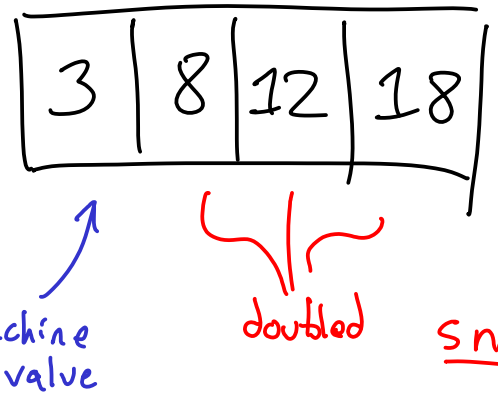
Egg-eater common questions

(4, 6, 9)



heap_cursor

- global variable
  - C param (snake_main)
  - C local var (snake_main)

Higher-order functions

Functional programming languages

✓ – everything is an expression

– partial application → `let f x y = x + y in let g = f 1 in ...`

– anonymous functions → `(fun x -> x + 1) 3`

– first-class functions (functions are values) → `f (fun x -> x + 1)`

	Python	C	OCaml	Java
partial application	No	No	Yes	No
anonymous functions	Yes	No	Yes	8?
first-class functions (functions are values)	Yes	Sort of. Not really	Yes	8?

Python: 2 params `f(1,2)`
2 curried `f(1)(2)`

`lambda x: x+1`

Examples of

anonymous fns

(fun : x y) = x+y

fun x y → x+y

fun x → fun y → x + y

ELambda of string expr *

first-class fns

list_map is_even my_list

this is a func

partial application

OCaml

```
let rec eval' (env :environment) (e :expr ) =
    let recurse = eval' env in
```

```
let sum_list =
  List.fold_left
    (fun a e → a + e)
    0
```

```
def add x y =
  x + y
end
def increase x =
    add 1 x
end
```

let increase = add 1 in

```
let f x y = x + y in
let g = f 1 in
let twice h z = h (h z) in
twice g 2 4
```

f sums args

g increments

twice : applies another function, but twice

$h^2(z)$

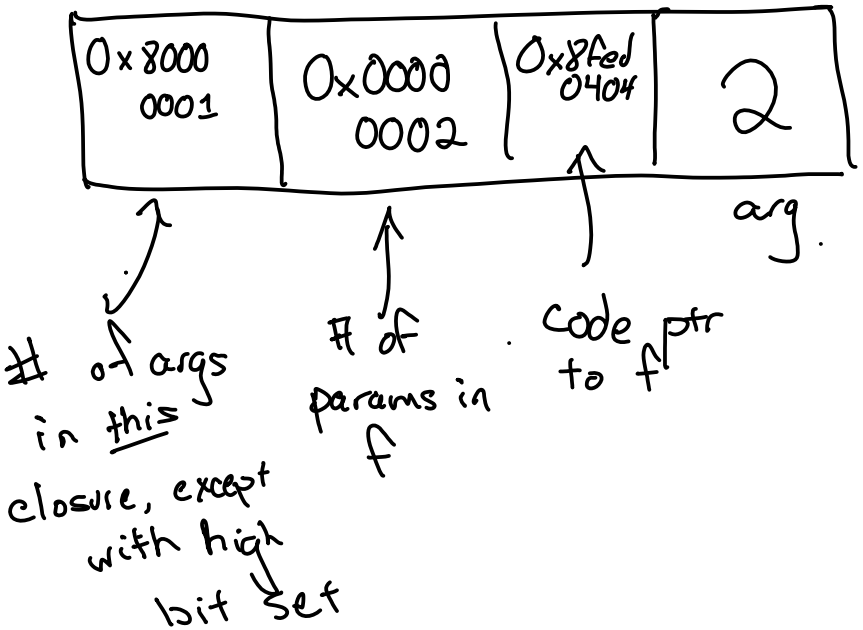
WHAT IS g?

- where the original code is
- what arguments have been applied

} closure

EAppl of expr * expr (f 1) z

EAppl (EAppl (*f*, EInt 1), EInt 2)



f 1

```
mov eax, _f
call eax
```