Programming Abstractions

My name is

BOB Geitz

About the class:

This is a course about Programming Languages

We will use the language Scheme to discuss, analyze and implement various aspects of programming languages.

The course has 4 parts:

A. Scheme and things you can do with it (5 weeks)
B. Logic Programming, and Prolog (2 weeks)
C. Implementing Scheme and other languages (4 weeks)
D. Advanced issues (delayed evaluation, continuations, etc.) (2 weeks)

A Quick History of Scheme

- John McCarthy invented LISP at MIT around 1960 as a language for AI.
- LISP grew quickly in both popularity and power. As the language grew more powerful it required more and more of a system's resources. By 1980 5 simultaneous LISP users would bring a moderately powerful PDP-11 to its knees.
- Guy Steele developed Scheme at MIT 1975-1980 as a minimalist alternative to LISP.
- Scheme is an elegant, efficient subset of LISP. It has some nice properties that we will look at that allow it to be implemented efficiently. For example, most recursions in Scheme turn into loops.

Why Scheme for CSCI 275?

- All LISP-type languages have lists as their main data structure; their programs are written in lists. This means it is easy to have programs take other programs as input. Scheme programs can reason about other programs. This makes Scheme useful for thinking about programming languages in general.
- Scheme is a different programming paradigm. Python, Java, C and other languages are <u>imperative</u> languages. Programs in these languages do their work by changing data stored in variables. Scheme programs can be written as <u>functional</u> programs -- they compute by evaluating functions and avoid variable assignments.
- Scheme is very elegant. It is much less verbose than Java, which means it is easier to see what is happening in a Scheme program.
- It is fun!

The course

- We will have about 10 labs (I call them labs; they are assignments you do on your own). You will have between 7 and 10 days for each lab. We will have one exam in early October, another in mid-to-late November, and a final exam.

- It is very important that you do the labs on time. This course differs from other CS classes in that each lab has a lot of small, independent parts. You can get some of it done and running even if it isn't all done. So hand the lab in on time, even if it isn't completely finished.   If you fall behind in the labs you won't follow the lectures and you will be lost.

We will use Dr. Racket as our Scheme interpreter. The history of this is rather tortured. First there was a group called PLT (Programming Language Theory) containing Matthias Felleisen, Shriram Krishnamurthy, Robby Findler and others. They produced a popular Scheme interpreter called Dr. Scheme. Over the years a catfight developed over what features belonged in "real" Scheme. Eventually the PLT group started calling its language "Racket" instead of Scheme. The interpreter changed its name from "Dr. Scheme" to "Dr. Racket". Feelings were hurt. No one says "Racket is really Scheme", though it is.

You can obtain Racket for free (Windows, Mac or Linux). See
www.racket-lang.org

Look at the syllabus.

The class website is www.cs.oberlin.edu/~bob/cs275

For this week:

- Read Chapters 1 - 3 of The Little Schemer

- Do Lab 1; this is due on Thursday of next week, September 12. In this class, when I say something is due on day X, that means it is due at 11:05PM at the end of day X.