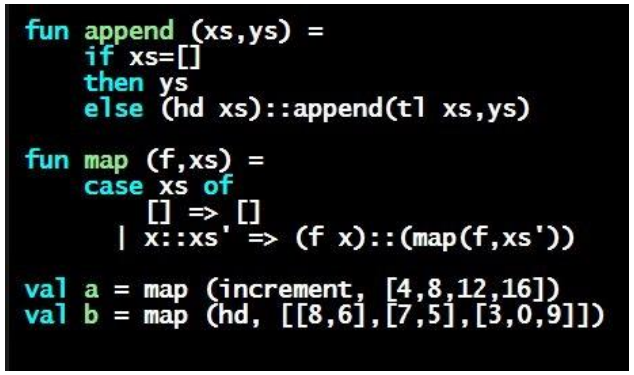

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Welcome to the Class!

Welcome!

Challenging opportunity to learn *the fundamental concepts* of programming languages

With hard work, patience, and an open mind, this course makes you a much better programmer

- Even in languages we won't use
- Learn the core ideas around which *every* language is built, despite countless surface-level differences and variations
- *Poor* course summary: "Uses ML, Racket, and Ruby"

Introductory material

This video:

- Course mechanics, structure, etc.

Other introductory videos

- Who I am and staff acknowledgements
- Initial “why study programming languages this way”
  - Real course-motivation in 2-3 weeks
- Discussion of recommended background
- Why “the course” is divided into Parts A, B, and C
- Grading policies
- Overall “roadmap” particularly upcoming weeks

Concise to-do list

1. Get familiar with all the materials on the webpage
   - Whether or not this is your first Coursera course
   - Several things different/unusual

2. Get set up using Emacs [optional] and SML [required]
   - Installation/configuration/use instructions on web page
     - And videos if you need
   - Essential; non-intellectual
     - No reason to delay!

3. Optional “dummy” Homework 0

Course materials

- Video lectures: most necessary, others labeled “optional”
- Reading notes
  - Same material as lectures, but more precise and complete
  - So: optional but recommended
- Lecture code
  - To avoid having to re-copy from .mp4/.pptx/.pdf
- Homework assignments
  - Graded against test cases *and* peer graded
- Exams
  - Cover topics harder to re-enforce with “just programming” because this is not just a programming course

Homework

- One per major-course section (minus 1)
  - Each homework submitted “all at once”
  - Much more like a challenging university course than many MOOCS [10-12 hours / week?]
- Doing the homework involves:
  1. Understanding the concepts being addressed
  2. Writing code demonstrating understanding of the concepts
  3. Testing your code to ensure you understand and have correct programs
  4. “Playing around” with variations, incorrect answers, etc.

  Only (2) is graded, but focusing on (2) makes homework harder
- Challenge problems: Low points/difficulty ratio
- Do not make solutions publicly available

Note my writing style

- Homeworks tend to be worded very precisely and concisely
  - I am a computer scientist and I write like one (a good thing!)
  - Technical issues deserve precise technical writing
  - Conciseness values your time as a reader
  - You should try to be precise too
- *Skimming or not understanding why a word or phrase was chosen can make the homework harder*
- By all means ask others if a problem is confusing
  - Being confused is normal and understandable – even encouraged
  - Once you're unconfused, you might agree the problem wording didn't cause the confusion

Welcome!

- Great adventure for all of us
  - Hope to support an active, energetic community
  - More fun together even if I cannot meet all of you

- This MOOC is one of "the neatest things I've ever done"
  - I hope you'll think so too

- Hope to stretch your mind and challenge you at every step
  - While giving you everything you need to succeed