Scheme

Douglas Blank

Scheme Syntax

Scheme Things

- Variables (can be string, char, int, list, etc)
- Expressions
- Functions (procedures)
- Symbols
- Numbers
  - Exact, Inexact, Complex, Imaginary, Ratios, Bignums, etc.
- Special Forms

Variables and Assignments

- (define x 5)
- (define x “hello world”)
- (set! x 6)
- (set! x “watch out!”)

Expressions

- (+ 1 2)
  3
- (* (+ 5 7) (- 7 3))
  48
- (+ 1 2 3 4 5 6)

Functions

- (define function
  (lambda (x)
  (+ x 1)))

- (function 6)
  7

Special Forms

- (or #t #f)
  #t
- (or #f #f “banana”)
  “banana”
- (and (not (= x 0)) (= (/ 1 x) ¼))
  short circuit

Special Forms

- (if test-exp then-exp else-exp)
- (if (equal? “hello” “there”)
    'same
    'diff)
- (cond
    (test-exp1 then-exp1)
    (test-exp2 then-exp2)
    ...
    (else else-exp))

Symbols

- 'hello
- (quote hello)
- '(this is a list of quoted things)

Lists

- (list 1 2 3)
- (list)
- '()
- (list 1 2 3 (list 4 5 6))
- '(1 2 3 (4 5 6))
- (append '(1 2 3) '(4 5 6))
  (1 2 3 4 5 6)

Cons Cells

- (cons 1 2)
  (1 . 2)
- (car (cons 1 2))
  1
- (cdr (cons 1 2))
  2

Lists from Cons Cells

- (cons 1 '())
  (1)
- (cons 2 (cons 1 '()))
  (2 1)
- (list? (cons 1 '()))
  #t
- (pair? (cons 1 2))
  #t
- (list? (cons 1 2))
  #f

Recursion

- Python:

```
def loop():
    loop()
```

- Scheme:

```
(define loop
    (lambda ()
        (loop)))
```

Recursion

- (define fact
  (lambda (n)
    (if (= n 1)
        1
        (* n (fact (- n 1))))))
- (define fib
  (lambda (n)
    (cond
      ((= n 1) 1)
      ((= n 2) 1)
      (else (+ (fib (- n 1)) (fib (- n 2)))))))

Side Effect vs Return Value

- (set! x 6)
- (display “hello”)
- (fib 5)
- (fact 5)