Comparative Programming Languages

CSC 372
Spring 2014

psst...Sign up for Piazza while you're waiting!

Instructor

William Mitchell (**whm**)

Lecturer, not a professor.

Education:

BS CS (North Carolina State University, 1981)

MS CS (University of Arizona, 1984)

Not "Dr. Mitchell" or "Professor Mitchell"

1993-2009, and August 2013-?

Consultant/contractor doing software development and training of software developers. Lots with Java, C++, C, ActionScript, and Icon.

2009-August 2013:

Research Programmer for SISTA. Worked on AnimalWatch, Teach Ourselves, and AnimalWatch Visually Impaired Suite.

Occasionally teach a CS course. (CSC 337, 352, 372, and others)

cs.arizona.edu says...

CSC 372: Comparative Programming Languages

This course is a study of several modern programming languages and the programming paradigm that each language strives to accommodate. Functional programming is studied with ~~ML~~ Haskell. Logic programming is studied with Prolog. ~~Icon~~ Ruby is studied to provide an alternative perspective on traditional ~~procedural~~ OO programming.

For each language we will study data types, control structures, syntax and semantics, idiomatic constructs, translation into executable units, and the run-time environment. In some cases we will go behind the scenes to examine implementation of language elements.

An emphasis of the course will be to understand the design philosophy of each language and how that philosophy is exhibited in the elements of the language.

Topic Sequence

- Functional programming with Haskell
- Imperative and object-oriented programming using dynamic typing with Ruby
- Logic programming with Prolog
- Whatever else in the realm of programming languages that we find interesting and have time for.

Note: The languages are vehicles for exploring different ways of programming. We'll cover representative elements of the languages, not everything.

Syllabus Highlights

Prerequisites

- CSC 127B or CSC 227

Piazza

- Our forum
- Sign up if you haven't already!

Teaching Assistants

- Yunhao Xu
- Illyoung Choi
- Learn their names; they'll know yours!

Syllabus, continued

Textbooks...

- No texts are required!
- Lectures, handouts, and Piazza postings might be all you need.
- Syllabus has recommendations for supplementary texts.

Syllabus, continued

Grading

- Assignments 55%
- Pop quizzes 5%
- Two mid-terms 20% (10% each)
- Final 20%

Ten-point scale: >= 90 is A, etc. Might go lower.

Syllabus, continued

Assignments—things like:

- Coding in the various languages
- Short answer and essay questions
- Diagrams
- Maybe stuff on D2L or elsewhere

Late assignments are not accepted!

No late days!

But, extensions for situations beyond your control.

Syllabus, continued

Bug Bounties

- One assignment point for each

Original Thoughts

- Half-point on final average for each

Syllabus, continued

Office Hours:

- I love office hours!
- Open-door policy in general
- Guaranteed hours posted on Piazza
- In-person is most efficient
- Skype preferred for IM
- **join.me** preferred for screen sharing
- OK to call my mobile but don't leave voice mail! (Send e-mail instead.)

NO CHEATING!

Capsule summary:

Don't cheat in my class!

Don't make it easy for anybody else to cheat!

One strike and you're out!

First offense? Plan on this:

Failing grade for course

Permanent transcript annotation

Disallowance of GRO for failing grade

Recommendation for one semester suspension

A typical first step on the road to ruin is sharing your solutions with your best friend, roommate, etc., who swears to just learn from your work and absolutely not turn it in as their work.

My Teaching Philosophy

- I work for you!
- My goal: everybody earns an "A" and averages 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.
- Everything I'll expect you to know on exams will be covered in class or on assignments.

READ THE SYLLABUS!

Assignment 0

Assignment 0

- On Piazza
- Due Friday, January 17, 10:00am
- Worth10 points (1% of final grade)
- Maybe 10 minutes to complete
- "Tell me about yourself..."—builds networks
- Thanks for doing it!

Pictures & name memorization

Basic questions about programming languages

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.

Q: What language is most commonly mis-listed on resumes as a programming language?

Does language choice matter?

Two extreme views on languages:

- If you've seen one language you've seen them all. Just pick one and get to work.

- Nothing impacts software development so much as the language being used.

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?

How old are programming languages?

Plankakül 1945
Short Code 1949
FORTRAN 1957
ALGOL 1958
COBOL 1959
LISP 1960
BASIC 1964
PL/I 1965
SNOBOL4 1967
SIMULA 67 1967
Pascal 1971
C 1972
Prolog 1972
Smalltalk 1972
ML 1977
Icon 1979

Ada 1980
C++ 1983
Objective-C 1983
Perl 1987
Haskell 1990
Python 1990
Java 1994
JavaScript 1995
Ruby 1995
C# 2000
Scala 2003
F# 2005
Clojure 2007
Go 2008
Dart 2011
Corelet 2013

How many languages are there?

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

(650+/-)

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

"about 2500"

Bottom line: Nobody knows how many programming languages have been created!

A pretty good family tree of prominent languages:

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

How do languages help us?

Free the programmer from details

```
int i = 5;
x = y * z + q;
```

Detect careless errors

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

Provide constructs to succinctly express a computation

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

How 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 using a paradigm, such as functional, object-oriented, or logic programming.

How are 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.

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

How are languages specified?

Consider this expression:

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

What are some languages in which it is syntactically valid?

In each of those languages, what is the meaning of it?

Building blocks

What are the building blocks of a language?

- Data types
- Operators
- Control structures
- Support for encapsulation
  - Functions
  - Abstract types (classes in Java)
- Error/exception handling
- Standard library

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 popularity?

- Available implementations
- Documentation
- Community
- Vectors of “infection”
- Ability to occupy a niche
- Availability of supporting tools, like debuggers and IDEs
- Cost

The philosophy of a language

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 both machine and 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?

UA's language heritage

The UA CS department was founded by Ralph Griswold in 1971. (Hint: know this!)

Griswold was Head of Programming Research at Bell Labs before coming to UA.

Griswold and his team created the SNOBOL family of languages at Bell Labs, culminating with SNOBOL4.

UA's heritage, continued

In the 1970s and 1980s, UA CS was recognized worldwide for its research in programming languages.

Here are some of the languages created here:

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

Along with language design, lots of work was focused on language implementation techniques, too.