Course Information and Overview

Software Installation

- ✓ **Reading:** Part C Software Installation and Use: Ruby and irb 30 min
- ✓ **Reading:** Installing and Using SML and Emacs

Section-Wide Items

- ✓ **Reading:** Section 8 Welcome Message 7 min
- **Reading:** Section 8 Reading Notes
- **Reading:** Code Files for All Section 8 Videos

Video Lectures

Homework 6 Instructions

Homework 6 Peer Review

Community-Contributed Resources

Homework 6 (Ruby 2.0)

Homework 6 (Ruby 2.1)

Homework 6 (Ruby 2.2)

Homework 6 (Ruby 2.3)

Section 8 Welcome Message

In case it has been a while since you completed the modules in Part B, remember we aim to provide various materials that will help you:

- Video lectures covering all the topics you need for the homework
- Reading notes that cover the same material in a written format
- Tips and extra practice problems contributed by others who have taken the course

As we start learning to program in Ruby, the focus will be on two topics that distinguish Ruby from ML and (how we used) Racket:

- Ruby is *object-oriented.* We will discuss what this means: that all values are objects, where an object is something whose behavior is defined in terms of *methods*, which are like functions except that have access to the "[state of the] object they are a part of" (called `self` in Ruby and `this` in various other languages like Java and C#). No prior experience with object-oriented programming and methods is needed, but, conversely, even if you have substantial experience, you will probably learn much from our precise, from-the-beginning approach to the key ideas -- and how they form the essential difference from functional or procedural programming.
- Ruby is a very *dynamic language*. In addition to being dynamically typed, much like Racket but "even more so" in certain ways, Ruby allows an executing program to change definitions that in many languages would be "fixed at compile time".

This week's programming assignment is quite a bit different from the assignments in Parts A & B. As it explains, much of the challenge is understanding the substantial provided code and then using object-oriented programming features to reuse much of that code without modifying it. "Spending more time reading code than writing code" is a common "real-world" approach to software development, and so it makes for a nice experience for those of you have not done it much. For others who do often do this, you may find this assignment a little "less rewarding" or "easier". Even so, next week's more challenging homework gets back to some really juicy programming-languages things (another intrepreter!), so you can look forward to that...