Week 1 Lecture 1

Theory

Getting Ready

- Install MIT Scheme
- Install emacs
- Read SICP Chapter 1 in its entirety (to know what's coming this week)
- Read SICP Section 1.1 closely

Resources (these help)

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

What's in this lecture?

- Intro to SICP
- Intro & First Steps in Scheme

What is SICP?

- Why is SICP important?
  - How to think in a structured way
  - How to turn thoughts into programs
  - How to build better programs

What if we skipped SICP?

- Would still learn all the applied stuff
- The different applied languages for the applied stuff would look foreign / scary
- Programs will be messier
- Programs will take longer to write
- Programs will be less correct

Basic Scheme Expressions

```
1 ]=> 486
;Value: 486

1 ]=> "hi there"
;Value 11: "hi there"

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

1 ]=> (* (+ 1 2) (- 4 2))
;Value: 6
```

Conditions

```
1 ]=> (> 1 0)
;Value: #t

1 ]=> (>= 2 3)
;Value: #f

1 ]=> (= 0 0)
;Value: #t
```

Conditional Expressions

```
1 ]=> (if (> 1 0) 3 4)
;Value: 3

1 ]=> (if (= 1 0) 3 4)
;Value: 4

1 ]=> (cond
        ((= 4 4) 4)
        ((> 3 4) 7)
        (else 5))
;Value: 4
```

Define

```
1 ]=> x
;Unbound variable: x
;To continue,...
```

```
1 ]=> (define x 5)
;Value: x

1 ]=> x
;Value: 5

1 ]=> (+ x 1)
;Value: 6
```

Scheme: Functions

```
1 ]=> (define (add5 x) (+ x 5))
;Value: add5

1 ]=> (add5 1)
;Value: 6

1 ]=> (add5 5)
;Value: 10

1 ]=> (define (square x) (* x x))
;Value: square

1 ]=> (square -1)
;Value: 1

1 ]=> (square 3)
;Value: 9
```

Scheme: Output

```
1 ]=> (write-line "hello user")
"hello user"
;Unspecified return value
```

```
1 ]=> (write-line (string-append "hello " "world"))
"hello world"
;Unspecified return value

1 ]=> (number->string 5)
;Value 12: "5"

1 ]=> (string->number "7")
;Value: 7
```

Scheme: Input

```
1 ]=> (read-line)
foo
;Value 13: "foo"

1 ]=> (define (try-unlock)
  (if (string=? (read-line) "opensesame")
    (write-line "unlocked!")
    (write-line "still locked.")))
;Value: try-unlock

1 ]=> (try-unlock)
hi
"still locked."
;Unspecified return value

1 ]=> (try-unlock)
opensesame
"unlocked!"
;Unspecified return value
```

Samples: Old or Young?

```
1 ]=> (define (is-old)
  (if (> (string->number (read-line)) 30)
    (write-line "You're old!")
    (write-line "Not old yet.")))
;Value: is-old

1 ]=> (is-old)
24
"Not old yet."
;Unspecified return value

1 ]=> (is-old)
64
"You're old!"
;Unspecified return value
```

Samples: F to C

```
1 ]=> (define (f-to-c x) (* (/ 5.0 9.0) (- x 32.0)))
;Value: f-to-c

1 ]=> (f-to-c 98.6)
;Value: 37.

1 ]=> (f-to-c 32)
;Value: 0.

1 ]=> (f-to-c 212)
;Value: 100.
```

Exercises

- Write c-to-f function
- SICP 1.1, 1.2, 1.3, 1.8