Presentation 10

Memory in Python

Announcements For This Lecture

Assignment 1

- Work on your revisions
  - Read feedback carefully
  - Want done by tomorrow
  - Partial credit after Wed.
- **Survey**: 688 responded
  - Deadline is tomorrow
  - **Avg Time**: 6.4 hours
  - **STD Dev**: 3.5 hours

More Assignments

- Assignment 2 **TONIGHT**
  - Scan and submit online
  - Upload before midnight
  - **Late:** -10% per day
  - No lates after FRIDAY
- Assignment 3 is posted
  - Due week from Friday
  - Last material for exam
  - Graded by Sun morning

Announcements For This Lecture

Assignment 1

- Work on your revisions
  - Read feedba
  - Want done
  - Partial cre
- **Survey**: 676
  - Deadline i
  - **Avg Time**: 6.5 hours
  - **STD Dev**: 3.4 hours

More Assignments

- Assignment 2 **TONIGHT**
  - Scan and submit online
  - fore midnight
  - % per day
  - ter THURS
  - 3 is posted
  - from Friday
  - Last material for exam
  - Graded by Sun morning

Video Lessons

- **Lesson 12** for today
- **Lessons 13, 14** next time

The First Prelim!

Prelim Details

- **Oct 18th 7:30-9:00**
  - Material up October 13th
  - Lists: **yes**, For-loops **no**
  - Study guide next week
- **Conflict with time?**
  - Submit to assign. Prelim 1 Conflict on CMS
  - Don't submit if no conflict

Online Exams

- For **online students** only
  - We have a list of you
- Onsite, but have excuse?
  - Submit as a **conflict** form
  - Give reason; will review
- Must take a **Mock Exam**
  - To ensure you are set up
  - Proctor will contact you

The First Prelim!

Prelim Details

- **Oct 18th 7:30-9:00**
  - Material u
  - Lists: **yes**,
  - Study guic
- **Conflict wi**
  - Submit to
    Conflict o
  - Don't submit if no conflict

Online Exams

- For **online students** only
  - list of you
  - have excuse?
  - a **conflict** form
  - n; will review
  - **Mock Exam**
  - you are set up
  - Proctor will contact you

SDS Student?

- **Submit a conflict form**
  - Whether online or onsite
  - Tell us your constraints
- **Else take normal exam**

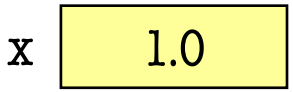
Activity Time: A Module

```
1 def polar_to_x(r,ang):
2     import math
3     return r*math.cos(ang)
4
5 def xy_to_rad(x,y):
6     return math.sqrt(x*x+y*y)
7
8 x = polar_to_x(1,0)
9 r = xy_to_rad(0,1)
```

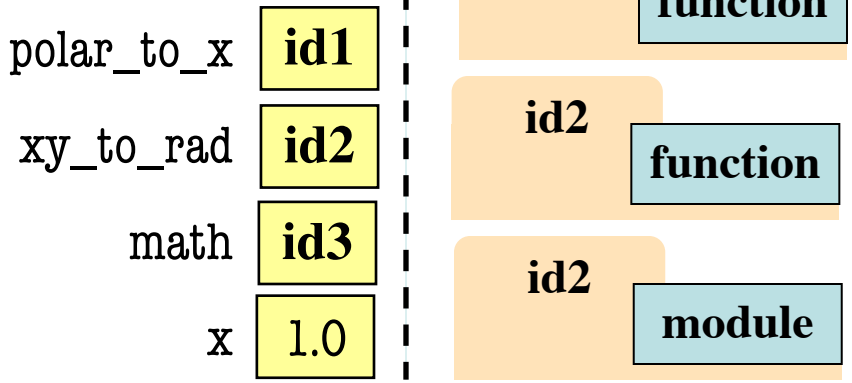
What is **global space**/**the heap** *after* line 8?

Which One is Closest to Your Answer?

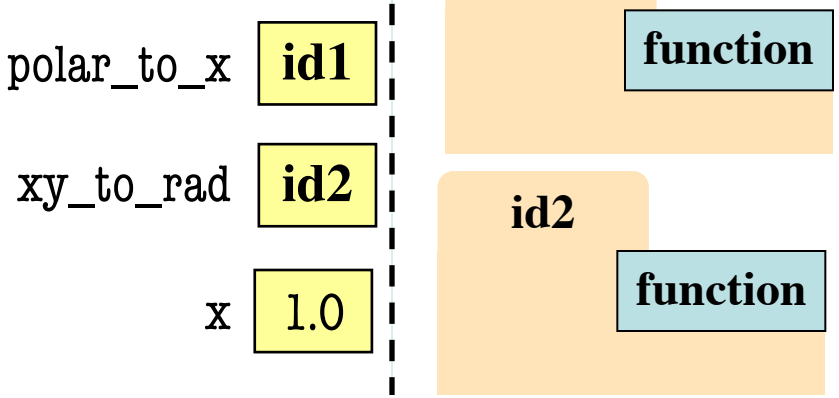
A:



B:



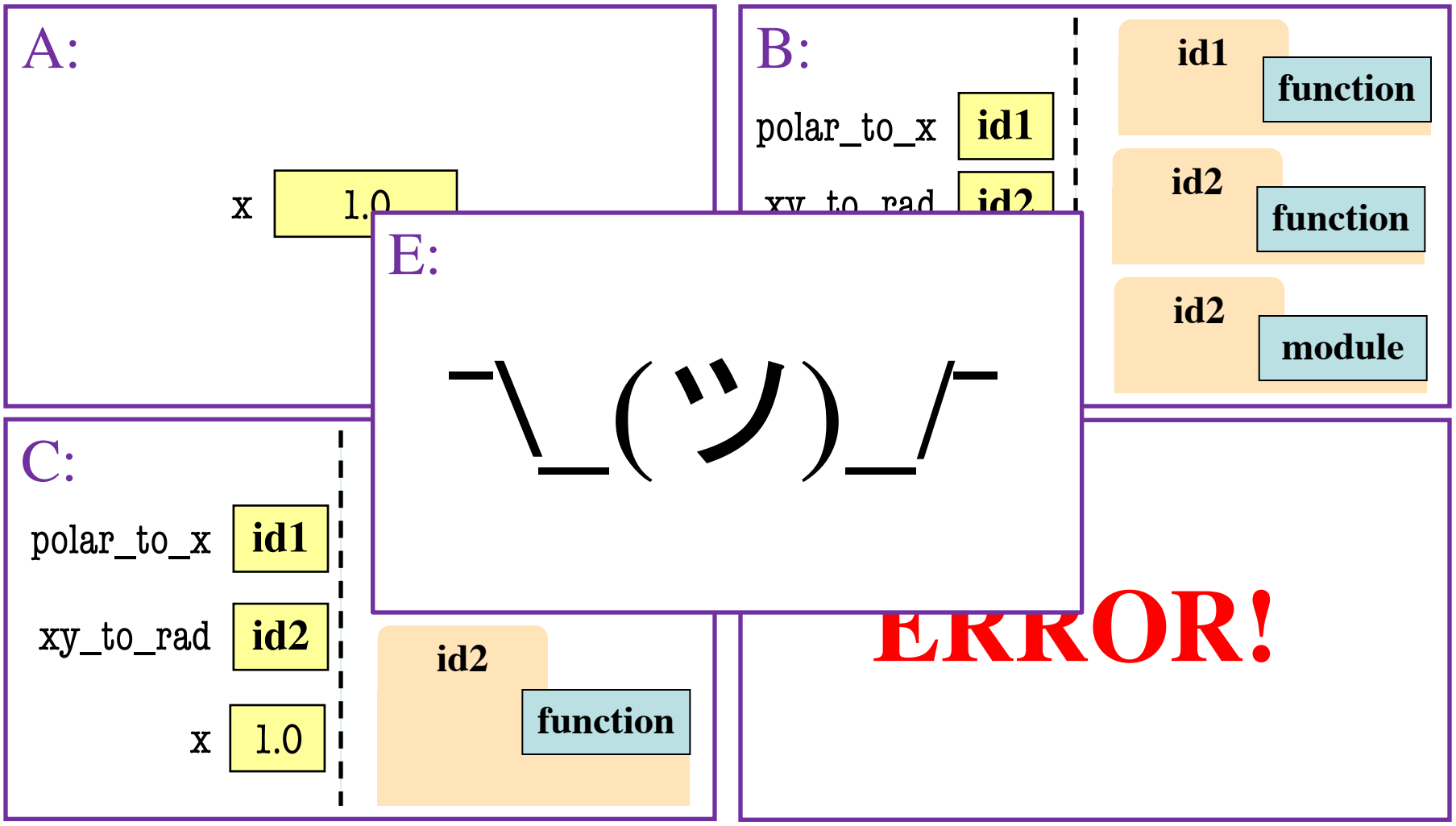
C:



D:

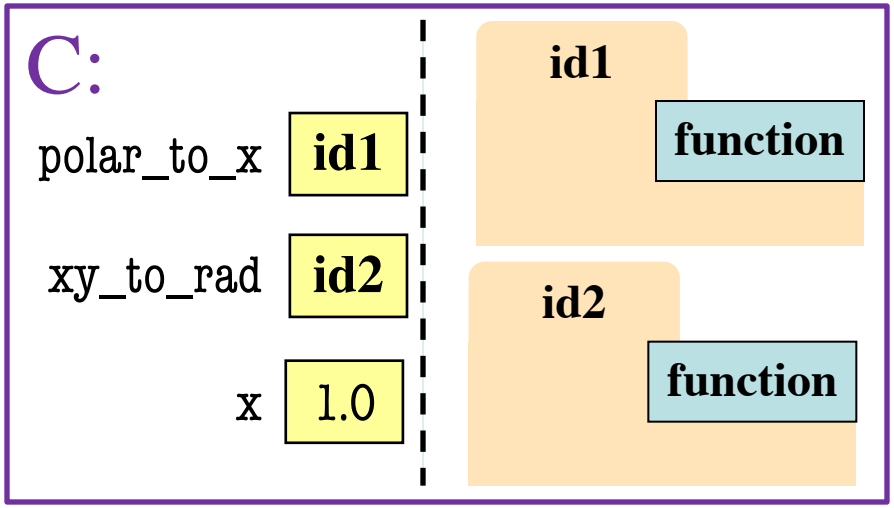
ERROR!

Which One is Closest to Your Answer?



Activity Time: A Module

```
1 def polar_to_x(r,ang):
2     import math
3     return r*math.cos(ang)
4
5 def xy_to_rad(x,y):
6     return math.sqrt(x*x+y*y)
7
8 x = polar_to_x(1,0)
9 r = xy_to_rad(0,1)
```



What is **global space**/**the heap** *after* line 8?

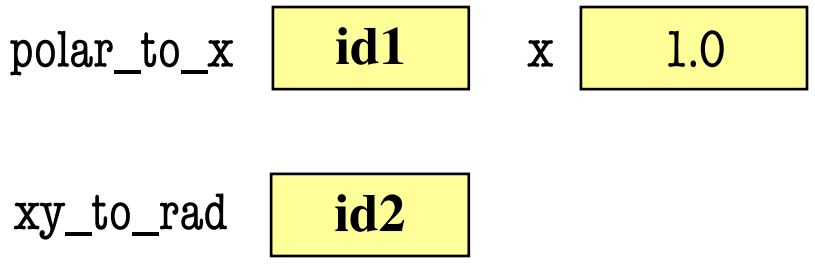
Activity Time: A Module

```
1 def polar_to_x(r,ang):
2     import math
3     return r*math.cos(ang)
4
5 def xy_to_rad(x,y):
6     return math.sqrt(x*x+y*y)
7
8 x = polar_to_x(1,0)
9 r = xy_to_rad(0,1)
```

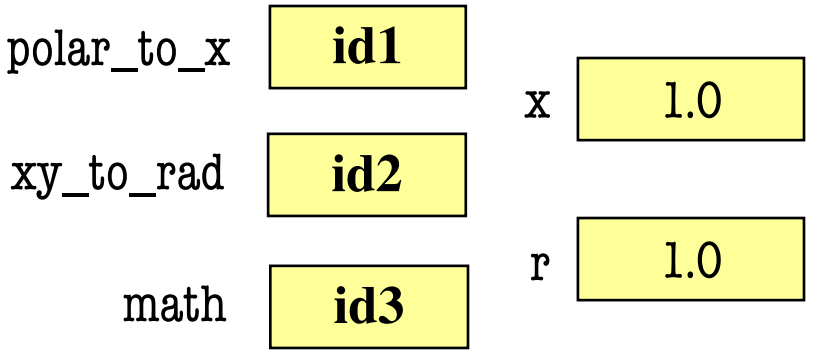
What is in **global space** *after* line 9?

Which One is Closest to Your Answer?

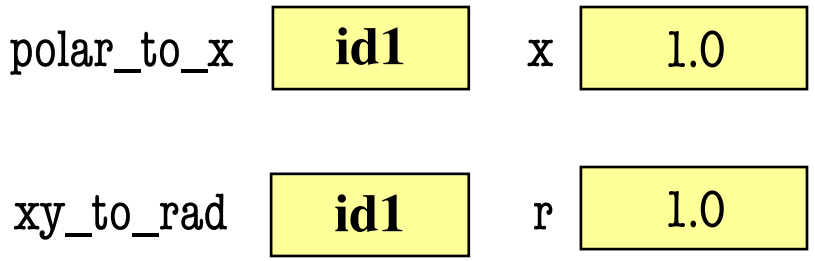
A:



B:



C:



D:

ERROR!

Activity Time: A Module

```
1 def polar_to_x(r,ang):
2     import math
3     return r*math.cos(ang)
4
5 def xy_to_rad(x,y):
6     return math.sqrt(x*x+y*y)
7
8 x = polar_to_x(1,0)
9 r = xy_to_rad(0,1)
```

D:

ERROR!

What is in **global space** *after* line 9?

Activity Time: The Call Stack

Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return y+1
```

Function Call

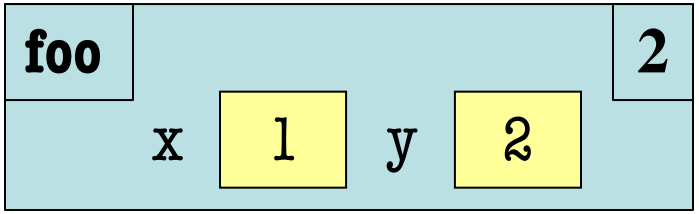
```
>>> foo(1)
```

What does the call stack look like at the **start**?

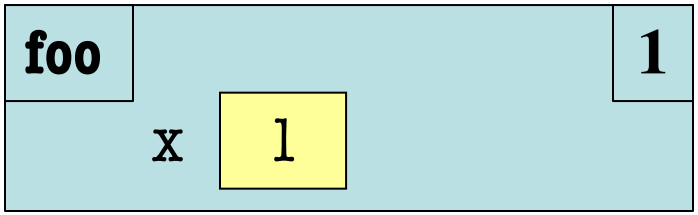
Which One is Closest to Your Answer?

A:

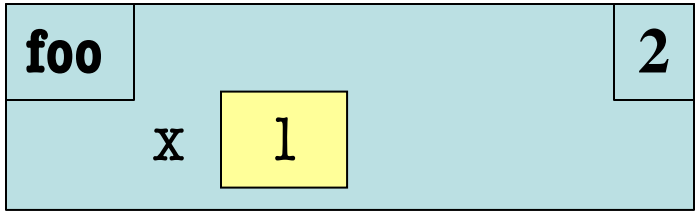
B:



C:



D:



Activity Time: The Call Stack

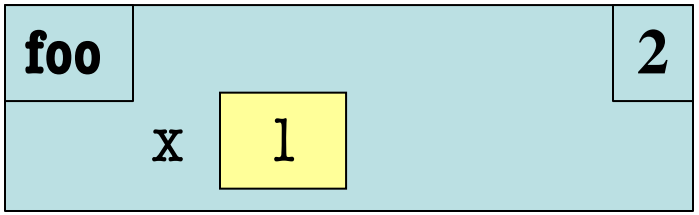
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return y+1
```

Function Call

```
>>> foo(1)
```

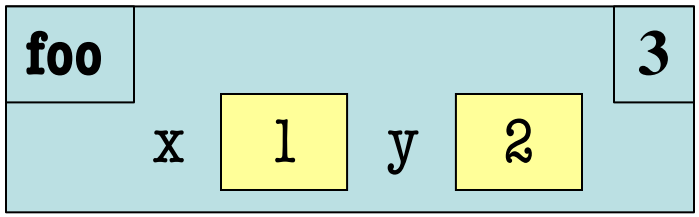
D:



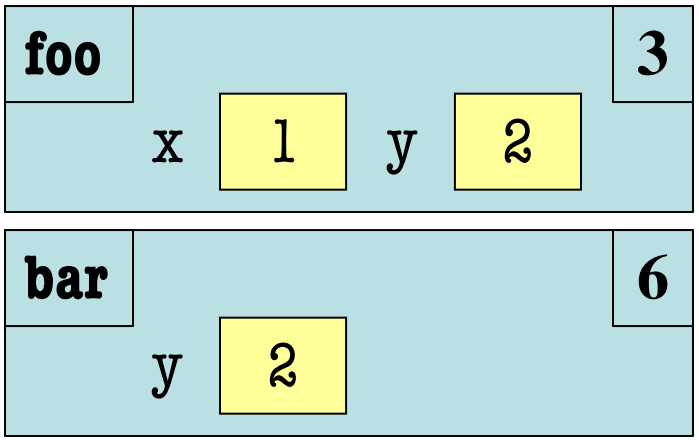
What is the **next step**?

Which One is Closest to Your Answer?

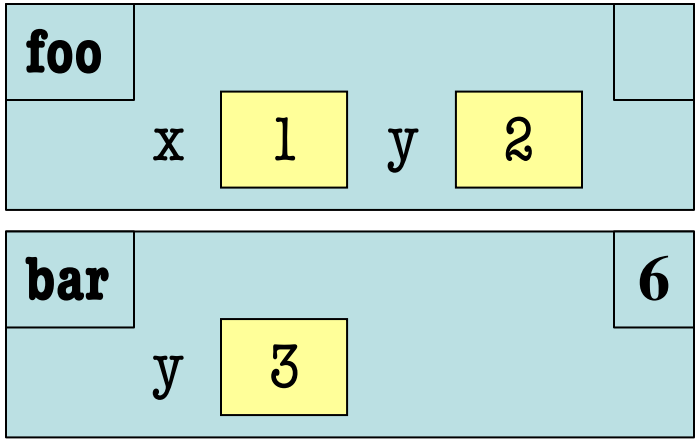
A:



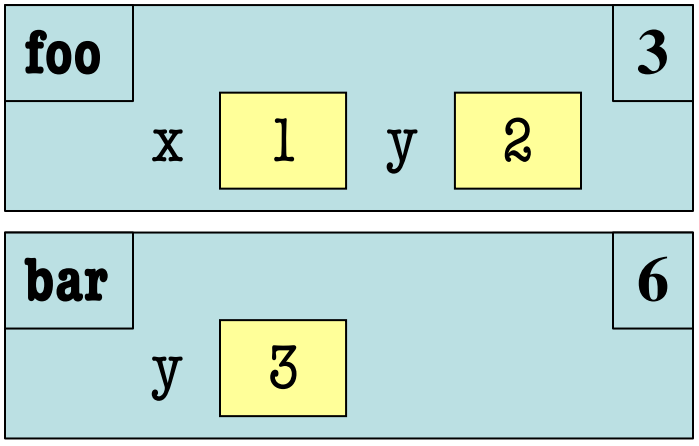
B:



C:



D:



Activity Time: The Call Stack

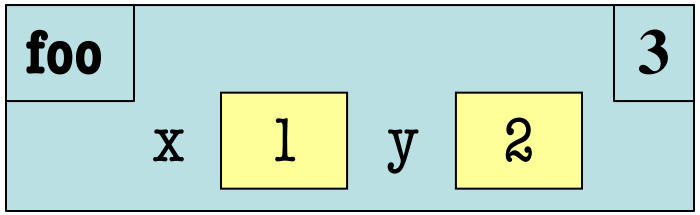
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return y+1
```

Function Call

```
>>> foo(1)
```

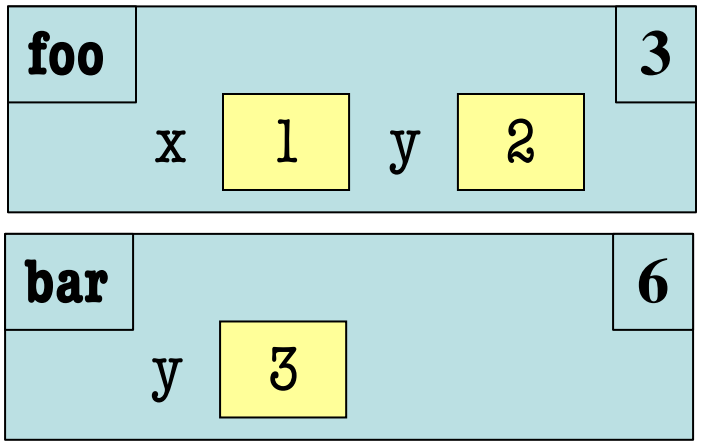
A:



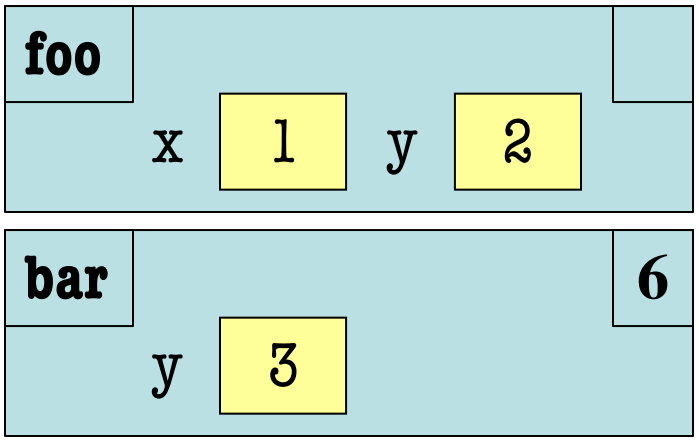
What is the **next step**?

Which One is Closest to Your Answer?

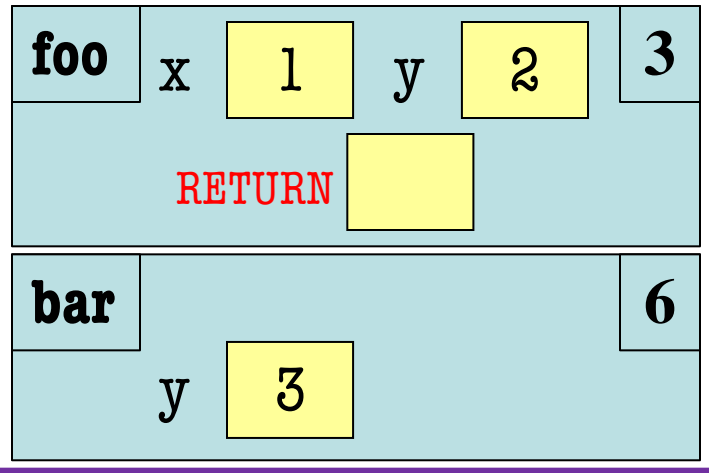
A:



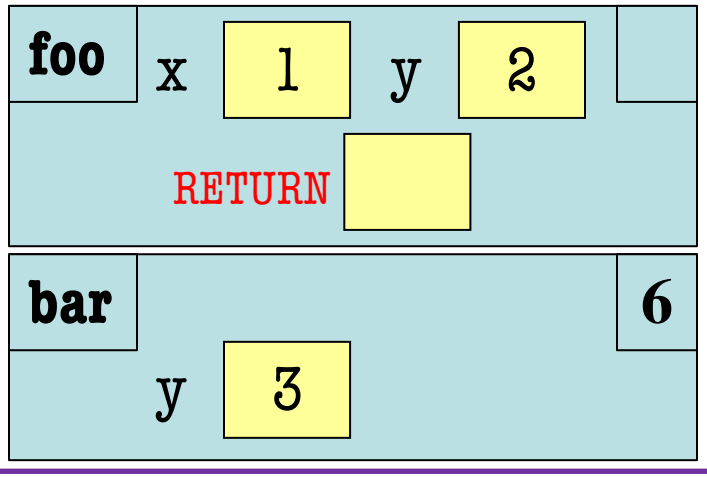
B:



C:



D:



Activity Time: The Call Stack

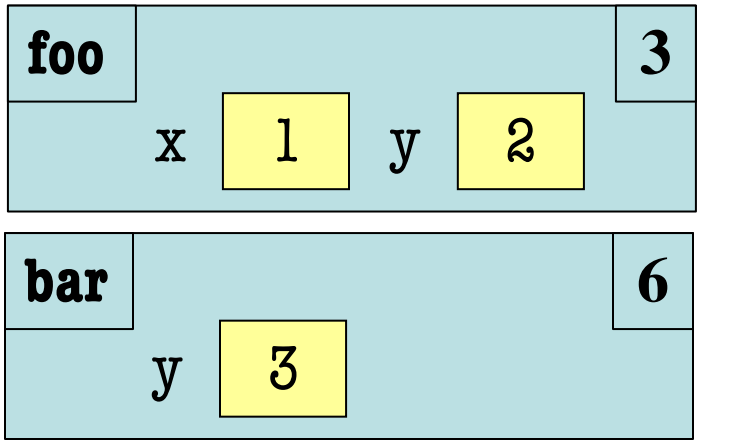
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return y+1
```

Function Call

```
>>> foo(1)
```

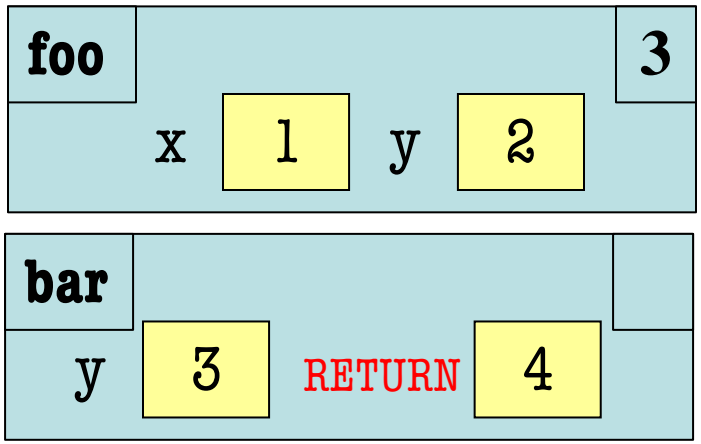
A:



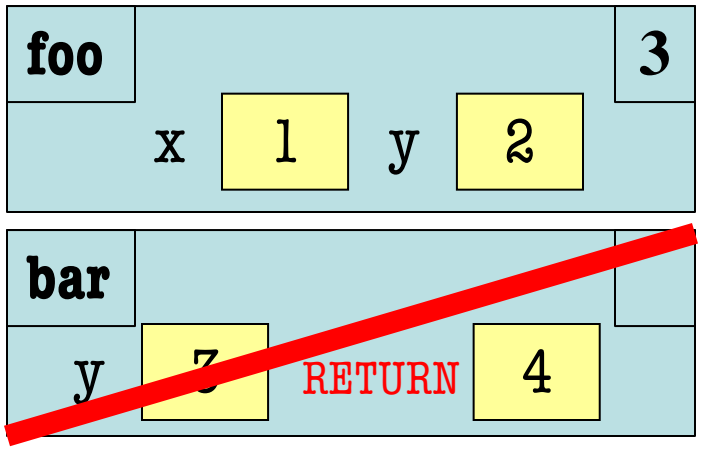
What is the **next step**?

Which One is Closest to Your Answer?

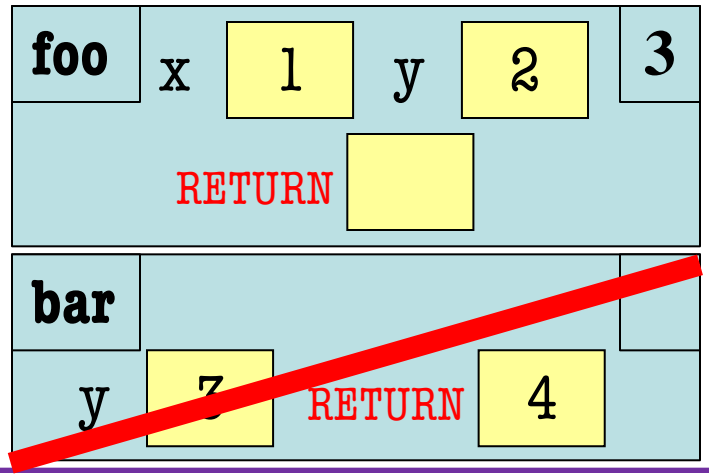
A:



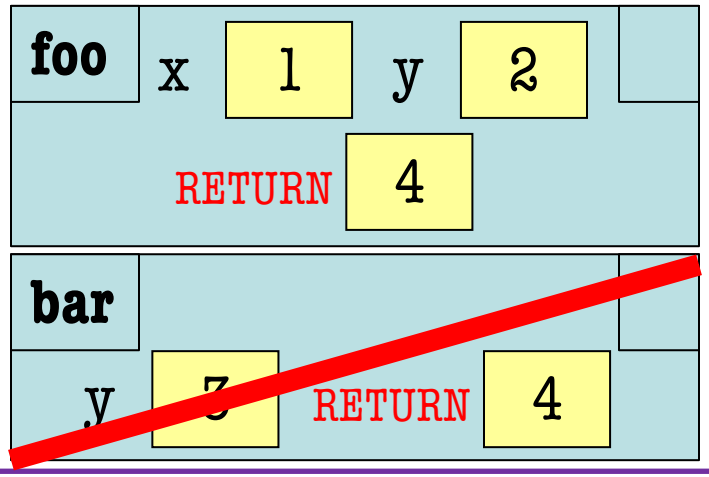
B:



C:



D:



Activity Time: The Call Stack

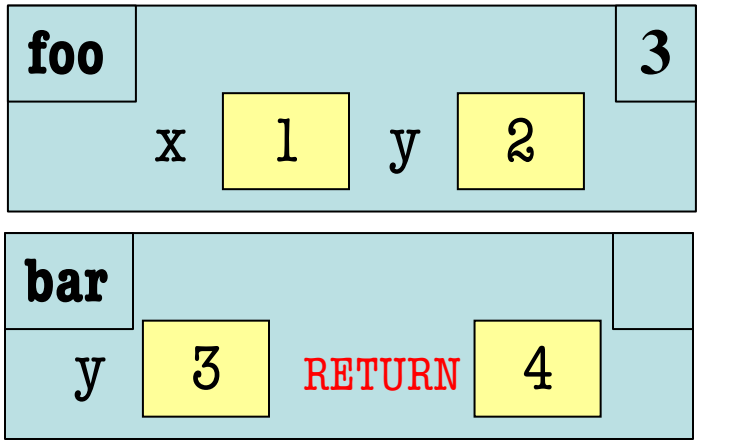
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return y+1
```

Function Call

```
>>> foo(1)
```

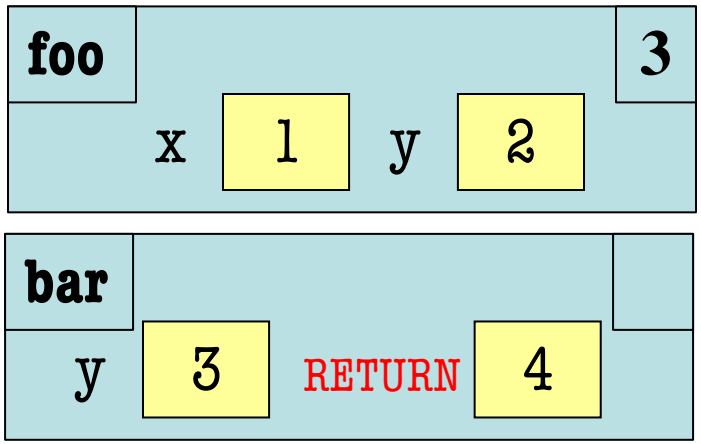
A:



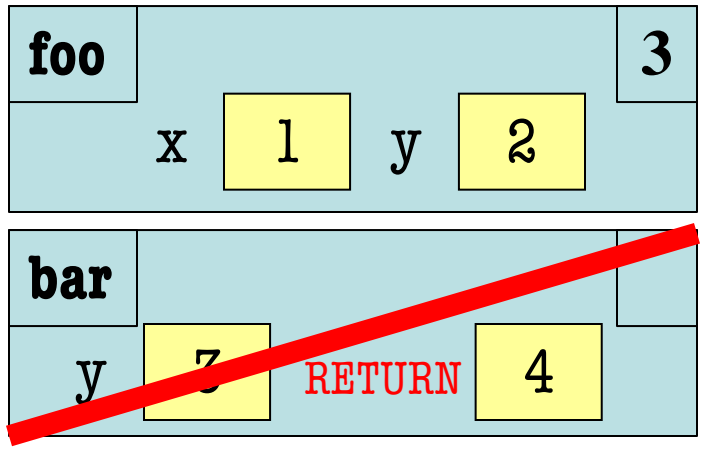
What is the **next step**?

Which One is Closest to Your Answer?

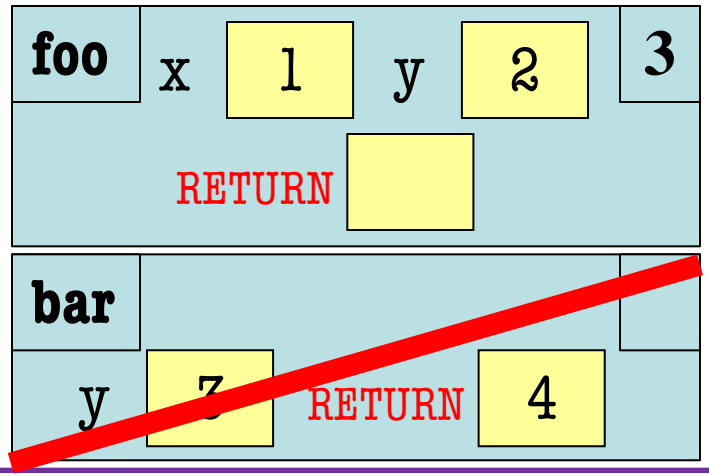
A:



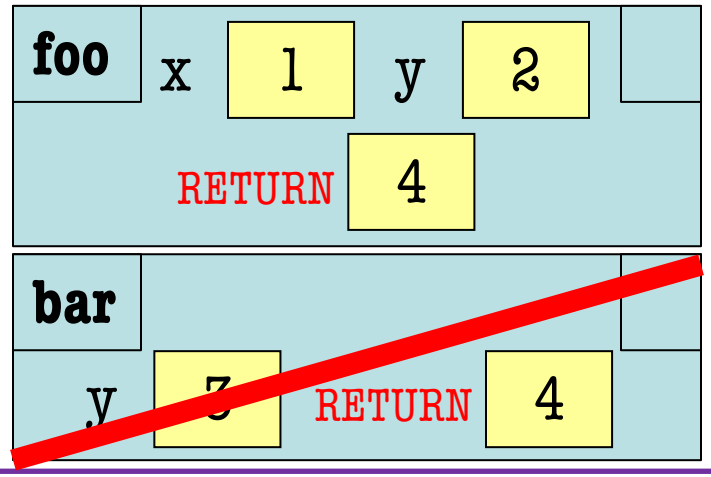
B:



C:



D:



Activity Time: The Call Stack

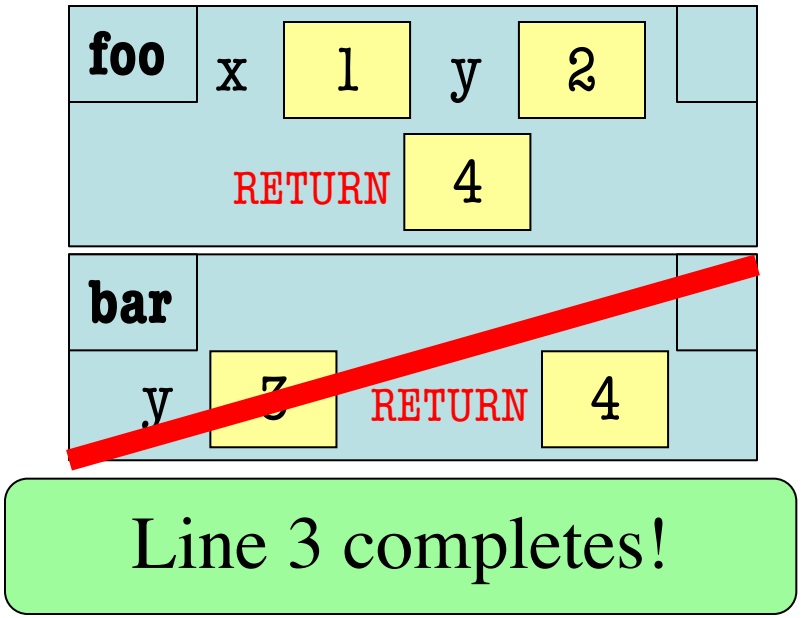
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return y+1
```

Function Call

```
>>> foo(1)
```

D:



Activity Time: The Call Stack

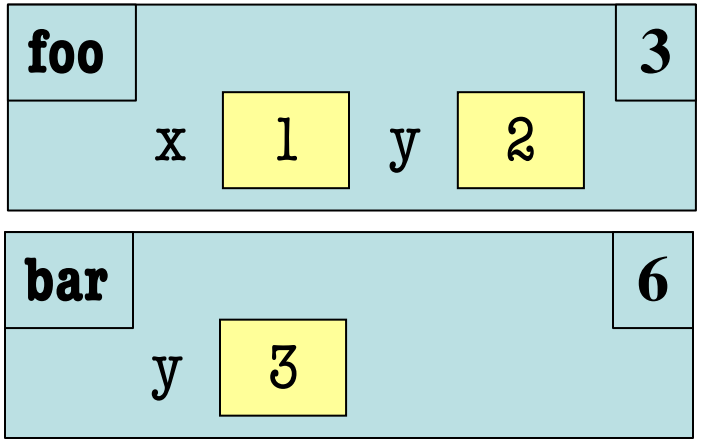
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return foo(y-1)
```

Function Call

```
>>> foo(1)
```

Assume we are here:



Activity Time: The Call Stack

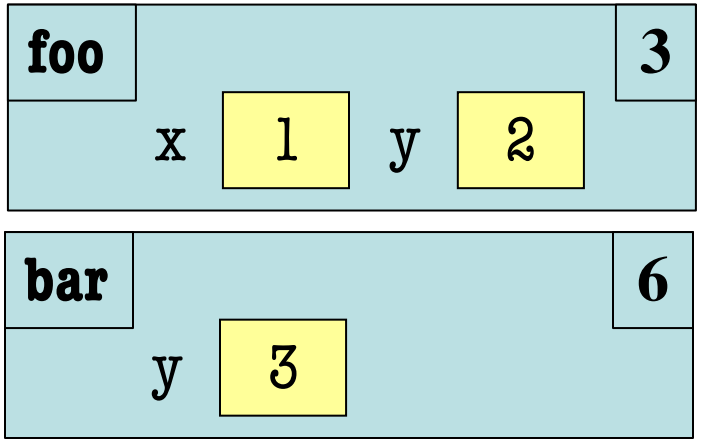
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return foo(y-1)
```

Function Call

```
>>> foo(1)
```

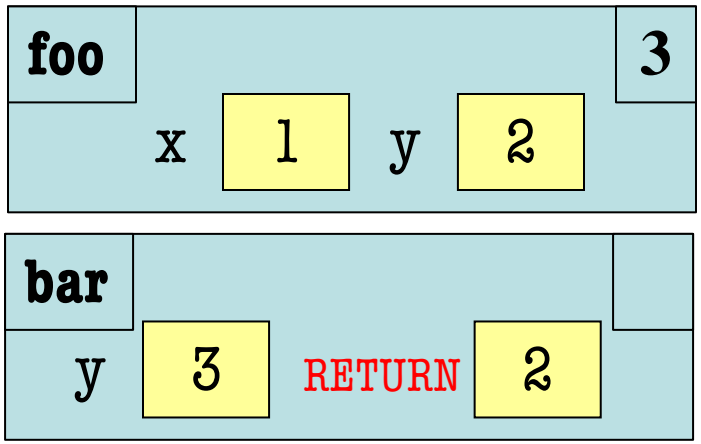
Assume we are here:



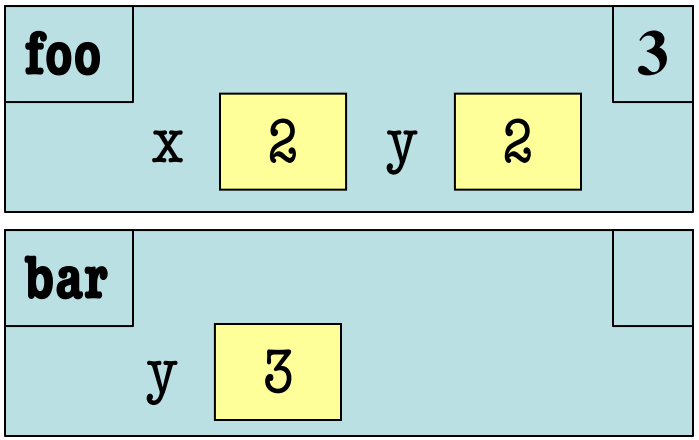
What is the **next step**?

Which One is Closest to Your Answer?

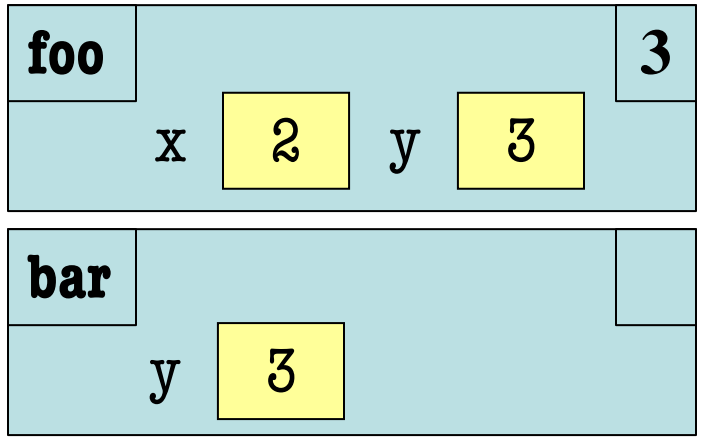
A:



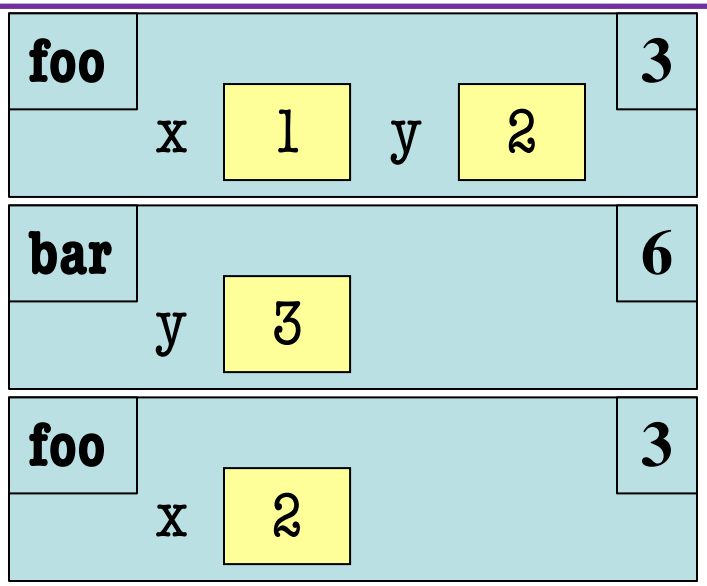
B:



C:



D:



Activity Time: The Call Stack

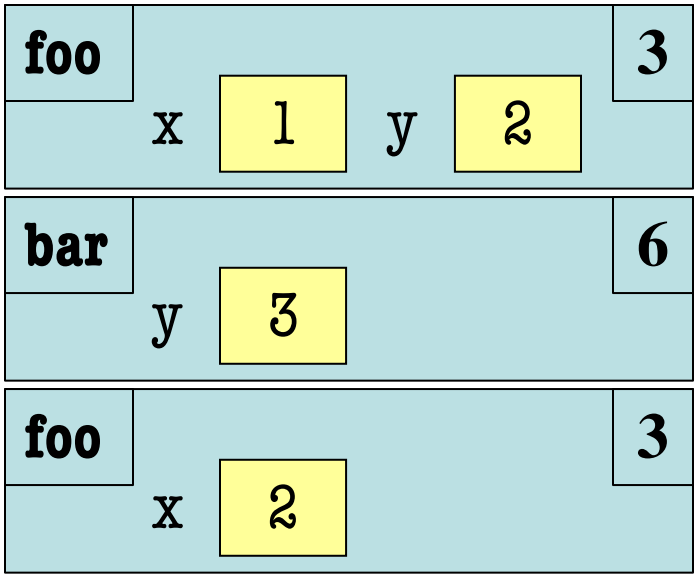
Function Definitions

```
1 def foo(x):
2     y = x+1
3     return bar(y+1)
4
5 def bar(y):
6     return foo(y-1)
```

Function Call

```
>>> foo(1)
```

D:



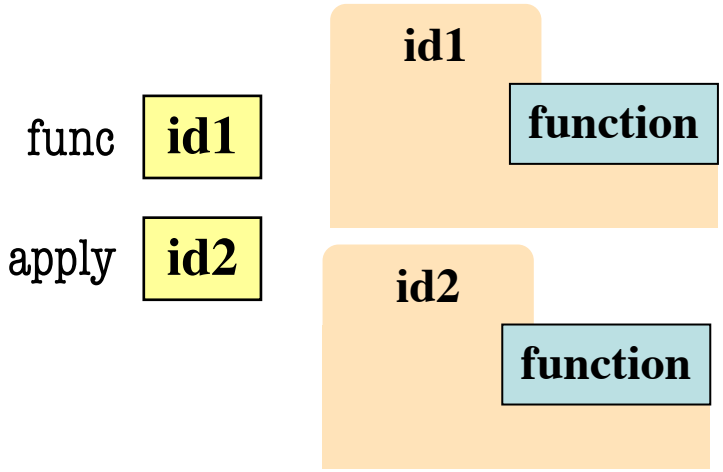
Will see this later on.

Activity Time: Stack + Heap

Function Definitions

```
1 def func(x):
2     return x+1
3
4 def apply(x,f):
5     return f(x)
```

Global Space/The Heap



Activity Time: Stack + Heap

Function Definitions

```
1 def func(x):
2     return x+1
3
4 def apply(x,f):
5     return f(x)
```

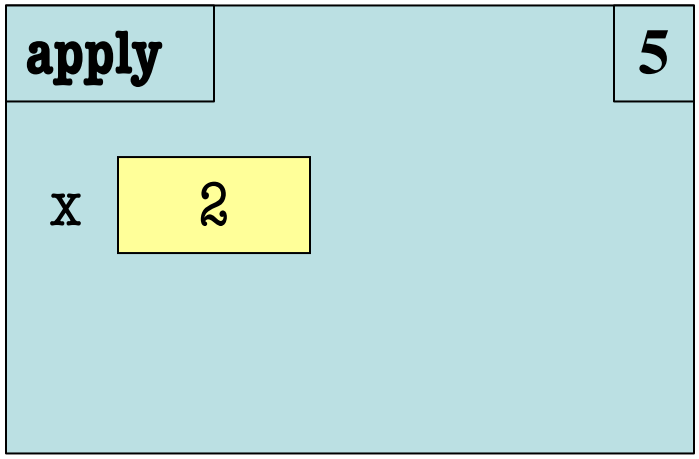
Function Call

```
>>> a = apply(2,func)
```

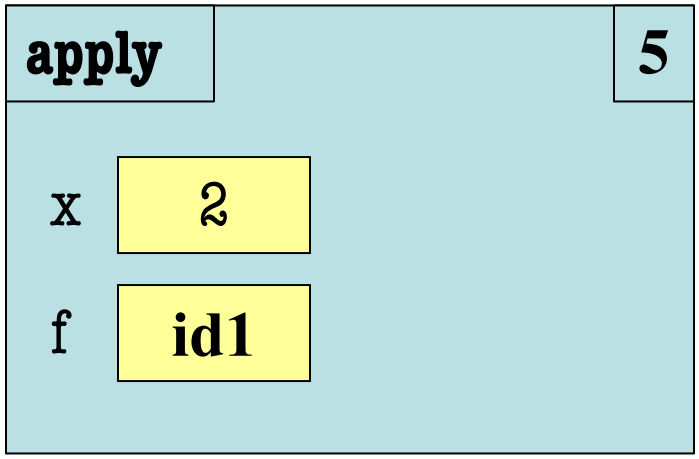
What does the call stack look like at the **start**?

Which One is Closest to Your Answer?

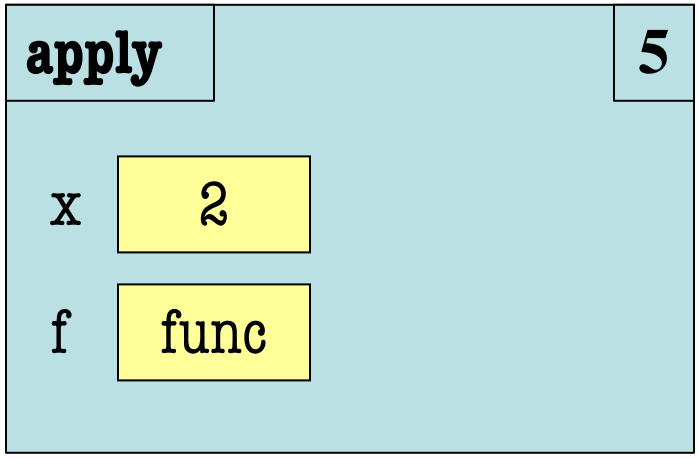
A:



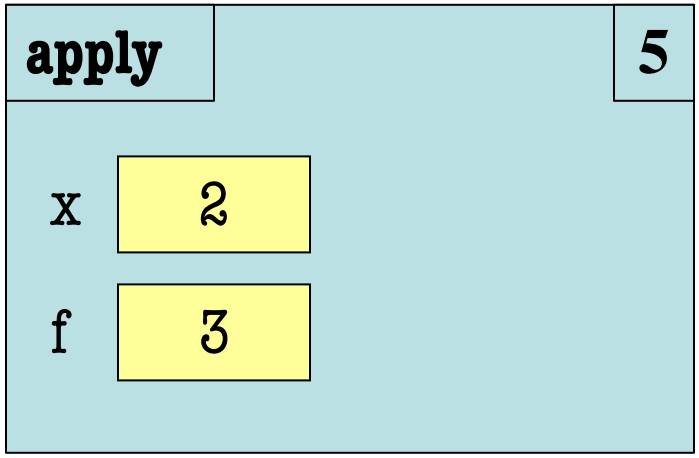
B:



C:



D:



Activity Time: Stack + Heap

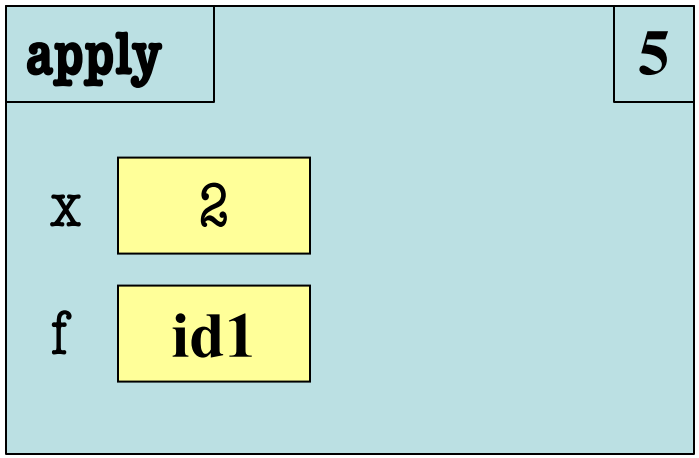
Function Definitions

```
1 def func(x):
2     return x+1
3
4 def apply(x,f):
5     return f(x)
```

Function Call

```
>>> a = apply(2,func)
```

B:



Questions?