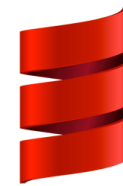
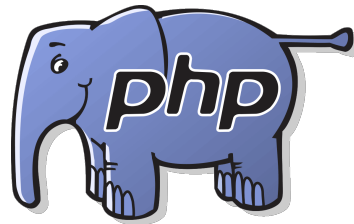


CS 360

Programming Languages

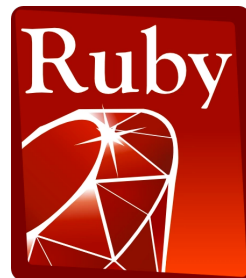
Day 10 - Motivation



Scala



Swift



Course Motivation

(Did you think I forgot? 😊)

- Why learn languages that are quite different from Python or C++?
- Why learn the fundamental concepts that appear in all (most?) languages?
- Why focus on functional programming?

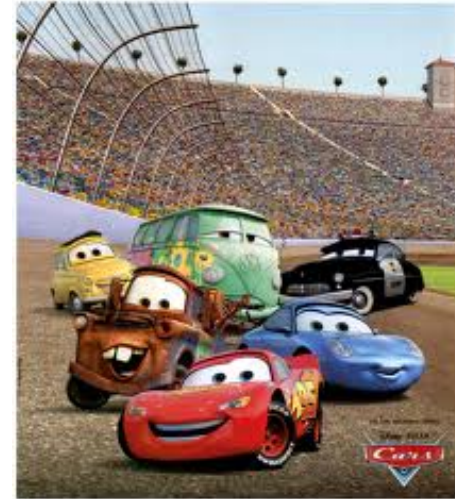
What is the best kind of car?

What is the best kind of shoes?

Cars / Shoes

Cars are used for rather different things:

- Winning the Indy 500
- Taking kids to soccer practice
- Off-roading
- Hauling a mattress
- Getting the wind in your hair
- Staying dry in the rain



Shoes:

- Playing basketball
- Going to a dance
- Going to the beach



More on cars

- A good mechanic might have a specialty, but also understands how “cars” (not 2014 Honda Civics) work.
 - And that the syntax---I mean upholstery color---isn’t essential
- A good mechanical engineer really knows how cars work, how to get the most out of them, and how to design better ones.
- To learn how cars work, it may make sense to start with a classic design rather than the latest model.
 - A popular car may not be a good car for learning how cars work.

All cars are the same

- To make it easier for everyone to drive a car, it's great that they all have common components, like steering wheels, brakes, windows, headlights, etc.
 - Yet it's still uncomfortable to learn a new one.
- And maybe PLs are more like cars, trucks, boats, and bikes.
- So are all PLs really the same...

Are all languages the same?

Yes:

- Any input-output behavior implementable in language X is implementable in language Y [Church-Turing thesis]
- Python, C++, Racket, and a language with one loop and three infinitely-large integers are “the same”
- Beware “the Turing tarpit”

Yes:

- Same fundamentals reappear: variables, abstraction, recursive definitions, ...

No:

- The primitive/default in one language is awkward in another

A note on reality

Reasonable questions when deciding to use/learn a language:

- What libraries are available for reuse?
- What can get me a summer internship?
- What does my boss tell me to do?
- What is the de facto industry standard?
- What do I already know?

CS 360 by design does not deal with these questions.

- You have the rest of your life for that.
- And the answers will be different in a few years anyway.

Why semantics and idioms

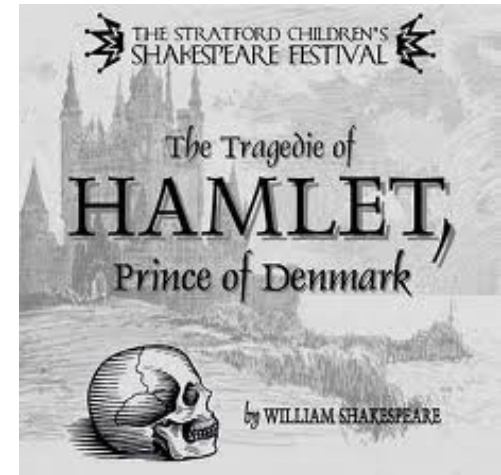
This course focuses as much as it can on semantics and idioms.

- Correct reasoning about programs, interfaces, and interpreters or compilers *requires* a precise knowledge of semantics.
 - Not “I *think* that conditional expressions might work like this.”
 - Not “I like curly braces more than parentheses.”
 - Much of software development is designing precise interfaces; what a PL means is a *really* good example.
- Idioms make you a better programmer.
 - Best to see in multiple settings, including where they shine.
 - See future languages in a clearer light.

Hamlet

The play *Hamlet*:

- Is a beautiful work of art.
- Teaches deep, eternal truths.
- Is the source of some well-known sayings.
- Makes you a better person.



Continues to be studied (even in college) centuries later even though:

- The syntax is really annoying to many (yet rhythmic).
- There are more popular movies with some of the same lessons (just not done as well).
- Reading *Hamlet* will not get you a summer internship.

Functional Programming

Okay, so why do we spend so much time with *functional languages*, i.e., languages where:

- Mutation is unavailable or discouraged
- Recursion expresses all forms of looping and iteration
- Higher-order functions are very convenient

Because:

1. These features are invaluable for correct, elegant, efficient software (great way to think about computation)
2. Functional languages have always been ahead of their time
3. Functional languages well-suited to where computing is going

Most of course is on (1), so a few minutes on (2) and (3) ...

Ahead of their time

All of these were dismissed as “beautiful, worthless, slow things PL professors make you learn in school”

- Garbage collection (now used in Python, Java, and most modern languages other than C/C++)
- Collections (i.e., lists) that can hold multiple data types at once (Python, Java via generics, C++ through templates)
- XML for universal data representation (like Racket/Scheme/LISP)
- Higher-order functions (Python, Ruby, JavaScript, more recent versions of C++, ...)
- Recursion (a big fight in 1960 about this – I’m told 😊)

Somehow nobody notices the PL people were right all along.

Recent Surge



CUFP 2017

Commercial Users of Functional Programming

SEPTEMBER 7TH-9TH — OXFORD, UNITED KINGDOM

C7: Owl - Data Science in OCaml

Using Haskell to run a datacenter

Using Functional Programming to
Accelerate Translational Research at
Pfizer

Why a surge?

My best guesses:

- Concise, elegant, productive programming.
- JavaScript, Python, Ruby helped break the Java/C/C++ hegemony.
 - And these functional languages do some things better.
- Avoiding mutation is *the* easiest way to make concurrent and parallel programming easier.
- Sure, functional programming is still a small niche, but there is so much software in the world today even niches have room.

Is this real programming?

- The way we're using Racket in this class can make the language seem almost “silly” precisely because lecture and homework focus on interesting language constructs.
- “Real” programming needs file I/O, string operations, floating-point, graphics, project managers, testing frameworks, threads, build systems, ...
 - Functional languages have all that and more.
 - If we used C++ or Python the same way, those languages would seem “silly” too.

Summary

- No such thing as a “best” PL.
- There are good general design principles for PLs.
- A good language is a relevant, crisp interface for writing software.
- Software leaders should know PL semantics and idioms.
- Learning PLs is not about syntactic tricks for small programs.
- Functional languages have been on the leading edge for decades
 - Ideas get absorbed by the mainstream, but very slowly.
 - Meanwhile, use the ideas to be a better programmer in C++ and Python.