# Miscellaneous Lesson #1: On the Design of this PL Course

## What's It All About?

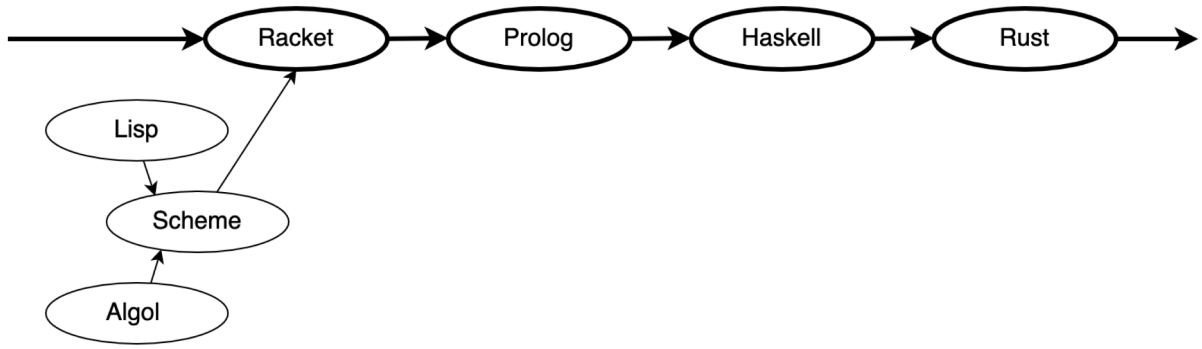
This "lesson" is intended to communicate something about the design of this course. Perhaps knowing something about its design will help you to see some big picture aspects of the course that you are taking.

## Design of the Course

One of the great pleasures of being a professor is that you get to **create** the courses that you teach! For me, creation is a process governed by **constraints**. (For more on this, you might like to read "Creativity from Constraint: The Psychology of Breakthrough", by Patricia Stokes. Or, if you want to go deep, you might read some of what the really creative people say about creativity and constraint, like Stravinsky with respect to music, or Poincare with respect to mathematics.) In order to guide my course creation, I determined to adopt four constraints to serve as guiding principles for the programming language tale that I intend to tell you: The "programming languages constraint", the "educational philosophy constraint", the "big ideas constraint", and the "historical narrative" constraint.

## The Programming Languages Constraint

The following image stands for the actual programming languages that I have determined to foreground in this course.




What do the arrows mean? The arrows with the narrower shafts stand for "influenced". Thus, you can see from the image that Scheme influenced Racket, and both Lisp and Algol influenced Scheme. The arrows with the broader shafts stand quite simply for "the next language to be considered (for reasons that the attentive student may discern as the course unfolds) in this course" within the sequence of four real world languages that will be foregrounded in the course.

## The Educational Philosophy Constraint

The teaching model for this course will feature three (sometimes competing) values:

1. Central control of content, as presented by the unfolding course narrative.
2. Collective focus of attention on the featured material.
3. Constructionist engagement with respect to learning about the featured languages and concepts.

## The Big Ideas Constraint

Programming languages are composed of constructs grounded in concepts. The course will incorporate, among others, the following languages, constructs, and concepts:

> Algol – anonymous functions – assembly language – BNF – closures – compiler – context free languages – curried functions – dynamic scoping – dynamic typing – Fortran – functional programming – garbage collection – Haskell – higher order functions – imperative programming – interpreter – Java – lazy evaluation – Lisp – lexical/static scoping – lexical/static typing – literate programming – logic programming – machine language – object-oriented programming – ownership – parser – parse tree – Prolog – Racket – recursion – regular expressions – resolution – run time heap – run time stack – Rust – scanner – Scheme – Smalltalk – token – type – unification

Some of the terms will merely mentioned in passing, since you will already possess considerable knowledge surrounding them. Other terms will be considered at length.

## The Historical Narrative Constraint

One approach to historical narrative is to tell an engaging story of historical events throught the eyes of a fictional character in a manner that is coherent, yet strives not to distort the facts. The story of programming languages told in this course purports to be one of historical fiction. Time will be used as an organizing theme. Individuals who contributed in one way or another to programming language ideas and constructs will make appearances. Care will be taken not to distort the historical record. If there should be some question about fidelity to real history as the story unfolds, please blame it on the fictional character, a ragged old computer science professor, not me.

## One Last Thought

The theme of this little lesson is "design by constraint". By way of example, I did my best to suggest how the course that I have designed for you this semester is grounded in four constraints. But it can be helpful to think about any number of things with this **design by constraint** idea in mind, programming languages being a case in point.

Programming language design is largely a exercise in (1) establishing a set of constraints, and then (2) designing a language that is consistent with the constraints. **As we proceed to consider specific programming languages in this course, you might like to think about what constraints the designer (sometimes a single human, sometimes a committee) might have adopted as a framework for the design of the language.**