Computer Science 112: Comparative Programming Languages
Spring 2003

The primary course goals are:
- Develop an understanding different programming language paradigms
- Study the major principles which underlie all programming languages
- Make learning new programming languages easier
- Enable fuller use of existing programming languages
- Provide a foundation in support of programming language selection, design and development

This course has two tightly coupled components, each of which comprises about one half on the course content. One component consists of a study of programming language paradigms, principles underlying programming languages and language features. The second component is detailed discussions of and hands-on experience with four or five selected programming languages. The selected languages include:
- Scheme, a dynamically typed functional language
- SML, a type-inferred, strongly typed functional language
- Smalltalk, a “pure” object-oriented language
- Prolog, a logic language

The principles component of the course includes:
- The lambda calculus
- Denotational semantics
- Type systems
- Scope
- Higher order functions
- Sequential control
- Modularity
- Object-oriented programming
- Concurrent and distributed programming

There are two midterms and a final examination. Course grades are based on five programming assignments and the examinations.