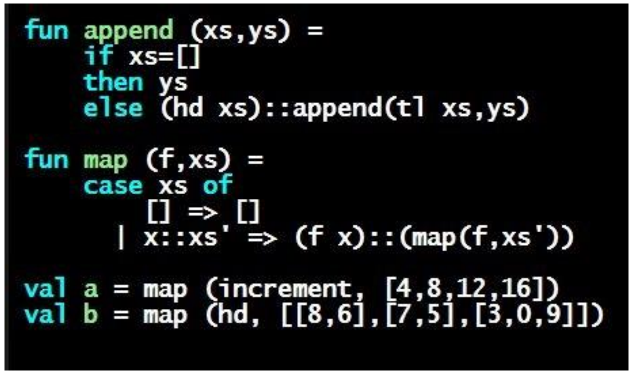
```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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
University of Washington

[Placeholder for] Course Motivation

What this course is about

- Many essential concepts relevant in any programming language
  - And how these pieces fit together
- Use ML, Racket, and Ruby languages:
  - They let many of the concepts “shine”
  - Using multiple languages shows how the same concept can “look different” or actually be slightly different
  - In many ways simpler than Java, C#, Python, …
- Big focus on *functional programming*
  - Not using *mutation* (assignment statements) (!)
  - Using *first-class functions* (can’t explain that yet)
  - But many other topics too

Why learn this?

This is the "normal" place for course motivation

- Why learn this material?

But in my experience, we don't have enough *shared vocabulary*

- So delay full motivation until after function closures (Section 3)
- (Will motivate immutable data at end of section 1)

In the meantime, have to "assert things" without much evidence

- Except lots of prior students
- Example feedback: "I had tried to learn _____ several times before. But after your course, I had no trouble working through the tutorials quickly…"

My claim

Learning to think about software in this "PL" way will make you a better programmer even if/when you go back to old ways

It will also give you the mental tools and experience you need for a lifetime of confidently picking up new languages and ideas

[Somewhat in the style of *The Karate Kid* movies (1984, 2010)
- http://www.imdb.com/title/tt0087538/
- http://www.imdb.com/title/tt1155076/

]

A strange environment

- Next 4-5 weeks will use
  - ML language
  - Emacs editor
  - Read-eval-print-loop (REPL) for evaluating programs

- Need to get things installed and configured
  - See written instructions (read carefully; feedback welcome)
  - Optional: videos showing Windows installation

- Only then can you focus on the “real content” and Homework 1

- Working in strange environments is a computing life skill

Not about the languages

- Cannot emphasize enough that ML (Part A), Racket (Part B), and Ruby (Part C) are “means to other ends”
  - Chosen as “particularly good fits for the topics”
  - Other choices possible

- Also readily admit ML in particular no longer used-much-for-real
  - Closely related languages are: OCaml, F#, Scala, Haskell
  - Arguably a feature and not a bug:
    - Nobody distracted by “already knowing it”
    - More effective for “Karate Kid” approach
    - Focus on core features, not libraries, fancy stuff, etc.
    - A very “clean, compositional, elegant” language

Enough text already

- Non-introductory lectures will write code
  - Plus switch to slides for key concepts
  - Plus “in-video questions”

- Much better than these “introduction” videos
  - So forgive the “boring intro stuff”