Administrivia

- Project 3 Part A due 3/29 (Monday after SB)
  Part B due 4/5 (a week after)
  - Everyone with a partner that wants a partner?
  - Extra Office Hours on Sunday 3/28 in C50 from 1pm
- Midterm 3 on 4/14 (appr. 1 week after Part B is due)
  - Covers OOP to Concurrency (week 8-11)
- What to expect on week we come back…
  - Scheme-2 (the last of the interpreters before MCE)
  - Vectors/Arrays
  - Mutation…the list kind ☺

Environment Diagrams…

Practice…it's on!

Agenda

- Step by Step Environment Diagram Stuff
- Practice, Practice, Practice

The Rules

- What are **"The Rules"**?
  - They're a set of guidelines to follow when doing environment diagrams
  - Follow **The Rules** and you'll be an environment diagram MASTER!
  - Once you've mastered it, you never need to look at them again ☺ (But keep them for reference)
  - Remember…
    DON'T THINK, JUST DO! ☺

The Rules

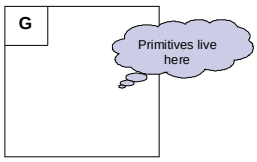
- EVERY expression typed into Scheme is either an **atom** or a **list**

So believe it or not…

STk > (define (a) 3) ;;← THIS is a LIST!

The Rules

- There is **ALWAYS** a current frame.
- Initially it's the **global environment**



The Rules: Atoms

- **Self-Evaluating:** Numbers, Strings, #t, #f

  Example:

  STk > 1 ;; no need to do anything

  1

- **Symbols:** (aka variables) look for first binding

  Example:

  STk > x ;; say if (define x 3)

  3

The Rules: Lists (aka Compound Expressions)

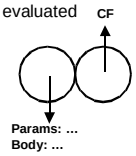
- Take the **car** of it, if it's a special form go to SPECIAL FORMs of **The Rules**
  i.e. (define (foo x) 3)
- Otherwise you're calling a procedure!
  i.e. (square 3)
  - So evaluate **ALL** subexpressions by **The Rules** then...
  - If **car** is primitive → apply by magic
    i.e. (+ 2 3) → *poof* it returns 5
  - If **car** is a λ then…
    - Create frame, f
    - Point f to where λ points
    - Bind formal parameters of λ in f & make f the current frame
    - Use **The Rules** to evaluate the body of λ in f

The Rules: Special Forms

- DESUGAR!
  - Define: (define (foo x) …) → (define foo (λ (x) …))
    - Write variable name in current frame
    - Evaluate body by **The Rules** in CF (current frame)
    - Point 1 (variable name) → 2 (evaled body)
  - Let
    1. ((λ (args) body) vals) ← just evaluate again by **The Rules**

The Rules: Special Forms

- λ → procedure (λ (params) body)
  - Draw Bubbles!
  - Left Bubble points to parameters and body
  - Right Bubble points to CF (current frame) where it's being evaluated



Super Simple Example

- What happens when we type:
  STk > (define x 3)

  First off everything from the STk prompt will be evaluated starting from the global environment. So this expression is saying…
  "evaluate the expression (define x 3) in the global environment"

Super Simple Example

- So what's next?
  STk > (define x 3)
- Let's look at **The Rules**
- Is it an Atom? **No!**
- Is it a List? **YES!**

Super Simple Example

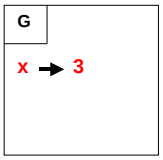
- So let's take the **car** of the expression **define** ← lookie it's a special form!
- Go to the Special Form section of **The Rules!**
- No need for any desugaring because the first argument to define is a variable not another list like (define (x) 3), so let's continue on…

Super Simple Example

(define x 3)
So it says in **The Rules** write the variable name in the current frame so…

then evaluate the body…

Then point 1→2



Tada!

- So that's an easy variable binding example.
- Let's do one more easy procedure and then we'll do more problems!

Another Example

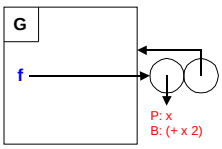
(define (f x) (+ x 2))

- So what do we do first with this expression?
- First off, it's a list, second off the car is **define** so…
- DESUGAR!

(define f (λ (x) (+ x 2)))

Another Example

1. Next Write the variable name in the current frame
2. Evaluate the body by **The Rules**
   1. (λ (x) (+ x 2))
   2. It's a list, the car's a λ so….
   3. Draw Bubbles!
3. Now Point 1→2

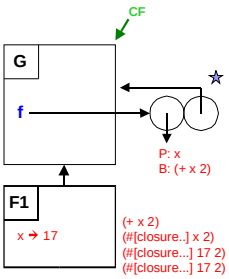


Another Example…Call

- So let's call the procedure
  STk > (f 17)
  So what happens now?
  - It's a list, the car is **NOT** a special form, so evaluate all the subexpressions
  - f is a procedure (let's call this ☆)
  - 17 is self-evaluating
  - So now you have (☆ 17)

Another Example…Call

- So take the car, it's just a λ, so
  - Create a frame, F1
  - Point F1 to where the right bubble of ☆ points
  - Bind formal parameters of ☆ in F1, make F1 the current frame
  - Use **The Rules** to eval the body of ☆ in F1
  - Returns 19 magically by '+'



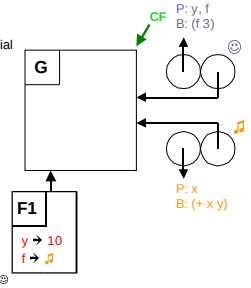
So LET's do this…

- Remember that you couldn't do
  (let ((y 10)
  (f (λ (x) (+ x y))))
  (f 3))
  just through intuition…but now see the REAL reason why…let's draw the environment diagram for this expression.

So LET's do this…

(let ((y 10) (f (λ (x) (+ x y))))
(f 3))

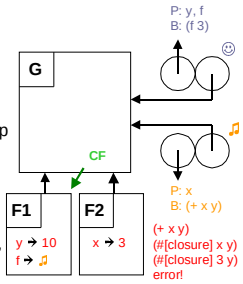
- It's a list! Take the **car**, it's the special form LET so…
- DESUGAR!
  ((λ (y f) (f 3))
  10
  (λ (x) (+ x y)))
- Evaluate all subexpressions
  - (λ (y f) (f 3)) → ☺
  - 10 → 10
  - (λ (x) (+ x y))) → ♫
- Now call the procedure ☺
  - Create a frame, F1
  - Point F1 to where ☺'s right bubble points
  - Bind formal parameters y & f in F1, make F1 the current frame
  - Use **The Rules** to eval the body of ☺ in F1



So LET's do this…

((λ (y f) (f 3))
10
(λ (x) (+ x y)))

- So the body of ☺ is: (f 3)
- It's a list, the car's not a special form, eval all subexp
  - (♫ 3)
- Now call ♫ on 3
  - Create a frame, F2
  - Point F2 to where ♫'s right bubble points
  - Bind formal parameters in F2, make F2 the current frame
  - Use **The Rules** to eval the body of ♫



Okay that's enough of that…

- So hopefully you're comfy with easy problems.
- Now let's do some more…don't you love me ☺