Advanced Progamming Languages	**Lecture 1**
CS 6110 Spring 2015	**Wed. Jan. 21, 2015**

Lecture 1

Topics

1. Course mechanics
2. Course content - overview, schedule

Course Mechanics

- 42 lecture times scheduled, one for midterm, 3 guest slots.
- *Abhishek Anand* is the teaching assistant, he will give at least one guest lecture.
- Assignments:
  - Problems sets, one may involve reading a classic paper and relating it to the course.
  - Final exam.
  - In-class midterm exam, Friday March 6.
  - Options to use the Coq proof assistant and programming language
  - Planned dates of problem set assignments:
    1. Jan 26
    2. Feb 9
    3. March 9
    4. March 23
    5. April 13
    6. April 27, May 6 due date for last problem set

What is this course about? Why do we offer it?

Textbooks

1. Simon Thompson *Type Theory and Functional Programming*, free at course website
2. Other widely used textbooks

   Pierce *Types and Programming Languages*, 2002

   Harper *Practical Foundations for Programming Languages*, 2013

   Mitchell *Foundations for Programming Languages*, 1996

What do these books say about the importance of the subject?
Why is such a course offered at Standford, Berkeley, CMU, Cornell, Illinois, Texas, Harvard, Princeton, Yale, Stony Brook, etc., as well as in the UK, France, Israel, etc.

Thompson says "correctness of progress is a key issue in computer science."

I would say this: Programming languages are at the core of computer science because we made them the central tool for doing our work of building software systems. The process of creating programming languages and using them is *autocatalytic*. The better they are, the better they become.

We use them to carry out the key work of computer science, *implementing intellectual processes and studying them*. This core activity now lies at the heart of all sciencee and engineering and most scholarship, especially mathematics.

As the key work of computer science expands to include facilitating social processes and providing access to most of organized human knowledge, the role of programming languages is expanding.

Can we imagine a point where the role of PL will change? Are we building toward a singularity? What programming methods will get there first? How will this change PL?

The *Handbook of Theoretical Computer Science*[1] divides theory into two parts, A and B. This course is mainly area B, but we will make connections to A as well. The US and Israel dominated area A and the EU area B in the early days. Researchers are making the claim that *type theory*, an area B topic, will provide an excellent foundation for both areas A and B.

[1] *Handbook of Theortical Computer Science* J. von Leeuwen
Vol. A *Algorithms*
Vol. B *Formal Methods and Semantics*

Standard Topics

1. The lambda calculus and pure foundational programming.
2. Lambda calculus mechanisms in widely used languages.
3. Applied lambda calculus.
4. Typed lambda calculus.
5. Partial types.

   Polymorphic typing, normalization
6. Evidence and proofs.

   Propositions as types
7. State/reference types.
8. Inductive and co-inductive types.
9. Molecules and classes.
10. CS-Logic correspondence.
11. Type theory.
12. Processes and event types.