Office Hours: You Should Go!

You are not alone!
Office Hours: You Should Go!

You are not alone!
Office Hours: You Should Go!

You are not alone!

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

Expected score vs. Number of dice rolled:
- 1 dice: 3.5
- 2 dice: 5.9
- 3 dice: 7.4
- 4 dice: 8.2
- 5 dice: 8.6
- 6 dice: 8.7
- 7 dice: 8.5
- 8 dice: 8.2
- 9 dice: 7.8
- 10 dice: 7.3
The Game of Hog

### Expected Score

<table>
<thead>
<tr>
<th>Number of Dice Rolled</th>
<th>Expected Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.5</td>
</tr>
<tr>
<td>2</td>
<td>5.9</td>
</tr>
<tr>
<td>3</td>
<td>7.4</td>
</tr>
<tr>
<td>4</td>
<td>8.2</td>
</tr>
<tr>
<td>5</td>
<td>8.6</td>
</tr>
<tr>
<td>6</td>
<td>8.7</td>
</tr>
<tr>
<td>7</td>
<td>8.5</td>
</tr>
<tr>
<td>8</td>
<td>8.2</td>
</tr>
<tr>
<td>9</td>
<td>7.8</td>
</tr>
<tr>
<td>10</td>
<td>7.3</td>
</tr>
</tbody>
</table>

### Chance of Scoring 10+

<table>
<thead>
<tr>
<th>Number of Dice Rolled</th>
<th>Chance of Scoring 10+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16%</td>
</tr>
<tr>
<td>2</td>
<td>19%</td>
</tr>
<tr>
<td>3</td>
<td>23%</td>
</tr>
<tr>
<td>4</td>
<td>28%</td>
</tr>
<tr>
<td>5</td>
<td>33%</td>
</tr>
<tr>
<td>6</td>
<td>40%</td>
</tr>
<tr>
<td>7</td>
<td>48%</td>
</tr>
<tr>
<td>8</td>
<td>48%</td>
</tr>
<tr>
<td>9</td>
<td>48%</td>
</tr>
<tr>
<td>10</td>
<td>48%</td>
</tr>
</tbody>
</table>
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
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:**
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!
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
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:**
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
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
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
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

Example: http://goo.gl/Gbtc5
Names Bound to Functional Arguments

```python
1  def apply_twice(f, x):
2      return f(f(x))

4  def square(x):
5      return x * x

7  result = apply_twice(square, 2)
```

Example: [http://goo.gl/Gbtc5](http://goo.gl/Gbtc5)
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)
Names Bound to Functional Arguments

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

Example: http://goo.gl/tgT5H
Non-Nested Functions Calls Have One Local Frame

```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)
```

Example: http://goo.gl/tgT5H
Non-Nested Functions Calls Have One Local Frame

```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)
```

Example: [http://goo.gl/tgT5H](http://goo.gl/tgT5H)
Non-Nested Functions Calls Have One Local Frame

Example:

```python
def f(x, y):
    return g(x)
def g(a):
    return a + y
result = f(1, 2)
```

Example: [http://goo.gl/tgT5H](http://goo.gl/tgT5H)
Non-Nested Functions Calls Have One Local Frame

Example: http://goo.gl/tgT5H
Non-Nested Functions Calls Have One Local Frame

Example: http://goo.gl/tgT5H
Non-Nested Functions Calls Have One Local Frame

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

Example: [http://goo.gl/tgT5H](http://goo.gl/tgT5H)
Non-Nested Functions Calls Have One Local Frame

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

Example: [http://goo.gl/tgT5H](http://goo.gl/tgT5H)
Non-Nested Functions Calls Have One Local Frame

An environment is a sequence of frames

Example: http://goo.gl/tgT5H
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: http://goo.gl/tgT5H
Environment Diagrams for Nested Def Statements

```python
1 def make_adder(n):
2     def adder(k):
3         return k + n
4     return adder
5
6 add_three = make_adder(3)
7 result = add_three(4)
```

Example: http://goo.gl/L9G2q
Environment Diagrams for Nested Def Statements

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

add_three = make_adder(3)
result = add_three(4)
```

Example: [http://goo.gl/L9G2q](http://goo.gl/L9G2q)
Environment Diagrams for Nested Def Statements

Nested def

1. def make_adder(n):
2.     def adder(k):
3.         return k + n
4.     return adder
5.
6. add_three = make_adder(3)
7. result = add_three(4)

Example: http://goo.gl/L9G2q
Environment Diagrams for Nested Def Statements

Example: [http://goo.gl/L9G2q](http://goo.gl/L9G2q)
Environment Diagrams for Nested Def Statements

Example:  http://goo.gl/L9G2q
Environment Diagrams for Nested Def Statements

**Nested def**

```python
1. def make_adder(n):
2.     def adder(k):
3.         return k + n
4.     return adder
5. 6. add_three = make_adder(3)
7. result = add_three(4)
```

Example: [http://goo.gl/L9G2q](http://goo.gl/L9G2q)
Environment Diagrams for Nested Def Statements

Example: [http://goo.gl/L9G2q](http://goo.gl/L9G2q)
Environment Diagrams for Nested Def Statements

Example: http://goo.gl/L9G2q
Every user-defined function has a parent frame

Example: http://goo.gl/L9G2q
• Every user-defined function has a parent frame
• The parent of a function is the frame in which it was defined

Example: http://goo.gl/L9G2q
Environment Diagrams for Nested Def Statements

- 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.

Example: [http://goo.gl/L9G2q](http://goo.gl/L9G2q)
• 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
• 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
Environment Diagrams for Nested Def Statements

- 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)
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
The Structure of Environments

```
Global frame
    make_adder
    add_three

f1: make_adder
    n 3
    adder
    Return value

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

```python
func make_adder(n)
    func adder(k) [parent=f1]
```
The Structure of Environments

Global frame
- `make_adder`
- `add_three`

f1: `make_adder`
- n
- adder
- Return value

adder [parent=f1]
- k
- Return value
The Structure of Environments

Global frame
make_adder
  add_three

f1: make_adder
  n
  adder
Return value

adder [parent=f1]
  k
  Return value

func make_adder(n)
  func adder(k) [parent=f1]
The Structure of Environments
A frame extends the environment that begins with its parent.
A frame extends the environment that begins with its parent.
The Structure of Environments

A three-frame environment

A frame *extends* the environment that begins with its parent
The Structure of Environments

A frame extends the environment that begins with its parent.

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.
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.
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.
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 three-frame environment

A two-frame environment

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

When defining a function:
How to Draw an Environment Diagram

When defining a function:

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

Example: http://goo.gl/2IuE0
The Environment for Function Composition

```python
1 def square(x):
2     return x * x

4 def make_adder(n):
5     def adder(k):
6         return k + n
7     return adder

9 def compose1(f, g):
10    def h(x):
11        return f(g(x))
12    return h
14 compose1(square, make_adder(2))(3)
```

Example: [http://goo.gl/2IuE0](http://goo.gl/2IuE0)
The Environment for Function Composition

```
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)
```

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

Example: [http://goo.gl/2IuE0](http://goo.gl/2IuE0)
The Environment for Function Composition

Example: http://goo.gl/2IuE0
The Environment for Function Composition

```
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)
```

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

Return value of make_adder is an argument to compose1
The Environment for Function Composition

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

Example: http://goo.gl/2IuE0
Return value of make_adder is an argument to compose1
The Environment for Function Composition

1. def square(x):
   2.   return x * x
3. return adder

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

Example: http://goo.gl/2IuE0
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: [link](http://goo.gl/2IuE0)
The Environment for Function Composition

1. def square(x):
   return x * x

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

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

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

Return value of make_adder is an argument to compose1

Example: http://goo.gl/2IuE0
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)
```

Example: [http://goo.gl/2IuE0](http://goo.gl/2IuE0)
The Environment for Function Composition

```
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)
```

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

>>> ten = 10
Lambda Expressions

>>> ten = 10

>>> square = x * x
Lambda Expressions

```python
>>> ten = 10

>>> square = x * x

An expression: this one evaluates to a number
```
Lambda Expressions

```python
>>> ten = 10

>>> square = x * x

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

An expression: this one evaluates to a number
Lambda Expressions

```python
>>> ten = 10
```

An expression: this one evaluates to a number

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

Also an expression: evaluates to a function

```python
>>> square = lambda x: x * x
```
Lambda Expressions

```python
>>> ten = 10

>>> square = x * x

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

An expression: this one evaluates to a number

Also an expression: evaluates to a function

A function
Lambda Expressions

>>> ten = 10

An expression: this one evaluates to a number

>>> square = \(x \times x\)

Also an expression: evaluates to a function

>>> square = lambda \(x\): \(x \times x\)

A function with formal parameter \(x\)
Lambda Expressions

```python
>>> ten = 10

>>> square = x * x  

An expression: this one evaluates to a number

>>> square = lambda x: x * x

Also an expression: evaluates to a function

A function

with formal parameter x
and body "return x * x"
```
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"
```
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"

Notice: no "return"

Must be a single expression
```
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

>>> 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

Notice: no "return"

Must be a single expression

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

(Demo)