

```
fun append (xs,ys) =  
  if xs=[]  
  then ys  
  else (hd xs)::append(tl xs,ys)  
  
fun map (f,xs) =  
  case xs of  
    [] => []  
  | x::xs' => (f x)::(map(f,xs'))  
  
val a = map (increment, [4,8,12,16])  
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```

Programming Languages

Dan Grossman

Dynamic Typing

Dynamic typing

Major topic coming later: contrasting static typing (e.g., ML) with dynamic typing (e.g., Racket)

For now:

- Frustrating not to catch “little errors” like `(n * x)` until you test your function
- But can use very flexible data structures and code without convincing a type checker that it makes sense

Example:

- A list that can contain numbers or other lists
- Assuming *lists or numbers* “all the way down,” sum all the numbers...

Example

```
(define (sum xs)
  (if (null? xs)
      0
      (if (number? (car xs))
          (+ (car xs) (sum (cdr xs)))
          (+ (sum (car xs)) (sum (cdr xs))))))
```

- No need for a fancy datatype binding, constructors, etc.
- Works no matter how deep the lists go
- But assumes each element is a list or a number
 - Will get a run-time error if anything else is encountered