

# CS 581: Programming Language I

## Course Introduction

Fall 2019

# Introduce yourself!

Rough estimate in class

- Taken CS 381?
- Experience with Haskell or other functional language?

By Monday, 5pm

Submit a PDF (through TEACH) containing:

- Name, year/status, research group (if applicable)
- Photo, ideally with a view of your face
- Answers to a few questions
- An interesting fact



# Outline

About this course

Why study programming languages?

Languages are at the heart of computer science

Good languages really matter

How to study programming languages

Course logistics

## About this course

CS 381:



CS 581:



- Functional programming
- Syntax
- Denotational semantics
- Lambda calculus

# Outline

About this course

Why study programming languages?

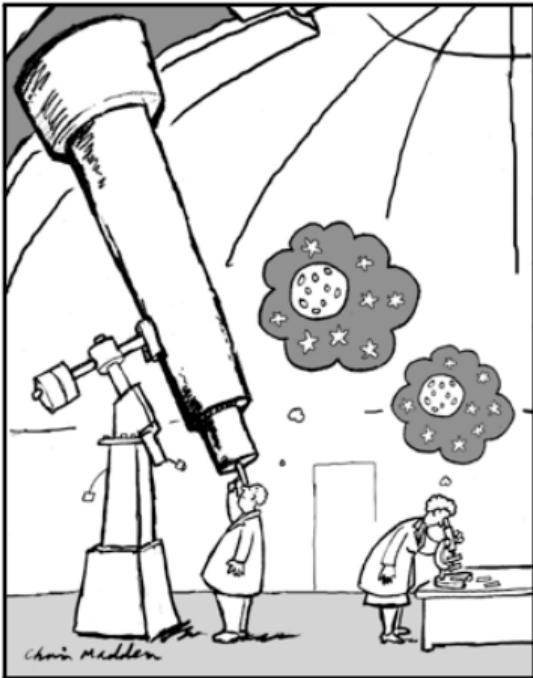
Languages are at the heart of computer science

Good languages really matter

How to study programming languages

Course logistics

# What is computer science?



*Computer science is no more about computers than astronomy is about telescopes.*

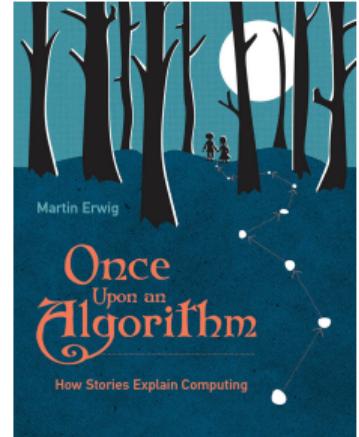
*—Edsger Dijkstra*

Computer Science = the science of **computation**

# What is computation?

Computation = **systematic transformation of representation**

- **Systematic:** according to a fixed plan
- **Transformation:** process that has a changing effect
- **Representation:** abstraction that encodes particular features



Languages play a central role:

- The “fixed plan” is an **algorithm**, which is described in a **language**
- The “representation” is **data**, which is also often described in a **language**

# What about software engineering?

## Science vs. Engineering

**Science:** tries to understand and explain

**Engineering:** applies science to build stuff

### Science

physics

chemistry

“computing”

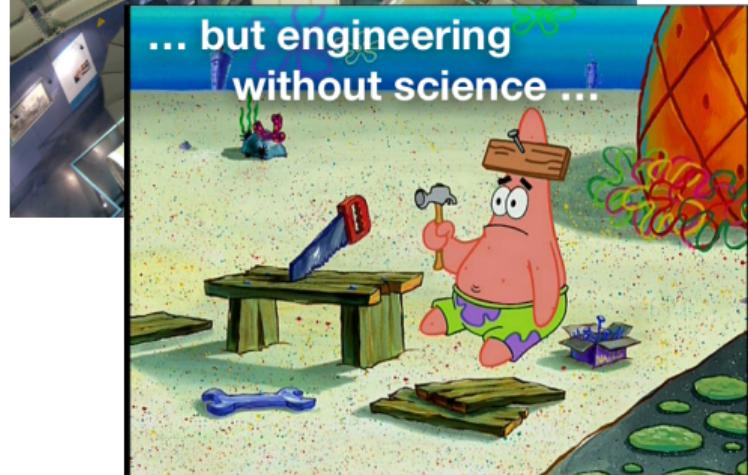
### Engineering

structural engineering, ...

chemical engineering, ...

software engineering, ...

Both are part of “computer science”



# Central role of PL in CS

PL supports both aspects of CS:

- to understand and explain (science)  
we need **languages** to describe and reason about computations for ourselves
- to build cool stuff (engineering)  
we need **languages** to describe computations for a computer to execute

# Outline

About this course

Why study programming languages?

Languages are at the heart of computer science

Good languages really matter

How to study programming languages

Course logistics

# Why good languages matter: preventing bugs

Good languages can help **prevent bugs**

- Mars Climate Orbiter failure, 1998
  - caused by mismatched units between ground and spacecraft
  - lost \$327.6 million + years of effort
- Heartbleed bug in SSL, 2012–2014
  - caused by missing bounds check
  - huge violations of privacy, including 4.5 million medical records
  - estimated \$500 million in damage
- Steam's Linux client deletes root, 2015
  - caused by silent failure of a directory lookup operation
  - offending line commented by “Scary!”... :-/



# Why good languages matter: managing complexity

Large-scale software systems are complex!

Good languages can help us **manage this complexity**



- “Structured programming”, 1950–1960s
  - problem: “spaghetti code” caused by GOTOs
  - solution: subroutines, conditionals, loops
- Rust programming language, Mozilla, 2010s
  - problem: managing memory in low-level, concurrent systems code
  - solution: ownership system

# Why good languages matter: medium of thought

## The languages we use ...

- influence our **perceptions**
- guide and support our **reasoning**
- enable and shape our **communication**

- What problems do we see? How do we reason about and discuss them?
- How do we develop, express, and share solutions?

*By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race.*

*—Alfred North Whitehead via Kenneth Iverson's  
ACM Turing Award Lecture, “Notation as a Tool of Thought”*

## Example: Positional number system

In the 13th century, this is how numbers were represented in Europe:

$$\text{MMCDXXXI} \div \text{XVII} = ? \quad :-)$$

...even basic arithmetic is hard!

Fibonacci popularized the Hindu-Arabic notation

- didn't just make arithmetic much more convenient ...
- completely changed the way people thought about numbers, revolutionizing European mathematics

$$\begin{array}{r} 143 \\ 17 ) 2431 \\ \underline{1700} \\ 731 \\ \underline{680} \\ 51 \\ \underline{51} \\ 0 \end{array}$$



# Example: Symbolic logic

For **over 2000 years** the European study of logic focused on syllogisms

*Every philosopher is mortal.*

*Aristotle is a philosopher.*

*Therefore, Aristotle is mortal.*

Only 256 possible forms ... field solved!

A couple of **notational** innovations in the 19th century cracked it wide open

- George Boole – Boolean algebra
- Gottlob Frege – *Beggriffsschrift* (symbolic predicate logic)



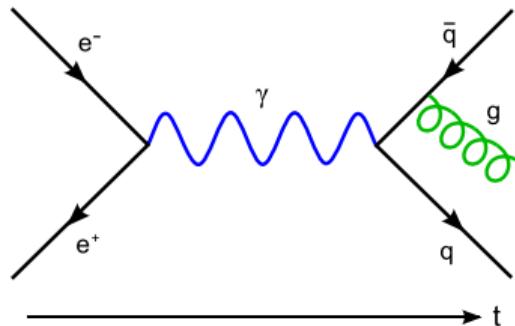
## Example: Feynman diagrams

Interactions of subatomic particles lead to brain-melting equations

- reasoning about interactions requires complex math
- high overhead to communicating problems and solutions

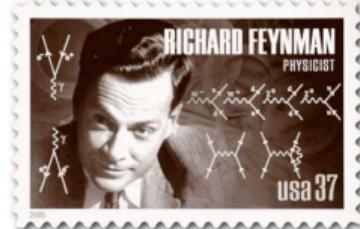
Only a handful of people can do this stuff!

In 1948, Richard Feynman introduced a **visual language** for representing interactions



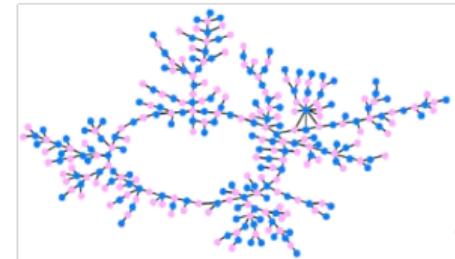
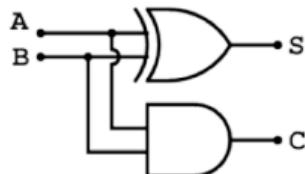
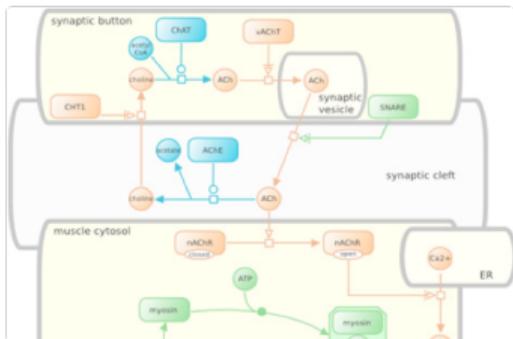
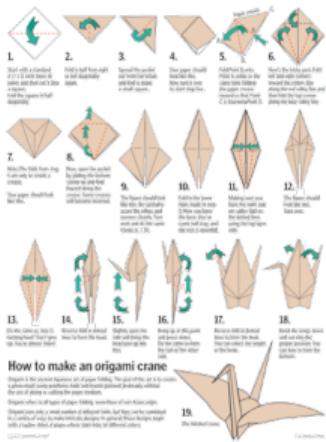
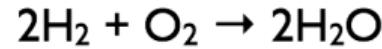
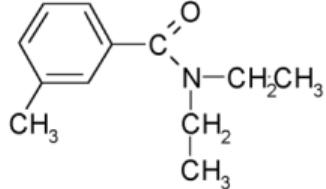
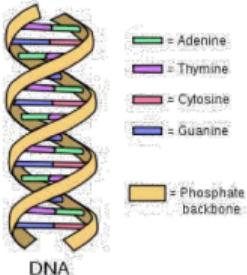
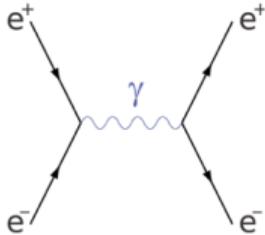
Raises level of abstraction

- eliminates *incidental complexity* (math)
- focus on *essential complexity* (interactions)
- supports communication, collaboration  
(undergrads can do it)



# Domain-specific languages

$$F = ma$$
$$E = mc^2$$



# Outline

About this course

Why study programming languages?

Languages are at the heart of computer science

Good languages really matter

How to study programming languages

Course logistics

# One idea: just try out a bunch of languages



Haskell



Racket



**Not this course!**

# Our focus: programming language concepts and theory

Focus on how to **define** programming languages

For several toy languages, we will:

- define the **structure** of its programs
- define the **meaning** of its programs
- identify the **features** that are common to many languages

# Role of metalanguages

**Metalinguage:** a language to define the structure and meaning of another language!

In this course:

- grammars
- mathematics
- inference rules
- Haskell
- English



# Summary of our strategy

Focus mostly on programming language **concepts**

1. define **abstract syntax** of languages
2. define **semantics** of languages
3. compare different **language features**
4. in-depth study of **lambda calculus**

We use **metalinguages** for examining these concepts

1. formal definitions using **grammars**, **mathematics**, and **inference rules**
2. interpreters in **Haskell**

# Outline

About this course

Why study programming languages?

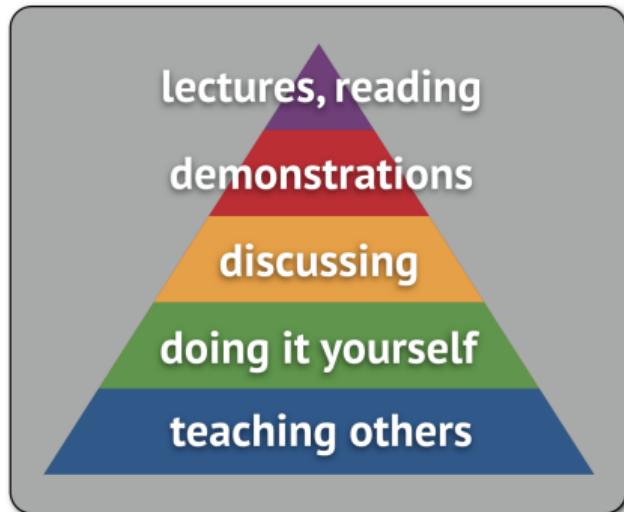
Languages are at the heart of computer science

Good languages really matter

How to study programming languages

Course logistics

# Learning strategy



“Learning pyramid”

## In class

- lectures
- demonstrations / live coding
- in-class exercises
- peer-feedback/discussion of homework and exercises

## On your own

- outside reading
- homework
- study for quizzes, exams