Start Here!

Thank you for signing up for Programming Languages, Part C on Coursera. We are very glad you are here!

First things first, let's be clear that this is Part C of a 3-part course. We will assume you have completed Parts A & B and are eager to continue our challenging and rewarding study of Programming Languages. There is not really a good way to describe the necessary background for Part C other than to say that it builds on the material in Parts A & B in many ways. The introductory material in Part A discussed why we created a 3-part course. We don't repeat that discussion here, but you may wish to (re-)visit it.

Because the format of Part C follows the now-familiar pattern, we will not have an "introductory week" like we did in Part A -- there wouldn't be very much to do. You do need to install the software for programming in Ruby, but this will hopefully not take very long.

Part C has three "weeks":

- Week 1 has a lot of content. First, we will cover various Ruby basics focusing on how Ruby is (1) a *pure object-oriented language* (every value is an object with a class that defines its methods) and (2) a dynamically-typed language, with even more dynamic features than Racket. We will then focus on Ruby's basics and its approach to object-oriented programming. Then we will focus on Ruby's *blocks*, which are *almost* function closures. Lastly and most importantly, we will focus on *subclassing, inheritance, and method overriding* -- key features that distinguish object-oriented programming from other approaches. The homework assignment is different from the others in Programming Languages: you will use subclassing to reuse the code in a provided, working application (a Tetris game) to create a version with a few different behaviors.
- Week 2 is primarily focused on comparing and contrasting functional programming and object-oriented programming by showing how they best support "exactly opposite" ways to decompose programs into pieces, and we learn the double-dispatch idiom for a way to decompose certain computations in a (very?) object-oriented way. We also compare and contrast three related "advanced" features for object-oriented programming: *multiple inheritance*, Ruby's *mixins*, and Java-style *interfaces*. The total length of the lectures is less than in most weeks, but the homework -- the last one in programming languages -- is more challenging, synthesizing a lot of ideas by having you port a small interpreter for a "domain-specific language" from ML to Ruby.
- Week 3's new content is more conceptual/theoretical, covering *subtyping*, how the general idea of subtyping naturally arises in statically-typed object-oriented languages, how subtyping contrasts with ML-style polymorphism (generic types), and how subtyping can be combined with generics synergistically. Week 3 has no programming assignment. Instead it has a "final exam" that focuses on the material in *Part B and Part C*. Like for the exam at the end of Part A, there is a reading with information about the exam, and there is a practice exam. And then we will be done, with a short wrap-up lesson congratulating you for reaching the very end of this three-part adventure!

Note that you may find Weeks 2 and 3 shorter than usual because, in fact, they used to be part of the same week. Since the topics are nicely separable, we have "spread them out" but, of course, you can finish in less than (or more than) 3 weeks if you prefer.

The introductory videos in this first "lesson" provide a little more detail welcoming you to "Part C", discussing the topics ahead, and describing the structure of the course.

[As a final detail, in Part C, each section (e.g., 8) has a homework number that is less than the section number (e.g., 6). This is because Section 4 of Part A and Section 7 of Part B do not have homeworks. We know this is a little confusing, but skipping a homework number is probably at least as confusing to some people.]