CSC 372, Spring 2025

Comparing Programming Languages

Michelle Strout

March 18, 2025

Plan

- **Announcements (=> Look at syllabus schedule)**
  - SA5, the MT2 review, is due Wednesday
  - Joe Strout telling us about MiniScript language this Thursday
  - LA2 is due Sunday
  - Midterm 2, MT2, is a week from today on March 24th
  - Final Project has been posted
- **Last time**
  - Man, Wolf, Goat, Cheese problem
  - Knapsack problem
  - Getting a job, how is groupwork relevant?
- **Today**
  - Comparing programming languages

TopHat Questions

- Prolog question related to Knapsack
- Questions about evaluating programming languages
- Possible questions about MiniScript
  - https://miniscript.org/
  - Started in 2016
  - Farmtronics Mod in Stardew Valley is built on MiniScript

Outline for rest of today

- One last Prolog use case
- Wrapping your head around a programming language
- Comparing and evaluating languages
  - Various rankings and the metrics they use
  - Evaluation Criteria from various instructors
  - Using LLMs to compare programming languages
- Final Project
  - Which evaluation criteria do you plan to use to indicate language preferences?

One Last Prolog Use Case

- https://shchegrikovich.substack.com/p/use-prolog-to-improve-llms-reasoning, Oct 13, 2024
  - "Just recently, developers started to use Prolog as an intermediate language to improve reasoning." of LLMs
  - "It's a declarative programming language good for symbolic reasoning tasks."
  - "Due to it's declarative nature it might be a bit easier for LLMs to generate code, because LLM doesn't need to generate precise control flow."
  - This blog post has a good diagram in it.

Outline for rest of today

- One last Prolog use case
- **Wrapping your head around a programming language**
- **Comparing and evaluating languages**
  - Various rankings and the metrics they use
  - Evaluation Criteria from various instructors
  - Using LLMs to compare programming languages
- **Final Project**
  - Which evaluation criteria do you plan to use to indicate language preferences?

For a new language, five powerful questions:

- **You can ask these questions about any language**
  1. What is the abstract syntax? Syntax categories?
  2. What are the values?
  3. What environments are there? What are names mapped to?
  4. How are terms evaluated?
  5. What's in the initial basis? Primitives and otherwise, what is built in?
- **Answer these questions for SML and Prolog**

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Ray Toal Suggestions

- **Organize the concepts of the language**
  - Naming of things in the language: binding, scope, visibility, etc.
  - Evaluation: how are expressions evaluated to produce values
  - Control flow: how are statements executed to produce effects?
  - Types: a value's type constrains how it can be used in a program
  - Functional abstraction: procedures, functions, higher-order functions, etc.
  - Data abstraction: how do we group together data and code?
  - Concurrency: is there only a single thread of execution or can events happen concurrently?
  - Metaprogramming: can the program refer to itself? Asking questions such as what type is this expression?

My Suggestions

- Look for some examples of people using the language
- Try it out in some online IDE
  - E.g., https://ato.pxeger.com/about
- Ask an LLM to compare the language with one you are familiar with

Outline for rest of today

- One last Prolog use case
- Wrapping your head around a programming language
- **Comparing and evaluating languages**
  - Evaluation Criteria from various instructors
  - Various rankings and the metrics they use
  - Using LLMs to compare programming languages
- **Final Project**
  - Which evaluation criteria do you plan to use to indicate language preferences?

Course Expected Learning Outcomes

- **By the end of this class, students should be able to**
  - describe various programming languages with appropriate vocabulary
  - compare different languages and paradigms in the context of real-world problems
  - compare different kinds of type systems
  - write programs in at least three different languages, including SML, Prolog, and Chapel
  - complete a major team project
  - write a lexer and recursive descent parser for a simple language
  - use AI generation tools for comparing and contrasting programming languages

Imperative vs. Declarative

- An *imperative* language is one in which you specify *how* to do a task. Examples include Java, C, and Python.
- A *declarative* language is one in which you describe the task rather than specifying how the task is to be done. An example of this is Prolog.

Slides Content Credit: Melanie Lotz

Paradigms

- Object-oriented
  - programs are organized around the concept of objects
  - Examples: Java, Ruby
- Functional
  - programs are organized around the concept of a mathematical function
  - tend to require a lot of recursion
  - Examples: SML, Haskell
- Logical
  - programs are organized around the idea of formal logic
  - declarative
  - Example: Prolog
- Scripting
  - high-level
  - good for quick development
  - good for short programs you want to get working quickly
  - tend not to be super picky syntax-wise
  - good for “gluing” other pieces of code together
  - Example: Python, Ruby

Generations

- first-generation–machine code
- second-generation–Assembly
- third-generation
  - high-level
  - multi-purpose
  - Examples: C, C++, Java, Python, Fortran, Lisp
- fourth-generation
  - special-purpose
  - Example: SQL (used for querying databases), Postscript (used for formatting text)
- fifth-generation
  - logic & constraint-based
  - Example: Prolog

Translation processes: compilers & interpreters

- A compiler translates the full source code before it can be run. One advantage of this is faster runtimes since the translation work is already done AND compilers often do some code optimization.
- An interpreter translates and runs code line by line.
- Examples of compiled languages: Java (though that's an oversimplification), C, SML
- Examples of interpreted languages: Python, Ruby

Type Systems & Type Safety

- strong vs. weak type checking
  - This distinction has to do with how strict the type system is done and how much parameter coercion is allowed.
  - SML, for example, is considered very strongly typed as it is very strict about types and it does little to no parameter coercion.
  - Java is also strongly typed, though not quite so strongly as SML, because it is strict about types but it does do some parameter coercion (e.g. int to double).
  - C is not very strongly typed as it allows things of one type to be put into spaces allotted for another type without the compiler complaining much.
  - Very few languages do no type checking at all. Bliss is one example.

- Is Prolog strongly or weakly typed? Dynamic or statically typed?

Type Systems & Type Safety

- static vs. dynamic type checking
  - This distinction has to do with WHEN the types are checked.
  - If a language does *static* type checking, it means that the types are checked at compile-time.
  - If a language does *dynamic* type checking, it means that the types are checked at runtime.

- Statically typed languages
  - <list in class>

- Dynamically typed
  - <list in class>

Type Systems & Type Safety

- type annotations vs. type inference
  - This distinction has to do with HOW types are determined and checked.
  - Java mostly relies on *type annotations*, which means that the programmer must declare the type of a variable, and the compiler uses that information to check that the usage is consistent.
  - SML, on the other hand, relies on *type inference* (although it does allow for type annotations), which means that the type of a variable is determined by *how it is used*.
  - This difference is important in understanding how best to debug type errors. In Java, if you have a type error, it helps to look carefully at where you declared the variable and then how you are using it. In SML, you should start by looking at how you are using the variable first because that determines its type (whether you used a type annotation or not).

Ray Toal Technical Criteria

- Easy to read?
- Easy to write?
- Highly expressive?
- Designed to catch mistakes?
- Designed to allow quick and incremental compilation?
- Amenable to efficient target code generation?
- Genuinely portable?

Dr. Strout's list

- **Ways to evaluate programming languages**
  - What kind of problems does the language help solve?
  - What is the community like in terms of responsiveness, friendlieness, helpfulness, etc.?
  - Performance, programmability, power usage, memory usage, compile time
  - How well does it interoperate with the language(s) already being used in the domain the problem is in? With the language(s) being used by the code you are modifying, maintaining, extending, etc.?
  - How well do LLMs understand code written in that language? How well do they do at generating the code? How well do they do at providing help understanding how to debug and/or understand error messages?
  - Will it help me get a job?
  - What tools exist to help one to debug, analyze, and program in that language?

Popularity is measured in rankings

- **Tiobe Rankings**
  - "The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings."
- **RedMonk Programming Language Rankings**
  - combines data from GitHub (usage) and Stack Overflow (discussion) to gauge language popularity
- **IEEE Spectrum Ranking**
  - the "Spectrum" ranking, weighted towards the profile of the typical IEEE member; the "Trending" ranking, which seeks to identify languages gaining momentum; and the "Jobs" ranking, reflecting employer demand

Outline for rest of today

- One last Prolog use case
- Wrapping your head around a programming language
- Comparing and evaluating languages
  - Various rankings and the metrics they use
  - Evaluation Criteria from various instructors
  - Using LLMs to compare programming languages
- **Final Project**
  - Which evaluation criteria do you plan to use to indicate language preferences?

Criteria Students Indicated are Important

- <fill in during class>