Goals for today

- more practice at parsing
- variable scoping

stack

(

define

sum

f

lambda a

b

a b

f

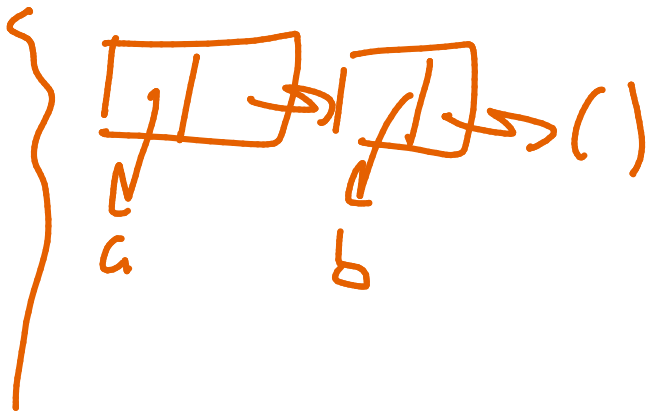
+

a

b

subtree → empty

a b



subtree → + a b

subtree → lambda a b + a b

lambda a b + a b

next page

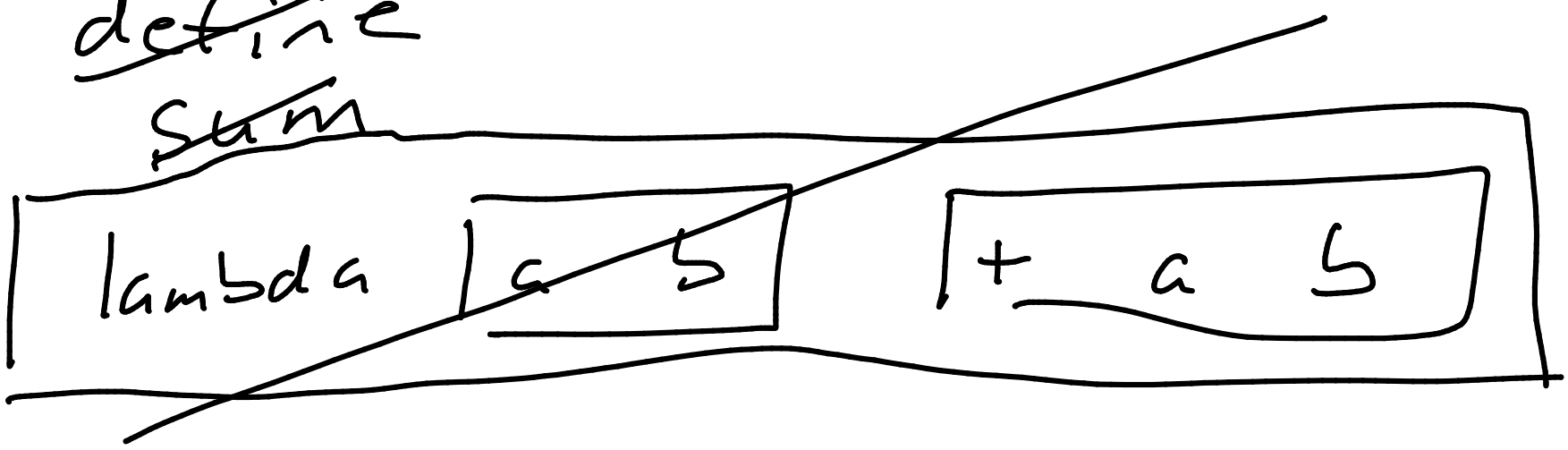
no work yet, just rewrite stack

stack

~~f~~

~~define~~

~~sum~~

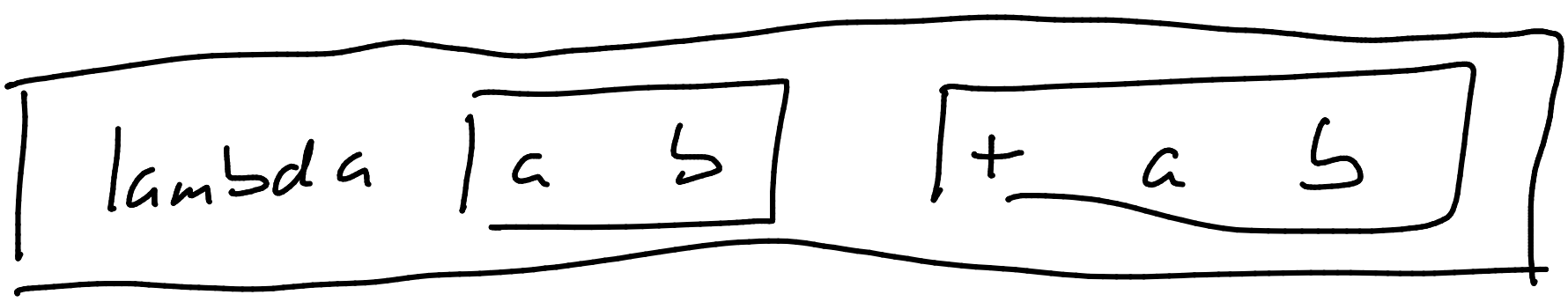


Last step: one more)

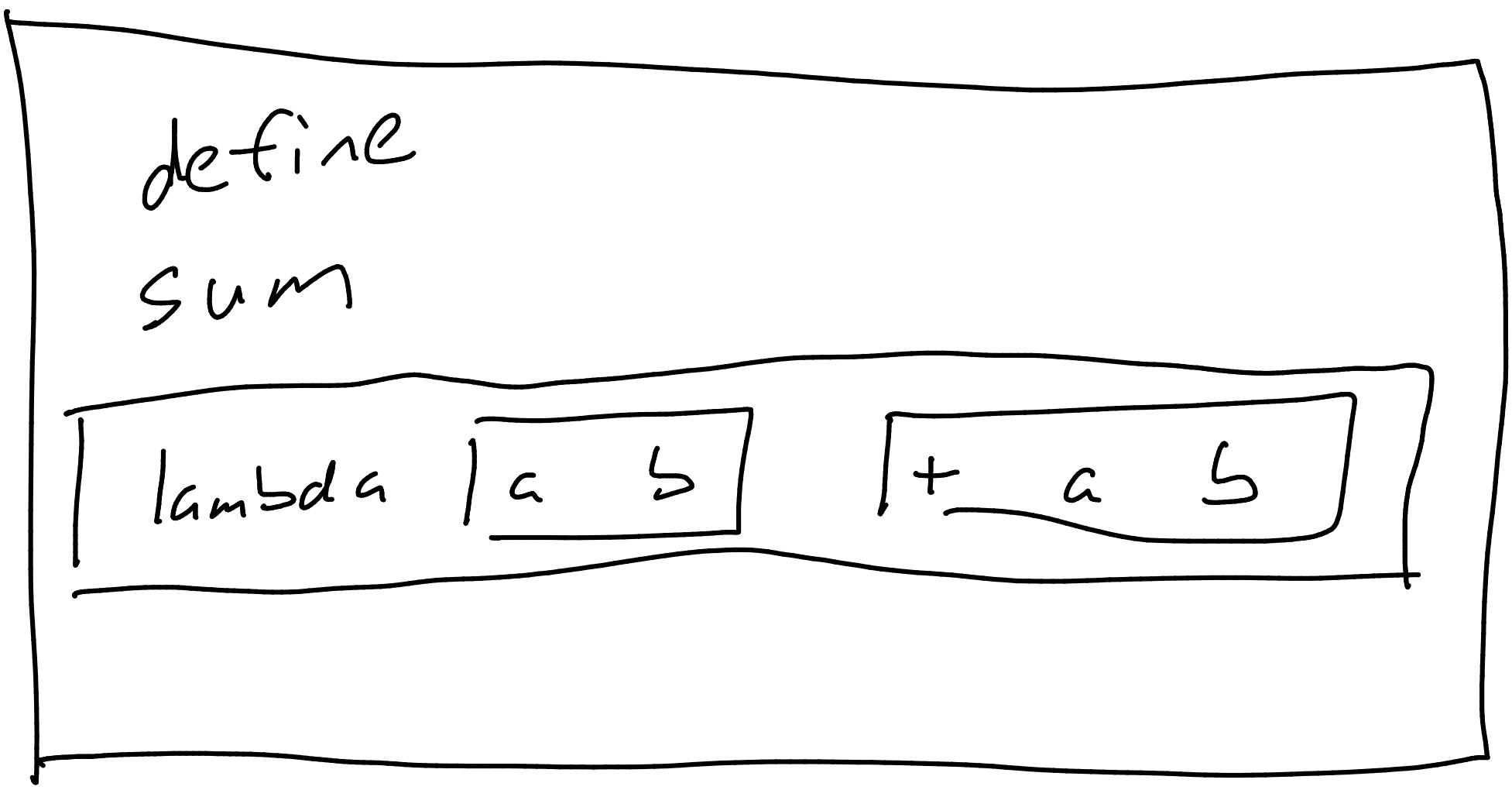
subtree ↓

define

sum



stack



Important detail: a Scheme program is a list of Scheme expressions.

Tokens:
a
b
(+ 3 5)
c

Variable scoping - which portions of code a variable is "alive" for.

Two main approaches

static scoping - a variable is visible based on structure of code (nested)

dynamic scoping - a variable is visible based on order of execution of code.

```
(define x 1)

(define fun1
  (lambda ()
    (let ((x 2))
      (fun2))))

(define fun2
  (lambda ()
    (display x)))

(fun1)
```

new local variable x

which x?

static scoping: global

dynamic scoping:
most recently seen
[in fun 1]