

```
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

*Classes and Objects*

# *The rules of class-based OOP*

In Ruby:

1. All values are references to *objects*
2. Objects communicate via *method calls*, also known as *messages*
3. Each object has its own (private) *state*
4. Every object is an instance of a *class*
5. An object's class determines the object's *behavior*
  - How it handles method calls
  - Class contains method definitions

Java/C#/etc. similar but do not follow (1) (e.g., numbers, null) and allow objects to have non-private state

# *Defining classes and methods*

```
class Name
  def method_name1 method_args1
    expression1
  end
  def method_name2 method_args2
    expression2
  end
  ...
end
```

- Define a new class called with methods as defined
- Method returns its last expression
  - Ruby also has explicit **return** statement
- Syntax note: Line breaks often required (else need more syntax), but indentation always only style

# *Creating and using an object*

- `ClassName.new` creates a new object whose class is `ClassName`
- `e.m` evaluates `e` to an object and then calls its `m` method
  - Also known as “sends the `m` message”
  - Can also write `e.m()`
- Methods can take arguments, called like `e.m(e1, ..., en)`
  - Parentheses optional in some places, but recommended

# Variables

- Methods can use local variables
  - Syntax: starts with letter
  - Scope is method body
- No declaring them, just assign to them anywhere in method body (!)
- Variables are mutable, **x=e**
- Variables also allowed at “top-level” or in REPL
- Contents of variables are always references to objects because all values are objects

# *Self*

- `self` is a special keyword/variable in Ruby
- Refers to “the current object”
  - The object whose method is executing
- So call another method on “same object” with `self.m(...)`
  - Syntactic sugar: can just write `m(...)`
- Also can pass/return/store “the whole object” with just `self`
- (Same as `this` in Java/C#/C++)