CS339: Abstractions and Paradigms for Programming

Introduction and Logistics

Manas Thakur
CSE, IIT Bombay

Autumn 2025

Natural Languages

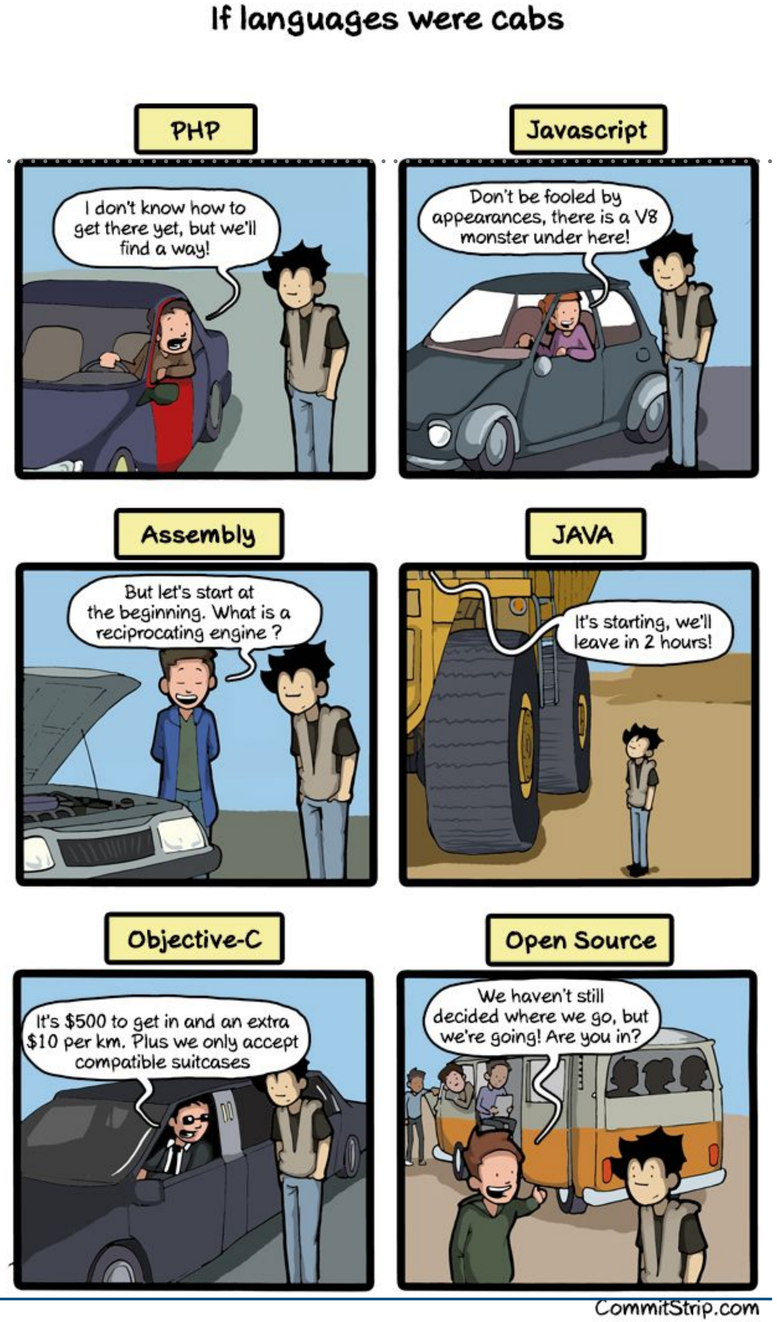
- Name the languages that you can speak/read/write.
- A unique advantage of living in India?
  - *The APP course in no other country can start with this slide!*

- Have you tried learning a new language?
- How different was that language from those you already knew?
- What was your strategy?

Programming Languages

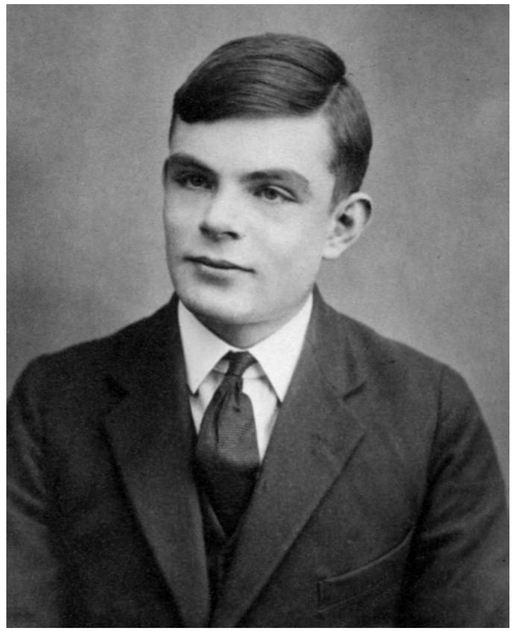
- Name the PLs that you know.
  - What do we mean by *know*?
- Name an English letter that's not a PL.
- Why do we have so many PLs?
- How did so many PLs evolve?

- Or aren't they so many, but just syntactic sugars over a set of fundamental *computational* constructs?



Standing on the shoulders of giants

- Who are these people?



Alan Turing

Alonzo Church

- Hints:
  - Both were computer scientists.
  - The most famous award in CS is named after the LHS.
  - RHS was the PhD advisor of the LHS.

Two models of computation

- The **Turing Machine** (1936) performs computations by:
  - Reading input
  - Modifying an internal memory using instructions
  - Producing output

If you don't remember TOC,
think of a computer that you are used to!!

- The **Lambda Calculus** (1936) performs computations by:
  - Evaluating expressions

We would learn more of it in this course.

- **Church-Turing Thesis** (1937): Both the models, and all other reasonable computation models, are equivalent!

Abstractions and Paradigms for Programming

- **WHAT** is the square root of a number X?

A number Y such that Y * Y == X.

- **HOW** to compute the square root of a number X?

- Make a guess Y.
- Improve the guess Y until its square is near enough to X.

A SQRT "**procedure**"

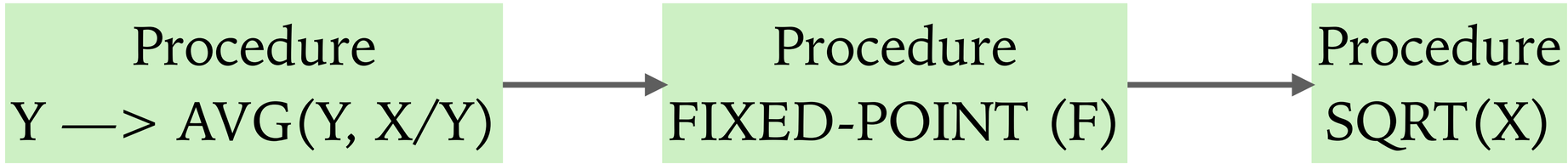
1. Procedural Abstraction

- Can we reuse the SQRT procedure as a blackbox?

```
Compute SQRT(A) + SQRT(B).
```

Functional Paradigm

- Can we consume and produce procedures?



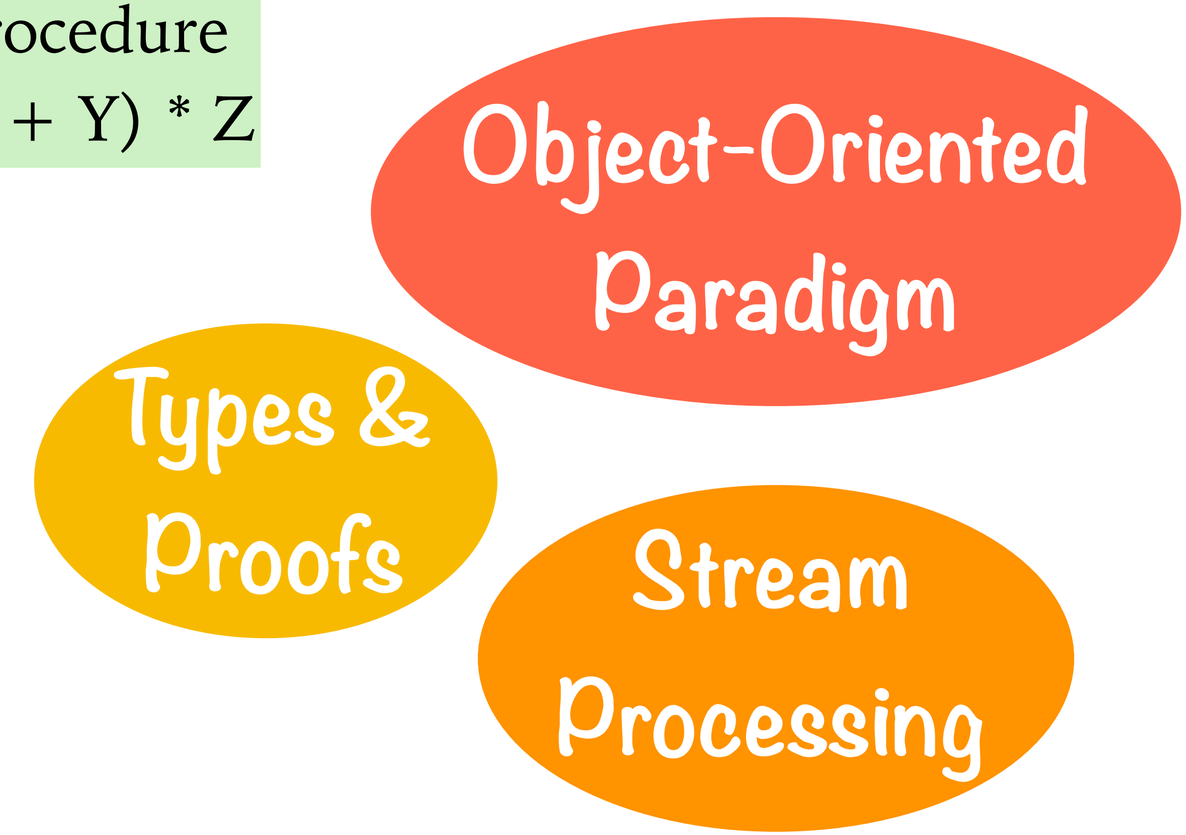
- Functions become real entities in our programs!

2. Data Abstraction

- Say we have a general procedure to add two things and multiply by another:

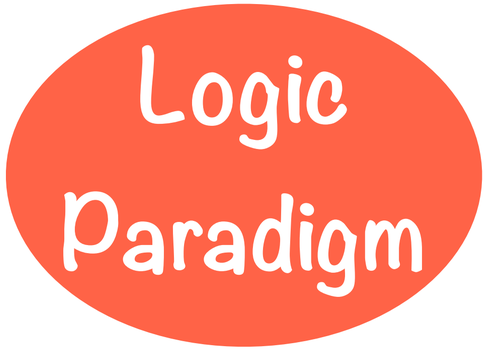
Procedure
(X + Y) * Z

- Can X, Y, Z be
  - numbers?
  - vectors?
  - polynomials?
  - signals?



3. Language Abstraction

- Can we design a language that gives us higher level primitives?
  - vectors? matrices? tensors?
  - images? videos?
- Can we modify an existing language to work differently?
  - lazy evaluation instead of eager?
  - automatic allocation instead of explicit types?
- Can we extend a language to support something more?
  - C to C++?



Abstractions and Paradigms for Programming

- Computation is much more fundamental (and older) than computers or computer science, and is very much about specifying the right abstractions.
- There are several paradigms (aka *ways*) of specifying computations.
- But not as many as there are programming languages.

- There is no way we can teach you the PL that you can keep using for the next 30-40 years. But if we teach you the ways of specifying and abstracting computations, and equip you with the skills to design your own PLs, you would have learnt enough to become a **successful computing engineer**.

The APP Course

- Empowers us to choose the right PL for the task at hand;
- Makes learning a new PL much easier;
- Equips us in designing a new PL and its interpreter;
- Is huge **fun**!

Side learnings:

- Ability to comprehend large programs
- Language technologies that everyone uses
- Crux of **future** programming languages
- Different *right* ways of thinking about the same thing
- Motivation to learn the art of cooking (CS302) and processing (CS614, CS618, CS6004) the food that everyone (CSXYZ) eats

Logistics, Evaluation, Rules

Logistics (CS339+CS355)

- Three classes per week in Slot 2: Mon 9:30am; Tue 10:35am; Thu 11:35am
- One lab per week in Slot L1: Mon 2pm, **starting next week**
- Instructor: Manas Thakur
- TAs: Aditya, Meetesh, Preet, Kush, Samarth, and many more to come…
- **Office hour**: Wed 4:00-5:00 pm (CC 308) — this week only 30 mins
- Course webpage (schedule, deadlines): https://tinyurl.com/app25-plan
- Moodle (material, submissions, marks)
- Grading would be relative, but you would definitely pass if you score >30

Logistics (CS355)

- You can come to the lab every week for practice
- But you will be **evaluated in alternate weeks:**
  - 23B0901–23B0996: A batch (When? …look at the course plan)
  - 23B0997–everybody else: B batch (When? …look at the course plan)
- As and Bs, as well as the TAs, will be rotated for fairness
- Stick to your batch unless you have a pink slip
- Different questions per batch, which means you will get double the number of questions for practice!

Evaluation (CS339)

- Exams
  - Quizzes (10 * 2)
  - Midsem (30)
  - Endsem (40)
- Participation (10)
  - Attendance
  - Activities (e.g. Pop Quizzes)
  - PCs (what are they?)

Evaluation (CS355)

- Practice (15)
  - 5 out of 6 labs * 3
- Attendance (5)
  - 5 out of 6 labs * 1
  - Yes, you can miss any one lab without any problem; if you don't, we would take the best 5 out of 6
- Exams (80)
  - Midsem: 40
  - Endsem: 40
  - Auto-graded; no re-evaluation; practice well

Missed-Evaluation Policy (CS339+CS355)

- All the below considerations would require prior permission
- CS339:
  - Missed Quiz: Scale from next Big Exam
  - Missed Midsem: Scale from Endsem
  - Missed Endsem: Ideally none, but can consider a supplementary exam in emergency situations
- CS355:
  - Missed Midsem: Scale from Endsem
  - Missed Endsem: Scale from Midsem

The Constitution

- **Rights:**
  - An in-depth understanding of the various topics
  - Acceptable answers to doubts/queries
  - Timely and complete evaluations
- **Duties:**
  - Sincerity and honesty
  - No hesitation in reaching out to the instructor/TAs
  - **No plagiarism** in anything

Reaching Out

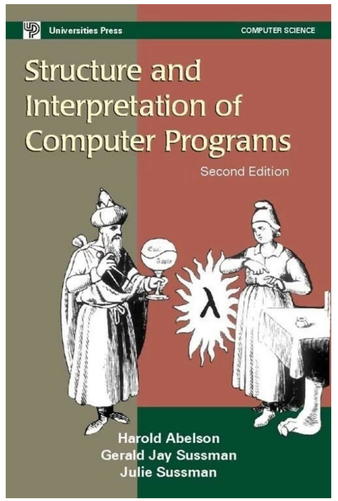
- All conversation (doubts, queries, musings) encouraged either in the lab (SL2) or in the office hour (recall: Wed 4-5 PM, CC 308)
- **Email the instructor only if unavoidable, with the following norms:**
  - Subject must start with [CS339] or [CS355]
  - Responses *usually* sent within 48 hours
  - Don't expect responses to *DOS attacks*
- **No emails/DMs to TAs** unless told by the instructor for something particular
- No deadlines/evaluation dates can be changed except in an emergency; make sure you keep following the course-plan page
- Be polite and considerate in all conversations with the teaching team (and even classmates)

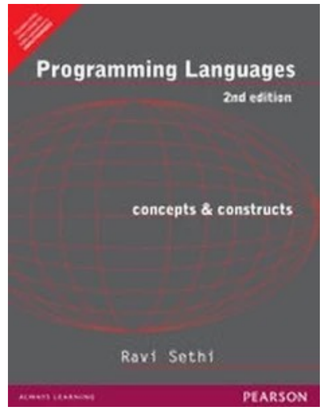
Learning Resources

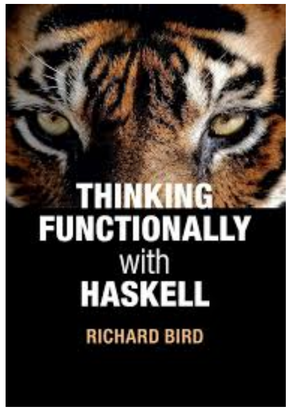
Most concepts and Scheme

Lambda calculus

Haskell, types, proofs



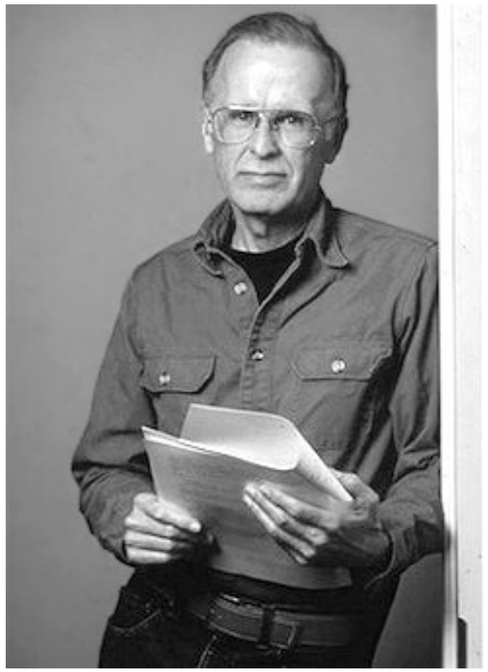




- Textbooks (**SICP**, PLCC, TFWH)
- Some videos and papers
- Some online resources
- Take notes in class (slides won't always be enough/available)
- Practice programs in the lab and at home

First Homework!

- **Activity 1:**
  - Read first seven pages of John Backus's Turing Award Lecture
  - Try finding out why was that lecture very interesting
  - Answer a pop quiz in the next class
- Install **DrRacket** with **sicp** package on your computer
  - https://download.racket-lang.org/
- Next class: Back to high-school maths!



The *spirit* of this course

Dedication text of ***Structure and Interpretation of Computer Programs*** *(SICP)*
Authors: Harold Abelson and Gerald Jay Sussman with Julie Sussman (MIT)

This book [course] is dedicated, in respect and admiration, to the spirit that lives in the computer:

"I think that it's extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don't think we are. I think we're responsible for stretching them, setting them off in new directions, and keeping fun in the house. I hope the field of computer science never loses its sense of fun. Above all, I hope we don't become missionaries. Don't feel as if you're Bible salesmen. The world has too many of those already. What you know about computing other people will learn. Don't feel as if the key to successful computing is only in your hands. What's in your hands, I think and hope, is intelligence: the ability to see the machine as more than when you were first led up to it, that you can make it more."

—Alan J. Perlis (April 1, 1922 – February 7, 1990), first Turing Award winner