61A Lecture 5

Wednesday, September 5
Office Hours: You Should Go!

You are not alone!

http://inst.eecs.berkeley.edu/~cs61a/fa12/staff.html
The Game of Hog

- **Expected score**
  - Number of dice rolled: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  - Expected scores: 3.5, 5.9, 7.4, 8.2, 8.6, 8.7, 8.5, 8.2, 7.8, 7.3

- **Chance of scoring 10+**
  - Number of dice rolled: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  - Chance of scoring 10+: 16%, 16%, 48%, 48%, 40%, 33%, 28%, 23%, 19%, 16%
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 environment model handles that already!

We'll discuss an example today

**Functions as return values:**

We need to extend our model a little

Functions need to know where they were defined

Almost everything stays the same (demo)
Names Bound to Functional Arguments

```python
1  def apply_twice(f, x):
2      return f(f(x))
3
4  def square(x):
5      return x * x
6
7  result = apply_twice(square, 2)
```

Example: [http://goo.gl/Gbtc5](http://goo.gl/Gbtc5)
Non-Nested Functions Calls Have One Local Frame

An environment is a sequence of frames.

An environment for a non-nested function (no def within def) consists of one local frame, followed by the global frame.

Example: [Link to Example](http://goo.gl/tgT5H)
• Every user-defined function has a *parent frame*
• The parent of a function is the frame in which it was *defined*
• Every local frame has a *parent frame*
• The parent of a frame is the parent of the function *called*

Example: [http://goo.gl/L9G2q](http://goo.gl/L9G2q)
The Structure of Environments

A frame extends the environment that begins with its parent

The global environment: the environment with only the global frame

When a frame or function has no label [parent=___], then its parent is always the global frame

A frame extends the environment that begins with its parent
How to Draw an Environment Diagram

When defining a function:

1. Create a function value with signature `<name>(<formal parameters>)`

2. For nested definitions, label the parent as the first frame of the current environment

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

When calling a function:

1. Add a local frame labeled with the `<name>` of the function

2. If the function has a parent label, copy it to this frame

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

4. Execute the body of the function in the environment that starts with this frame
The Environment for Function Composition

```python
1 def square(x):
2     return x * x
3
4 def make_adder(n):
5     def adder(k):
6         return k + n
7     return adder
8
9 def compose1(f, g):
10    def h(x):
11        return f(g(x))
12    return h
13
14 compose1(square, make_adder(2))(3)
```

Return value of `make_adder` is an argument to `compose1`

Example: [http://goo.gl/2IuE0](http://goo.gl/2IuE0)
Lambda Expressions

```python
>>> ten = 10
An expression: this one evaluates to a number

>>> square = x * x
Also an expression: evaluates to a function

>>> square = lambda x: x * x
A function with formal parameter x
and body "return x * x"

>>> square(4)
16
Must be a single expression
```

Lambda expressions are rare in Python, but important in general
More Higher-Order Function Examples

(Demo)