Week 1 Lecture 3

Theory

Getting Ready

- Feel good about Lecture 2
- Read SICP Seciton 1.3 closely

What’s in this lecture?

- Higher Order Functions
- Building more interesting programs

Resources (these help)

- MIT OpenCourseWare 6.001 2005SP
- Lecture Notes
- Video Lectures
- Development Tools

Scheme: Higher Order Functions

```
;; driver is a function that can call other functions

1 ]=> (define (driver f a b) (f a b))
;Value: driver

1 ]=> (driver + 1 2)
;Value: 3

1 ]=> (driver * 6 4)
;Value: 24
```

Sum-Of-Terms Driver

```
;; next is "increment" function
;; term is the "do stuff to term" function
(define (sum term a next b)
  (if (> a b) 0
    (+ (term a)
        (sum term (next a) next b))))
```

Sum-Of-Terms (Tail-Recursive)

```
;; next is “increment” function
;; term is the “do stuff to term” function
(define (sum term a next b)
  (sum-iter 0 term a next b))

(define (sum-iter accum term a next b)
  (if (> a b) accum
    (sum-iter (+ accum (term a)) term (next a) next b)
```

Adding filter

```
;; filter function takes a number & returns boolean
(define (sum term a next b filter)
  (sum-iter 0 term a next b filter))

(define (sum-iter accum term a next b filter)
  (if (> a b) accum
    (let ((tx (if (filter a) (term a) 0)))
      (sum-iter (+ accum tx) term (next a) next b filter))))

(define (odd-filter n) (= 1 (modulo n 2)))

(define (even-filter n) (= 0 (modulo n 2)))
```

Lambda 1

```
;; create a function that adds one to its argument
(lambda (x) (+ x 1))

;; a function that returns the function f(x, y) = ax + by^2
(define (axby2 x y)
  (lambda (a b) (+ (* a x) (*  b y y))))

(define (a7b92 a b) (xa2yb 7 9))

(a7b92 7 9)
```

Lambda 2

```
;; what do these do?

(define (chain f g) (lambda () (if (f) #t (g))))

(define (not-true) (lambda () #f))

(define (not-false) (lambda () #t))

((chain (not-true) (not-true)))

((chain (not-false) (not-true)))

((chain (not-true) (not-false)))
```

Exercises

- SICP 1.30, 1.31, 1.32, 1.33, 1.35, 1.36, 1.41