Interacting with a User

```
(define greet
  (lambda (name)
    (display (string-append "Hello " name "!"))
    (display " What is your favorite number?")
    (let ((num (read)))
      (if (equal? num 5)
          (begin
            (display "Great! ")
            (display num)
            (display " is my favorite number too."))
          (display (string-append  (number->string num) " is ok."))))))

(greet "Tom")
```

Scheme provides a procedure called read that reads one value of any type

begin when there's more than one

convert num to a string

Calculate the Average

Scheme provides a procedure called read that reads one value of any type

```
(define average
  (lambda ()                          ; no parameters
    (accumulator 0 0 (read))))        ; (read) reads anything

(define accumulator                   ; recursive
  (lambda (sum n next)
    (if (not (number? next))          ; use of a sentinel
        (compute-average sum n)
        (accumulator (+ next sum) (+ 1 n) (read)))))

(define compute-average
  (lambda (sum n)
    (if (> n 0)
        (/ sum n)
        "no number")))

(average)
```

Recursion to repeat!

ABS Control in Scheme

```
(define mile-inch 63360)
(define mypi 3.14159265)
(define wheel-diameter 15) ; inches

(define wheel-sensor (lambda ()
                       (begin (display "get rotations per second: ")
                              (read)) ))
(define wheel-velocity (lambda (rps) ; miles per hour
                         (/ (* mypi wheel-diameter rps 3600)
                            mile-inch) ))
(define body-velocity (lambda ()
                        (begin (display "get miles per hour: ")
                               (read)) ))

(define error-detection (lambda(wv bv)
                          (if (< (abs (- bv wv)) 0.01)
                              (write "no action")
                              (if (> bv wv)
                                  (write "reduce brake force!")
                                  (write "reduce accel force!"))) ))
(define start-engine (lambda ()
                       (error-detection (wheel-velocity(wheel-sensor))
                                        (body-velocity) )))
(define main (lambda ()
               (start-engine)))

(main) ; call the procedure
```