What Is Computer Science?

The Scientific Study of Computation

- Expressing or Describing
- Automating
- Understanding or Reasoning About

What Is Our Most Fundamental...

- Tool?
- Contribution?
- Activity?

My answer is probably obvious!

The History of Computer Science

- Parallels or
- Is demarcated by or
- Has advanced due to or
- Has been driven primarily by or
- Consists of little other than...

Development of Programming Languages!

Some Historical Milestones

- Machine language
- Assembly language
- FORTRAN
- Lisp
- Algol

Some More Historical Milestones

- Prolog
- C
- Simula
- C++
- ML
Still More Historical Milestones

- Ada
- Scheme
- Haskell
- Java
- C#

A Brief History of CMPSCI 630

- Pre-Java (the bad old days)
- Early Java
- Middle Java
- Harper and POPL
- Separation and Scala

Why Study Programming Languages?

Programming is an explanatory activity.

- To yourself, now and in the future.
- To other developers and maintainers.
- To the machine!

Why Study Programming Languages?

Therefore the language we use matters enormously.

- How to write a program to solve a problem?
- How to express your assumptions and guarantees?
- How to document the structure of a system?

Why Study Programming Languages?

There is (or ought to be) a close relationship between

- the code, which is executable, and
- its properties, which are descriptions of its behavior.

Good languages make it easier to establish, verify, and maintain the relationship between code and its properties.

The Science of Programming Languages

You may have heard that all programming languages are equal.

- All familiar languages are Turing equivalent — they express the same set of computable functions as each other.
- Therefore (?) there’s no difference between them. It’s all a matter of taste.
The Science of Programming Languages

Yet we all know that some languages are more equal than others.

- Do you want to build web pages in assembly language?
- Do you want to build a device driver in Perl?
- Do you want to use C as your markup language?

There is also a **scientific basis** for programming languages whose primary tools are:

- **Type theory.** Techniques for structuring languages to ensure safety and modularity of programs.
- **Operational semantics.** Techniques for describing the execution behavior of programs, at various levels of abstraction.
- **Mathematical logic.** Techniques for specifying and verifying programs.

The Goal Of This Course

The goal of CMPSCI 630 is to introduce the **fundamental principles** of programming language design and implementation.

- Emphasis on rigor and elegance.
- Emphasis on both theory and practice.

We will model a wide variety of programming concepts in the framework of type theory and operational semantics.

Some Topics We’ll Discuss

**Theoretical foundations.**

- Inductive definitions.
- Structural induction.

Syntactic structure.

- Concrete syntax: the strings you type.
- Abstract syntax: the “deep structure” of a language.
  1. First-order: tree structure.
Some Topics We’ll Discuss

Semantics of languages.
• Static semantics, or type systems.
• Dynamic semantics, or execution rules.
• What is a safe language?

Language concepts:
• Higher-order functions.
• Abstract machines.
• Imperative programming.

Some Topics We’ll Discuss

More language concepts:
• Continuations and concurrency.
• Dynamic typing.
• Lazy evaluation.
• Parallelism.

Yet more language concepts:
• Modularity and data abstraction.
• Polymorphism and parametricity.
• Inheritance and subtyping.

Some Things We Won’t Discuss

Taxonomy.
• Not a trip to the zoo.
• Not a survey of the “Top Ten” PL’s.
• Not a way to collect C.V. items.

What’s CMPSCI 63X Like?

It will treat both basic concepts and a real instance.
• Basic concepts via formalism and abstraction
• Real instance via an interesting new language: Scala

Emphasis on rigor, clarity, and elegance:
• Not bound by flaws or limit of specific languages
• But can draw conclusions about specific languages
What Will I Get Out of CMPSCI 630?

After taking this course you should be able to:

• Confidently critique existing languages
• Define and analyze your own language
• Prove properties of languages
• Avoid common mistakes and pitfalls

What Will I Get Out of CMPSCI 630?

After taking this course you should also be able to:

• Reflect more deeply on programming style
• Write better programs (?)
• Carry out research on programming languages

What's CMPSCI 630 Like?

Most of all, it’s fun!

• Elegant interplay between theory and practice.
• Lots of interesting and novel ideas.
• Plenty of scope for further work.

People

Professor:

Jack Wileden
206 Computer Science Building
wileden@cs.umass.edu
Hours: Monday 3:30-4:30 & Thursday 2:30-3:30 or by appointment

Teaching Assistant / Grader:

TBA

Web Pages

The course web site is the authoritative source for all course information.

• Course syllabus, including slides and notes.
• Assignments, due dates, submission instructions.
• Announcements.
• Course policies

URL: http://www-edlab.cs.umass.edu/cs630.

Primary Textbook

Practical Foundations for Programming Languages by Robert Harper.

• Working draft available on course web site.
• Subject to change as the semester progresses, so doled out piecemeal.
Scala Books

• **Scala By Example** by Martin Odersky.

• **Scala Language Specification** by Martin Odersky.

These and other Scala references available on course web site.

Homework Assignments

Expect six assignments.

• Approximately two weeks each.

• Some written assignments

• Some programming assignments

• Some may be both

Grades

Homework: 70%.

• Roughly half for written, half for programming.

Project: 30%.

• More details on this soon.

Academic Integrity

You must read the material on the web site to receive a grade in this course!

• Bottom line: all work must be solely your own.

• Must acknowledge you’ve read the rules; part of first homework assignment (Homework 0 – see the course web site).

Any Questions?