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

Environment diagrams visualize the interpreter’s process.

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

This is an example of a user-defined function:
```python
>>> def <name>(<formal parameters>):
    return <return expression>
```

Execution procedure for def statements:
1. Create a function value with signature `<name>(<formal parameters>)`
2. Bind `<name>` to that value in the current frame

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

Example: [http://goo.gl/boCk0](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/boCo0](http://goo.gl/boCo0)

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/boCo0](http://goo.gl/boCo0)

General Lookup Procedure

- 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

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

Multiple Environments in a Diagram

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.

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

Formal Parameters

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

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

Formal parameters have local scope

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

Life Cycle of a User-Defined Function

Def statement:
- Name
- Formal parameter
- Return expression

What happens?
- Function created
- Name bound

Def statement:
- Body (return statement)

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

Op's evaluated
- Function called with argument(s)

New frame!
- Params bound
- Body executed

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