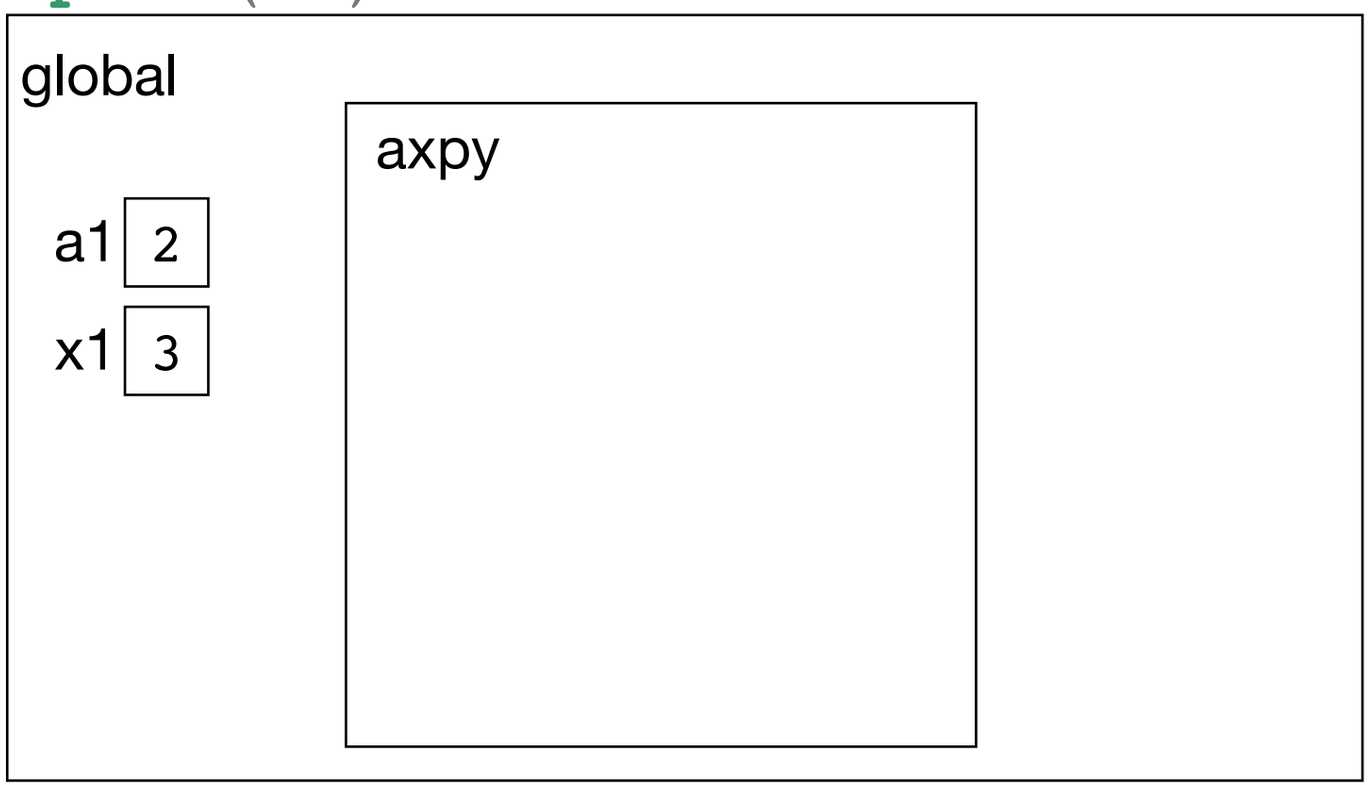
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



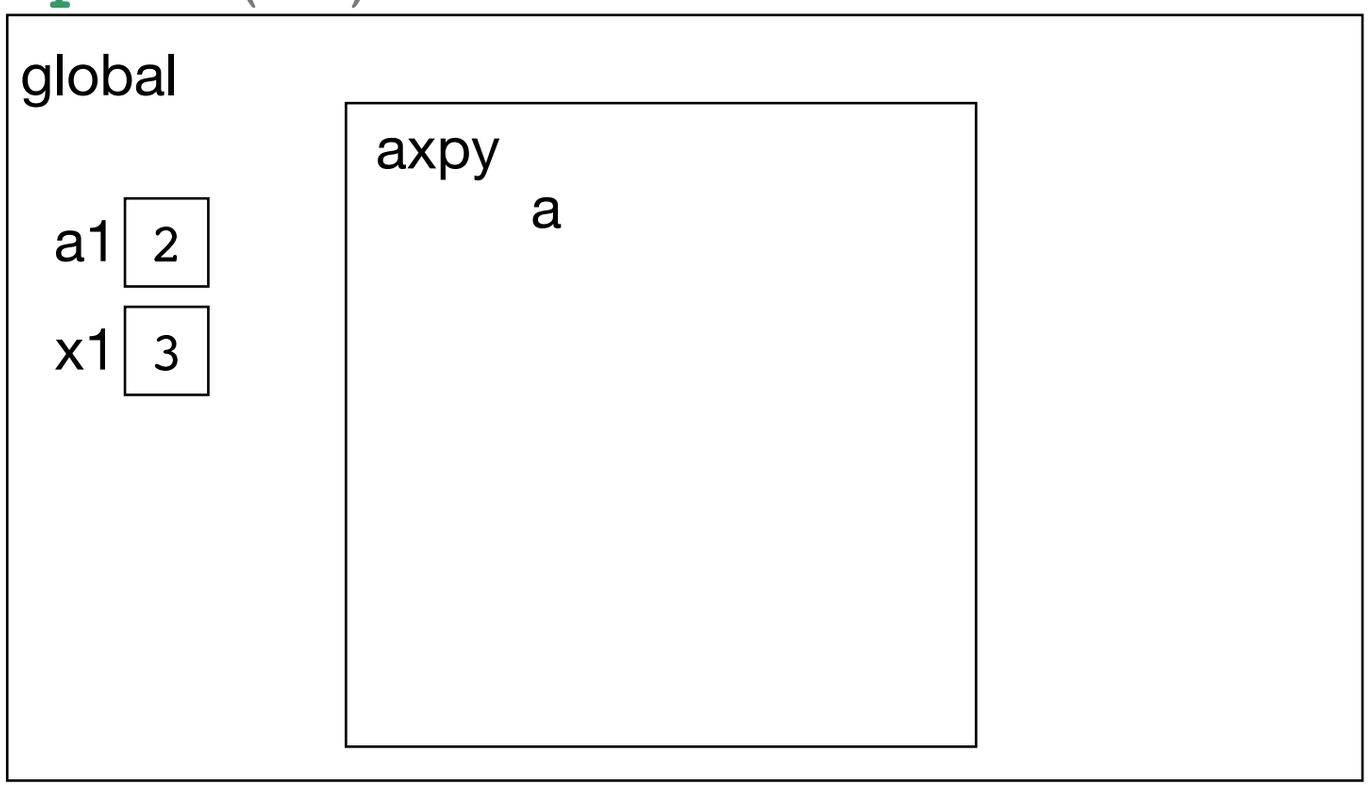
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



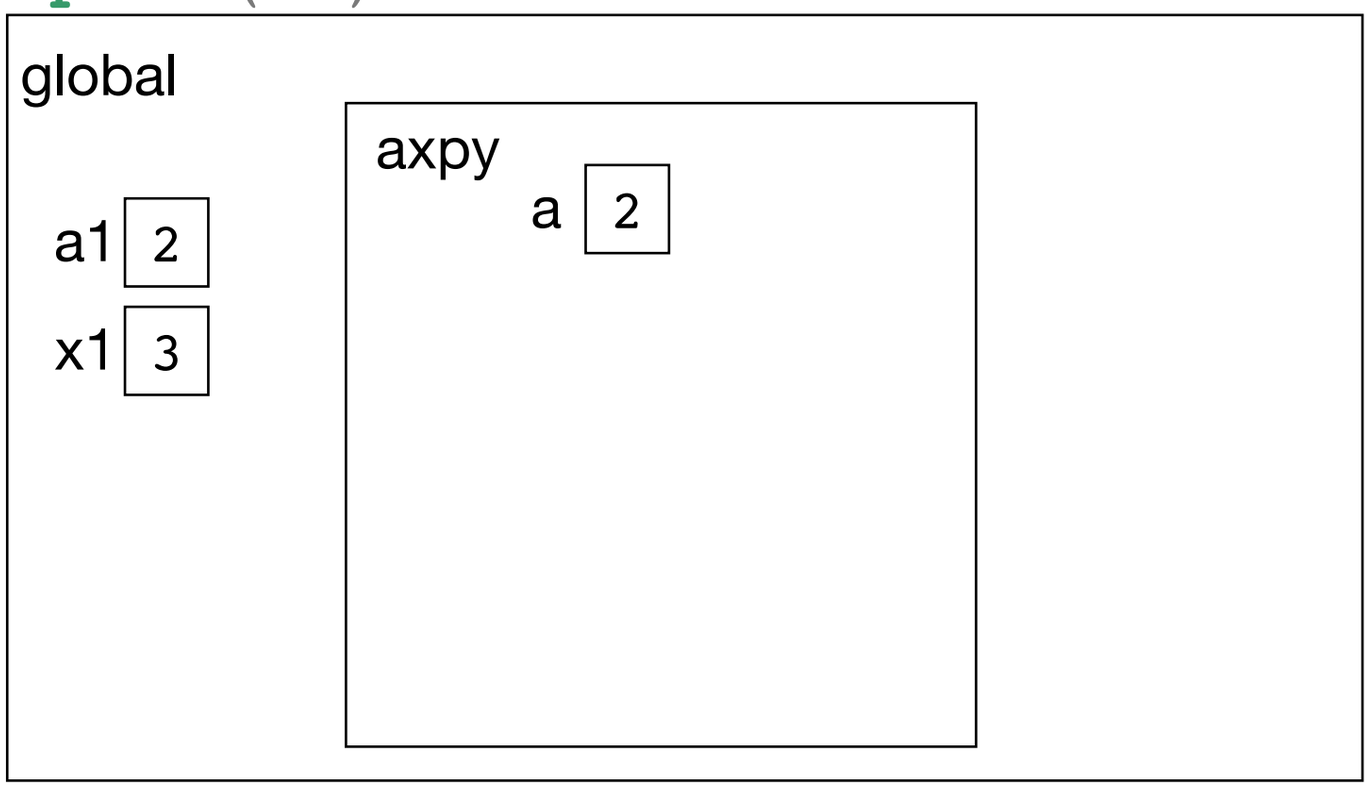
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



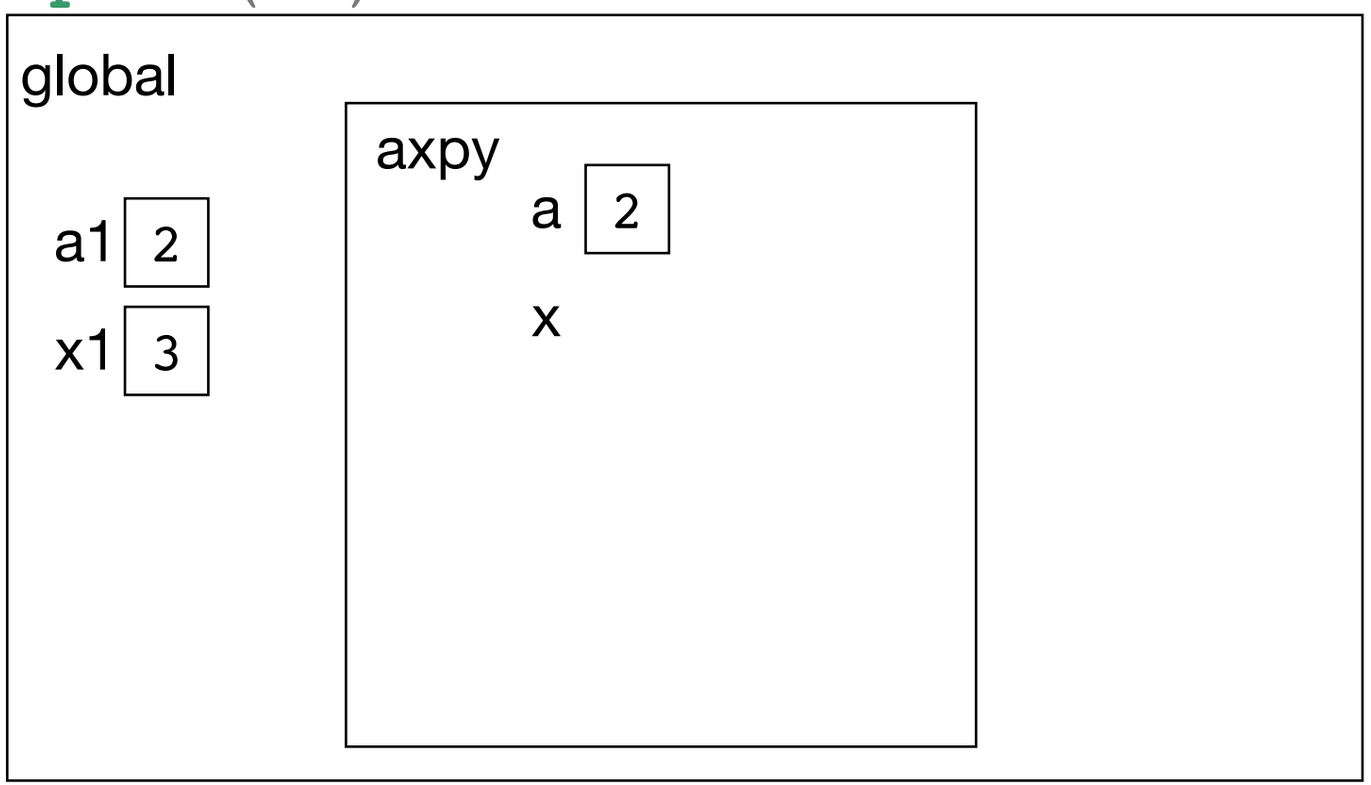
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



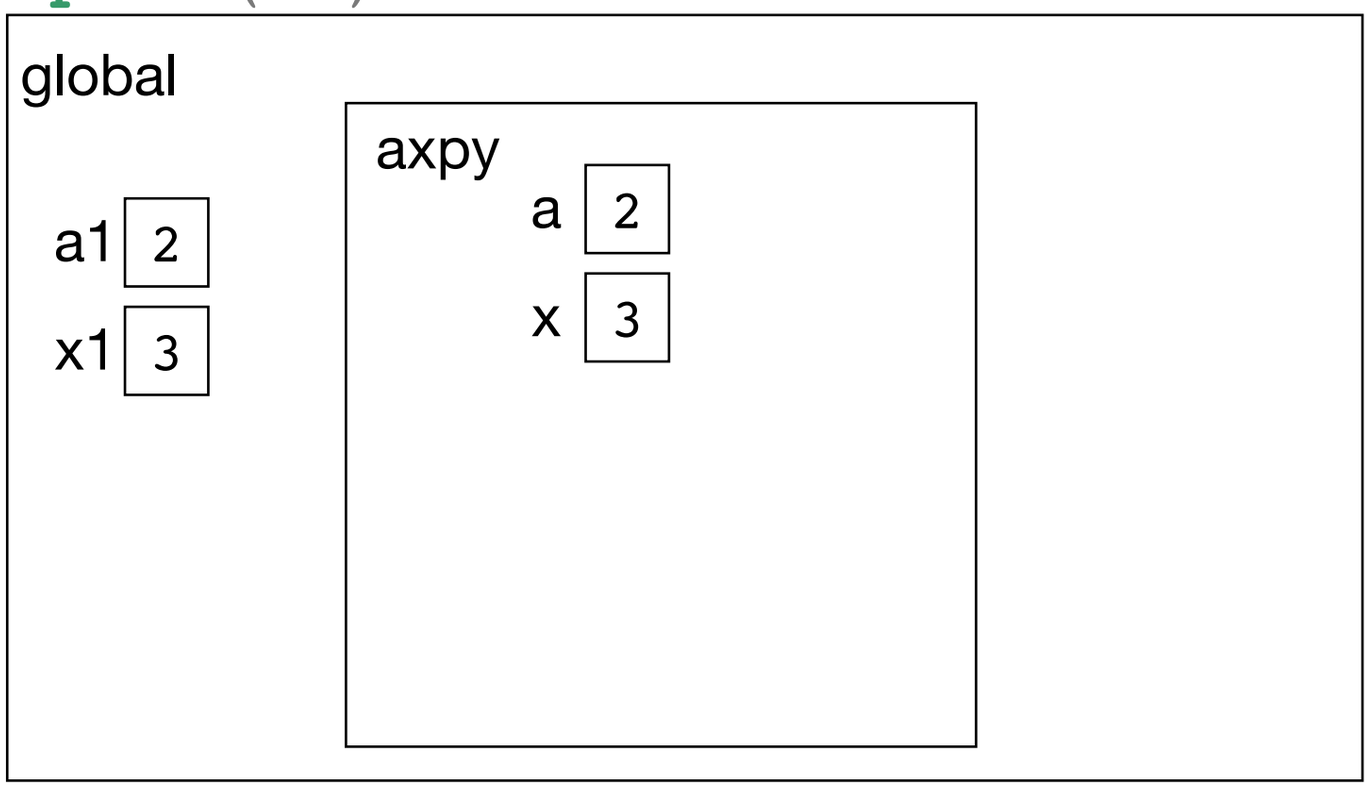
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



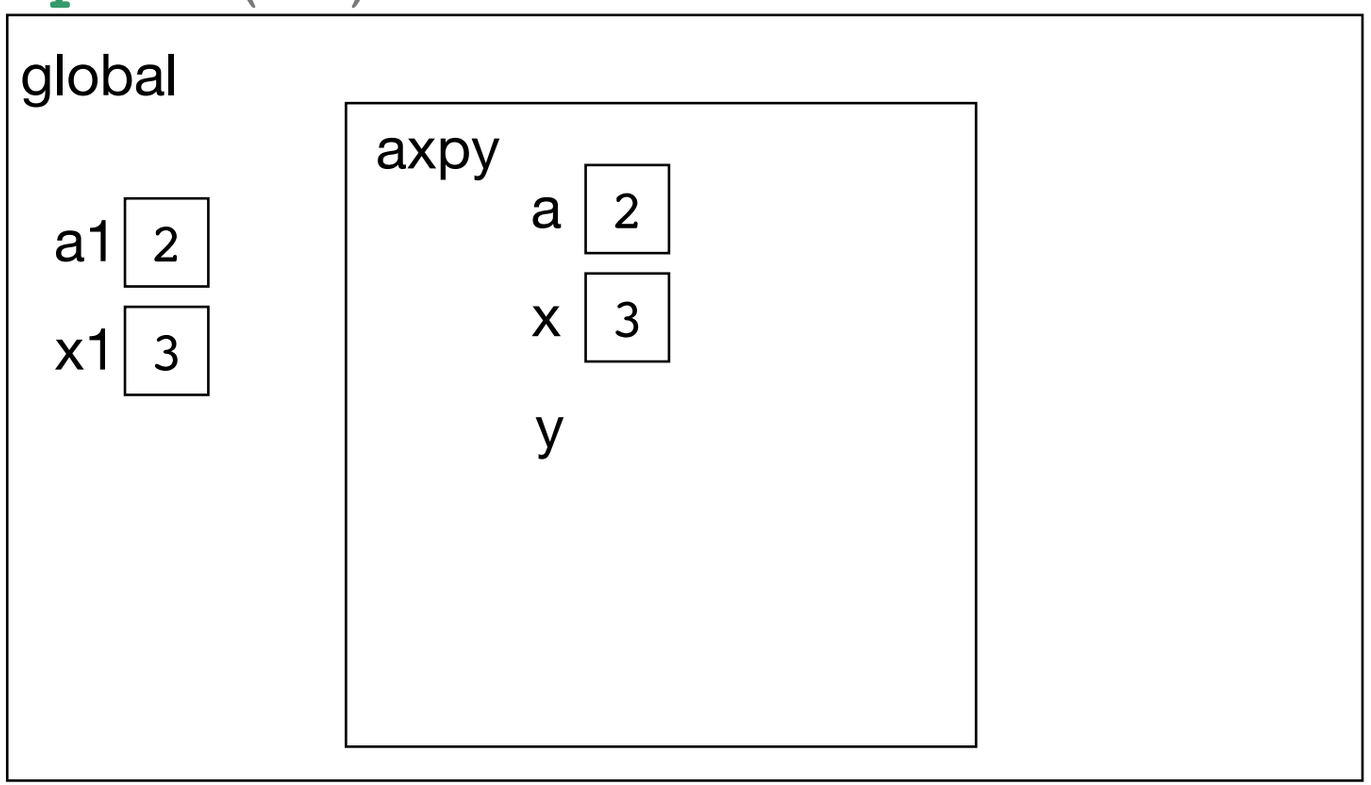
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



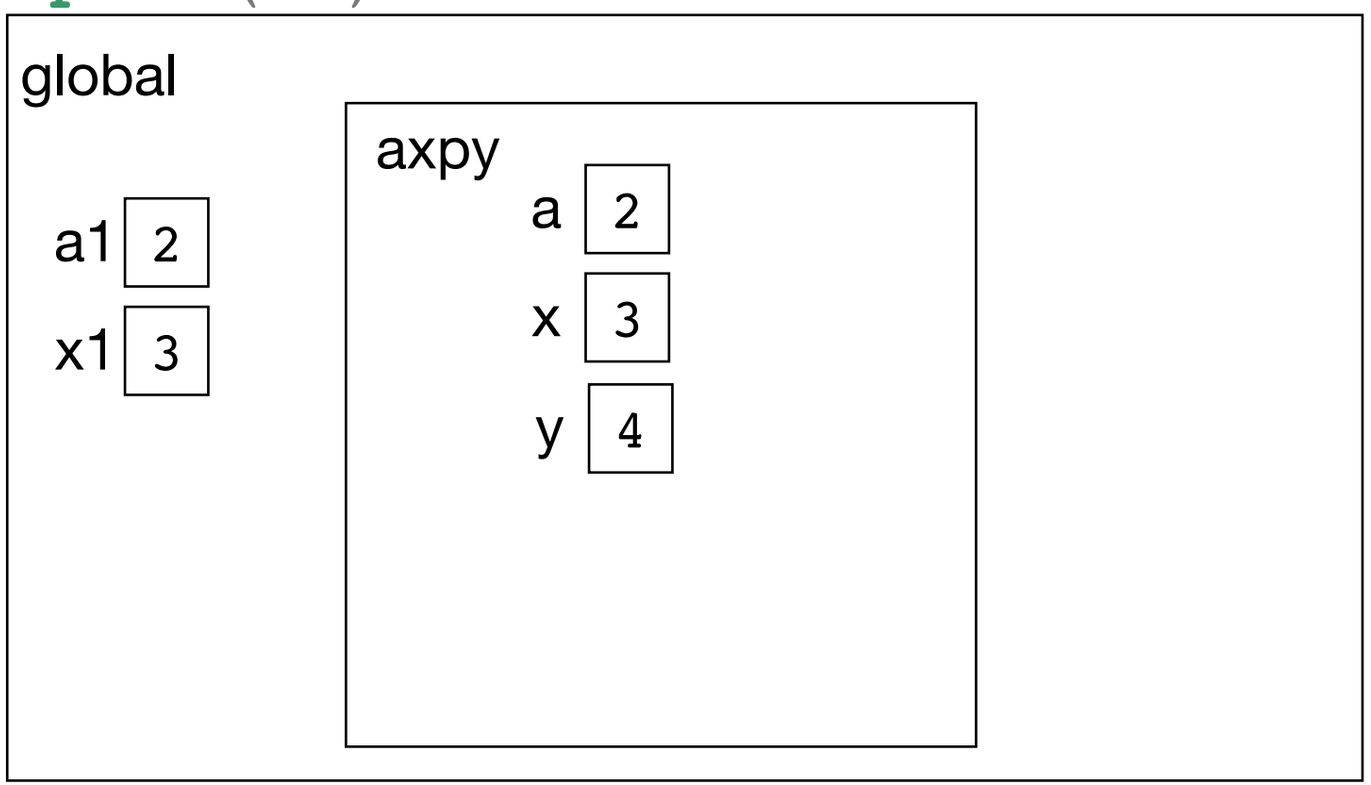
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



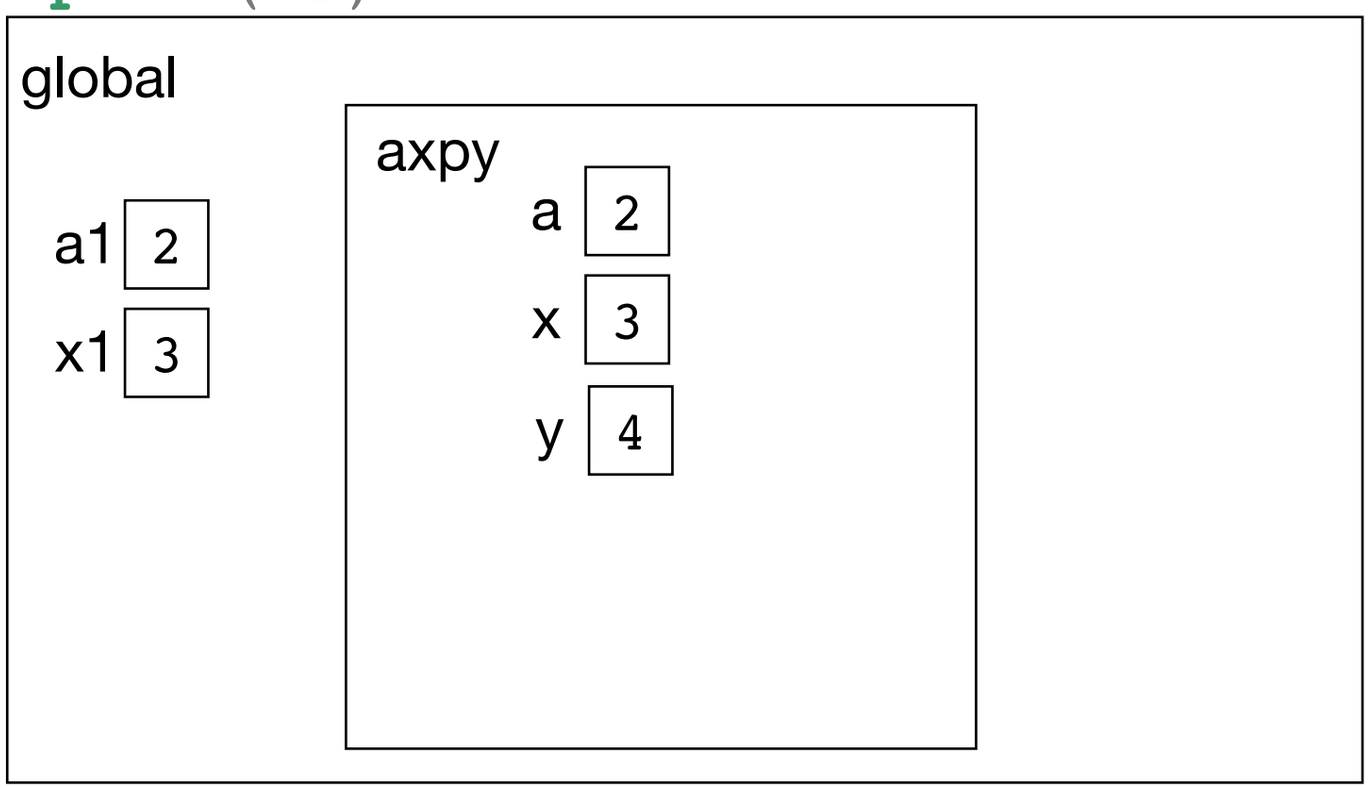
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



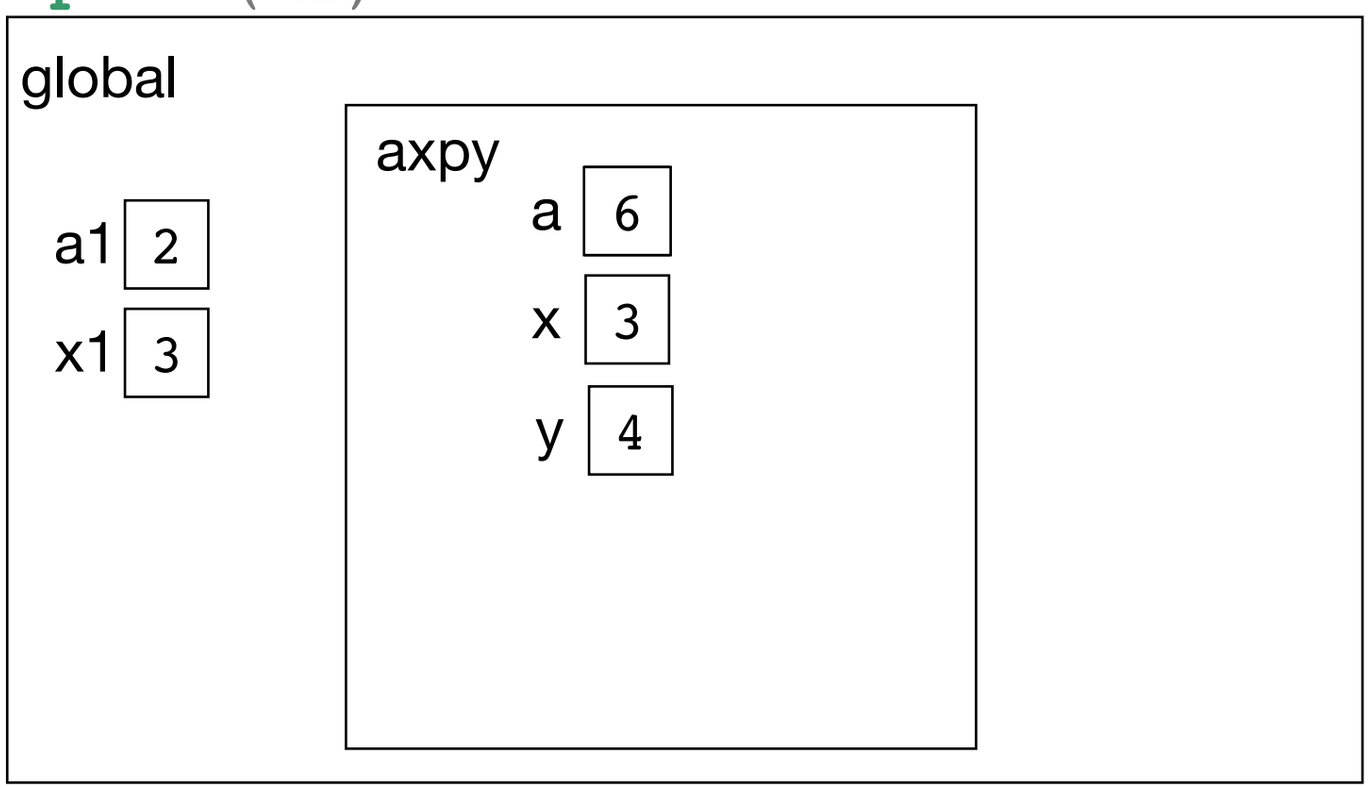
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



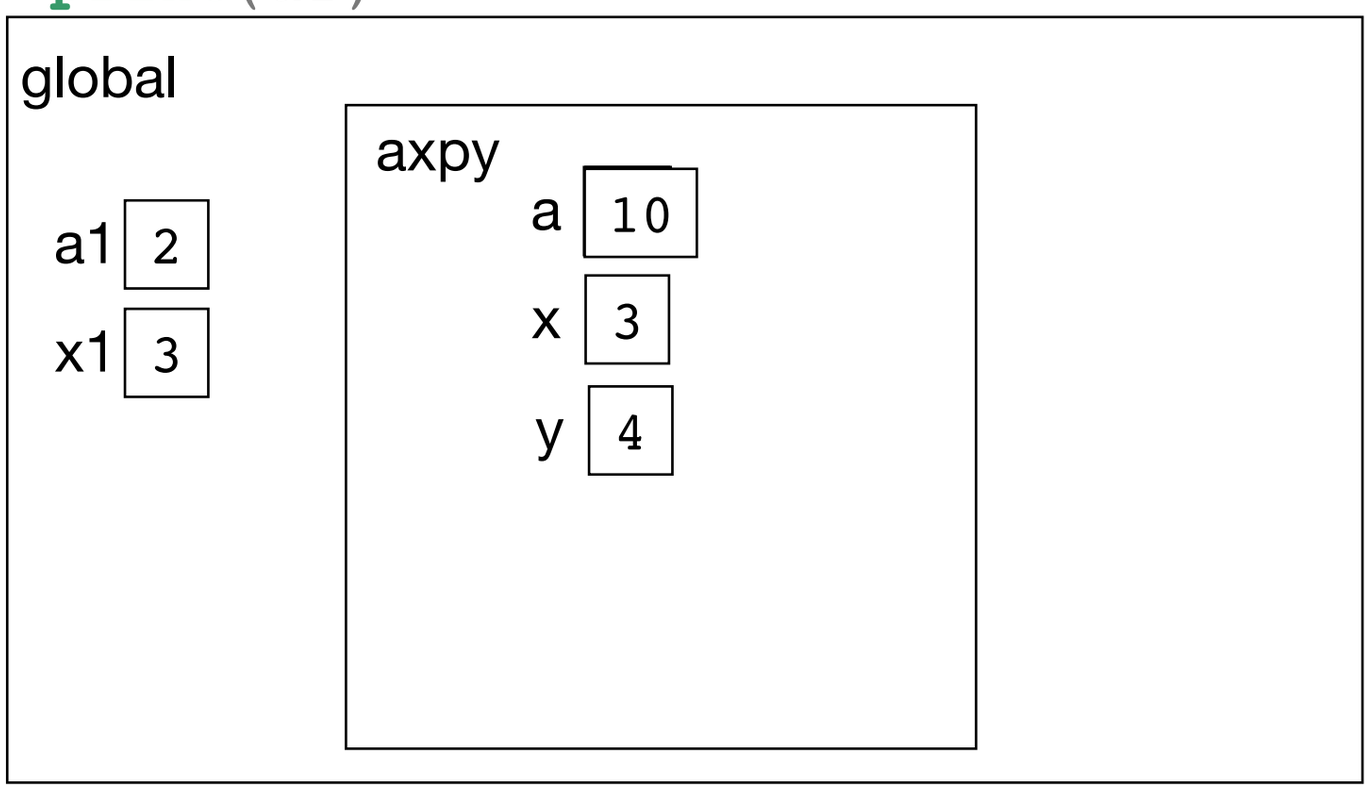
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



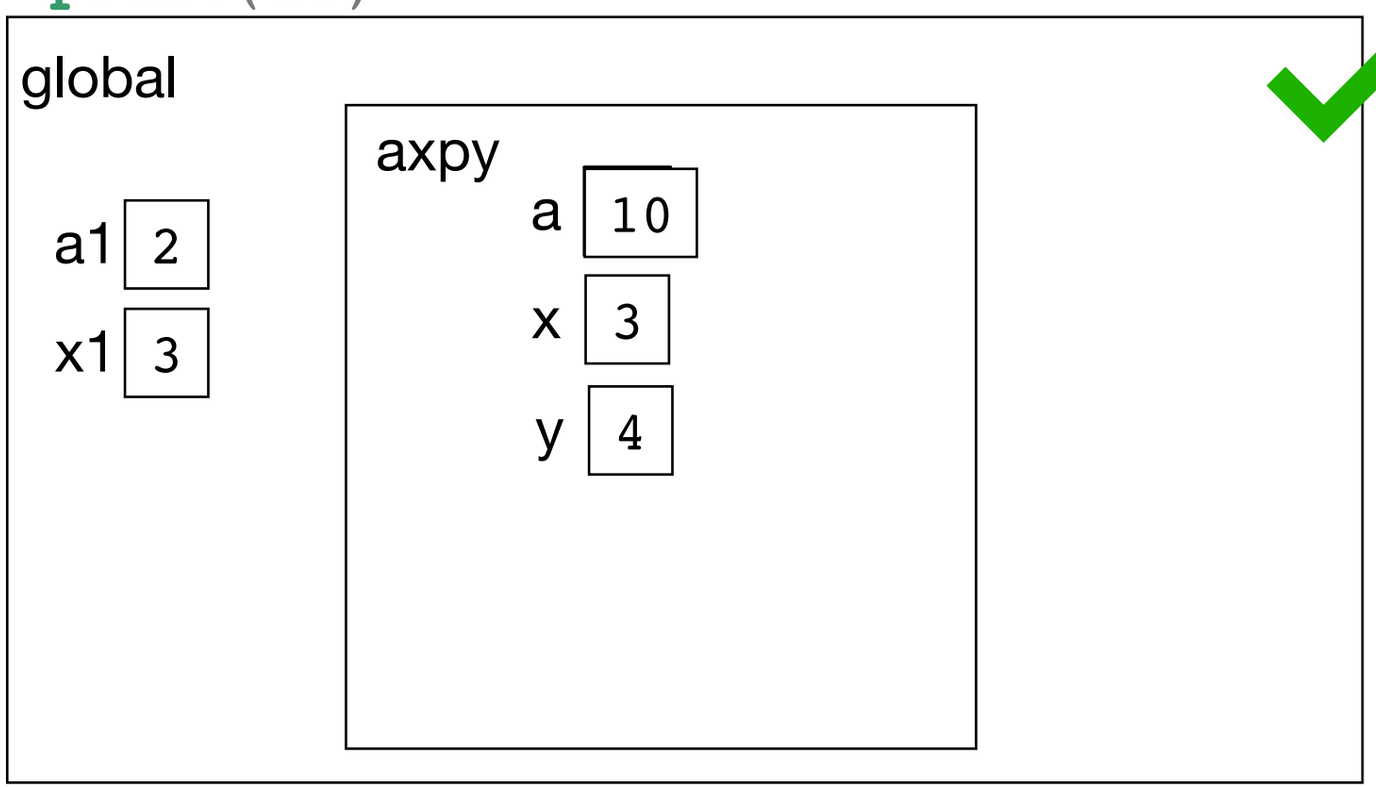
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



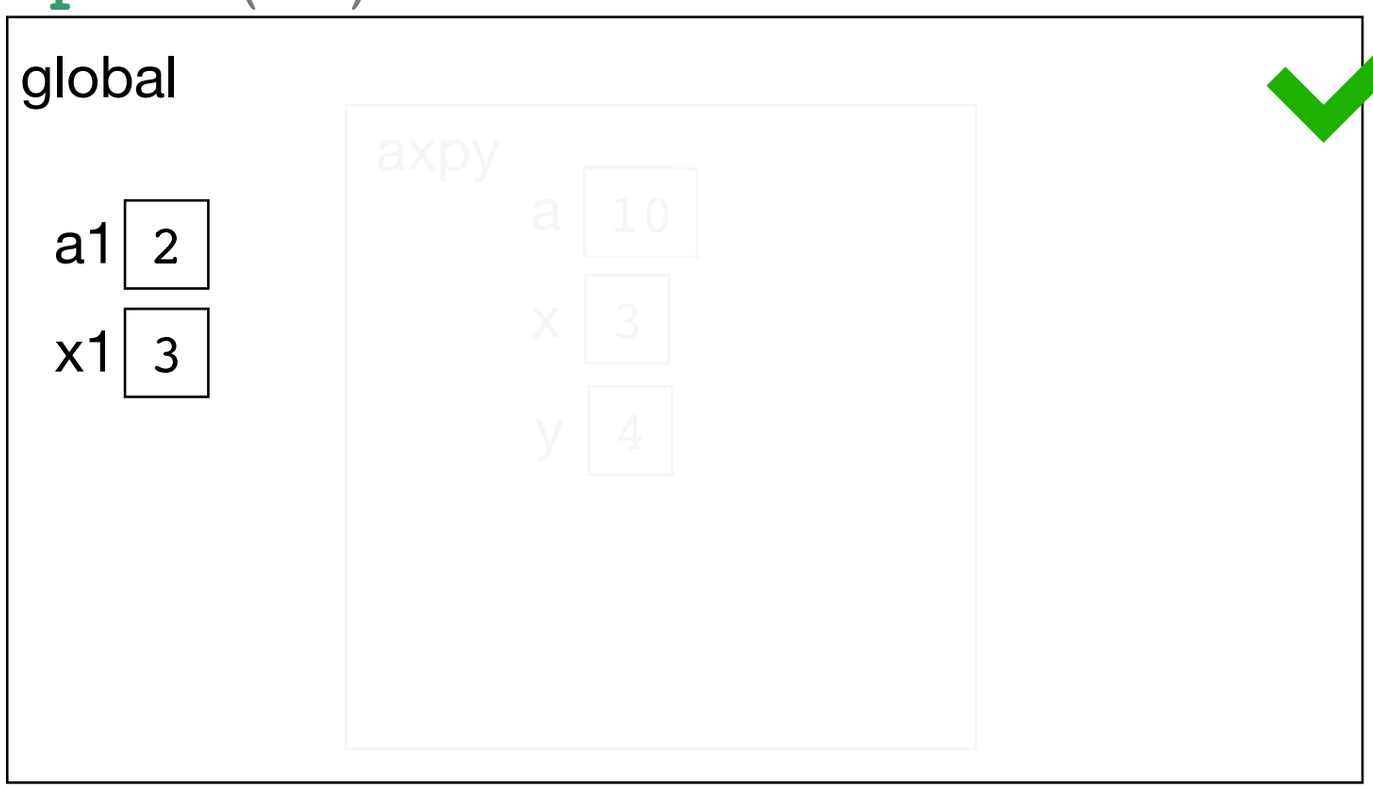
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



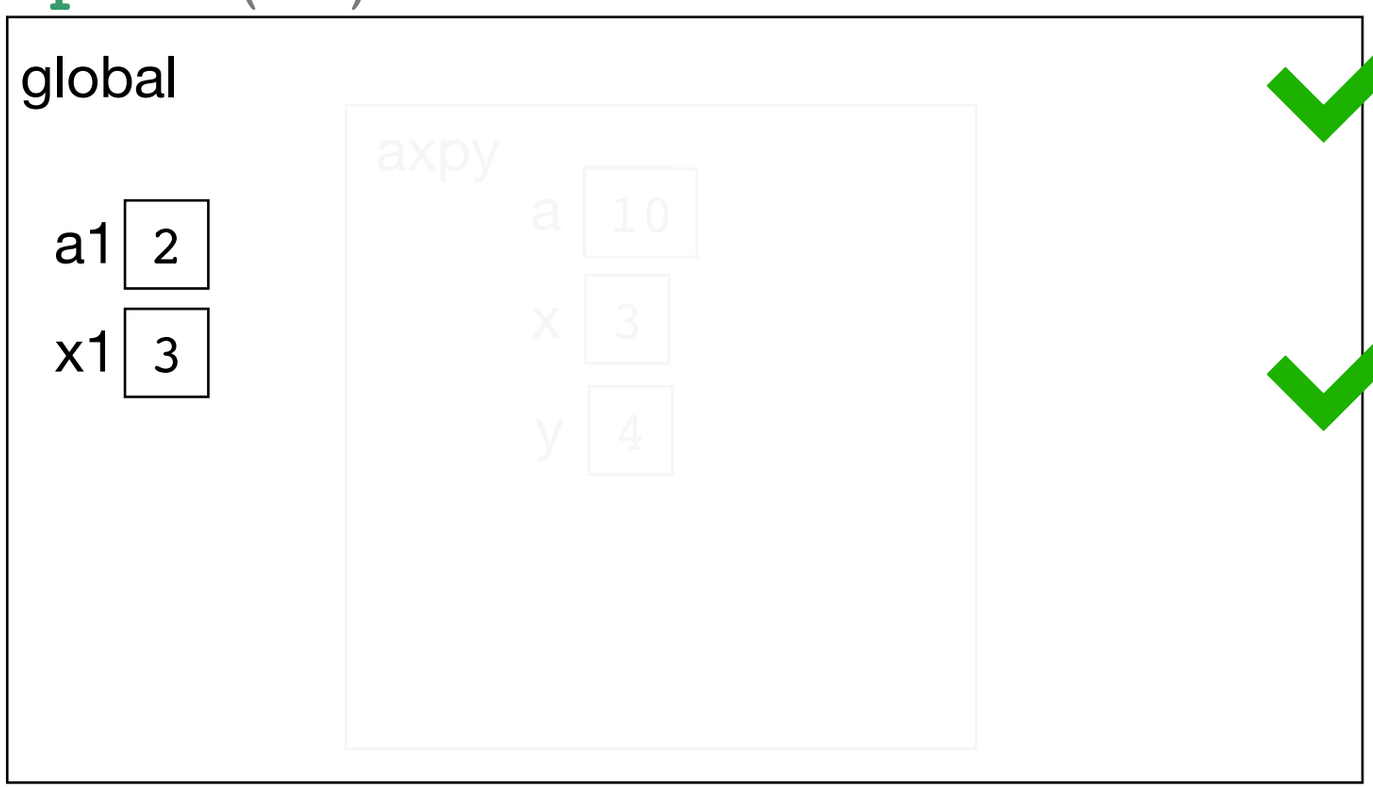
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



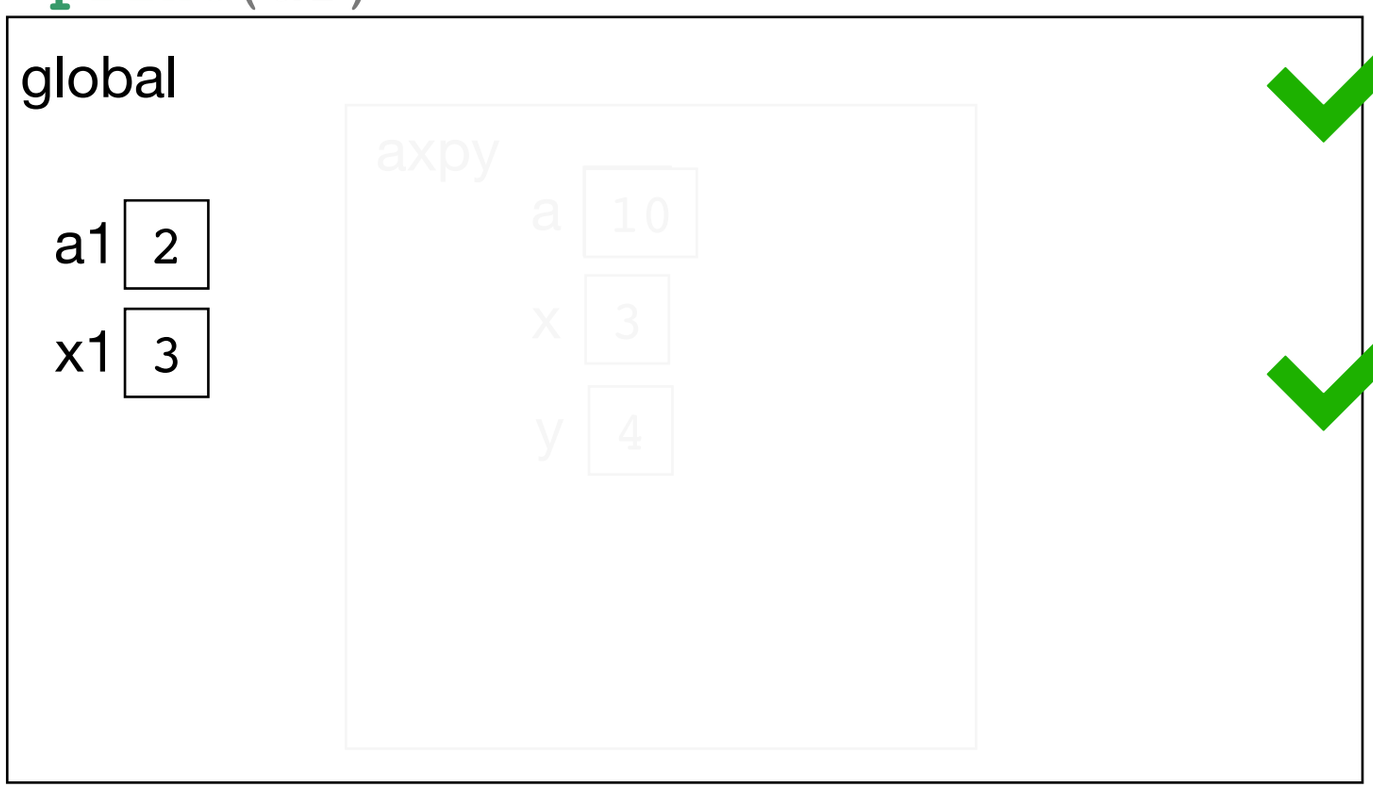
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, 3, 4))
print(a1)
```



1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a = a * x
    a += y
    return a

a1 = 2
x1 = 3
print(axpy(a1, 3, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

global

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```

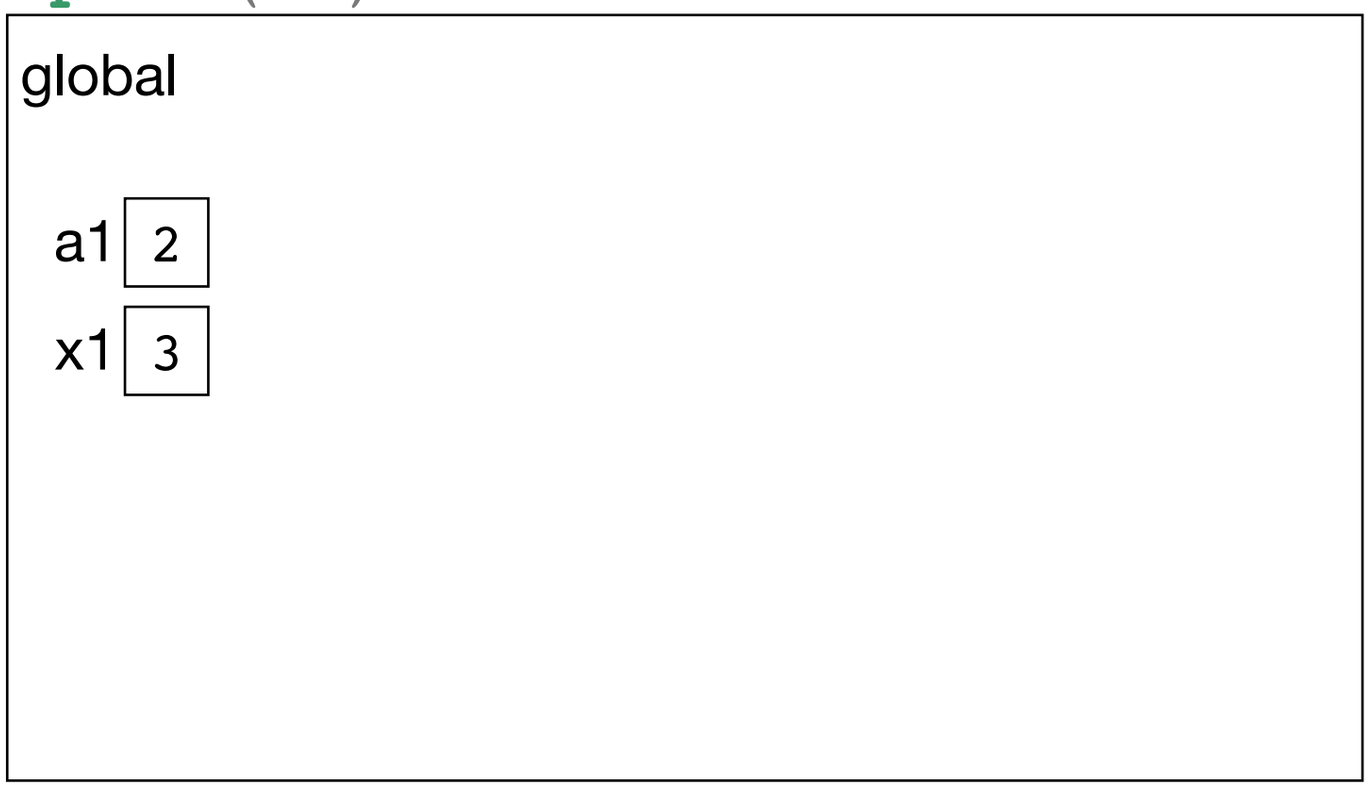
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



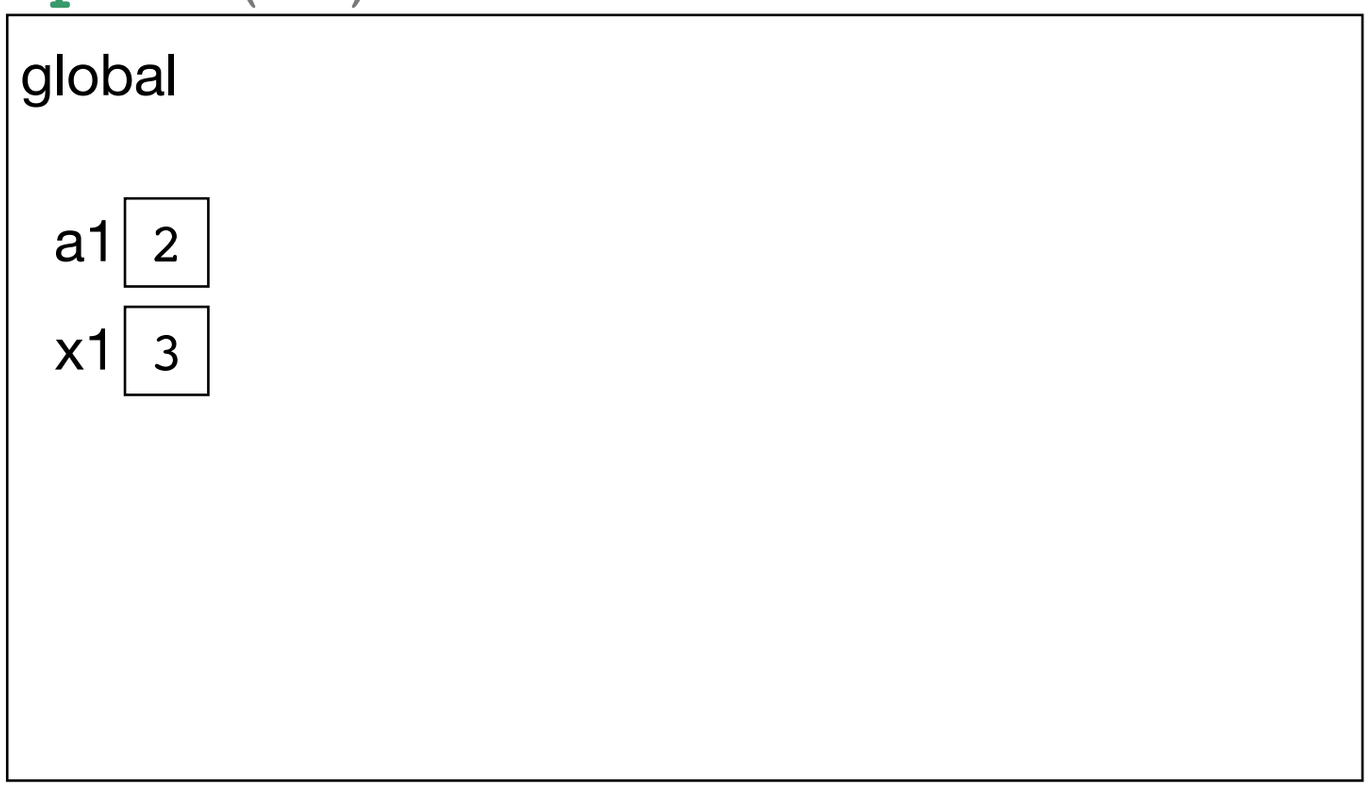
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, x1, 4))
print(a1)
```



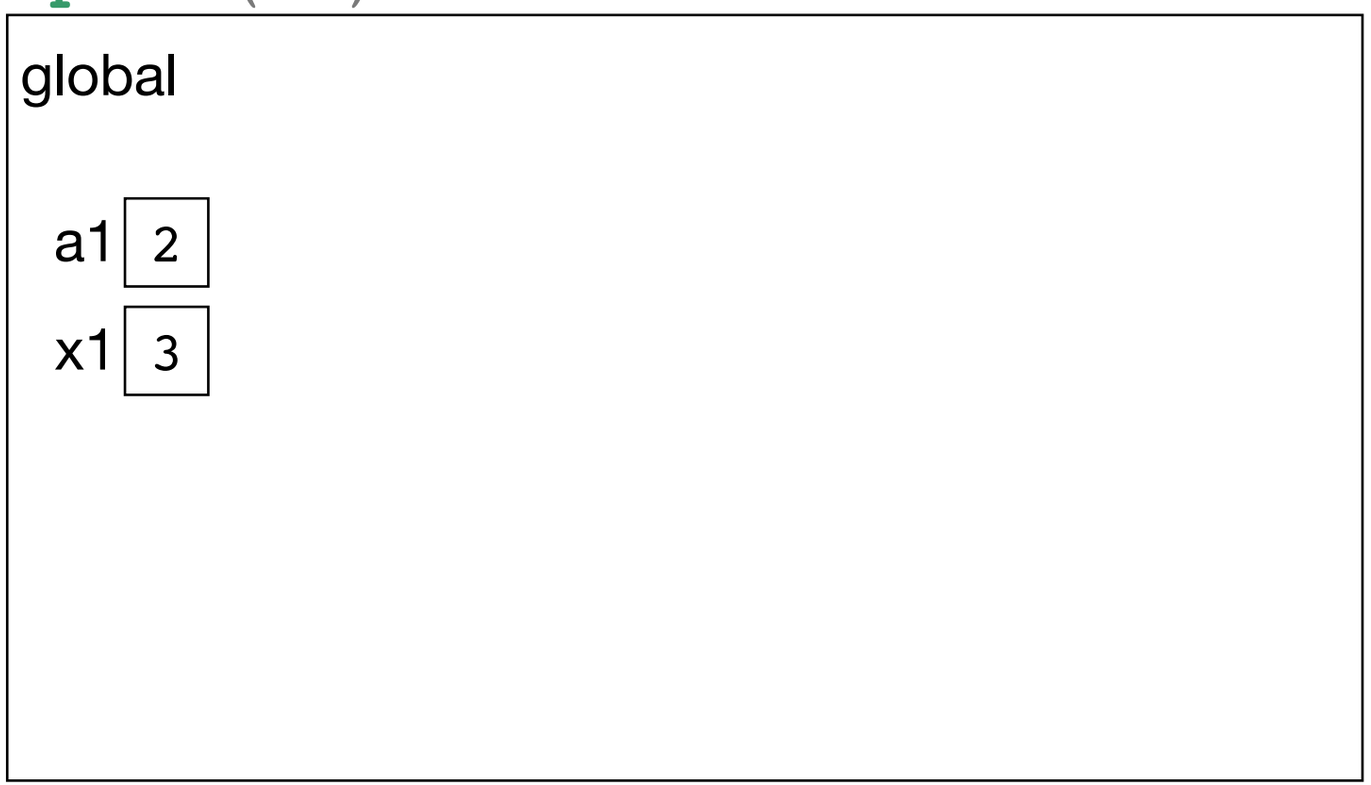
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



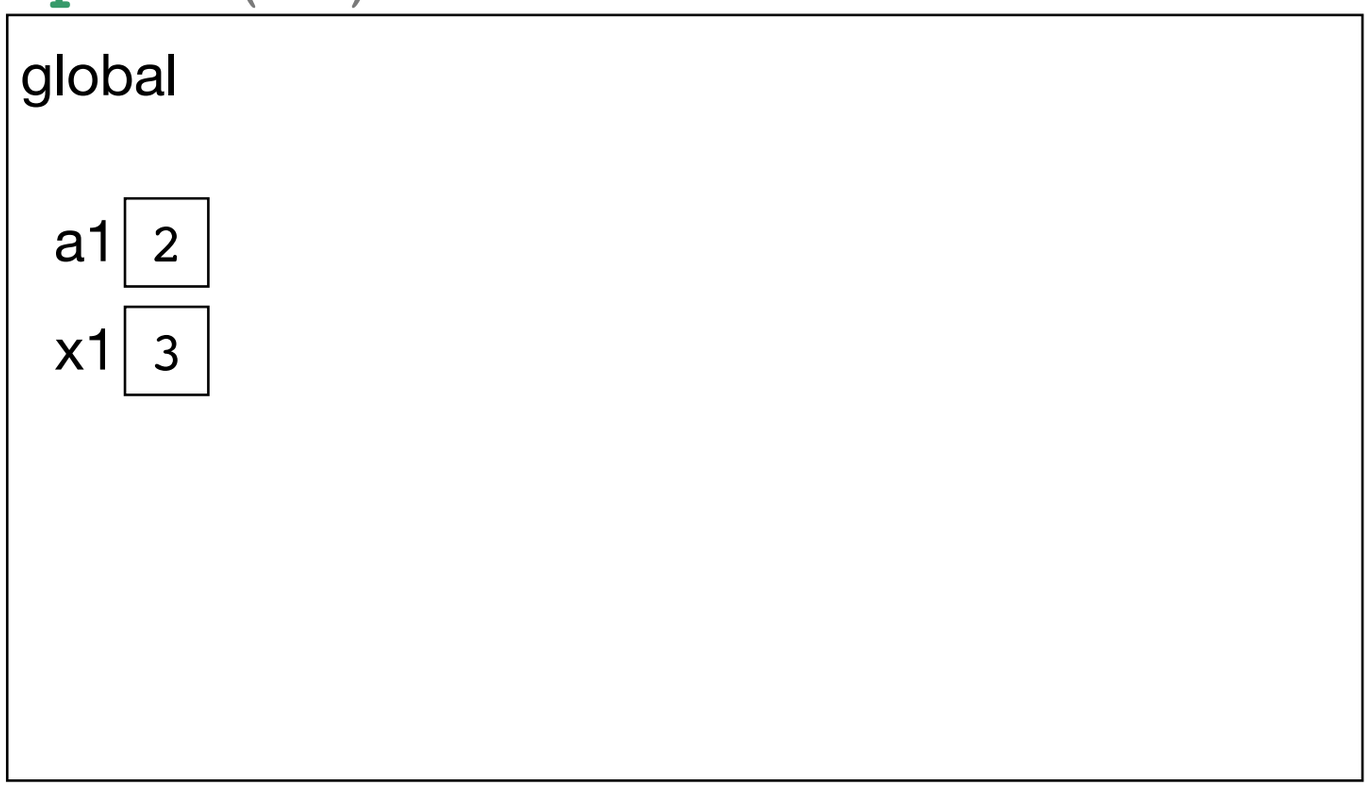
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



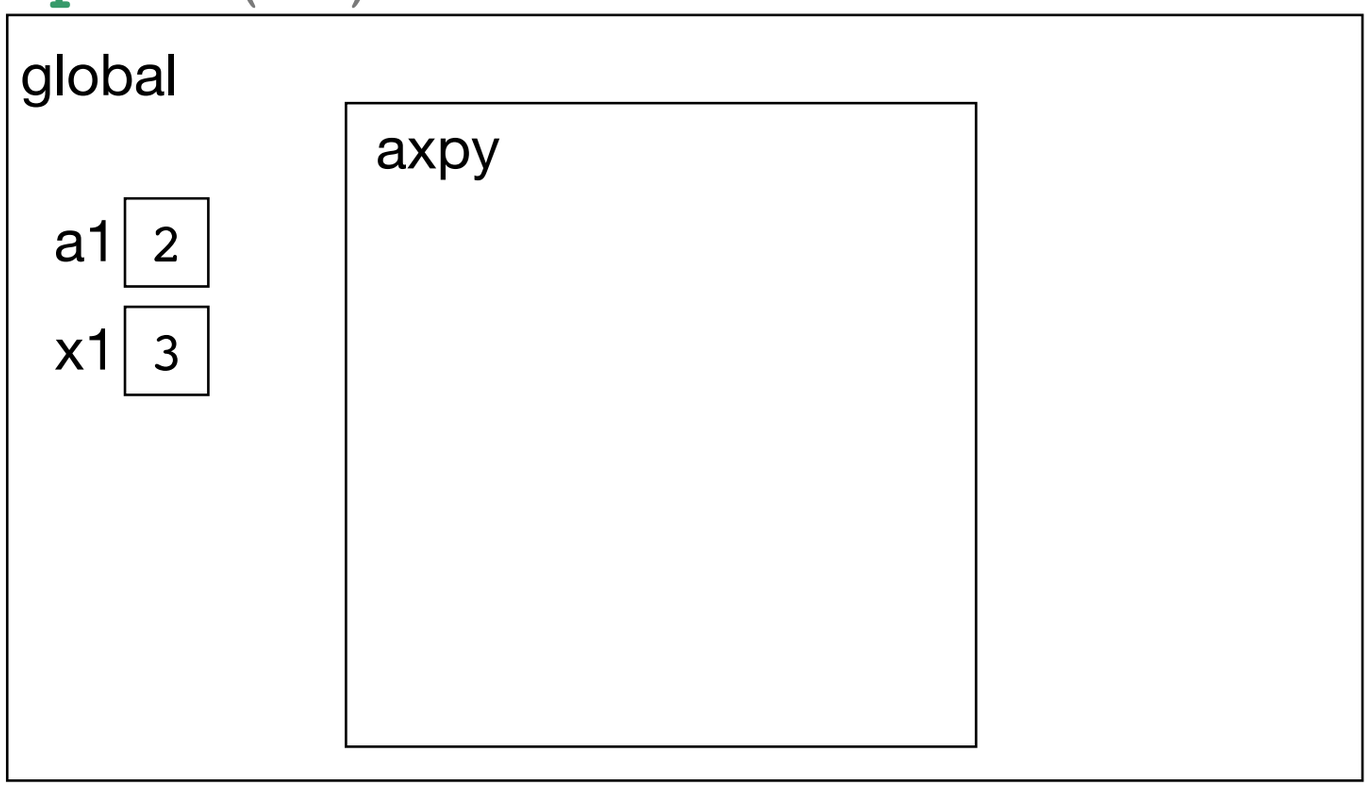
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



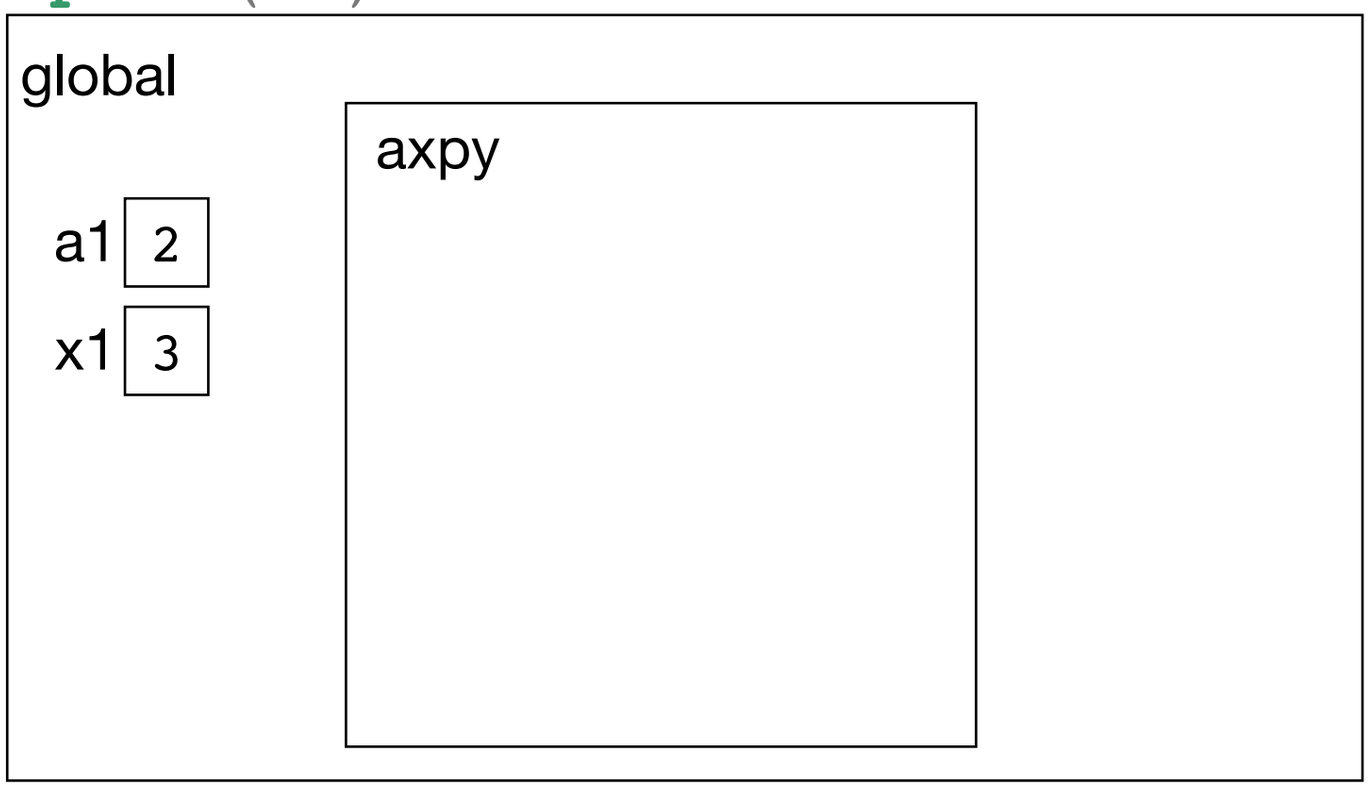
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



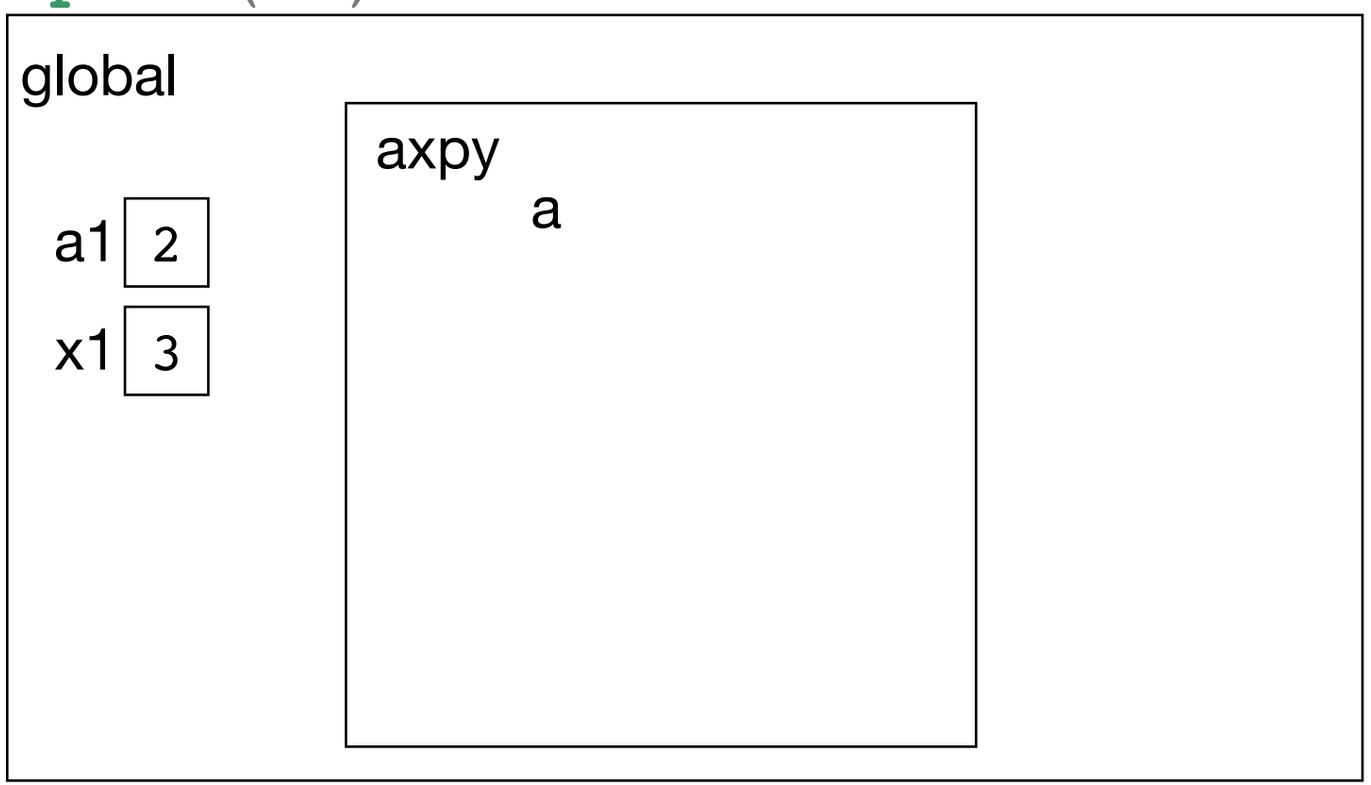
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



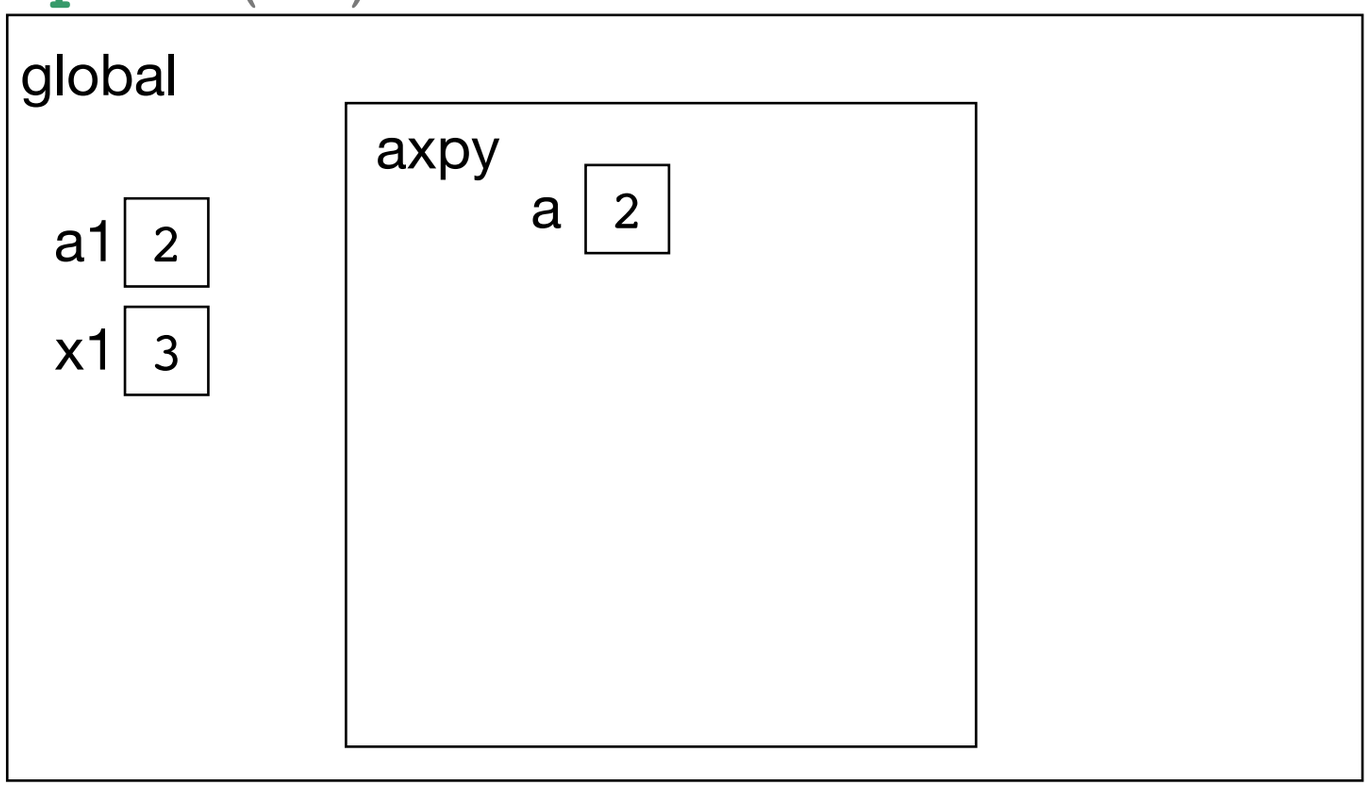
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



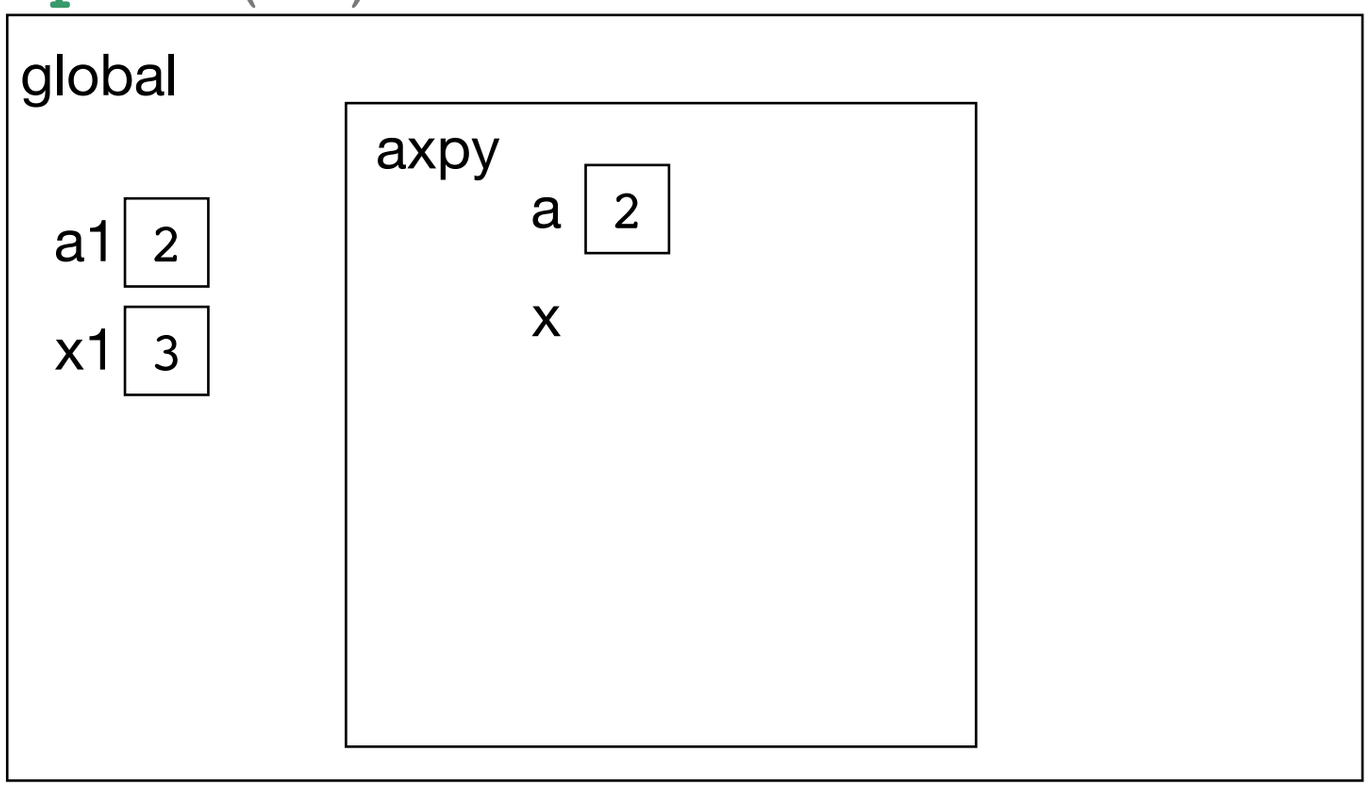
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



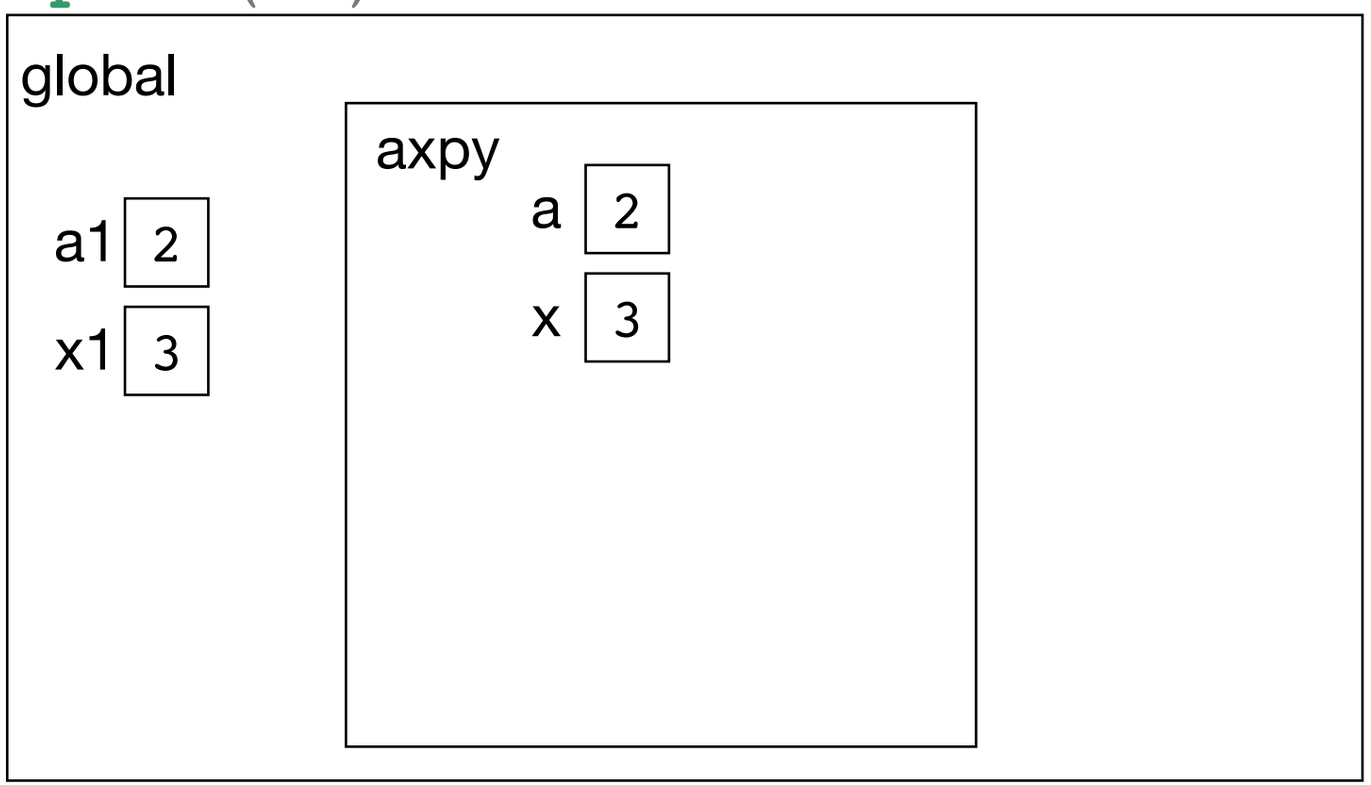
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



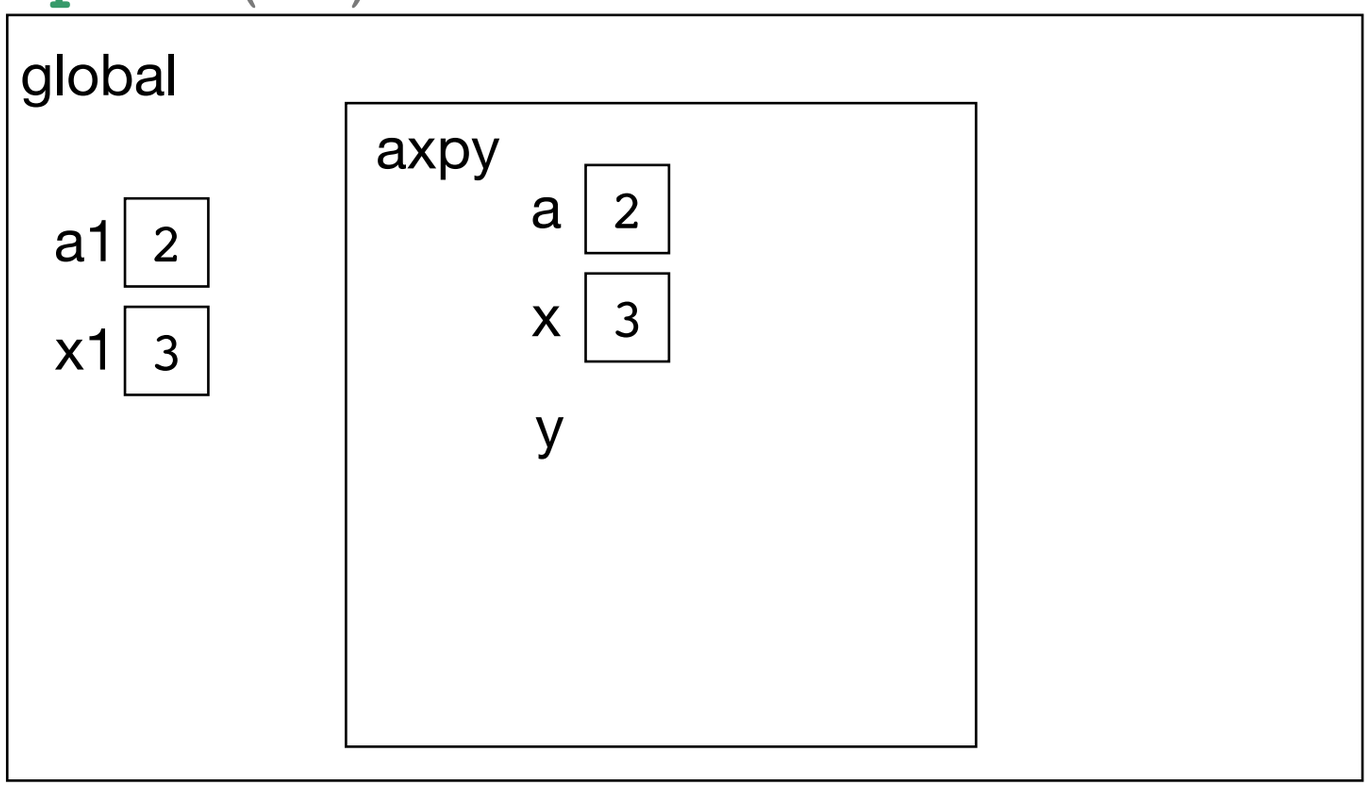
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



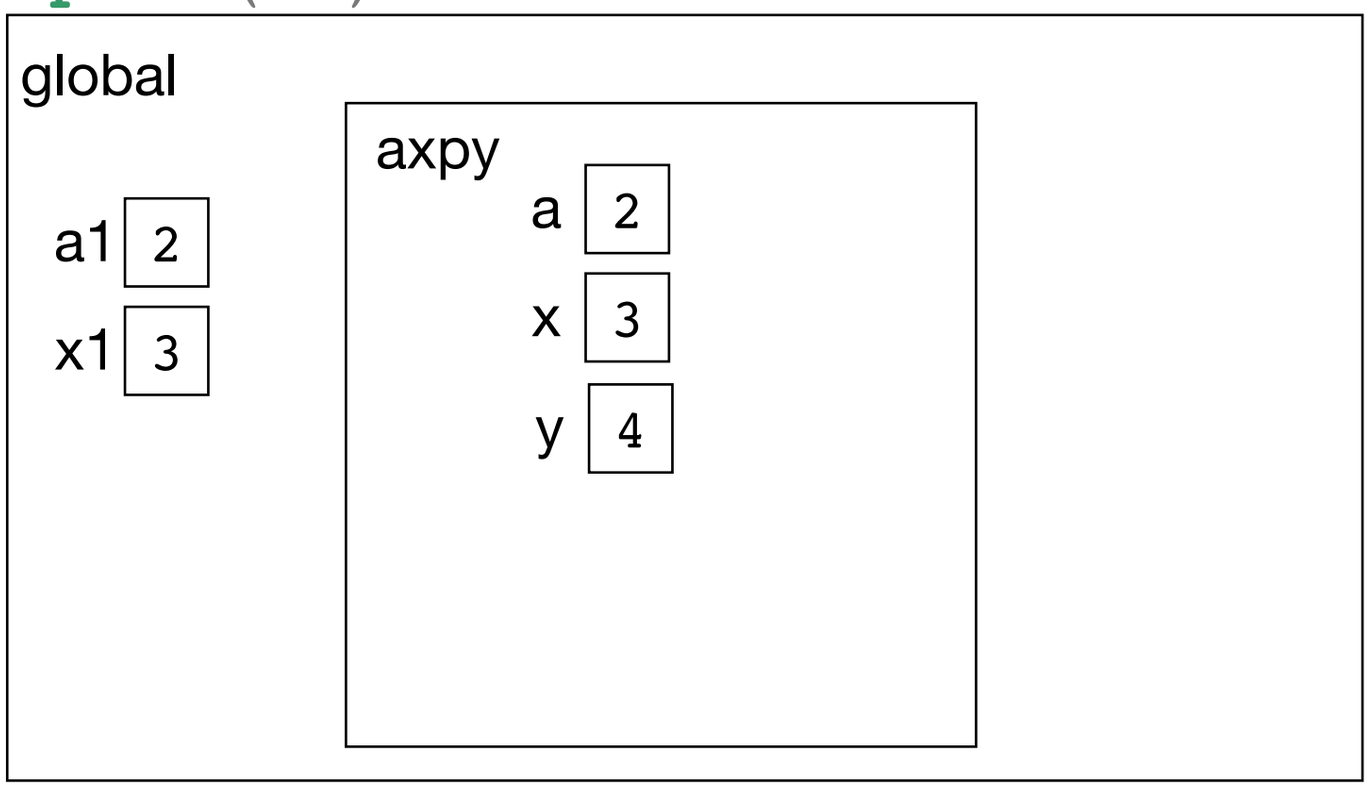
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



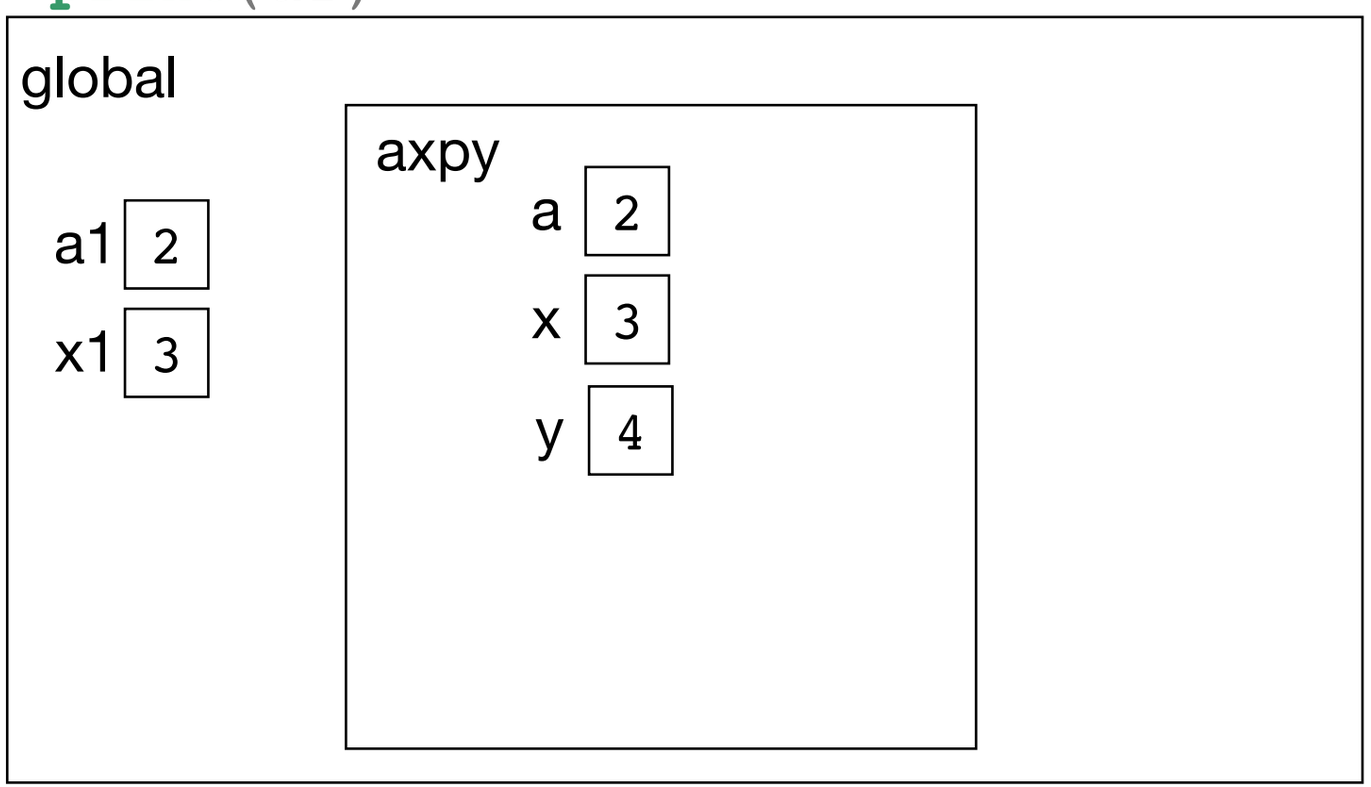
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



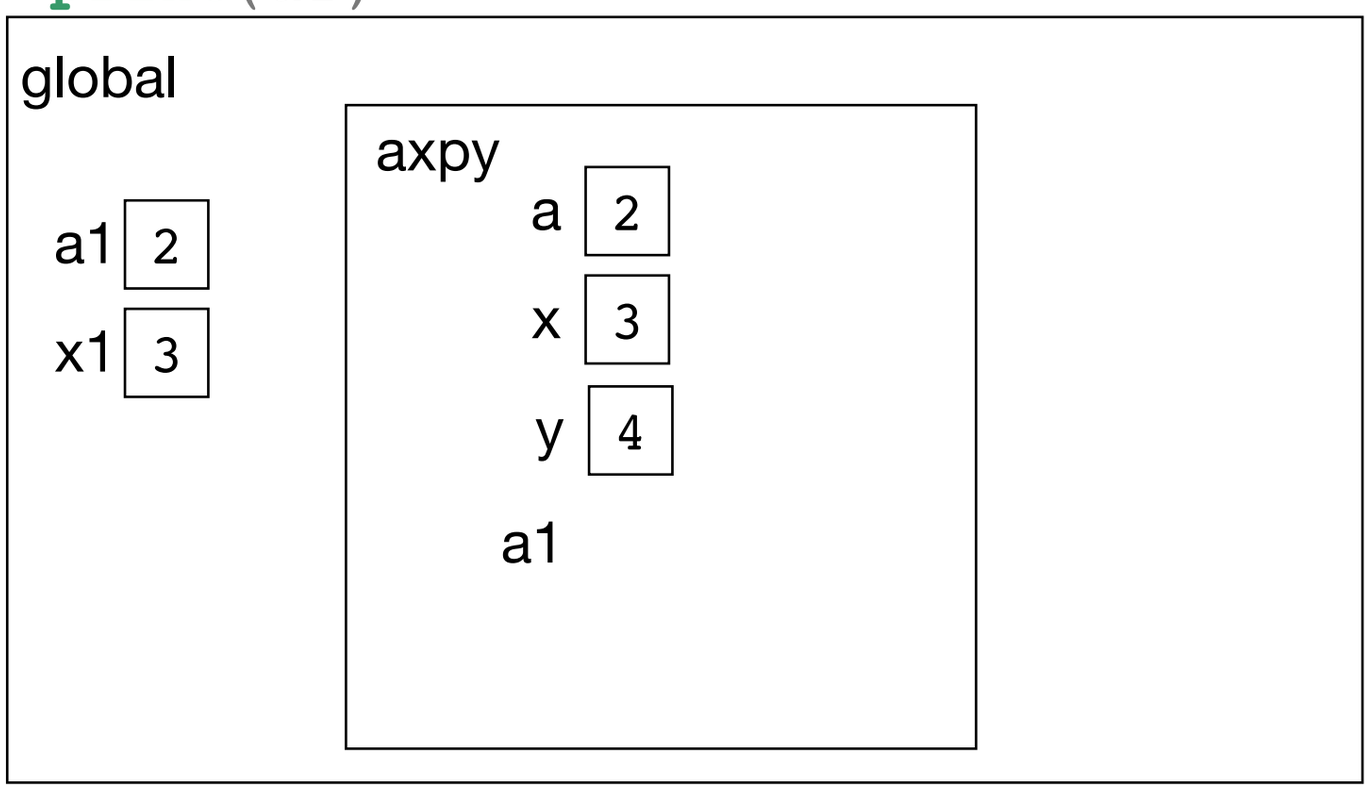
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



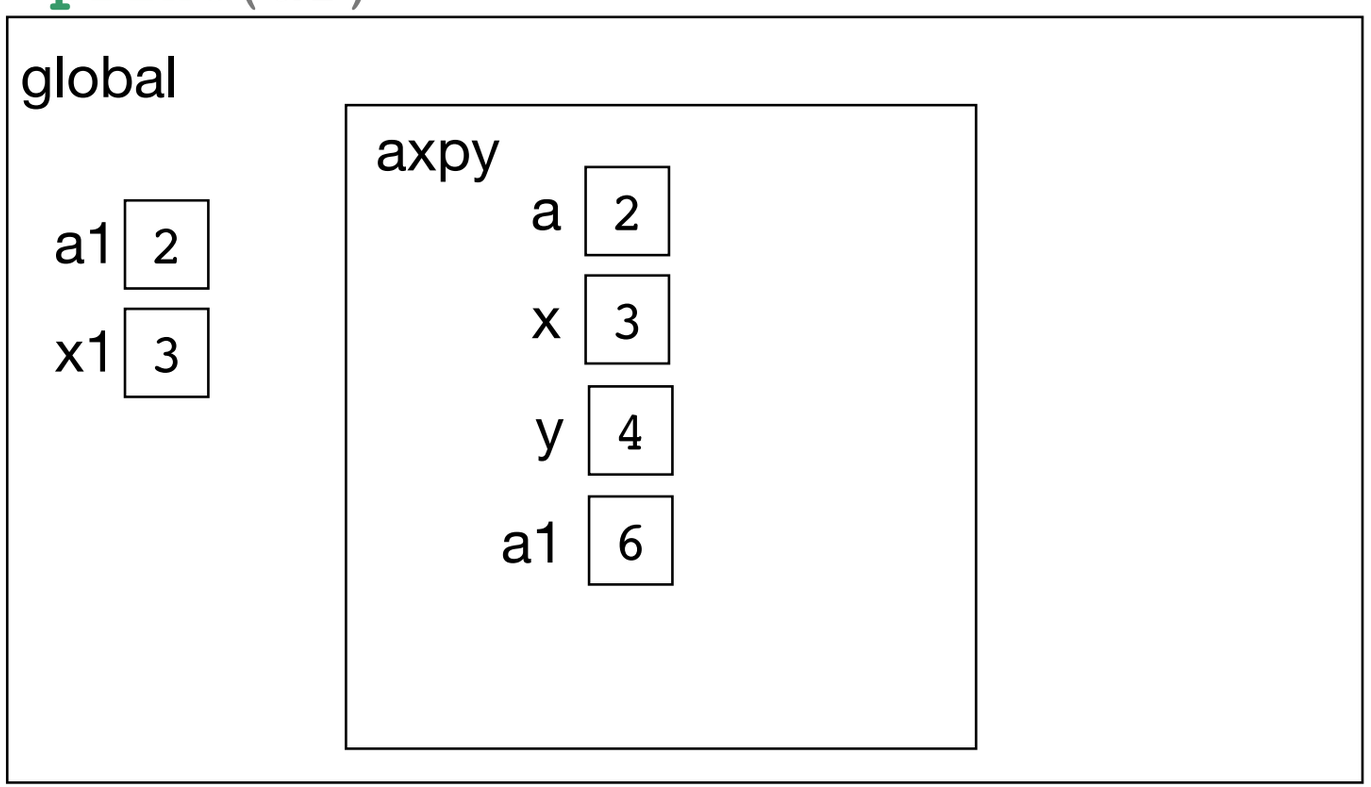
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



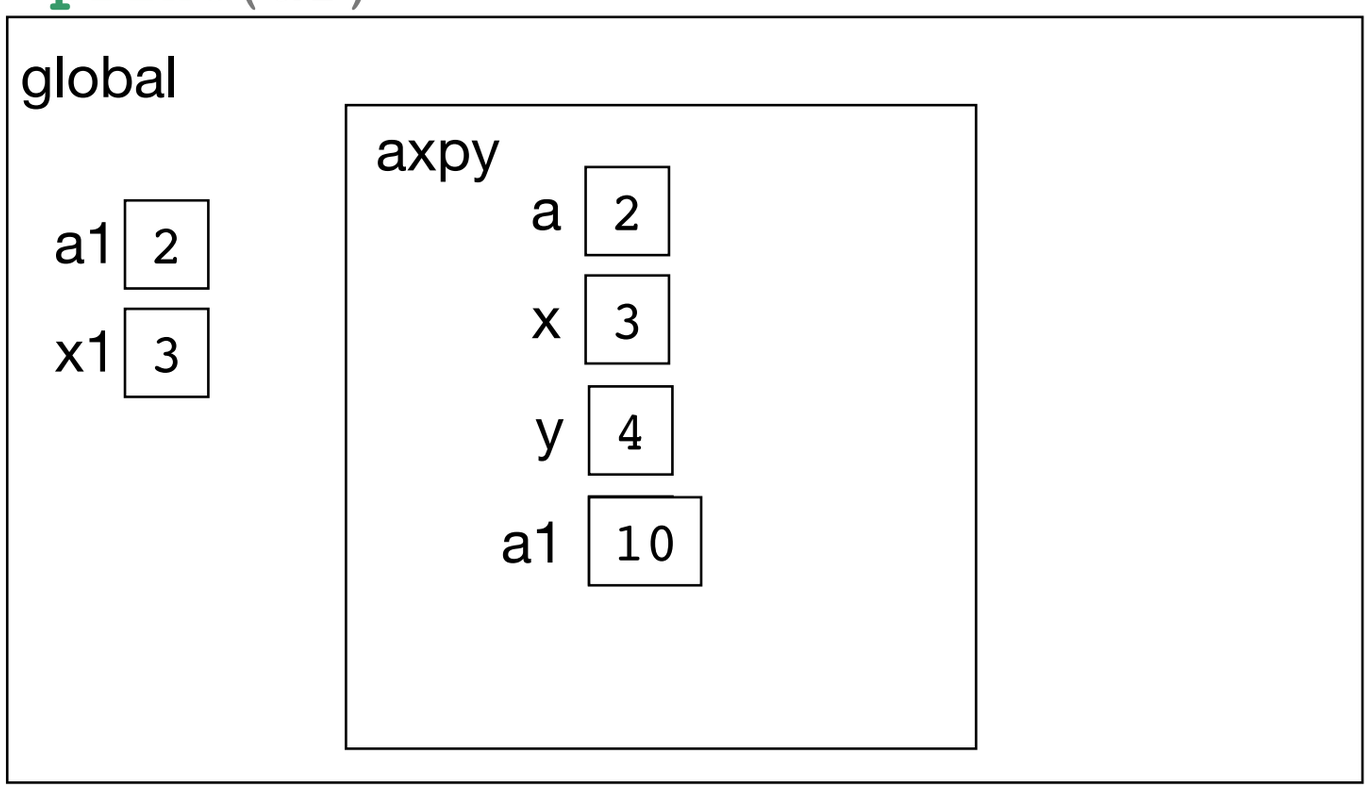
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



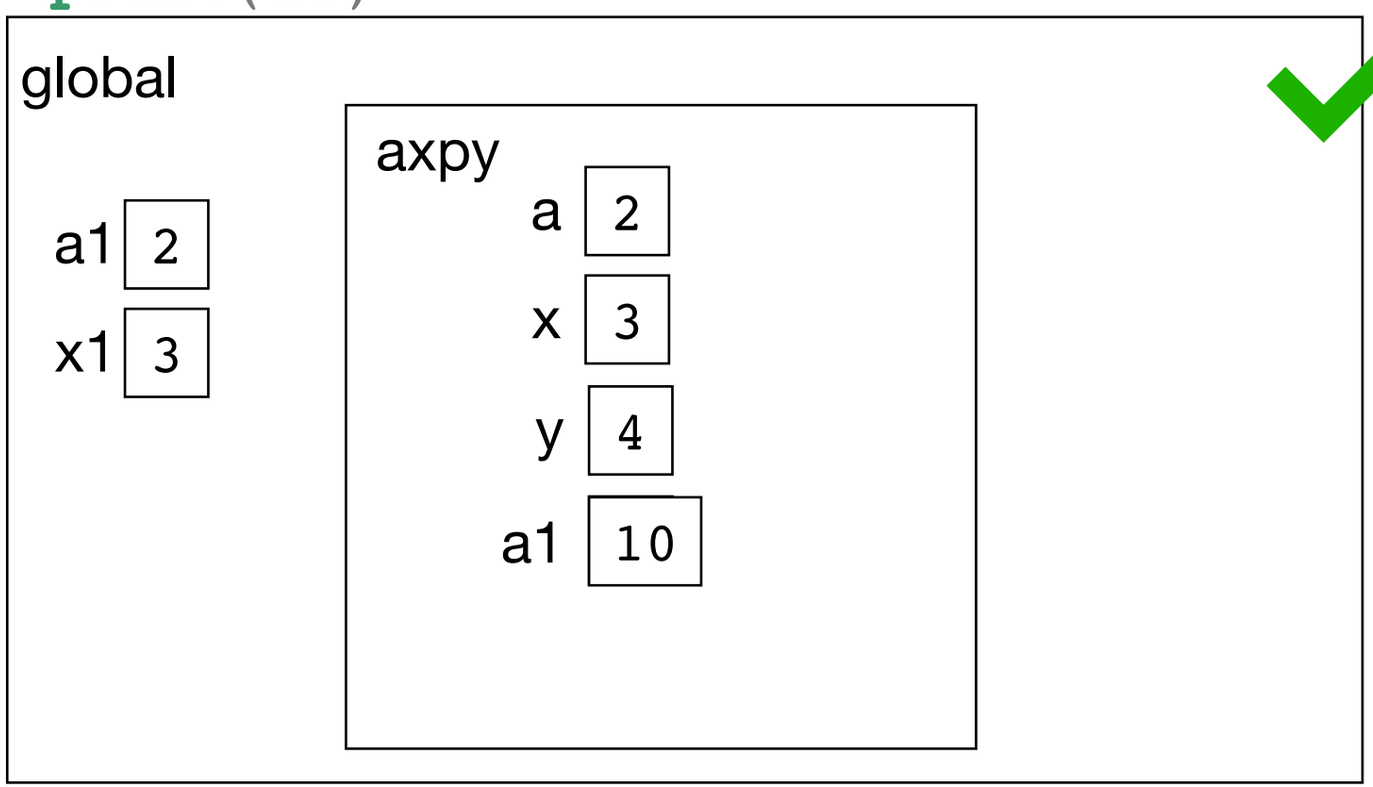
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



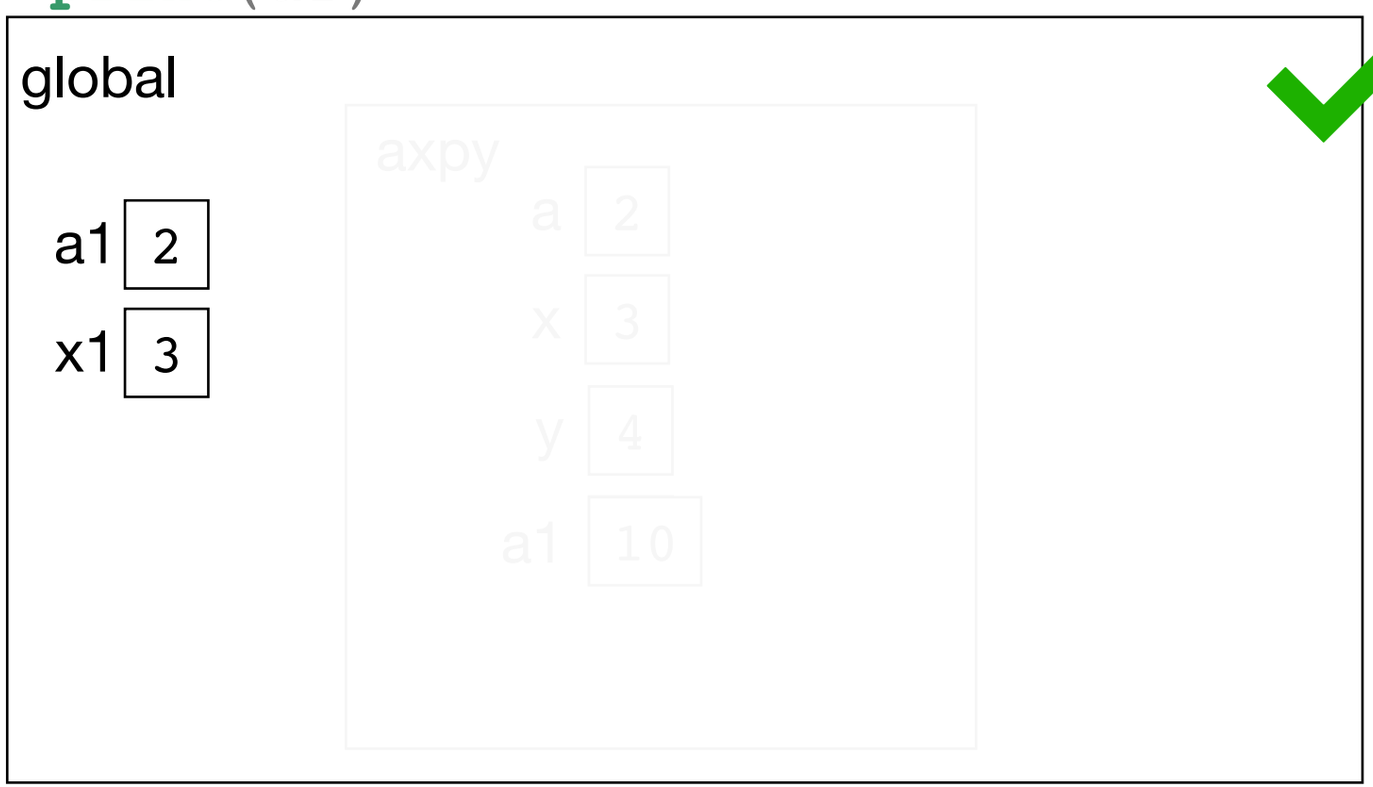
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



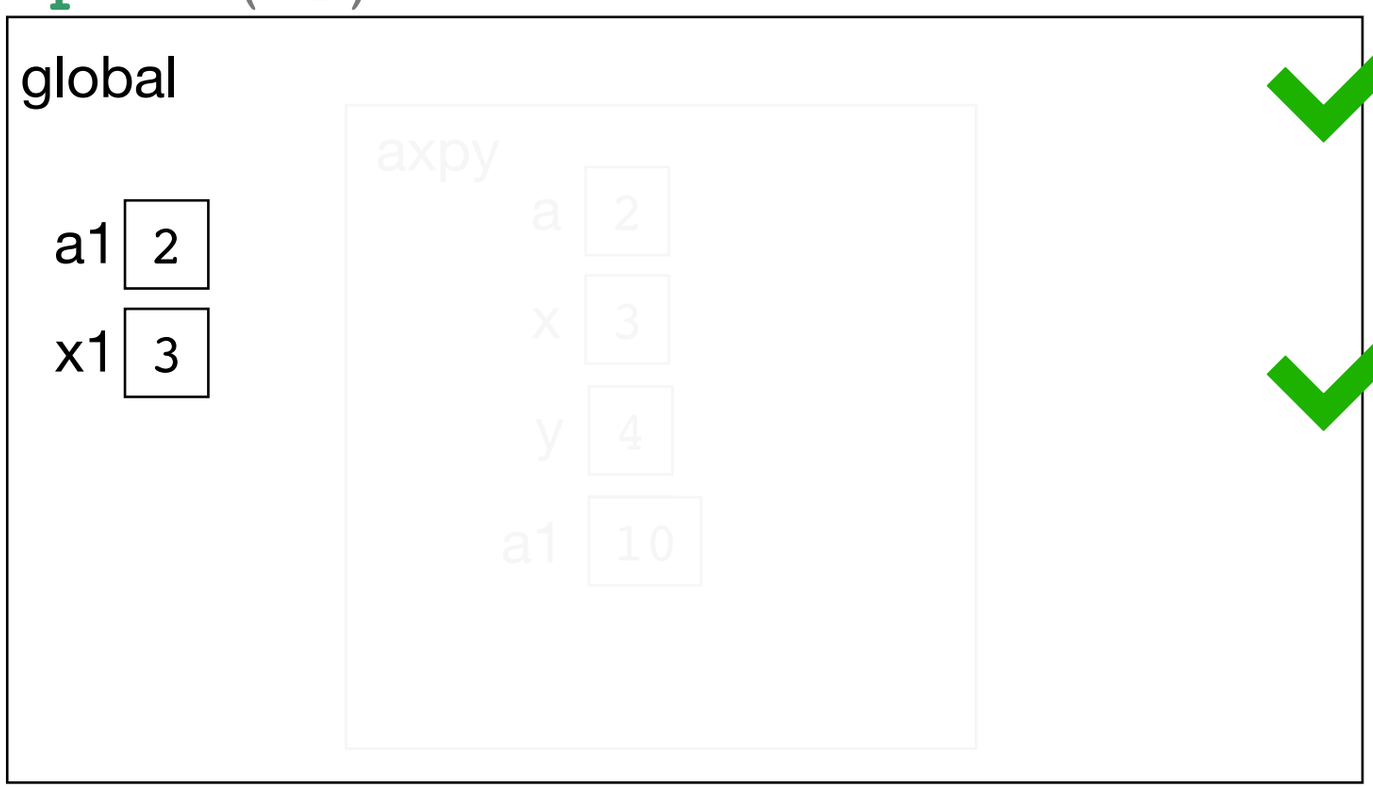
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(2, 3, 4))
print(a1)
```



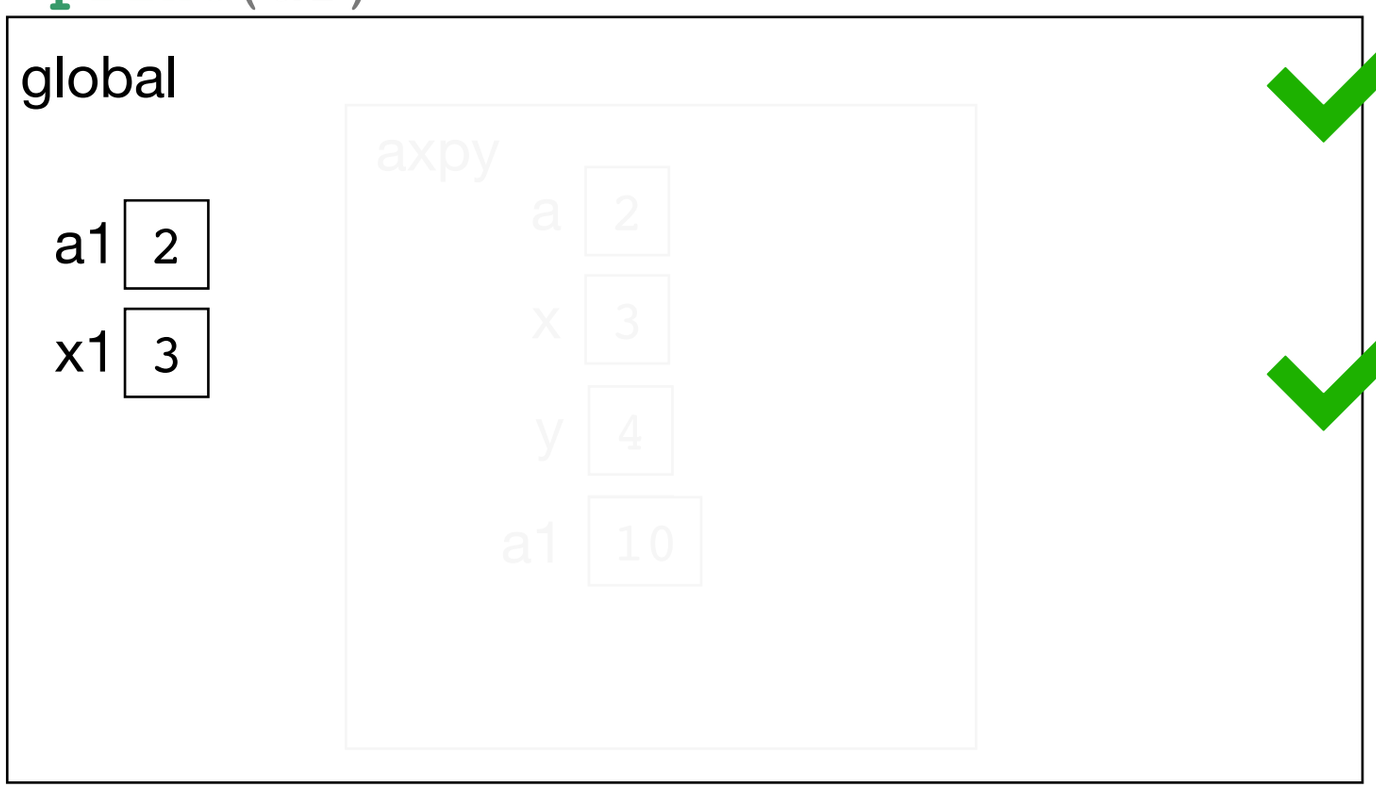
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, 3, 4))
print(a1)
```



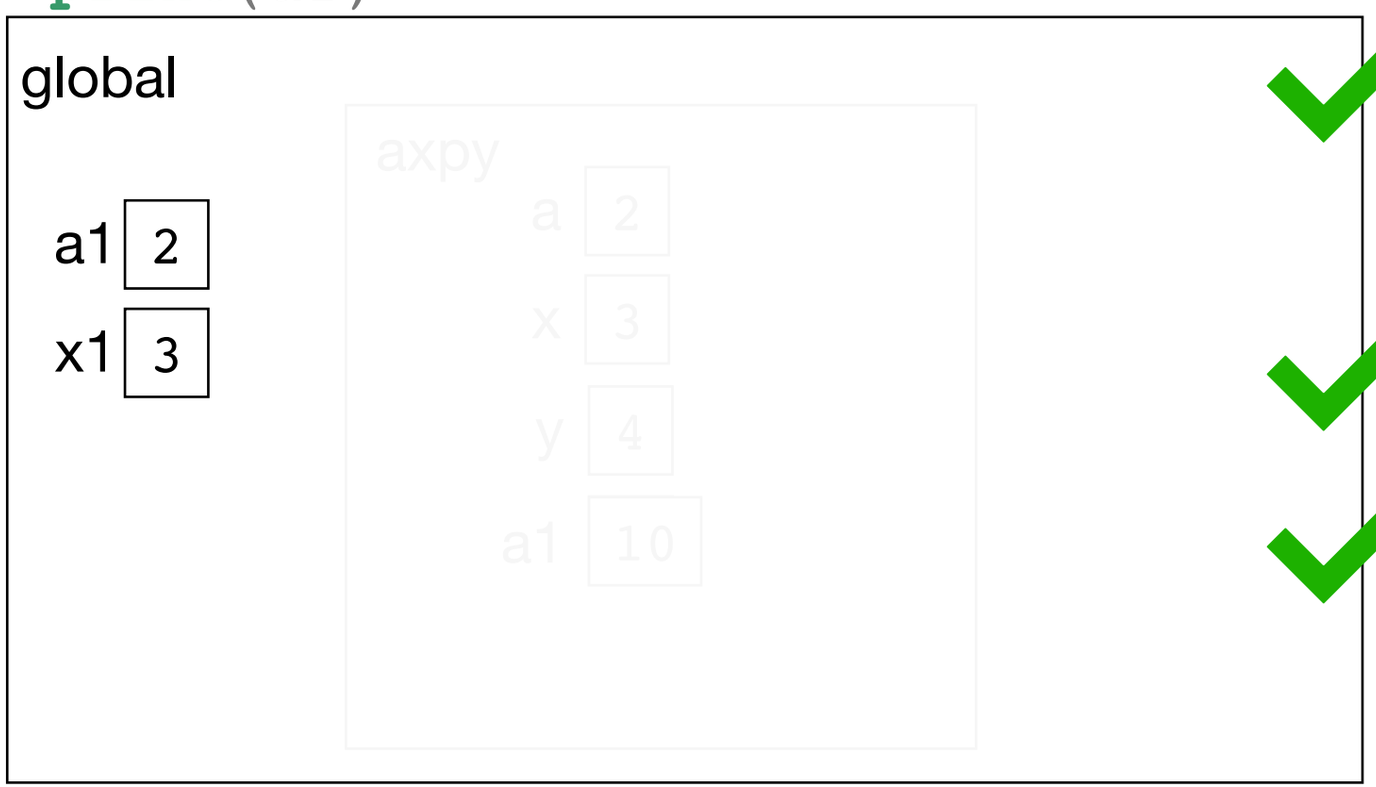
1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value

How to Execute Function Calls

```
def axpy(a, x, y):
    """ Return a*x + y """
    a1 = a * x
    a1 += y
    return a1

a1 = 2
x1 = 3
print(axpy(a1, 3, 4))
print(a1)
```



1. Evaluate all arguments
2. Draw a local "box" inside the global one
3. Assign argument values to parameter variables in the local box
4. Execute the function body

   If multiple variables exist with the same name, use the **innermost** one available.
5. When done, erase the local box
6. Replace the function call with its return value