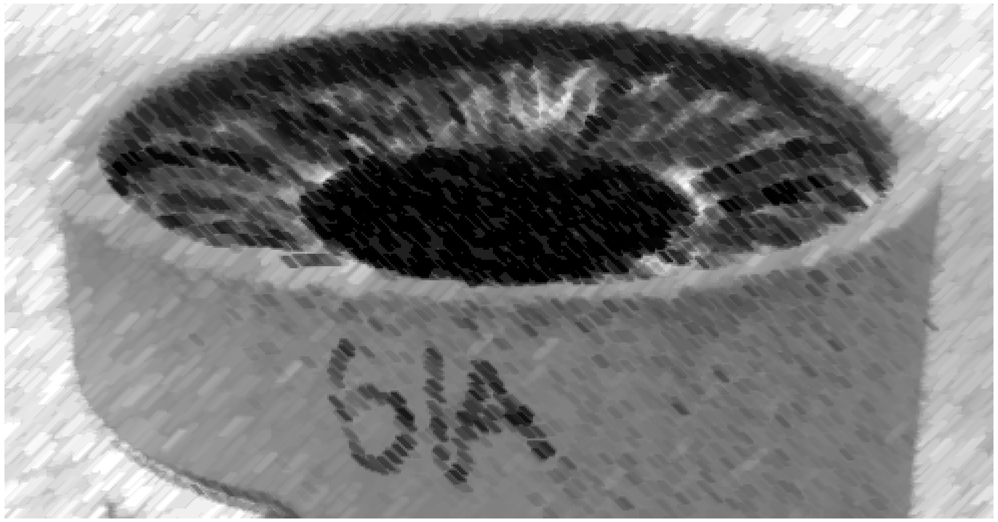


CS61A Lecture 5

Amir Kamil
UC Berkeley
February 1, 2013

Announcements

- Quiz today!
  - Only worth two points, so don't worry!
- Hog project
  - Get started early!
  - If you still don't have a partner (and want one), find one on Piazza
    - Use existing post; don't make a new one

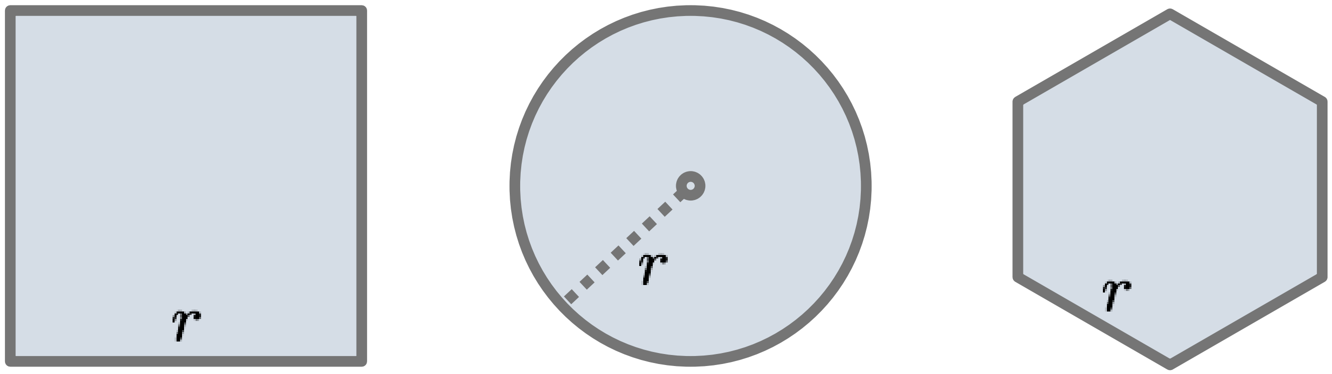
The Art of the Function

- Give each function exactly one job
- Don’t reapeat yourself (DRY).
- Don’t reapeat yourself (DRY).
- Define functions generally

Generalizing Patterns with Parameters

Regular geometric shapes relate length and area.

Shape:



Area: $\boxed{1} \cdot r^2$ $\boxed{\pi} \cdot r^2$ $\boxed{\frac{3\sqrt{3}}{2}} \cdot r^2$

Finding common structure allows for shared implementation

Generalizing Over Computational Processes

The common structure among functions may itself be a computational process, rather than a number.

$$\sum_{k=1}^{5} k = 1 + 2 + 3 + 4 + 5 \qquad = 15$$

$$\sum_{k=1}^{5} k^3 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 \qquad = 225$$

$$\sum_{k=1}^{5} \frac{8}{(4k-3)\cdot(4k-1)} = \frac{8}{3} + \frac{8}{35} + \frac{8}{99} + \frac{8}{195} + \frac{8}{323} \qquad = 3.04$$

Functions as Arguments

Function values can be passed as arguments

```
def cube(k):
    return pow(k, 3)

def summation(n, term):
    """Sum the first n terms of a sequence.

    >>> summation(5, cube)
    225
    """
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total
```

Function of a single argument (not called term)

A formal parameter that will be bound to a function

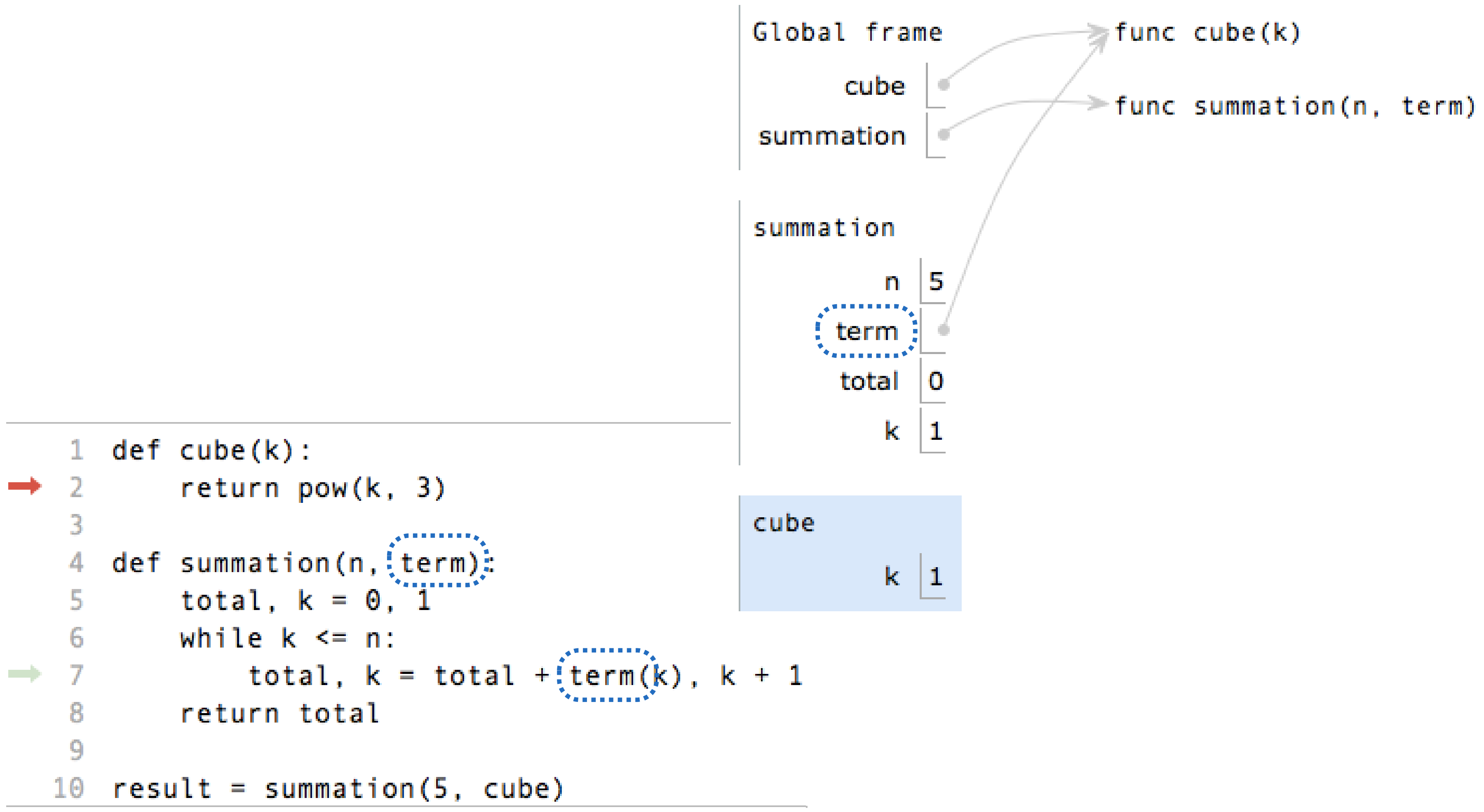
The cube function is passed as an argument value

The function bound to term gets called here

$0 + 1^3 + 2^3 + 3^3 + 4^3 + 5^5$

Function Values as Parameters

Parameters can be bound to function values



```
def cube(k):
    return pow(k, 3)

def summation(n, term):
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total

result = summation(5, cube)
```

Example: http://goo.gl/e4YBH

Functions as Return Values

Locally defined functions can be returned

They have access to the frame in which they are defined

A function that returns a function

```
def make_adder(n):
    """Return a function that adds n to its argument.

    >>> add_three = make_adder(3)
    >>> add_three(4)
    7
    """
    def adder(k):
        return add(n, k)
    return adder
```

The name add_three is bound to a function

A local def statement

Can refer to names in the enclosing function

Call Expressions as Operators

```
make_adder(1)(2)
```

```
make_adder(1)   (        2        )
```

Operator

Operand 0

An expression that evaluates to a function value

An expression that evaluates to any value

```
def make_adder(n):
    def adder(k):
        return add(n, k)
    return adder
```

Higher-Order Functions

Functions are first-class: they can be manipulated as values in Python

Higher-order function: a function that takes a function as an argument value or returns a function as a return value

Higher order functions:

- Express general methods of computation
- Remove repetition from programs
- Separate concerns among functions