CSc 372
Comparative Programming Languages

The University of Arizona
Fall Semester, 2006

Introduction

Instructor

Teaching philosophy

Course topics

Syllabus Highlights

Instructor

- William Mitchell
- Consultant/contractor specializing in software development and training of software developers. Practice includes Java, C++, C, Icon, object-oriented methods, and programming language implementation.
- Occasionally teach courses in programming languages at The University of Arizona (CSc 328, 352, 372, 451).
- Education: BSCS (North Carolina State University, 1981), MSCS (The University of Arizona, 1984).
- Lecturer, not a professor.

Teaching Philosophy

- I work for you!
- My goal: everybody earns an "A" and spends less than ten hours per week on this course, counting lecture time.
- Effective use of office hours, e-mail, IM, and the telephone can equalize differences in learning speed.
- I should be able to answer every pertinent question about course material.
- My goal is zero defects in slides, assignments, etc.

Course Content

At least three-fourths of our time will be spent studying three programming languages, and the programming paradigms they best support:

- Functional programming with ML.
- Fun with Ruby, a “scripting language”.
- Logic programming with Prolog.

The balance of the course will spent on various topics of interest in programming languages, possibly including material on Icon, Lisp, Scala, Aspect Oriented Programming, the “big picture”, and more (or less).

Syllabus Highlights

- Instructor
- Teaching Assistant
- Prerequisites
- Texts
- Grading Structure
- Assignments
- Bug Bounties
- Quizzes
- Computing Facilities
- Office Hours
- E-mail
- IM
- Telephone
- Mailing List
- Original Thoughts
- (**NO!**) Cheating

Important: Read through the syllabus before the second class meeting.

Basic Questions

What is a programming language?

Why study programming languages?

How many programming languages are there?

When did languages come into being?

...and more...

What is a programming language?

A simple definition:

A system for describing computation.

It is generally agreed that in order for a language to be considered a programming language it must be *Turing Complete*.

One way to prove that a language is Turing Complete is to use it to implement a *(Universal) Turing Machine*, a theoretical device capable of performing any algorithmic computation.

Why study programming languages?

- Learn new ways to think about computation.
- Learn to see languages from a critical viewpoint.
- Improve basis for choosing languages for a task.
- Add some tools to the “toolbox”.
- Increase ability to design a new language.

Speculate: How many programming languages is a typical software developer fluent in?

When did various languages come into being?

Plankakül	1945	Smalltalk	1972
Short Code	1949	ML	1977
FORTRAN	1957	Icon	1979
ALGOL	1958	Ada	1983
COBOL	1959	C++	1983
LISP	1960	perl	1987
BASIC	1964	Haskell	1988
PL/I	1965	Python	1990
SIMULA 67	1967	Java	1994
Pascal	1971	Ruby	1995
C	1972	C#	2000
Prolog	1972	Scala	2003

A pretty good family tree of prominent languages:

http://www.digibarn.com/collections/posters/tongues/

How many programming languages are there?

http://groups.google.com/groups/dir?&sel=33583294&expand=1
USENET comp.lang at Google

http://dir.yahoo.com/Computers_and_Internet/Programming_and_Development/Languages

http://en.wikipedia.org/wiki/Alphabetical_list_of_programming_languages

http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm

http://hopl.murdoch.edu.au/
HOPL: An interactive Roster of Programming Languages

Language development at UA

The University of Arizona Department of Computer Science has a history of developing programming languages. Here are some of them:

Cg	S
EZ	Seque
Icon	SIL2
Leo	SL5
MPD	SR
Ratsno	SuccessoR
Rebus	Y

How do programming languages help us create programs?

- Free the programmer from details

```
int i = 5;

x = y * z + q;
```

- Detect careless errors

```
int f(String s, char c);
...
i = f('i', "Testing");
```

- Provide constructs to succinctly express a computation

```
for (int i = 1; i <= 10; i++)
    ...
```

How do languages help..., continued

- Provide portability

  Examples: C provides moderate source-level portability. Java was designed with binary portability in mind.

- Provide for understanding by other persons

- Facilitate working in a particular style, such as imperative, functional, or object-oriented.

How are programming languages specified?

There are two facets to the specification of a language:

Syntax:
Specification of the sequences of symbols that are valid programs in the language.

Semantics:
Specification of the meaning of a sequence of symbols.

Consider this expression:

```
a[i] = x
```

What are some languages in which it is valid? What are the various meanings of it?

Some languages have specifications that are approved as international standards. Others are defined by little more than the behavior of a lone implementation.

How can languages be evaluated?

- Simplicity ("mental footprint")
- Expressive power
- Readability of programs
- Reliability of programs
- Run-time efficiency
- Practical development project size
- Support for a style of programming
- Popularity

What factors affect the popularity of a language?

- Available implementations
- Available documentation
- Vectors of “infection”
- Ability to occupy a niche
- Availability of supporting tools, like IDEs

Language philosophy

What is the philosophy of a language? How is that philosophy exhibited?

C

- Close to the machine
- Few constraints on the programmer
- High run-time efficiency
- "What you write is what you get."

C++

- Close to the problem being solved
- Support object-oriented programming
- "As close to C as possible, but no closer." — Stroustrup

PostScript

- Page description
- Intended for generation by machine, not humans

What is the philosophy of Java?