Comparative Programming Languages

CSC 372

Spring 2018

cs.arizona.edu/classes/cs372/spring18

Stranger Danger

Introduce yourself to your tablemates while we're waiting to launch!

Instructor

William Mitchell (**whm**)

- Consultant/contractor doing software development and training of software developers. Lots with Java, C++, C, Python, Ruby, Icon, and more. Linux stuff, too.
- Occasionally teach a CS course. (337, 352, 372, and others)
- Adjunct instructor, not a professor
- Education:

  BS CS (North Carolina State University, 1981)

  MS CS (University of Arizona, 1984)
- Incorrect to say "Dr. Mitchell" or "Professor Mitchell"!

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: We'll cover a selection of elements from the languages, not everything.

Themes running through the course

- Discerning the philosophy of a language and how it's manifested
- Understanding tensions and tradeoffs in language design
- Acquiring a critical eye for language design
- Assessing the "mental footprint" of a language
- Learning how to learn a language (LHtLaL)

Syllabus highlights

(cs.arizona.edu/classes/cs372/spring18/syllabus.html)

Prereqs, Piazza, Mail

Prerequisites

- CSC 120 or 127B or 227
- But, this is a 300-level class!

Piazza

- Our forum
- Sign up if you haven't already!
- Private posts disabled—use mail
- If staff starts a thread, the first post is required reading

Mail

- 372s18@cs.arizona.edu goes to whm and TAs
- **For anything more than "Thanks!" use "Reply All" to follow the Cc:'s**

Teaching Assistants

Undergraduate TAs

- Alex Koltz (**akoltz**)
- Michael Ordaz (**michaelaordaz**)

Office hours and contact information is on website

Textbooks

- No texts are required.
- Lectures, handouts, and Piazza postings might be all you need.
- Syllabus and slides have recommendations for supplementary texts, most of which are on *Safari Books Online*.
- Suggested supplementary readings will be provided (but alignment with our topics and sequencing is tough)

Handouts

Handouts of all slides will be provided

Three options for handouts with slides:

- Handouts with selected questions answered
- Handouts with selected questions unanswered (default)
- No handouts

Mail us to adjust your preference

Both versions, as both PDFs and `.pptx`s, on website

Grading

Grading

- Assignments 60%
- Pop quizzes 5%
- Mid-term exam 13%
- Final exam 22%

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

Original Thoughts

- Half-point on final average for each

Assignments

Assignments—things like:

- Coding in the various languages
- Short answer and essay questions
- Diagrams
- One video project

Late assignments are not accepted!

No late days!

But, extensions for situations beyond your control.

Collaborative Learning Exercises

"Collaborative learning is two or more students laboring together and sharing the workload equitably as they progress toward intended learning outcomes."—Barkley et al.

- This is my first use of CLEs. I welcome your feedback!
- Exercises typically done during the class period
- Group size will vary; groups formed in various ways
- Exercises will be graded
- Scores will be grouped with assignment points

We'll be grading on lectura

You can develop solutions on your own machines but they'll be graded on the CS machine "lectura", and thus should be tested there.

Mail us (372s18@cs) **TODAY** if you haven't worked on lectura or if you feel your knowledge of that eviroment is weak; we'll be happy to help you get up and running there.

If you haven't had 352, I recommend you skim through my UNIX slides:

http://cs.arizona.edu/classes/cs352/fall15/unix.pdf

Office hours

- I love office hours!
- Website has guaranteed hours
- Open-door policy otherwise
- In-person interaction is most effective
- Discord or Skype preferred for IM and screen sharing
- OK to call my mobile but don't leave voice mail! (Send e-mail instead.)

Suggestions for success

- Attend every lecture.
- Arrive on time for lectures.
- Work through all examples on the slides. Try some what-ifs, too.
- Read the write-up for an assignment the day it's handed out.
- Start on assignments early. Don't be a regular in the Thursday Night Club.
- Don't leave any points on the table.
- Don't hesitate to ask for help on an assignment.
- If you get behind, come to office hours right then!
- Don't make bad assumptions.

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!

For a first offense expect these penalties:

- Failing grade for course
- Permanent transcript annotation
- Recommendation for one semester university 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.

No asking the world for help!

The material covered in lectures, posted on Piazza, etc. should be all you need to do the assignments.

I challenge you to not search the web for solutions for problems on assignments!

Posting problem-specific questions on websites, IRC channels, mailing lists, etc. will be considered to be cheating!

Example: *I'm learning Haskell and trying to write a function that returns True iff the parentheses in a string are properly matched. Any suggestions?*

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, and IM 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.
  Bug Bounty: One assignment point
- Everything I'll expect you to know on exams will be covered in class, on assignments, or on Piazza.

READ THE SYLLABUS!

(cs.arizona.edu/classes/cs372/spring18/syllabus.html)

Assignment 1 and 2

Assignment 1

- It's a survey (on the class website)
- Due Sunday, January 14, 11:00pm
- Worth 10 points
- Maybe 10 minutes to complete
- Thanks for doing it!

Assignment 2

- On the website
- Due Sunday, January 21 at 11:00pm

Pictures & Name memorization

Basic questions about programming languages

What is a programming language?

A simple definition:

A system of rules for data manipulation.

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 a language is Turing Complete is to use it to implement a *Universal Turing Machine*, a theoretical device capable of performing any algorithmic computation.

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

Does it matter what language is used?

The two extremes:

- 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. We must choose very carefully!

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

"A programmer should learn a new language every year."
--Recommended in *The Pragmatic Programmer*

How old are programming languages?

Plankalkül 1945
Short Code 1949
FORTRAN 1957
ALGOL 1958
LISP 1958
COBOL 1959
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
Erlang 1986
Perl 1987
Haskell 1990
Python 1991
Ruby 2/24/93
Java 1995
JavaScript 1995
PHP 3 1998
C# 2000
D 2001
Scala 2003
Clojure 2007
Go 2008
Rust 2010
Kotlin 2011
TypeScript 2012
Julia 2012
Swift 2014
Goaldi 2015

How many languages are there?

The State of the Octoverse 2017 (octoverse.github.com)

wikipedia.org/wiki/Alphabetical_list_of_programming_languages

The Language List
people.ku.edu/~nkinners/LangList/Extras/langlist.htm

Online Historical Encyclopaedia of Programming Languages
hopl.info

How are languages related to each other?

Some of the many *attempts* at a family tree of languages:

digibarn.com/collections/posters/tongues

levenez.com/lang

rigaux.org/language-study/diagram.html

www.seas.gwu.edu/~mmburke/courses/csci210-su10/tester-endo.pdf

(Seems to be based on **hopl.info** data.)

Collaborative Learning Exercise

What languages are popular right now?

cs.arizona.edu/classes/cs372/spring18/cle-poplang.html

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 concisely 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.

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

How are languages specified?

The specification of a language has two key aspects:

- Syntax

  Specifies the sequences of symbols that are valid programs in the language.

- Semantics

  Specifies the meaning of a sequence of symbols.

Language specifications fall on a broad spectrum:

High end:

The specification is a published international standard

Low end:

The behavior of the lone implementation is the specification.

Syntax vs. semantics

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?

What are various meanings for these expressions?

```
x || y
x y
*x
```

Building blocks

What are the building blocks of a language?

- Data types
- Operators
- Control structures
- Support for encapsulation
  - Functions
  - Abstract types / Classes
  - Packages / Modules
- Error / Exception handling
- Standard library

What are qualities a language might have?

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

What are some tensions between these qualities?

What factors affect popularity?

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

The philosophy of a language

What is the philosophy of a language? How is it manifested?

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 machines, not humans

What is the philosophy of Java?

A Little U of A CS History

The Founding of UA CS

- The UA CS department was founded by Ralph Griswold in 1971. (Hint: know this! Mnemonic aid: ASCII 'G' is 71.)
- Griswold was Head of Programming Research and Development at Bell Labs before coming to UA.
- Griswold and his team at Bell Labs created the SNOBOL family of languages, culminating with SNOBOL4.
- Griswold's interest and prominence in programming languages naturally influenced the course of research at UA.

UA CS's language heritage

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

These are some of the languages created here:

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

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

My intersection with Griswold's work

I learned FORTRAN IV and BASIC in a summer school course at Wake Forest during the summer after high school.

In first trip to library at NCSU, took home a stack of books on programming languages, including SNOBOL4. Was totally mystified.

Learned PL/I in two-course introduction to computer science sequence.

Took a one-unit course on SNOBOL4 during sophomore year. Used SPITBOL whenever possible in courses thereafter.

Attended a colloquium at NCSU where Ralph Griswold presented a new programming language, named Icon.

Ported Icon to an IBM mainframe and DEC's VAX/VMS.

Went to graduate school here at UA, and worked on Icon as a graduate research assistant for Dr. Griswold.

Quiz!