How can we generate a vector for a sequence?

```scheme
> (make-vect-seq 10 (lambda (x) (* x 2)))
#(0 2 4 6 8 10 12 14 16 18)
> (make-vect-seq 10 even?)
#(t f t f t f t f t f)
> (make-vect-seq 10 (lambda (x) (random (+ 1 x))))
#(0 0 2 1 1 5 0 0 3 4)
```

**make-vect-seq** is built-in to Scheme as the **build-vector** function:

```scheme
> (build-vector 10 even?)
#(t f t f t f t f t f)
```
How could we add up all the elements in a vector?

```scheme
> (define sum-vector
  (lambda (vec)
    (let ((sum 0))
      (dotimes (i (vector-length vec))
        (set! sum (+ sum (vector-ref vec i))))
    sum))

> (sum-vector #(1 3 5 7 9))
25
> (sum-vector (vector))
0
```

We can use the `vector-swap!` procedure we defined previously:

```scheme
;; VECTOR-REVERSE!
;; INPUTS: VEC, a vector
;; OUTPUT: The destructively modified VEC that now has its elements in reverse order
;; SIDE EFFECT: VEC's elements are now in the reverse order
(define vector-reverse!
  (lambda (vec)
    (let ((len (vector-length vec)))
      (dotimes (i (floor (/ len 2)))
        (vector-swap! vec i (- len i 1))
      )
    vec))

> (define v (vector 1 2 3 4 5))
> v
#(1 2 3 4 5)
> (vector-reverse! v)
#(5 4 3 2 1)
> v
#(5 4 3 2 1)
```

How can we use iteration to (destructively) reverse a vector?

```scheme
> (vector-reverse! v)
i: 0
i: 1
#(5 4 3 2 1)
```
Iterating and recursing to solve problems

How could we print a beautiful checkerboard?

```scheme
> (print-checkerboard 5 8)
X _ X _ X _ X 
_ X _ X _ X _ 
X _ X _ X _ X _ 
_ X _ X _ X _ 
X _ X _ X _ X _ 
```

When we introduced iteration, we claimed it didn’t let us do anything we couldn’t do with recursion.

We can prove this is true by writing a recursive equivalent to our favorite iterative construct, `dotimes`. 
We now have recursive and iterative equivalents:

```lisp
> (dotimes-func 5
  (lambda (i)
    (printf "i = \(\text{~A}\)\)" i)))
i = 0
i = 1
i = 2
i = 3
i = 4
> (dotimes (i 5)
  (printf "i = \(\text{~A}\)\)" i)))
i = 0
i = 1
i = 2
i = 3
i = 4
```

Acknowledgments

This lecture incorporates material from:

Simon Ellis
Luke Hunsberger
Jennifer Walter