Comparative Programming Languages

CSC 372
Spring 2023

www2.cs.arizona.edu/classes/cs372/spring23

Stranger Danger

Introduce yourself to those around you while we're waiting to launch!

Instructor

William Mitchell (`whm`)

- Consultant/contractor doing software development and training of software developers. Lots with Python, Java, C++, C, Ruby, Icon, UNIX and more.

- Occasionally teach a CS course. (337, 352, 372, 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"!

"A language that doesn't affect the way you think about programming is not worth knowing."

– Alan J. Perlis

Is learning new languages worthwhile?

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

"Research results from the psychology of programming [Petre 89] indicate that expertise in programming is far more strongly related to the number of different programming _styles_ understood by an individual than it is to the number of years' experience in programming."
   —Tim Budd, *Multiparadigm Programming in Leda*, p. viii

[Petre 89]: Marian Petre, *Finding a Basis for Matching Programming Languages to Programming Tasks*, Ph.D. Thesis, Department of Computer Science, University College London, 1989.

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

Topic Sequence

- Functional programming with Haskell

- Logic programming with Prolog

- Racket, to study Scheme's power-via-simplicity approach. (Racket is a Scheme, and Scheme is a Lisp!)

- A day each on SNOBOL4 and Icon

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

Subplots

Language appreciation:

- Discerning the philosophy of a language and how it's manifested

- Recognizing choices, tensions and tradeoffs in language design

- Assessing the "mental footprint" of a language

Learning how to learn a language (LHtLaL)

Class Website

www2.cs.arizona.edu/classes/cs372/spring23

Syllabus Highlights!

(www2.cs.arizona.edu/classes/cs372/spring23/syllabus.pdf)

Prerequisites

- CSC major admission; CSC 210
- Knowledge of Python and Java is assumed

Piazza

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

Mail

- 372s23@cs.arizona.edu goes to whm and TAs
- **<u>For anything more than "Thanks!" use "Reply All" to follow the Cc:'s</u>**

Graduate TA: Sam

Samantha Cox (**samtastic8**)

- First year Master's student in Computer Science

- BS CS from UA, Spring 2022

- Minor in Information Science & eSociety

- 9th semester as a CS TA

- Interests: Languages & Syntax, Software Design/Development, gaming (Elden Ring lately), gym/weightlifting, looking for new music.

- From Surprise, Arizona.

- Assigned to 372 for 20 hours/week

- Contact information is on our website.

- linkedin.com/in/samtastically

Undergraduate TA: Martin

Martin Cox (`coxmp`)

- Senior in Computer Science (BS)

- First semester as a TA

- Fall '22 CSC 372 survivor

- Interests: Programming languages, rapid prototyping/microcontrollers, Formula 1 racing, woodworking, table-top games, politics.

- From Cambridge, England

- Assigned to 372 for 10 hours/week

- Contact information is on our website.

- linkedin.com/in/martin-cox-27163523/

Textbooks

- No texts are required.

- Lectures, handouts, and Piazza postings might be all you need.

- Syllabus and slides have recommendations for supplementary texts, some of which are on learning.oreilly.com

Handouts

- Handouts of all slides will be provided unless you tell me otherwise.

- Handouts will have selected questions unanswered.

- Versions with and without answers, as both PDFs and .**pptx**s, will be posted on our website

Grading

Grading
- Assignments   58%
- Pop quizzes   7%
- Midterm exam   12%
- Final exam   23%

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

Original Thoughts
- Half-point on final average for each

Assignments

Assignments—things like:

- Programming problems using the various languages
- Short answer and essay questions
- In-class activities
- One video project
- (Maybe) Question sets on D2L or `openclass.ai`

Late assignments are not accepted!

No late days!

But, extensions for situations beyond your control.
(See Extensions on Assignments in the syllabus.)

We'll be grading on lectura

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

Mail us (372s23@cs) **TODAY** if you haven't worked on **lectura** or if you feel your knowledge of that evironment 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 pretty old UNIX slides:

   https://www2.cs.arizona.edu/classes/cs352/fall15/unix.pdf

Office hours

- I love office hours!
- Our Google Calendar has guaranteed hours
- Open-door policy otherwise
- In-person interaction is most effective
- Discord preferred for IM
- Zoom preferred for screen sharing
- I'll usually answer questions after class for some amount of time.

Suggestions for success

- Attend every lecture.

- Arrive on time for lectures.

- <u>Try the examples on the slides yourself!</u>

- Try some what-ifs, too.

- Start on assignments early. Don't be a regular in the Friday 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 ASAP!

- 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 <u>university</u> 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 <u>not</u> search the web for solutions for problems on assignments!

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

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!

(www2.cs.arizona.edu/classes/cs372/spring23/syllabus.pdf)

Assignment 1

Assignment 1
- It's a survey (on the class website)
- Due Sunday, January 15 at 23:59:59
- Worth 10 points
- Maybe 10 minutes to complete
- Thanks for doing it!

A few 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!

Characterize yourself:
- I think language choice is not really a big deal.
- I'm very close to the middle.
- Language choice has great importance.

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

PostScript 1982

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)
Open source projects written in 337 unique languages

wikipedia.org/wiki/Alphabetical_list_of_programming_languages
Almost 700

The Language List
people.ku.edu/~nkinners/LangList/Extras/langlist.htm
"about 2,500" but no update since < 2003. On **archive.org**.

Online Historical Encyclopaedia of Programming Languages
hopl.info
8,945 but has things like "JAVA BEANS" and minor variants like both ANSI Pascal and ISO Pascal.

Bottom line: Nobody knows how many programming languages have been created but it's in the thousands.

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

.../cs372/spring23/archive/HOPL-Pigott.pdf
("HOPL, Diarmuid Pigott, Nov 7 2003")

How do languages help us?

- Free the programmer from details

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

- Express intention and point out deviation

  ```
  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++) ...
  { i:v for i,v in enumerate([10,20,30]) }
  ```

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")

- Expressiveness

- Readability of programs

- Orthogonality

- Extensibility

- Reliability of programs

- Run-time efficiency

What are some tensions between these qualities?

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?   How about Python?

Design decisions

How many decisions must be made when designing a language?

Hundreds, maybe thousands

Examples:

- Should negative indexing be allowed for strings?

- Can strings be concatenated with integers?

- Should **list(7)** mean anything in Python?

- How many decisions does Python's **print(…)** embody?

Thoughts on language design

"The language designer should be familiar with many alternative features designed by others, and should have excellent judgment in choosing the best and rejecting any that are mutually inconsistent... One thing he should not do is to include untried ideas of his own. His task is consolidation, not innovation." —Tony Hoare

"Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered." — Guido van Rossum

"I believe that the purpose of life is, at least in part, to be happy. Based on this belief, Ruby is designed to make programming not only easy but also fun. It allows you to concentrate on the creative side of programming, with less stress." —Yukihiro Matsumoto ("Matz")

"Design a language that *you* want to use." — Ralph Griswold

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 strongly 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 programming languages created here:

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

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 the summer after high school.

In first trip to the 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.