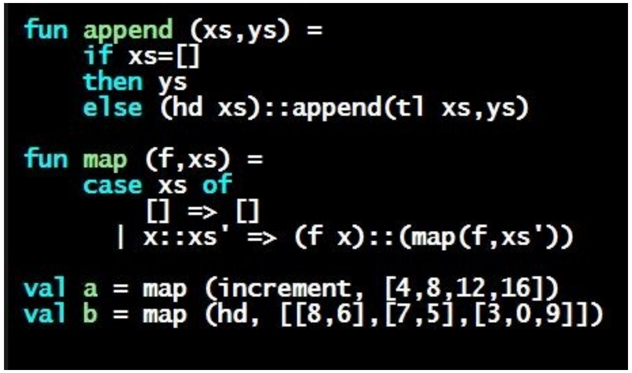

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Equivalence Versus Performance

What about performance?

According to our definition of equivalence, these two functions are equivalent, but we learned one is awful

- (Actually we studied this before pattern-matching)

```
fun max xs =
  case xs of
    [] => raise Empty
  | x::[] => x
  | x::xs′ =>
      if x > max xs′
      then x
      else max xs′
```

```
fun max xs =
  case xs of
    [] => raise Empty
  | x::[] => x
  | x::xs′ =>
     let
        val y = max xs′
     in
        if x > y
        then x
        else y
     end
```

Different definitions for different jobs

- PL Equivalence: given same inputs, same outputs and effects
  - Good: Lets us replace bad **`max`** with good **`max`**
  - Bad: Ignores performance in the extreme

- Asymptotic equivalence: Ignore constant factors
  - Good: Focus on the algorithm and efficiency for large inputs
  - Bad: Ignores "four times faster"

- Systems equivalence: Account for constant overheads, performance tune
  - Good: Faster means different and better
  - Bad: Beware overtuning on "wrong" (e.g., small) inputs; definition does not let you "swap in a different algorithm"

Claim: Computer scientists implicitly (?) use all three every (?) day