The order in which to do things in class for adding **let** to the interpreter.

1. Show the top-level eval slide, and write the rest on the board.

(define eval-exp

**(let ([identity-proc (lambda (x) x)])**

(lambda (exp **env**)

(cases expression exp

[lit-exp (datum) datum]

[var-exp (id) ; look up its value.

**(apply-env env**

**id**

**identity-proc ; procedure to call if var is in env**

**(lambda () ; procedure to call if var is not in env**

**(apply-env global-env ; was called init-env**

**id**

**identity-proc**

**(lambda ()**

**(error 'apply-env**

**"variable ~s is not bound"**

**id)))))]**

1. Add new case to eval-exp:

**[let-exp (syms exps bodies)**

**(let ([extended-env**

**(extend-env syms**

**(map (lambda (x)(eval-exp x env))**

**exps)**

**env)])**

**(let loop ([bodies bodies])**

**(if (null? (cdr bodies))**

**(eval-exp (car bodies) extended-env)**

**(begin (eval-exp (car bodies) extended-env)**

**(loop (cdr bodies))))))]**

1. Modifications to **app-exp** case: Add **env** to the calls to **eval-rands** and **eval-exp** .
2. Fix eval-rands

(define eval-rands

(lambda (rands **env**)

(map **(lambda (e)**

(eval-exp e **env**) rands)))

1. Ask whether any changes need to be made to **apply-proc** (the answer is no)

Add **if** to the interpreter.

**[if-exp (test true false) (if (eval-exp test** env**)**

**(eval-exp true** env**)**

**(eval-exp false** env**))]**

Show the transparency called **Interpreter after adding if and let**.

Add **lambda** to the interpreter.

1. In cases of eval-exp:

**[lambda (params body)** ; can handle multiple bodies as we did with let.

**(closure params body env)]**

1. **What should closure be?** Another case of proc-value

(define-datatype proc-val proc-val?

[prim-proc (name symbol?)]

**[closure (params body env)]**)

1. Enhance **apply-proc**

(define apply-proc

(lambda (proc-value args)

(cases proc-val proc-value

[prim-proc (op)

(apply-prim-proc op args)]

**[closure (params body env)**

**(let ([new-env (extend-env params args env)])**

**(eval-exp body new-env))]**

[else (error 'apply-proc

"Attempt to apply bad procedure: ~s"

proc-value)])))