Announcements

No exercise this week!

A1 sample tests posted (including corrected tests for ‘`unbound-name`)

Extra A1 office hours (Andrew, your awesome TA)

Friday Oct 12 1-3pm, BA4261

Monday Oct 15 3-5pm, BA3289

The problem of `self`

An instance method should be able to refer to the calling object.

Approach 1: `this` as a keyword

(e.g., Java, C++, Ruby)

Approach 2: `this` as an explicit parameter

(e.g., Python, Rust)

Implementation first try: `self` everywhere

Implementation second try: auto-binding `self`

WWPD? (What would Python do?)

Key technical steps

1. Split up methods into a separate __dict__
2. When looking up message, autobind `self` if message corresponds to an instance method.
3. Use `letrec` to be able to refer to the lambda in itself.

Lexical vs. dynamic scope, revisited

```
[(hash-has-key? self__dict__ attr)
 (hash-ref self__dict__ attr)]

[(hash-has-key? class__dict__ attr)
 ... (hash-ref class__dict__ attr))]
```

Chaining lookups in dictionaries is central to **inheritance** in “dynamic” object-oriented languages

Chaining lookups in dictionaries is central to **inheritance** in “dynamic” object-oriented languages

“OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late-binding of all things.”

–Alan Kay (inventor of Smalltalk)

“OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late-binding of all things.”

–Alan Kay (inventor of Smalltalk)

Manipulating control flow

In every programming language, function definition is non-strict.

```
(lambda (x) (/ 1 0))

\x -> error “David is not cool”

lambda x: raise ValueError
```

thunk: a nullary function used to delay evaluation of a value

```
(lambda () <expr>)
```

In Racket, thunk is defined as a macro, not a function. (Study question: why?)

In an eager language, we can simulate lazy arguments using thunks.

Suppose we have a lot of bulk data in a sequence.
Eager evaluation is not an option.

stream: an abstract model of a sequence of values over time

Next week, we'll look at implementing streams using lazy lists. Recall that a list can be defined recursively as:

- empty
- a value "cons" another list

The basic idea: make `cons` a lazy function!

Lab Announcement

All future labs will be held in BA3175 only.
(But don’t worry, but Simeon and Anthony will be there!)