CS61A Lecture 3

Amir Kamil
UC Berkeley
January 28, 2013
Announcements

- Reminder: hw0 due tonight, hw1 due Wed.

- In-class quiz on Friday
  - Covers through Wednesday’s lecture
  - Bring a writing implement

- Hog project out
  - Get started early!
  - More on hog next time
The Elements of Programming

- Primitive Expressions and Statements
  - The simplest building blocks of a language

- Means of Combination
  - Compound elements built from simpler ones

- Means of Abstraction
  - Elements can be named and manipulated as units
Environment diagrams visualize the interpreter’s process.

Code (left):
Statements and expressions
Next line is highlighted

Frames (right):
A name is bound to a value
In a frame, there is at most one binding per name

Example: [http://goo.gl/SK13i](http://goo.gl/SK13i)
User-Defined Functions

Named values are a simple means of abstraction

Named computational processes are a more powerful means of abstraction

Execution procedure for def statements:
1. Create a function value with signature
   \texttt{<name>\left(<\text{formal parameters}\right):}
   \texttt{return <return expression>}

Function “signature” indicates how many parameters

Function “body” defines a computational process
Procedure for applying user-defined functions (version 1):

1. Add a local frame
2. Bind formal parameters to arguments in that frame
3. Execute the body of the function in the new environment

Example: http://goo.gl/boCk0
Calling User-Defined Functions

Procedure for applying user-defined functions (version 1):
1. Add a local frame
2. Bind formal parameters to arguments in that frame
3. Execute the body of the function in the new environment

A function’s signature has all the information to create a local frame

Example: [http://goo.gl/boCk0](http://goo.gl/boCk0)
Looking Up Names

Procedure for looking up a name from inside a function (v. 1):

1. Look it up in the local frame
2. If not in local frame, look it up in the global frame
3. If in neither frame, generate error

Example: http://goo.gl/boCk0
Every expression is evaluated in the context of an environment.

So far, the current environment is either:
- The global frame alone, or
- A local frame, followed by the global frame

**Important properties of environments:**
- An environment is a sequence of frames
- The earliest frame that contains a binding for a name determines the value that the name evaluates to

The *scope* of a name is the region of code that has access to it.
Every expression is evaluated in the context of an environment.
The earliest frame that contains a binding for a name determines the value that the name evaluates to.

```
1 from operator import mul
2 def square(x):
3    return mul(x, x)
4 square(square(3))
```

Example: [http://goo.gl/hrfnV](http://goo.gl/hrfnV)
Formal Parameters

def square(x):
    return mul(x, x)

vs

def square(y):
    return mul(y, y)

Example: http://goo.gl/boCk0
Life Cycle of a User-Defined Function

**Def statement:**

- Formal parameter: square( x )
- Return expression: return mul(x, x)

**Call expression:**

- Operator: square
- Function: func square(x)

**Calling/Applying:**

- Argument: 4
- Signature: 4
- Return value: 16