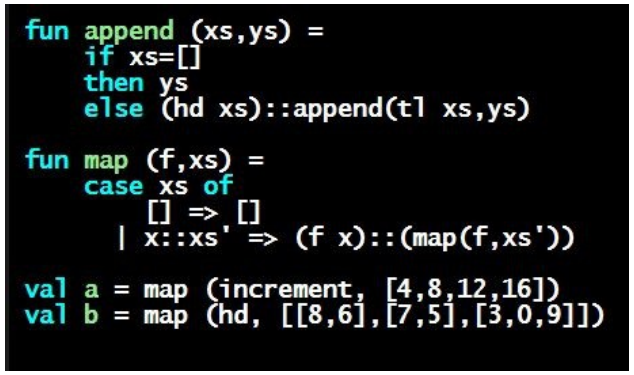

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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
2013

Optional: Course-Motivation Introduction

Course Motivation
(Did you think I forgot? ಠ_ಠ)

- Why learn the fundamental concepts that appear in all (most?) languages?
- Why use languages quite different from C, C++, Java, Python?
- Why focus on functional programming?
- Why use ML, Racket, and Ruby in particular?

Caveats

Will give some of my reasons in terms of this course

- *My* reasons: more personal opinion than normal lectures
  - Others may have equally valid reasons

- *Partial* list: surely other good reasons

- *In terms of course*: Keep discussion informal
  - Not rigorous proof that all reasons are correct

- Will not say one language is “better” than another
  - May omit “your favorite language” without malice

Summary

- No such thing as a “best” PL
- Fundamental concepts easier to teach in some (multiple) PLs
- A good PL is a relevant, elegant interface for writing software
  - There is no substitute for precise understanding of PL semantics
- Functional languages have been on the leading edge for decades
  - Ideas have been absorbed by the mainstream, but very slowly
  - First-class functions and avoiding mutation increasingly essential
  - Meanwhile, use the ideas to be a better C/Java/PHP hacker
- Many great alternatives to ML, Racket, and Ruby, but each was chosen for a reason and for how they complement each other