6.001 SICP – Basic Scheme

primitives / combinations / abstraction
some examples
Lambda
Special form cond / and / or / if
Evaluate lambda

Syntax

Things that make up scheme programs:
• self-evaluating 23, "hello", #t
• names +, pi
• combinations (+ 2 3)
(* pi 4)
• special forms (define pi 3.14)
(lambda _)
(if _)
Note that special forms are not combinations, even though they are syntactically similar.
A combination requires that the first subexpression be a procedure.

Language elements -- abstractions
• Need to capture ways of doing things – use procedures

Interaction of define and lambda
(lamba (x) (* x x))
===> #[compound-procedure 9]
(define square (lambda (x) (* x x)))
===> undef
(square 4)
===> 16
((lambda (x) (* x x)) 4)
===> 16

Fill in each box...
(define twice (lambda (x) (* 2 x)))
twice 2 ===> 4
twice 3 ===> 6

(define constant2 (lambda () 2))
(constant2) ===> 2

(define second (lambda (x y z) y))
(second 2 15 3) ===> 15
(second 34 -5 16) ===> -5

Examples...
1. (define six (lambda () 6)) undef
2. six #[compound-procedure 1] error
3. (+ six 1)
4. (+ (six) 1) 7
5. (define six 6) undef
6. six 6
7. (+ six 1) 7
8. (+ (six) 1) error
Syntactic Sugar

\[
\text{(define square (lambda (x) (* x x)))}
\]

\[
\text{===>}
\]

\[
\text{(define (square x) (* x x))}
\]

- a convenient shorthand
- this is a use of lambda!

If

- The special form if has the following form.
  \[
  \text{(if <pred1> <consequent> <alternative>)}
  \]
  - <consequent> is evaluated, if the predicate evaluate to #t and return that value.
  - Otherwise it evaluate the <alternative> and and return that value.
  - Eg. \((\text{if} (<a b) a b)\)
  - How to write an abs function?

Abs functions

\[
\text{(define abs (lambda (x)}
\]

\[
\text{(if (< x 0) (- x) x)))}
\]

And / or

\[
\text{(and <expr1> <expr2> ... <exprN>)}
\]

- Evaluates each argument in turn, immediately returns #f if found, otherwise returns last value.

\[
\text{(or <expr1> <expr2> ... <exprN>)}
\]

- Evaluates each argument in turn, immediately returns a non-false (#t) value when found, otherwise returns #f.

Cond

- Multiple expressions can be used in each clause
  \[
  \text{(cond (<pred1> <expr1a> ... <expr1z>)
          (<pred2> <expr2a> ... <expr2z>)
          ...}
  \]
  - The expressions in each clause are evaluated in sequence, and the value of the clause is the last expression.