Course Summary

• Programming Languages
• Translation of Programming Languages
• Tools
• Construction of Complex Software
• Other Applications of Static Analysis

Programming Languages

• Scope of declarations
• Scope vs. extent (lifetime) of variables
• Interactions between language design and runtime structures:
  - Function representation
    * Effects of recursion, variable-sized data, functional values
  - Inheritance
    * Single vs. multiple inheritance
    * Java-style interfaces
• Example: Pyth and Python: compare and contrast

Translation of Programming Languages

• Lexical analysis
  - regular expressions, finite automata
• Context-free syntax
  - BNF
  - Top-down, recursive descent
  - Bottom-up, shift-reduce parsing
  - Terminology: derivation
  - Syntax-driven translation
• Static semantics
  - Symbol tables, relation to block structure
  - Types, type inference

Translation of Programming Languages, contd.

• Code generation
• Runtime representations for "special effects"
  - Exceptions
  - Procedure calls
  - Object-oriented method dispatch
  - Garbage collection
• Optimization
  - Terminology: basic blocks, control-flow graph
  - "Classical" optimizations
  - Structure of flow analysis
Tools

- Lexer-generation, use of regular expressions and states
- Parser generators, rule-based programming

Construction of Complex Software

- Be familiar with project, including parts you didn't write.
- Concept of a "pass"
- Use of object-orientation to partition task
- Importance of intermediate forms; how used to reduce work of porting compilers

Other Applications of Static Analysis

- Verifying program properties.
- Insuring security of a platform (Java bytecode verifier)
- Analysis that improves performance.