Environment Diagrams

Slides adapted from Berkeley CS61a

What are Environment Diagrams?

- A visual tool to keep track of bindings & state of a computer program

- In this class, we use Python as our primary language
  - The diagrams we teach can be applied to similar languages

Why do we use Environment Diagrams?

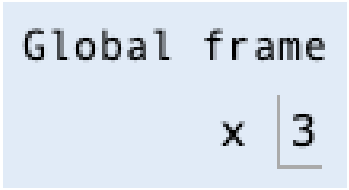
- Environment Diagrams are conceptual
  - understand *why* programs work the way they do
  - confidently predict how a program will behave
- Environment Diagrams are helpful for debugging
  - When you're really stuck,

    diagramming code > staring at lines of code
- Environment Diagrams will be used in future courses
  - CS 61C (Machine Structures)
  - CS 164 (Programming Languages and Compilers)

What do we've seen so far

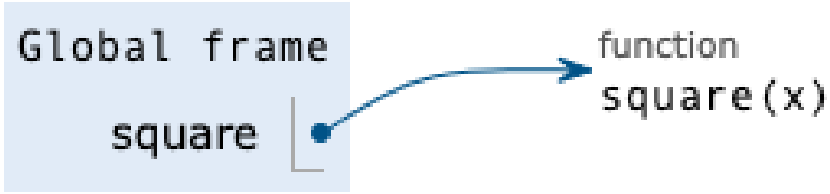
Assignment Statements

```
x = 1
x = x + x + x
```



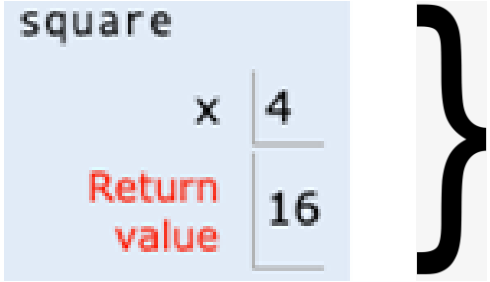
Def Statements

```
def square(x):
    return x * x
```

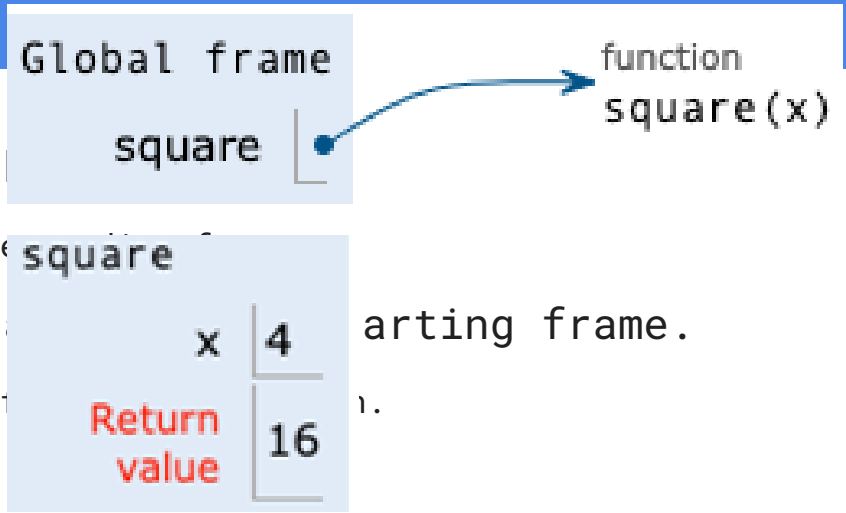


Call Expressions

```
square(4)
```



Terminology: Frames



A **frame** keeps track of varia

- Every call expression has a corre

Global, a.k.a. the global fr arting frame.

- It doesn't correspond to a speci .

Parent frames

- The parent of a function is the frame in which it was defined.
- If you can't find a variable in the current frame, you check it's parent, and so on. If you can't find the variable, NameError

Demo

Check Your Understanding

Draw the environment diagram

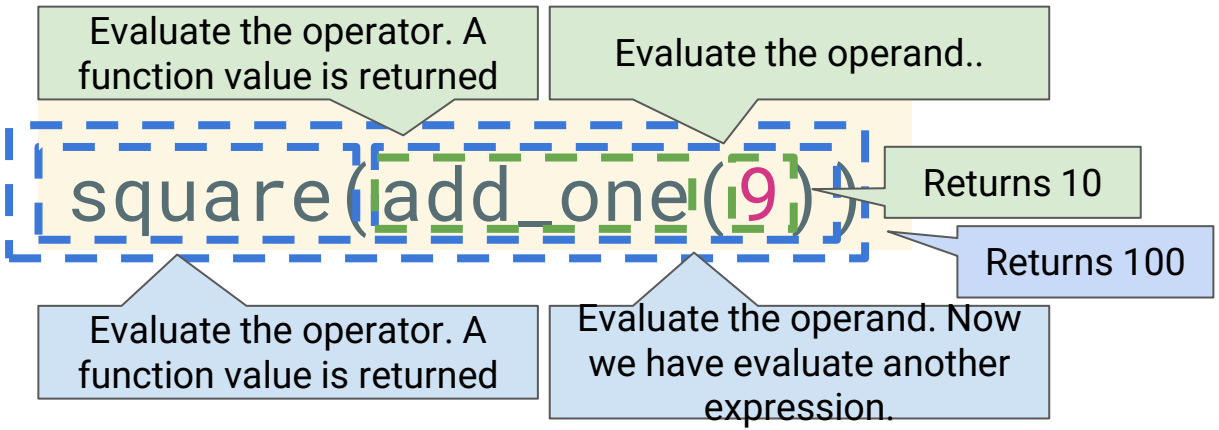
```
def square(x):
    return x * x

def sum_of_squares(x, y):
    return square(x) + square(y)

sum_of_squares(3, 4)
```

Review: Evaluation Order

Demo

Remember to evaluate the **operator**, then the **operand(s)**, then apply the **operator** onto the **operand(s)**.

```
def add_one(x):
    y = x + 1
    return y

def square(x):
    return x * x
```



What will the environment diagram look like? (**When are frames created?**)

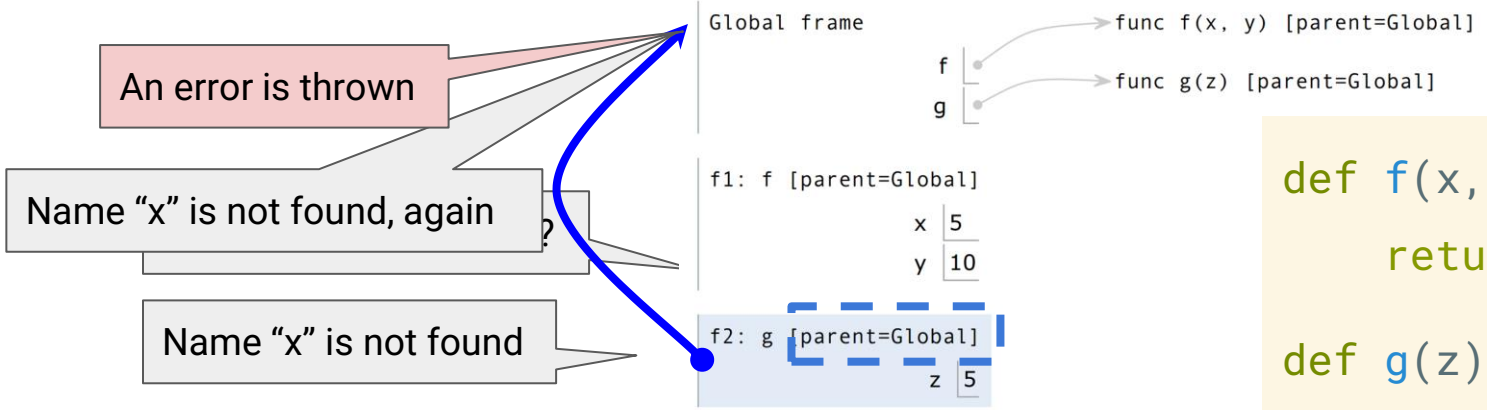
The environment diagram should reflect Python's evaluation.

Variable Lookup

Local Names

Variable Lookup:

- Lookup name in the current frame
- Lookup name in parent frame, its parent frame, etc..
- Stop at the global frame
- If not found, an error is thrown




```
def f(x, y):
    return g(x)

def g(z):
    return z + x

result = f(5, 10)
```

Important: There was no lookup done in f1 since the parent of f2 was Global

What happens here?

Evaluation vs Apply

Demo

```
def a_plus_bc(a, b, c):
    """
    >>> a_plus_bc(2, 3, 4) # 2 + 3 * 4
    14
    """
```

Apply operator a_plus_bc function to operand 4, 3, 81.

How many frames are created?
In what order?

```
a_plus_bc(square(2), 3, square(square(3)))
```

Apply operator square function to operand 2.

Apply operator square function to operand 9.

Apply operator square function to operand 3.

Break/Q&A

Lambda Expressions

Lambda Expressions

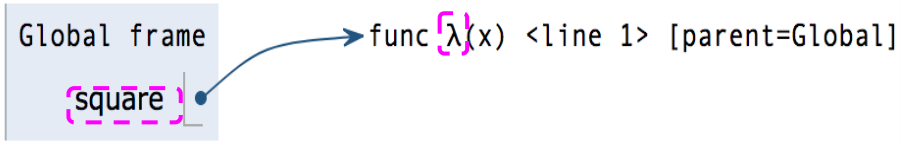
Expressions that evaluate to functions!

A function with parameter x that returns the value of x * x

```
>>> square = lambda x: x * x
>>> square
<function <lambda> ... >
>>> square(4)
16
>>> x = square(5)
>>> x
25
```

Lambda Expressions vs `def` Statements

```
square = lambda x: x * x
```




```
def square(x):
    return x * x
```

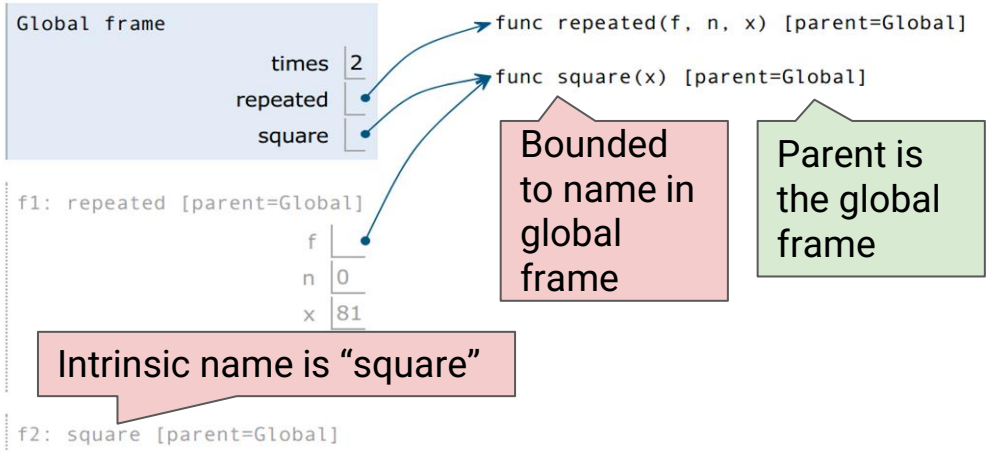
- Both create a function with the same behavior
- The parent frame of each function is the frame in which they were defined
- Both bind the function to the same name
- Only the `def` statement gives the function an intrinsic name

Environment Diagram

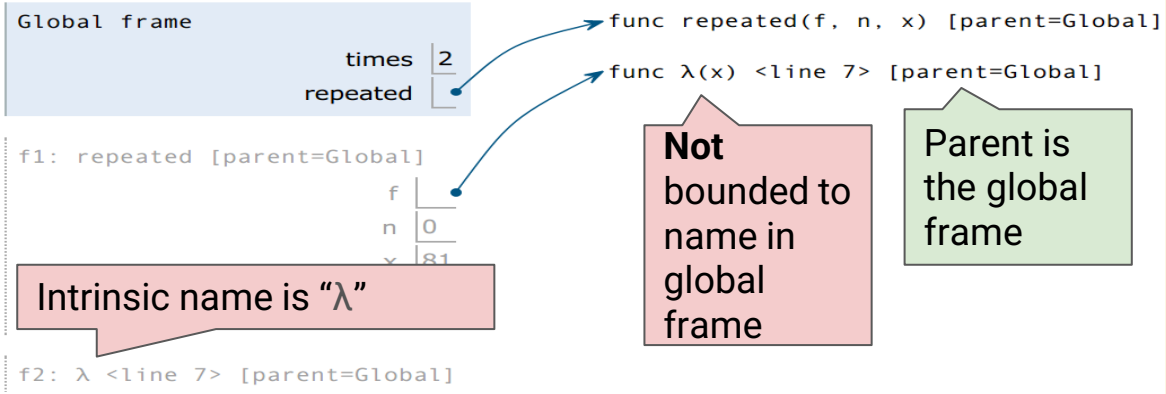
```
times = 2

def repeated(f, n, x):
        while n > 0:
                x = f(x)
                n -= 1
        return x

repeated(lambda x: x*x, times, 3)
```

```
repeated(square, times, 3)
```

Comparisons




```
times = 2
def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x
def square(x):
    return x * x
repeated(square, times, 3)
```




```
times = 2
def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x
repeated(lambda x: x * x, times, 3)
```

Higher Order Functions

Higher Order Functions

A function that ...

- takes a function as an argument value, and/or
- returns a function as a return value

You just saw this in

the previous example!

```
times = 2

def repeated(f, n, x):
        while n > 0:
                x = f(x)
        n -= 1
        return x

repeated(lambda x: x*x, times, 3)
```

Locally Defined Functions

```
>>> def make_greeter(name):
...                         return lambda greeting: print(greeting, name)
>>> greeter_function = make_greeter("Tiffany")
>>> greeter_function("Hey what's up, ")
```

Currying

```
>>> make_greeter("Tiffany")("Where's the party at, ")
```

Summary

- **Environment Diagrams** formalize the evaluation procedure for Python
  - Understanding them will help you think deeply about how the code that you are writing actually works
- **Lambda** functions are similar to functions defined with `def`, but are nameless
- A **Higher Order Function** is a function that either takes in functions as an argument (shown earlier) and/or returns a function as a return value (will see soon)