Computational Structures in Data Science

UC Berkeley EECS
Lecturer
Michael Ball

Lecture 4:
Lambda & Environments
Intro Recursion

Sept 30, 2019

http://cs88.org

Updates and Announcements

- Midterm 2 Weeks!
- Oct 14, 7-9pm
- Room: 155 Dwinelle
- Will release samples soon
- HW Party: Tues 8-10pm, “Woz” (430 Soda)
  - Lab 4 and HW4 Help
  - Lab 5 and HW5 out then, to get a start!
- Python Tutor, use https://tutor.cs61a.org

Computational Concepts Toolbox

- Data type: values, literals, operations,
  - e.g., int, float, string
- Expressions, Call expression
- Variables
- Assignment Statement
- Sequences: tuple, list
  - indexing
- Data structures
- Tuple assignment
- Call Expressions
- Function Definition Statement
- Conditional Statement
- Iteration:
  - data-driven (list comprehension)
  - control-driven (for statement)
  - while statement
- Higher Order Functions
  - Functions as Values
  - Functions with functions as argument
  - Assignment of function values
- Lambda - function valued expressions
- Recursion
  - Next week!

Universality

- Everything that can be computed, can be computed with what you know now.

- Poorly or Well

Today's Lecture

- Review
  - Higher Order Functions
  - Environments
- Lambda
- Some recursion + HOFs

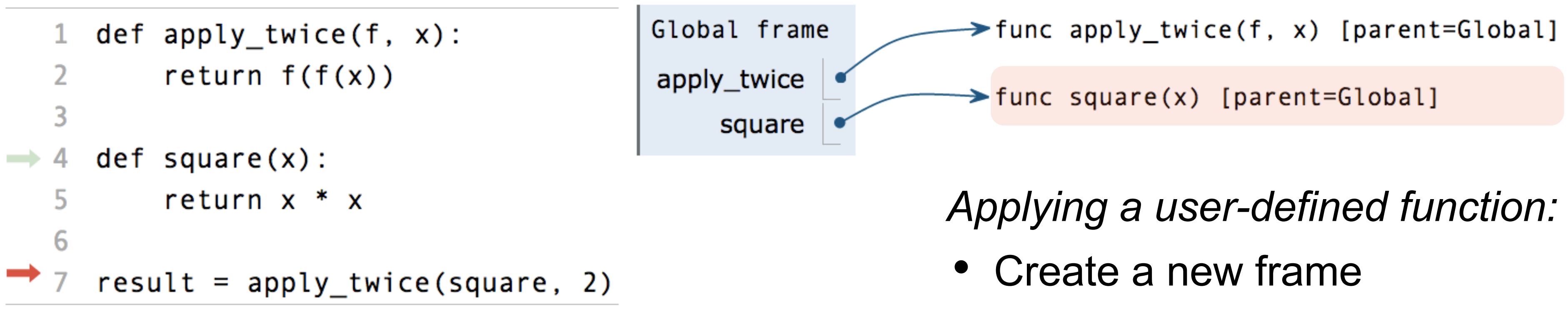
What would Python Display?

```
def summation(n, func): # Sum from 1 to N.
    total = 0
    for i in range(1, n + 1):
        total = total + func(i)
    return total

def cube(x):
    return x*x*x

def sum_cubes(n):
    return summation(n, cube)

sum_cubes(3)
```

- A) 6
- B) 9
- C) 27
- D) 36
- E) An Error Occurs

Python Tutor Link

Names can be Bound to Functional Arguments

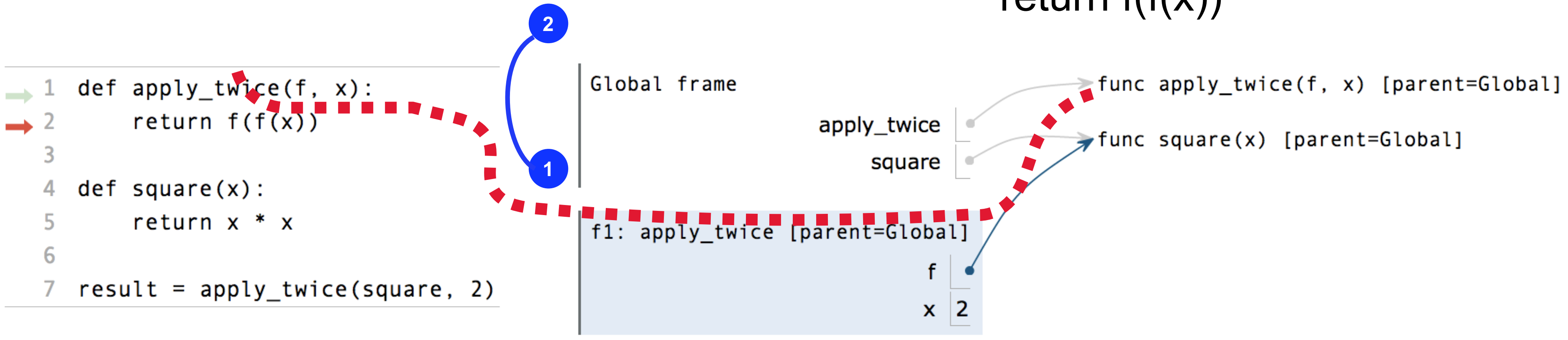


```
def apply_twice(f, x):
    return f(f(x))

def square(x):
    return x * x

result = apply_twice(square, 2)
```

```
Global frame
   apply_twice  ──>  func apply_twice(f, x) [parent=Global]
        square  ──>  func square(x) [parent=Global]
```

Applying a user-defined function:

- Create a new frame
- Bind formal parameters (f & x) to arguments
- Execute the body: return f(f(x))



```
def apply_twice(f, x):
    return f(f(x))

def square(x):
    return x * x

result = apply_twice(square, 2)
```

```
Global frame                          func apply_twice(f, x) [parent=Global]
          apply_twice
               square                 func square(x) [parent=Global]

f1: apply_twice [parent=Global]
                    f
                    x  2
```

pythontutor.com/composingprograms.html#code=def%20apply_twice%28f,%20x%29%3A%0A%20%20%20%20return%20f%28f%28x%29%29%0A%20%20%20%20%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%20%20%20%20%0Aresult%20%3D%20apply_twice%28square,%202%29&mode=display&origin=composingprograms.js&cumulative=true&py=3&rawInputLstJSON=[]&curInstr=0

Lambda Expressions

- **Function expression**
  - **“anonymous” function creation**
  - **Expression, not a statement, no return or any other statement**

`lambda` <arg or arg_tuple> : <expression using args>

```
add_one = lambda v : v + 1
```

```
def add_one(v):
    return v + 1
```

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

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

Important: No "return" keyword!

A function

with formal parameter x

that returns the value of "x * x"

Must be a single expression

```
>>> square(4)
16
```

Lambda expressions are not common in Python, but important in general

Lambdas

```
>>> def inc_maker(i):
...     return lambda x:x+i
...
>>> inc_maker(3)
<function inc_maker.<locals>.<lambda> at 0x10073c510>

>>> inc_maker(3)(4)
7
>>> map(lambda x:x*x, [1,2,3,4])
<map object at 0x1020950b8>

>>> list(map(lambda x:x*x, [1,2,3,4]))
[1, 4, 9, 16]
>>>
```

What would Python Display?

```
high_ord_func = lambda x, func: x + func(x)

high_ord_func(2, lambda x: x + 3)
```

- **A) 5**
- **B) 7**
- **C) 8**
- **D) <function <lambda> at 0x10b859710>**
- **E) An Error Occurs**

Python Tutor Link

Demo

- Acronym
  - Filter
  - Map
  - Reduce
  - ‘The University of California at Berkeley’ → ‘UCB’