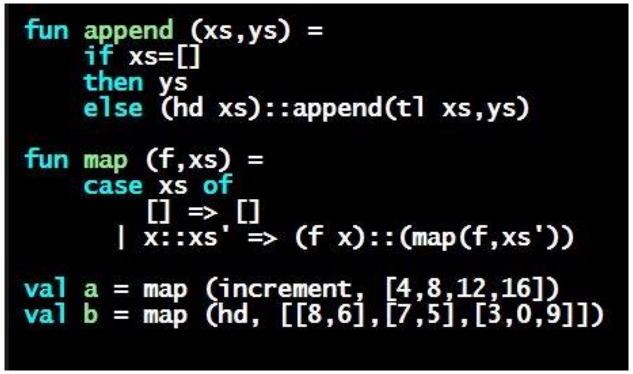

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Optional: Why ML, Racket, and Ruby?

The languages together

SML, Racket, and Ruby are a useful *combination* for us

	dynamically typed	statically typed
functional	Racket	SML
object-oriented	Ruby	Java/C#/Scala

ML: polymorphic types, pattern-matching, abstract types & modules

Racket: dynamic typing, “good” macros, minimalist syntax, eval

Ruby: classes but not types, very OOP, mixins

[and much more]

Really wish we had more time:

Haskell: laziness, purity, type classes, monads

Prolog: unification and backtracking

[and much more]

But why not…

Instead of SML, could use similar languages easy to learn after:

- OCaml: yes indeed but would have to port all my materials 😐
  - And a few small things (e.g., second-class constructors)
- F#: yes and very cool, but needs a .Net platform
  - And a few more small things (e.g., second-class constructors, less elegant signature-matching)
- Haskell: more popular, cooler types, but lazy semantics and type classes from day 1

Admittedly, SML and its implementations are showing their age (e.g., **`andalso`** and less tool support), but it still makes for a fine foundation in statically typed, eager functional programming

But why not…

Instead of Racket, could use similar languages easy to learn after:

- Scheme, Lisp, Clojure, …

Racket has a combination of:

- A modern feel and active evolution
- "Better" macros, modules, structs, contracts, …
- A large user base and community (*not* just for education)
- An IDE tailored to education

Could easily define our own language in the Racket system

- Would rather use a good and vetted design

But why not…

Instead of Ruby, could use another language:

- Python, Perl, JavaScript are also dynamically typed, but are not as “fully” OOP, which is what I want to focus on
  - Python also does not have (full) closures
  - JavaScript also does not have classes but is OOP

- Smalltalk serves my OOP needs
  - But implementations merge language/environment
  - Less modern syntax, user base, etc.

Is this real programming?

- The way we use ML/Racket/Ruby can make them seem almost “silly” precisely because lecture and homework focus on interesting language constructs

- “Real” programming needs file I/O, string operations, floating-point, graphics, project managers, testing frameworks, threads, build systems, …
  - Many elegant languages have all that and more
    - Including Racket and Ruby
  - If we used Java the same way, Java would seem “silly” too

A note on reality

Reasonable questions when deciding to use/learn a language:

- What libraries are available for reuse?
- What tools are available?
- What can get me a job?
- What does my boss tell me to do?
- What is the de facto industry standard?
- What do I already know?

Our course by design does not deal with these questions

 - You have the rest of your life for that
 - And technology *leaders* affect the answers