Announcements

• Take-home quiz released Wednesday 9/11 at 1pm, due Thursday 9/12 at 11:59pm.
  ▪ http://inst.eecs.berkeley.edu/~cs61a/fa13/hw/quiz1.html
  ▪ 3 points; graded for correctness.
  ▪ Submit in the same way that you submit homework assignments.
  ▪ If you receive 0/3, you will need to talk to the course staff or be dropped.
  ▪ Open-computer: You can use the Python interpreter, watch course videos, and read the online text (http://composingprograms.com).
  ▪ No external resources: Please don't search for answers, talk to your classmates, etc.

• Homework 2 due Tuesday 9/17 at 5pm.

• Project 1 due Thursday 9/19 at 11:59pm.

• Solutions to homeworks: http://inst.eecs.berkeley.edu/~cs61a/fa13/hw/solutions
Office Hours: You Should Go!

You are not alone!

http://inst.eecs.berkeley.edu/~cs61a/fa13/staff.html
The Purpose of Higher-Order Functions

**Functions are first-class:** Functions can be manipulated as values in our programming language.

**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
Environments for Higher-Order Functions
Environments Enable Higher-Order Functions

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

Functions as arguments:

Our current evaluation rules handle that case already!

We'll discuss an example today

Functions as return values:

We need to extend our rules a little

Functions need to know where they were defined

Almost everything stays the same
Names can be Bound to Functional Arguments

• Functions are values.
• Names can refer to functions (just as they can refer to any values).
• Multiple names can all refer to the same function, even in different frames.

Example: http://goo.gl/mwVuIF
Discussion Question

What is the value of the final expression below?

```python
def repeat(f, x):
    while f(x) != x:
        x = f(x)
    return x

def g(y):
    return (y + 5) // 3

repeat(g, 5)
```

If you think there's an error

Example: http://goo.gl/EDiOIr
Environments for Nested Definitions

(Demo)
Environment Diagrams for Nested Def Statements

- Every user-defined function has a parent frame (often global).
- The parent of a function is the frame in which it was defined.
- Every local frame has a parent frame (often global).
- The parent of a frame is the parent of the function called.

Example:

```python
def make_adder(n):
    def adder(k):
        return k + n
    return adder

add_three = make_adder(3)
add_three(4)
```
An Environment is a Sequence of Frames

A local frame extends the environment that begins with its parent.

A two-frame environment

The global environment

A three-frame environment

Global frame

```plaintext
func make_adder(n)
    func adder(k) [parent=f1]
```

When a frame or function has no parent label

```
[parent=___]
```

then its parent is the global frame

```
name: make_adder
n: 3
adder: ...
Return value
```

```
adder [parent=f1]
k: 4
Return value: 7
```

We don't bother to label frames that aren't parents.
How to Draw an Environment Diagram

When a function is defined:

1. Create a **function value**:  `func <name>(<formal parameters>)`

2. If the **parent frame** of that function is not the global frame, add matching **labels** to the **parent frame** and the **function value** (such as `f1`, `f2`, or `f3`).

   ```
   f1: make_adder        func adder(k) [parent=f1]
   ```

3. Bind `<name>` to the **function value** in the first frame of the current environment.

When a function is called:

1. Add a **local frame**, titled with the `<name>` of the function being called.

2. If the function has a parent label, copy it to the **local frame**: `[parent=<label>]`

3. Bind the `<formal parameters>` to the arguments in the **local frame**.

4. Execute the body of the function in the environment that starts with the **local frame**.
Local Names

(Demo)
Local Names are *not* Visible to Other (Non-Nested) Functions

```python
1  def f(x, y):
2      return g(x)
3
4  def g(a):
5      return a + y
6
7  result = f(1, 2)
```

- An environment is a sequence of frames.
- The environment created by calling a top-level function (no def within def) consists of one local frame, followed by the global frame.

Example: [http://goo.gl/b6WvUc](http://goo.gl/b6WvUc)
Function Composition
def square(x):
    return x * x

def make_adder(n):
    def adder(k):
        return k + n
    return adder

def compose1(f, g):
    def h(x):
        return f(g(x))
    return h

compose1(square, make_adder(2))(3)

Return value of make_adder is an argument to compose1
The Game of Hog