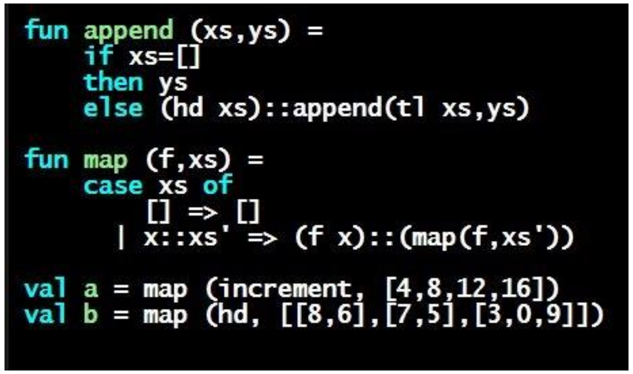

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Object State

Objects have state

- An object's state persists
  - Can grow and change from time object is created

- State only directly accessible from object's methods
  - Can read, write, extend the state
  - Effects persist for next method call

- State consists of *instance variables* (also known as fields)
  - Syntax: starts with an `@`, e.g., **`@foo`**
  - "Spring into being" with assignment
    - So mis-spellings silently add new state (!)
  - Using one not in state not an error; produces **`nil`** object

Aliasing

- Creating an object returns a reference to a new object
  - Different state from every other object

- Variable assignment (e.g., **`x=y`**) creates an alias
  - Aliasing means same object means same state

Initialization

- A method named **`initialize`** is special
  - Is called on a new object before **`new`** returns
  - Arguments to **`new`** are passed on to **`initialize`**
  - Excellent for creating object invariants
  - (Like constructors in Java/C#/etc.)

- Usually good *style* to create instance variables in **`initialize`**
  - Just a convention
  - Unlike OOP languages that make “what fields an object has” a (fixed) part of the class definition
    - In Ruby, different instances of same class can have different instance variables

Class variables

- There is also state shared by the entire class
- Shared by (and only accessible to) all instances of the class
- Called *class variables*
  - Syntax: starts with an `@@`, e.g., **`@@foo`**
- Less common, but sometimes useful
  - And helps explain via contrast that each object has its own instance variables

Class constants and methods

- *Class constants*
  - Syntax: start with capital letter, e.g., **`Foo`**
  - Should not be mutated
  - Visible outside class **`C`** as **`C::Foo`** (unlike class variables)

- *Class methods* (cf. Java/C# static methods)
  - Syntax (in some class **`C`**):

```
def self.method_name (args)
   …
end
```

  - Use (of class method in class **`C`**):

```
C.method_name(args)
```

  - Part of the class, not a particular instance of it