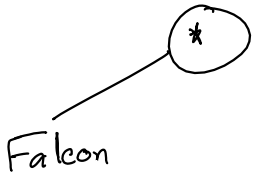
Falcon

Dove / Eagle don't have:

- pattern matching / alternatives / variants / unions
- anonymous functions / currying, partial application / first-class functions



(fun x -> x)

```
let add x y = x+y;;
let inc = add 1;;
add : int -> int -> int
let inc' y = add 1 y;;
```

```
Cff
int sum(int x, int y) {
    return x+y;
}
plus = sum;
```

"Functional language"	OCaml	C	Python	Java	Rust	Cff	Falcon
First-class functions	Yes	Sort of?	Yes	8?	Yes	11?	Yes
Partial application	Yes	No	No	No	No	No	Yes
Anonymous functions	Yes	No	Yes	8?	Yes	11?	No *

FCF in OCaml

```
let x=5 in
fun y -> x+y
```

```
fun x -> x+1
λx. x+1
```

Syntax

Dove

```
def f(a,b)
    a * b
end
```

```
f(2,3)
```

```
f(1)
```

Falcon

```
def f a b =
    a*b
end
```

```
f 2 3
```

```
f 1
```

Semantics

Dove

declare func called f

call fn named f

error

Falcon

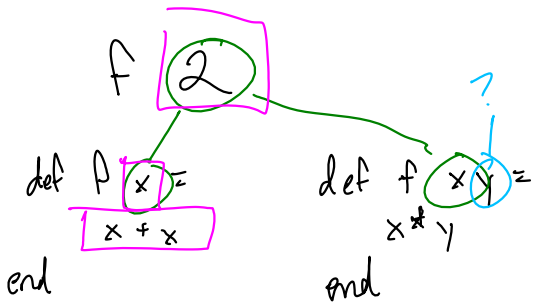
declare a func and store in a variable called f

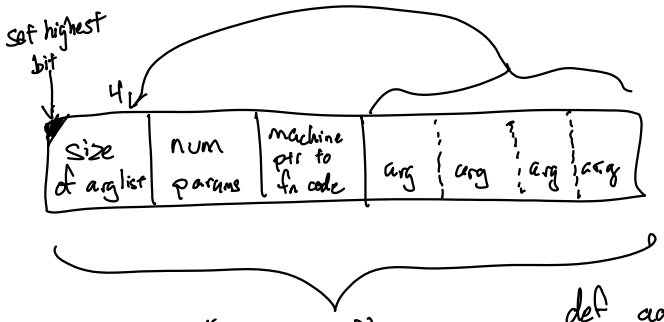
call fn in the f variable

produces a fn waiting for 2nd arg

How will we compile Falcon?

- how many args I have so far
- how many parameters does fn want?
- code to run
- whatever args have already been passed





"closure"

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

```
let inc = add 1 in
```