How’s it going?

Use the provided forms to give your feedback about CS 42. Here are some additional questions:

1. The pace of this class is...
   1 = way too slow; 4 = just right; 7 = way too fast

2. I’m learning a lot in CS 42.
   1 = strongly disagree; 4 = neither agree nor disagree; 7 = strongly agree

3. CS 42 is interesting.
   1 = strongly disagree; 4 = neither agree nor disagree; 7 = strongly agree

4. I know I can get help / support from (e.g., Prof. Ben, the grutors, Piazza), if I need it.
   1 = strongly disagree; 4 = neither agree nor disagree; 7 = strongly agree

5. When it comes to workload, so far, this is my hardest course this semester.
   1 = strongly disagree; 4 = neither agree nor disagree; 7 = strongly agree

6. I find the “sketches” helpful.
   1 = strongly disagree; 4 = neither agree nor disagree; 7 = strongly agree

Functional programming

Read: Can Programming Be Liberated from the von Neumann Style?...

What’s something you agree with?

What’s something you disagree with?

What’s something you don’t (yet) understand?

Why do you think the highlighted text is highlighted?

Can you sum up the reading in at most six words?
Some key features of functional programming

- No mutable state (no “assignments”).
- It’s all about evaluating expressions.
- Functions help us manage complexity.

Racket

What’s it like to learn a new programming language?

What kinds of things do we want to know first?

What does it feel like as you learn a new PL?

Dr. Racket, an Integrated Development Environment (IDE)

“Primitive” values

- integers
- booleans
- only false is false
- real numbers
- strings
- (lots more, non-primitive values)...

Examples / Questions:
**s-expressions**

\[(op \ arg_1 \ arg_2 \ ... \ arg_n)\]

**Rules:**

- the operation always comes first
- its arguments (if there are any) follow the operation
- no commas between arguments
- everything goes between parentheses

**Common mistakes:**

- forgetting parentheses
- rational vs. integer division (/ vs. quotient)
- equality (= vs. equal?)

**EXAMPLES / QUESTIONS:**

---

**“Variables”**

They’re called variables, but we won’t vary them (i.e., their values are constant).

\[(let* (\[var_1 \ expr_1\]
    ...
    \[var_n \ expr_n\])
    body-expr)\]

**Bindings:** an “assignment”. We say: “bind a value to a variable” or “bind a variable to a value”.

The body is the **scope** of the variables that are bound in the first part of the **let**.

**EXAMPLES / QUESTIONS:**
**Conditionals**

\[
\text{(if } \text{conditional-expr} \\
\qquad \text{true-expr} \\
\qquad \text{false-expr)}
\]

\[
\text{(cond} \quad [\text{condition}_1 \ \text{expr}_1] \\
\qquad \ldots \\
\qquad [\text{condition}_n \ \text{expr}_n] \\
\qquad [\text{else} \ \text{else-expr}])
\]

If you have more than one condition, use cond; otherwise use if.

**Examples / Questions:**

---

**Functions**

\[
\text{(define} \quad (\text{function-name} \ \text{parameter}_1 \ \ldots \ \text{parameter}_n) \\
\qquad \text{body-expr})
\]

Next time: lists in Racket