

CSCI 121: Frames & Environments

Executing a script

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Executing a script

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

-7

6

-42

Executing a script

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

How does Python do this work?

How does it track variables? Where does it store them?

How are they organized?

Executing a script

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
...
```

Questions:

How does Python do this work?

How does it track variables? Where does it store them?

How are they organized?

Answer:

Python uses *variable frames*

It organizes an *execution environment* full of them.

Rules for global frame

When a script is executed:

→ A *global frame* is constructed to hold script variables.

→ A *slot* is added for each *newly assigned* variable.

```
x = 35
```

→ The variable's slot stores its current value.

→ Python checks the slot for that variable's value.

```
print("The value is " + str(x) + ".")
```

→ A variable reassignment updates that slot's contents.

```
x = x + 1
```

Activity within the global frame

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

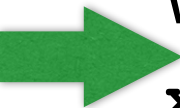
Activity within

#0

<global>

frame

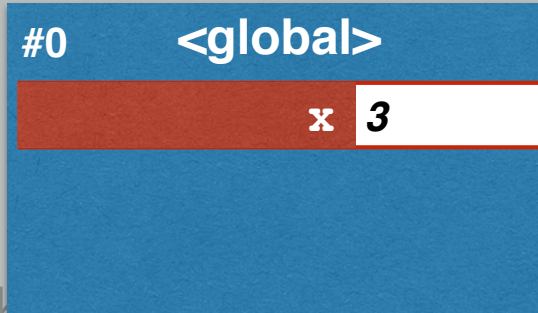
What does this script do when executed?



```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```


Output to the console:

Activity within



frame

What does this script do when executed?



```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

Activity within

#0	<global>
x	3
y	6

frame

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

Activity within

#0	<global>	
	x	-7
	y	6

frame

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

Activity within

#0	<global>	
	x	-7
	y	6

frame

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

-7

Activity within

#0	<global>	
	x	-7
	y	6

frame

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

-7
6

Activity within

frame

#0	<global>	
	x	-7
	y	6
	z	-42

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```

Output to the console:

-7
6

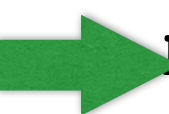
Activity within

frame

#0	<global>	
	x	-7
	y	6
	z	-42

What does this script do when executed?

```
x = 3
y = x + 3
x = x - 10
print(x)
print(y)
z = x * y
print(z)
```



Output to the console:

```
-7
6
-42
```

What about functions?

What happens when this script is executed?

```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```

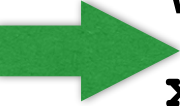

Rules for defining functions

Functions are treated like values, but in a special way.

- ➔ A *def* statement is like an assignment statement.
- ➔ A slot is created for that function.
- ➔ Its name refers to a new *function object*, constructed for that definition.
- ➔ Evaluation of *lambda* also constructs a new function object.
- ➔ A function object holds info about the definition so its code can be executed later (when it is called).

Function variables

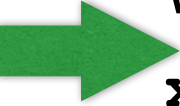
What happens when this script is executed?



```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```

Function

What happens when this so




```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```

#0

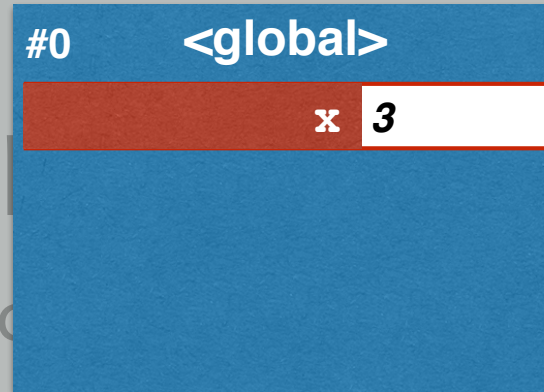
<global>

Function

What happens when this sc



```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```



Function

What happens when this sc

```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```

#0	<global>
x	3
y	6

Function

What happens when this script

```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```

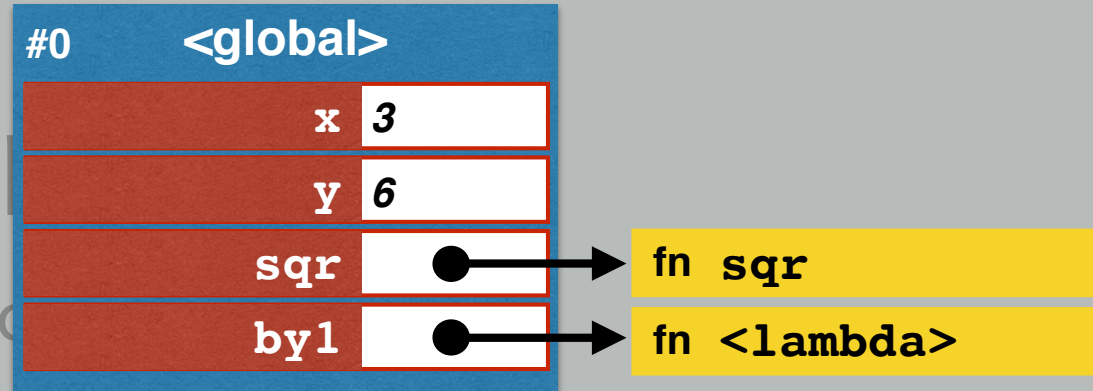
#0 <global>	
x	3
y	6
sqr	● →

fn sqr

Function

What happens when this script is executed?

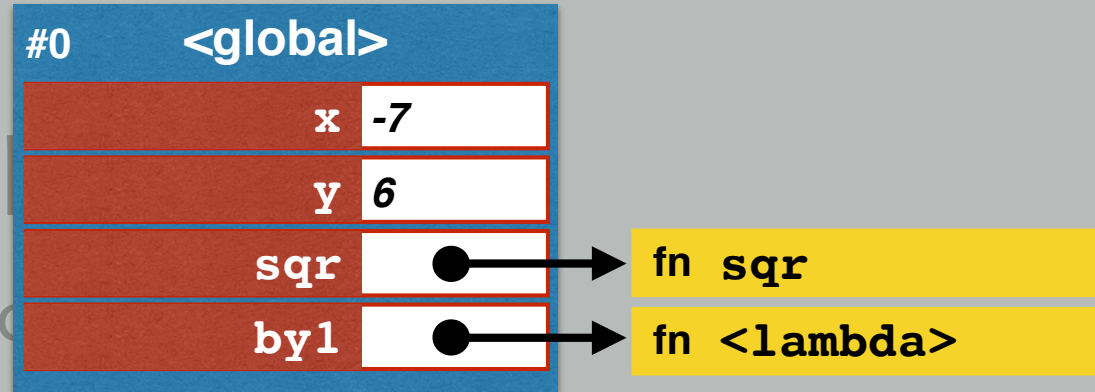
```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```



Function

What happens when this script is executed?

```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
```



What about function calls?

What happens when this script is executed?

```
x = 3
y = x + 3
def sqr(x):
    z = x * x
    return z
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```

Rules for executing functions

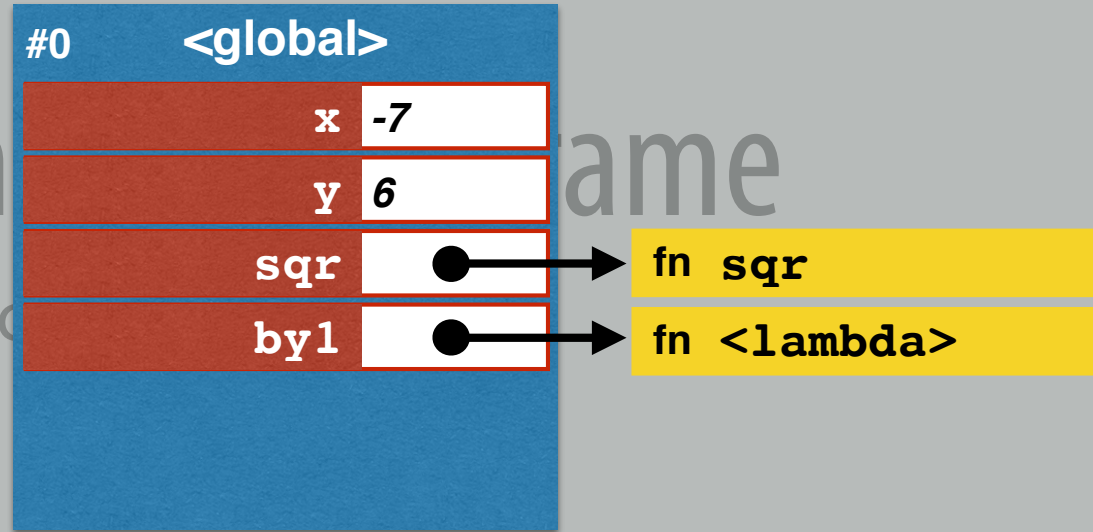
When a function is called:

- ➔ A new *local frame* is created when a function is called.
- ➔ It holds the *local variables* for that function.
- ➔ Slots are added for each *parameter variable*.
- ➔ They are set to the *values passed to that function*.
- ➔ Assignments add slots to that local frame.
- ➔ Python checks local slots for local variables' values.
- ➔ Reassigning updates a local slot's contents.

Activity with same

What happens when this script

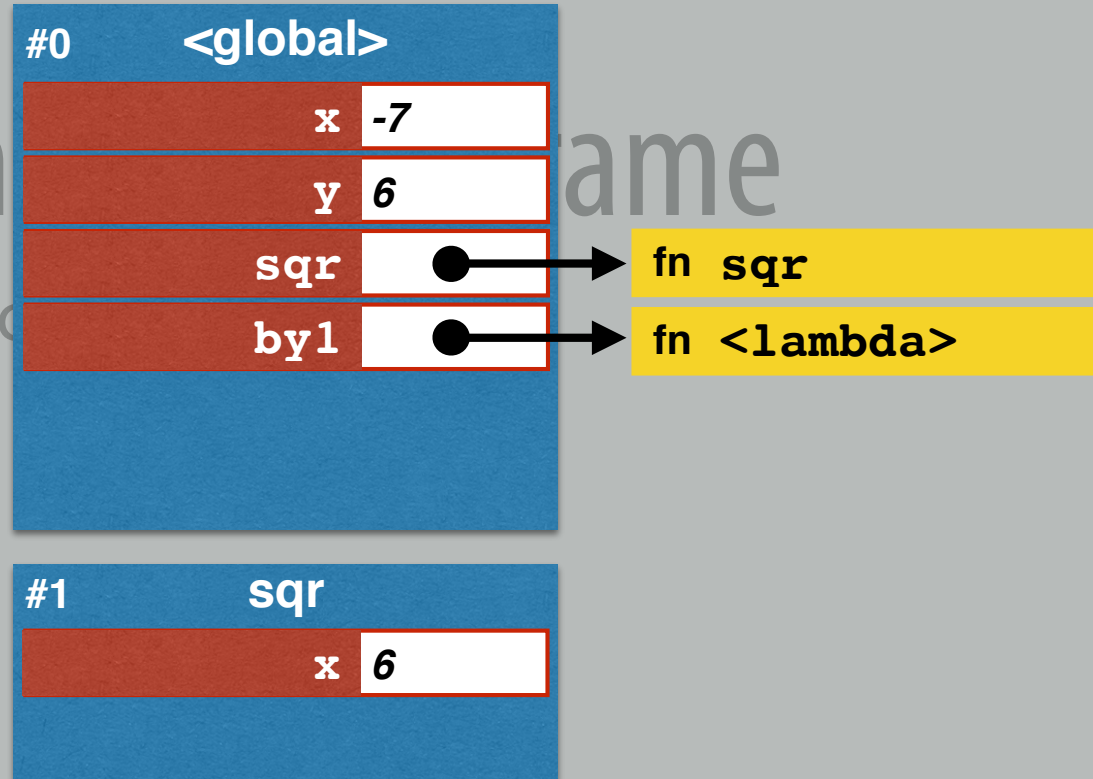
```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```



Activity with name

What happens when this script

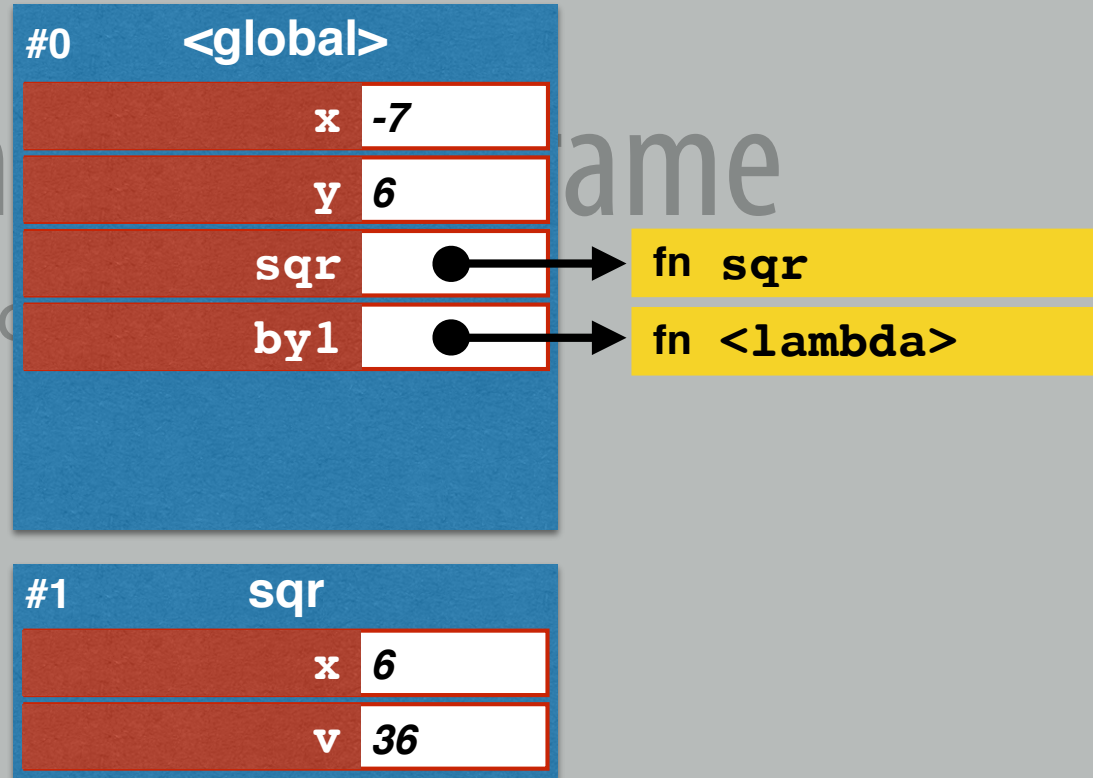
```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```



Activity with name

What happens when this script

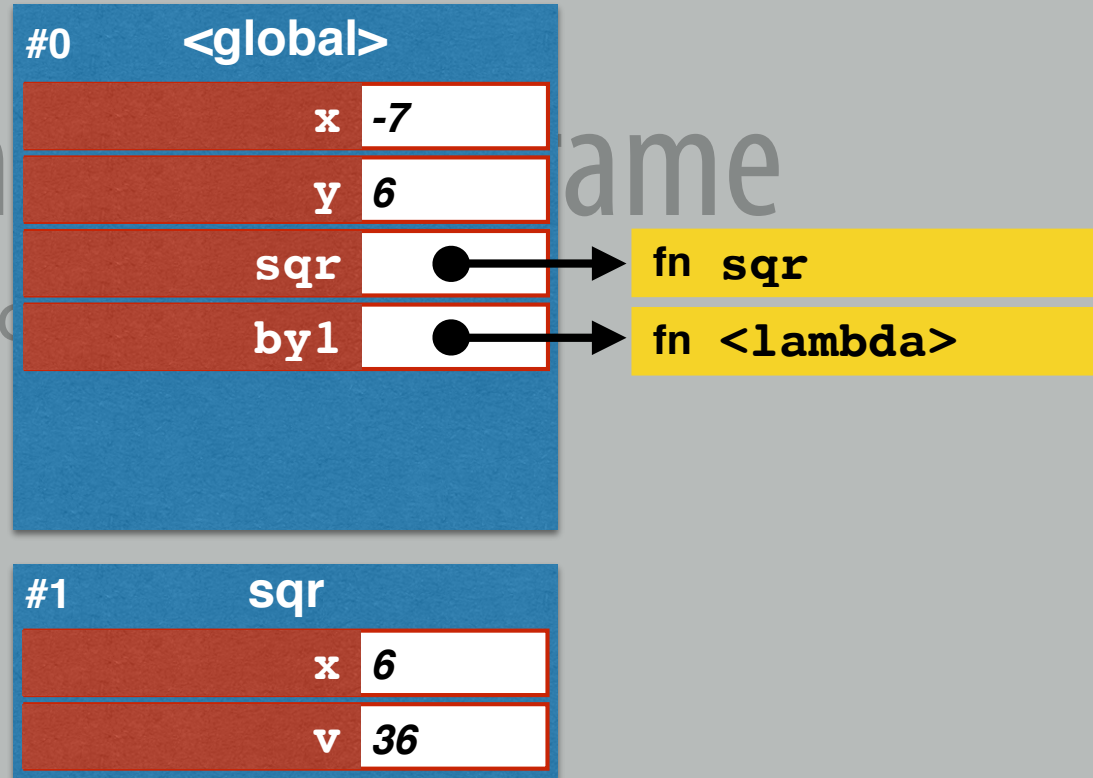
```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```



Activity with name

What happens when this script

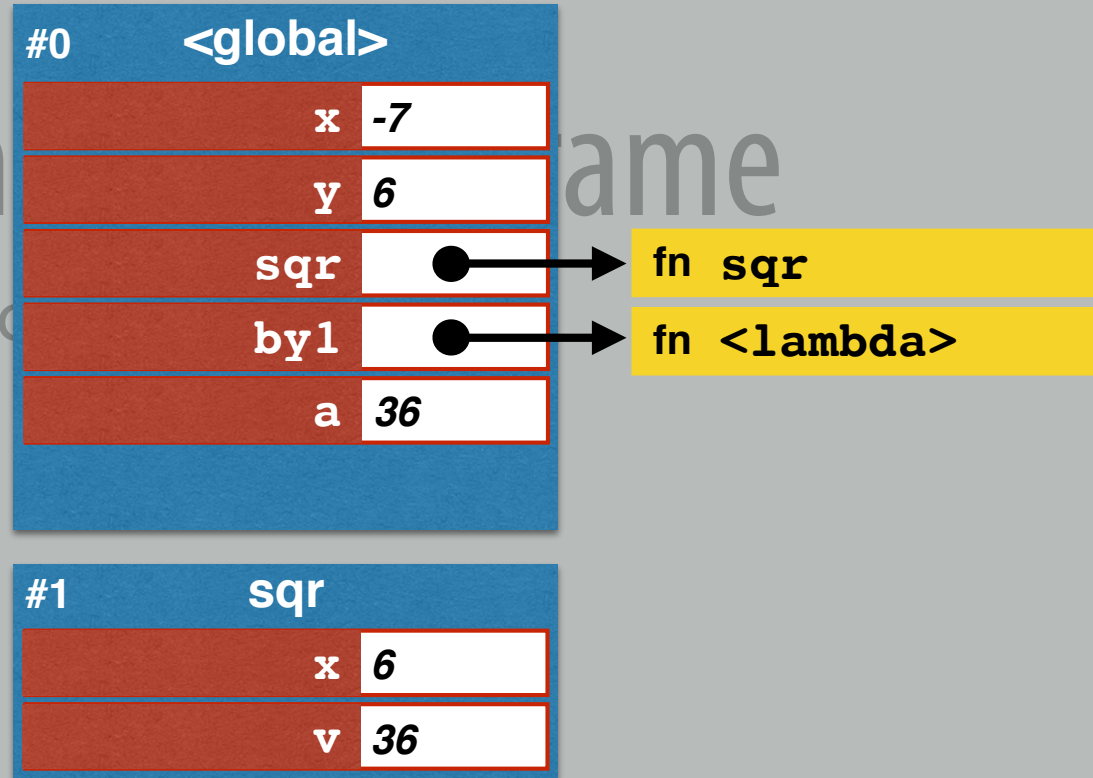
```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```



Activity with name

What happens when this script

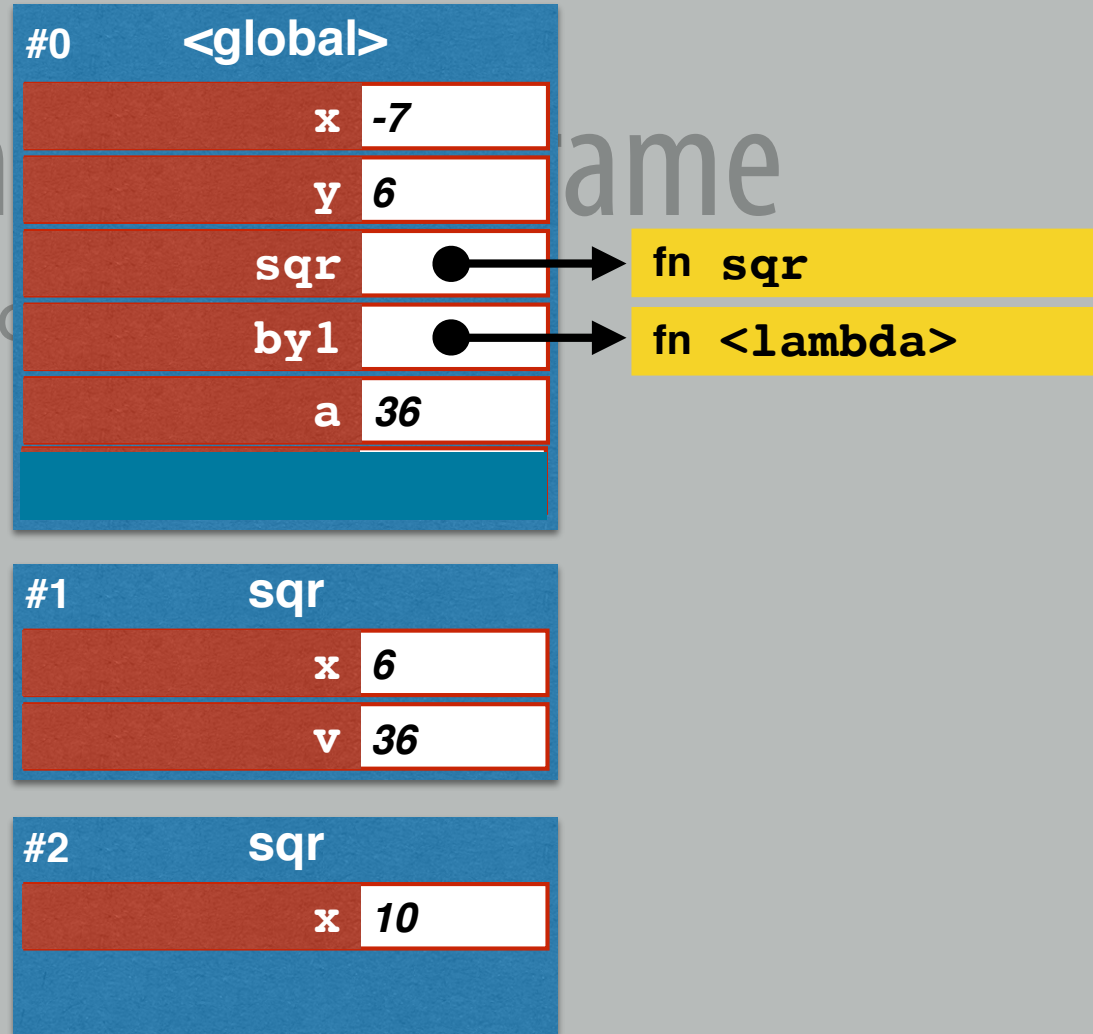
```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```



Activity with name

What happens when this script is executed?

```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```



Activity with name

What happens when this script is executed?

```
x = 3
y = x + 3
def sqr(x):
    v = x * x
    return v
by1 = (lambda x: x+1)
x = x - 10
a = sqr(y)
b = sqr(10)
```

#0 <global>	
x	-7
y	6
sqr	● →
by1	● →
a	36

fn **sqr**

fn **<lambda>**

#1 sqr	
x	6
v	36

#2 sqr	
x	10
v	100