

```
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

The REPL and Error Messages

# *Pragmatics*

Last two segments have built up key conceptual foundation

But you also need some pragmatics:

- How do we run programs using the REPL?
- What happens when we make mistakes?

Work on developing resilience to mistakes

- Slow down
- Don't panic
- Read what you wrote very carefully

*use*

`use "foo.sml"` is an unusual expression

It enters bindings from the file `foo.sml`

Result is `()` bound to variable `it`

– Ignorable

# *The REPL*

- Read-Eval-Print-Loop is well named
- Can just treat it as a strange/convenient way to run programs
  - But more convenient for quick try-something-out
  - Then move things over to a testing file for easy reuse
- For reasons discussed in next segment, do *not* use `use` without restarting the REPL session
  - (But using it for multiple files at beginning of session is okay)

# *Errors*

Your mistake could be:

- Syntax: What you wrote means nothing or not the construct you intended
- Type-checking: What you wrote does not type-check
- Evaluation: It runs but produces wrong answer, or an exception, or an infinite loop

Keep these straight when debugging even if sometimes one kind of mistake appears to be another

# *Play around*

Best way to learn something: Try lots of things and don't be afraid of errors

Maybe watching me make a few mistakes will help...