CS 61A Discussion 8

March 17, 2016

Scheme

“The single most important thing to realize is that you’re learning a new language. A lot of your questions will be along the lines of ‘*what happens if…*’, and you should definitely test these things out in the interpreter. This will allow you to quickly build the intuition necessary to use Scheme effectively.”

Thanks to Jack, it’s easy now! (scheme.cs61a.org)

Lambdas in Scheme

+ Lambda expressions always create function objects!

```
(define (sq x) (* x x))
```

- “Define a function...”
- Here is its name, here are its arguments, here’s what to do with those arguments when you call it

```
(define sq (lambda (x) (* x x)))
```

- “Make a binding from the name `sq` to whatever the expression is”
- Here, the expression happens to evaluate to a lambda function

WWSP, pt. 2

```
((lambda (x) (x x)) (lambda (y) 4))
```

WWSP, pt. 2

```
((lambda (x) (x x)) (lambda (y) 4))
4
```

Lists in Scheme...

Lists in Scheme are actually linked lists (“pairs”). Make sure you understand the following mapping:

Scheme lists	**Python** linked lists
`cons`	`Link(...)`
`car`	`.first`
`cdr`	`.rest`
`'(), nil`	`Link.empty`