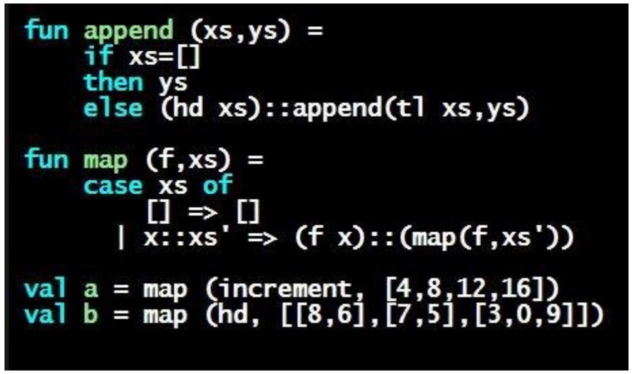

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Hashes and Ranges

More collections

- *Hashes* like arrays but:
  - *Keys* can be *anything*; strings and symbols common
  - No natural ordering like numeric indices
  - Different syntax to make them

  Like a dynamic record with anything for field names
  - Often pass a hash rather than many arguments

- *Ranges* like arrays of contiguous numbers but:
  - More efficiently represented, so large ranges fine

Good style to:

- Use ranges when you can
- Use hashes when non-numeric keys better represent data

Similar methods

- Arrays, hashes, and ranges all have some methods other don't
  - E.g., **`keys`** and **`values`**
- But also have many of the same methods, particularly iterators
  - Great for duck typing
  - Example

```
def foo a
  a.count {|x| x*x < 50}
end

foo [3,5,7,9]
foo (3..9)
```

Once again separating "how to iterate" from "what to do"