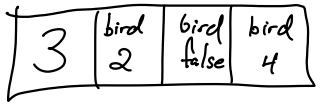
Remember Eagle 2

Features: tuples
heap

Implementation:

- malloc a big chunk
- pass ptr to bird_main
- store ptr in heap_cursor
- use heap incrementally
- heap layout: use heap memory in a consistent way



Falcon

Features:
- first-class function
- partial application
- anonymous functions

"functional language" (I don't like that term)

fns are values

fns can be given fewer arguments

fns don't need names

```
List.map (compile_expression env)
List.map (fun x -> x+1) [2; 3; 4]
```

	OCaml	Python	Javascript	Java	C++	C	Falcon	
First-class fns	Yes	Yes	Yes	8?	11?	No*	Yes	
Partial application	Yes	No	No	No	No	No	Yes	
Anonymous functions	Yes	Yes	Yes	8?	11?	No	No	

"lambda" → λ-calculus

Syntax

<program> ::= <decl_list> <expr>

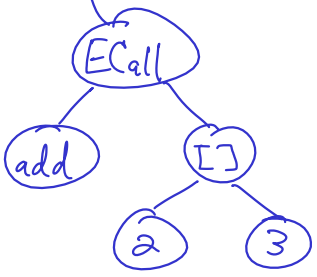
<decl> ::= 'def' <Identifier> <identifier_list> '=' <expr> 'end'

<expr> ::= ... } Cardinal exprs + tuples + released
| <expr> <expr>

EXAMPLES

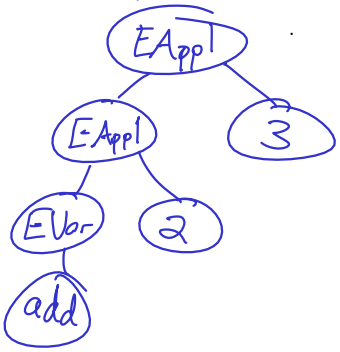
Dove

```
def add(x,y)
    x+y
end
add(2,3)
```



Falcon

```
def add x y =
    x+y
end
add 2 3
```



Semantics

Dove

At compile-time, create a definition for "add"

call add with args 2 and 3

Falcon

At compile-time, create a function value referring to defn of "add" and store in "add"

calling fn in the "add" variable with 2

calling result with 3

What information do I need to store to represent a function?

- Where are my instructions? call rax
- How many params do I want?
- How many args do I have?
- Arguments values

tuple

closure