CS 381: Programming Language Fundamentals

Course Introduction

Summer 2015

Outline

Why study programming languages?

- Languages are at the heart of computer science
- Good languages really matter
- Language design can have a hug impact

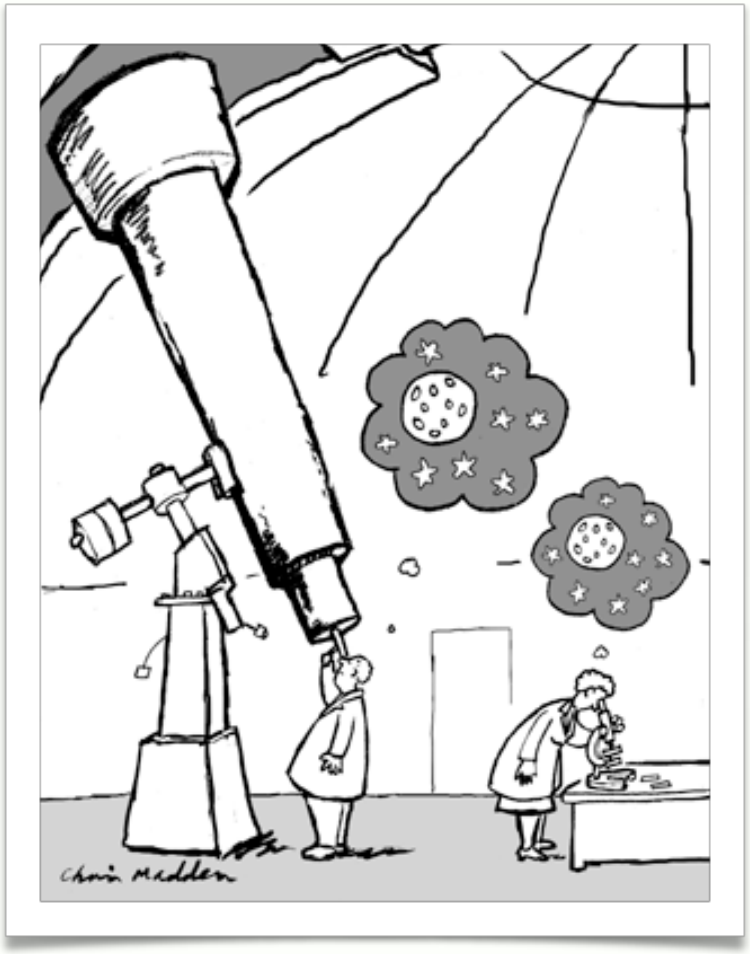
How to study programming languages

Course logistics

Common ~~Expectations~~ Myths

- ~~I will learn how to program~~
- ~~I will learn the details about particular programming languages~~
- ~~I will see a comparison of different programming languages~~
- ~~I am taking a functional programming course~~

What is computer science?



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

— Edsger Dijkstra

Computer Science = the science of **computing**

Oregon State UNIVERSITY

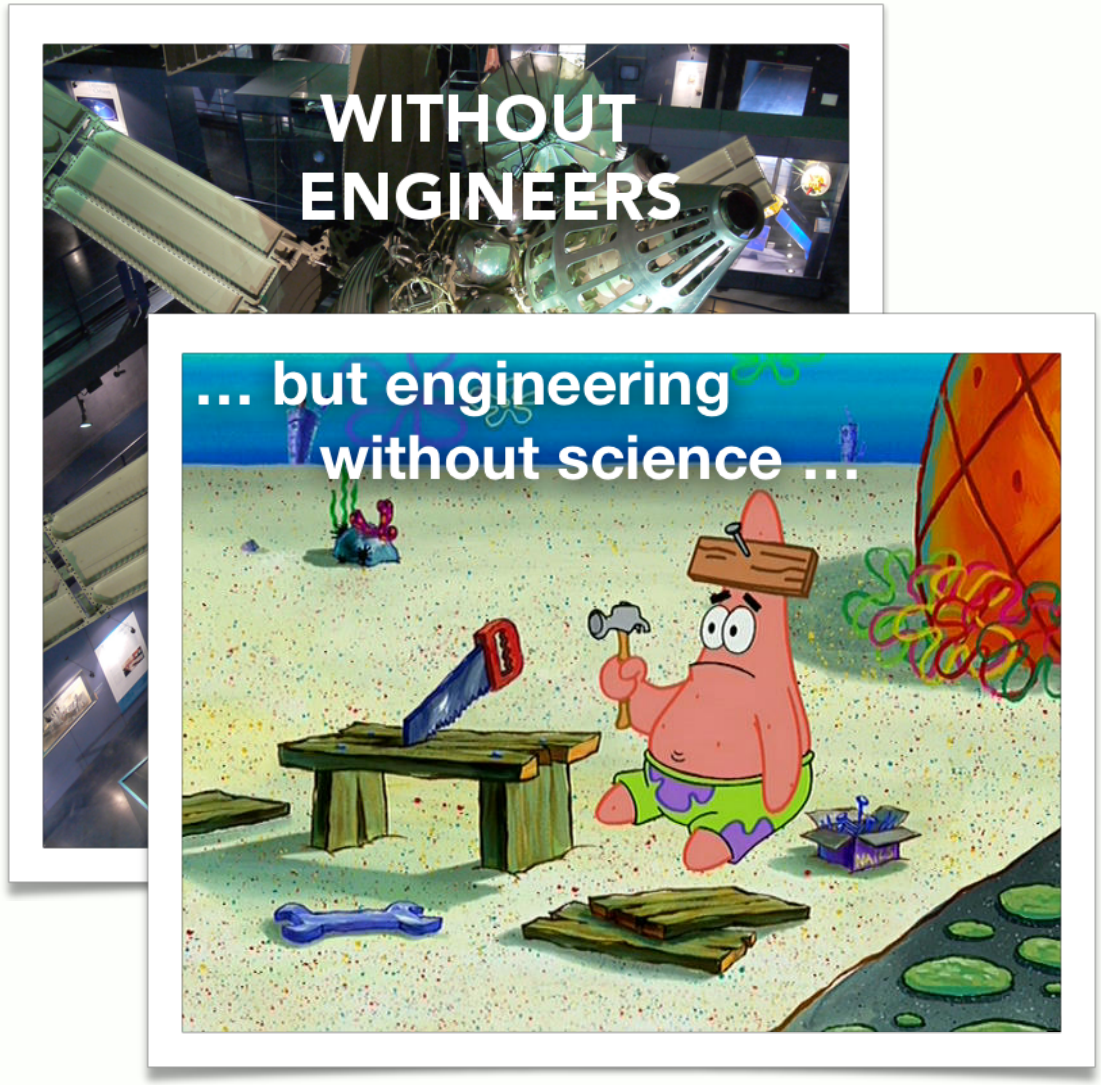
What is computer science?

Science and Engineering

Science: tries to understand and explain
Engineering: applies science to make stuff

Science	**Engineering**
physics	structural engineering, …
chemistry	chemical engineering, …
“computing”	software engineering, …

“Computer science” conflates **two fields**



Oregon State
UNIVERSITY

What is computation?

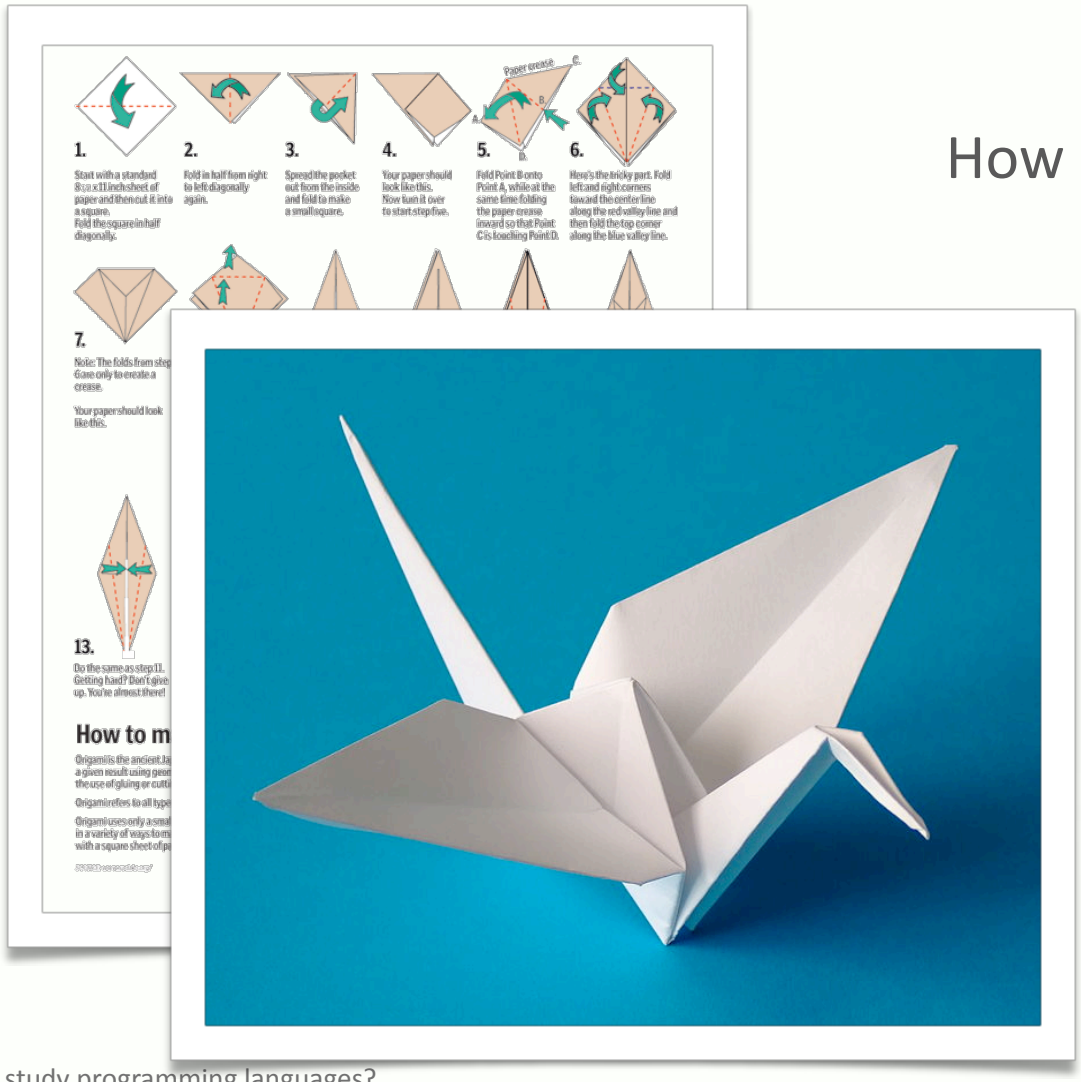
Computation is the **systematic transformation of representation**

- **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**
- Usually, the **representation** is also described in a **language**

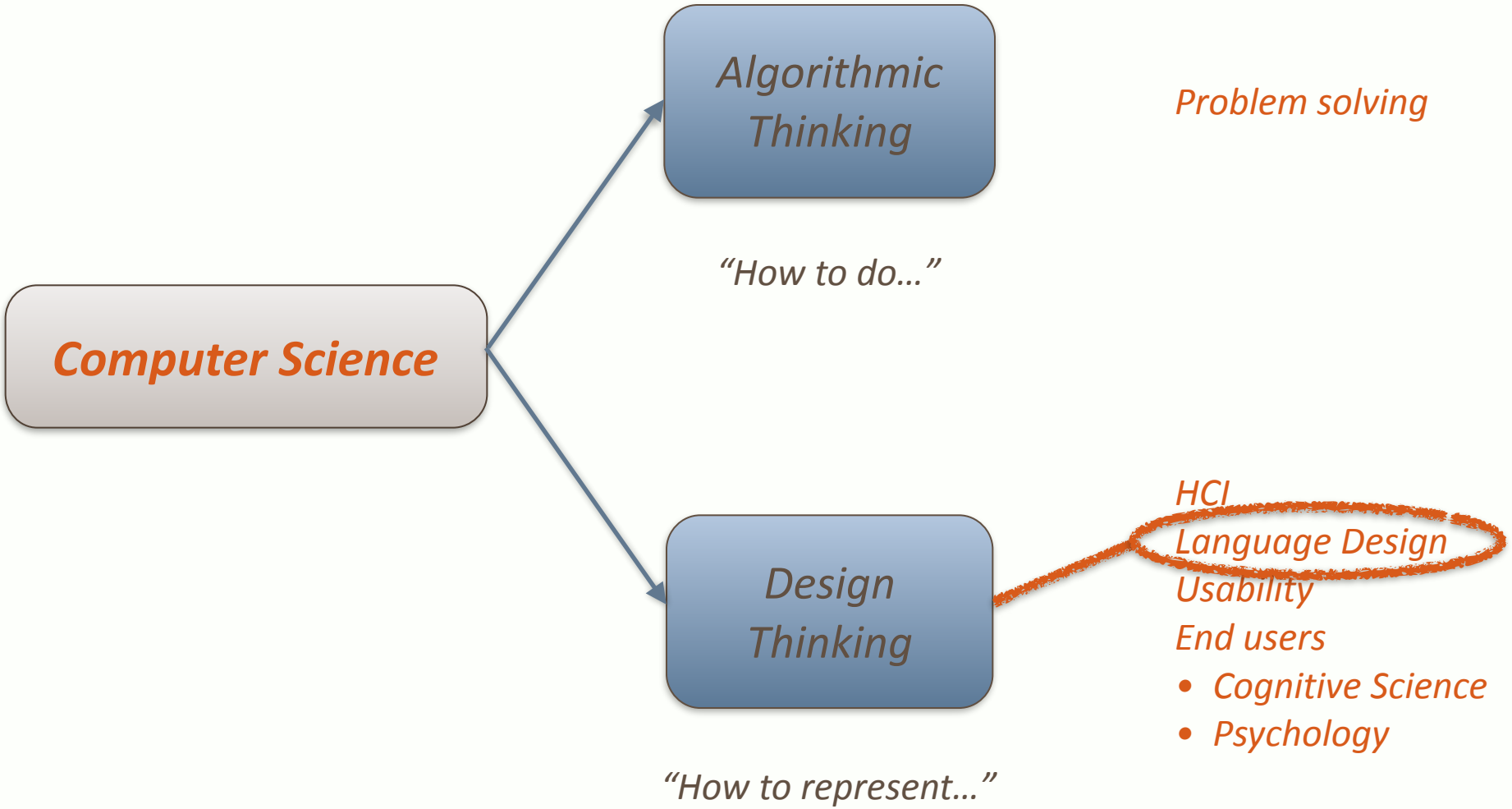
“Meatspace” computations: origami



How is this computation?

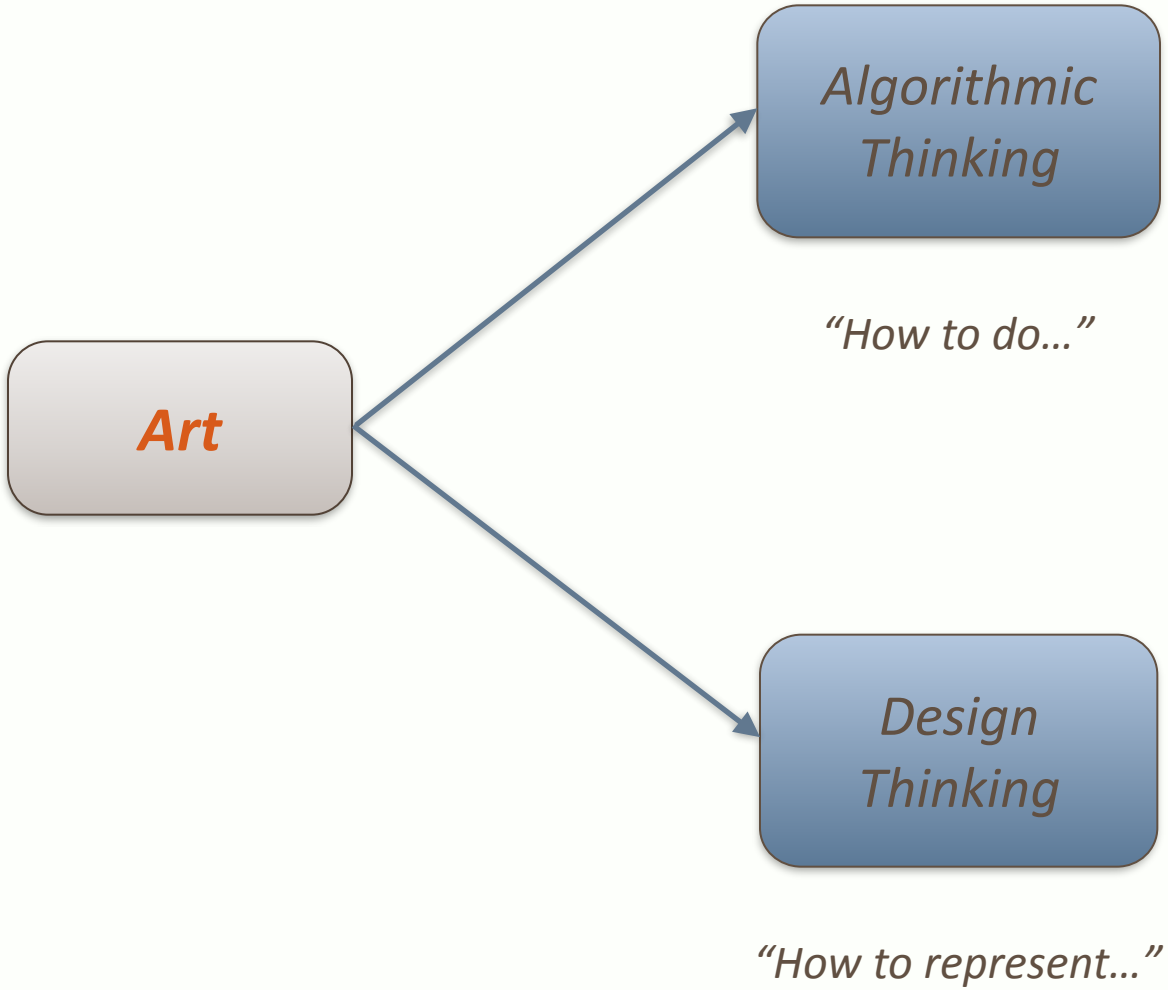
- Paper **represents** an object
- Instructions describe an **algorithm** for **systematically transforming** a given piece of paper into a particular **representation**
- The **language** in this case is a set of pictograms with English instructions

Two aspects



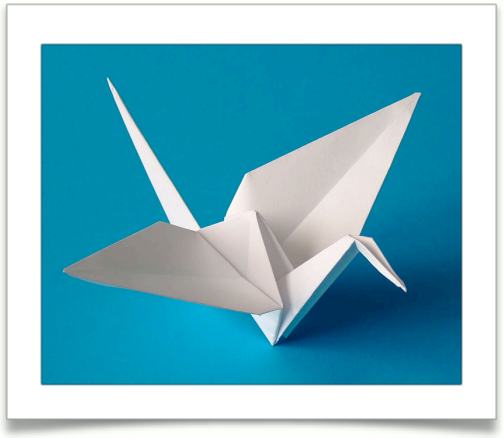
Oregon State UNIVERSITY

Algorithms and design: origami



Rules for folding paper

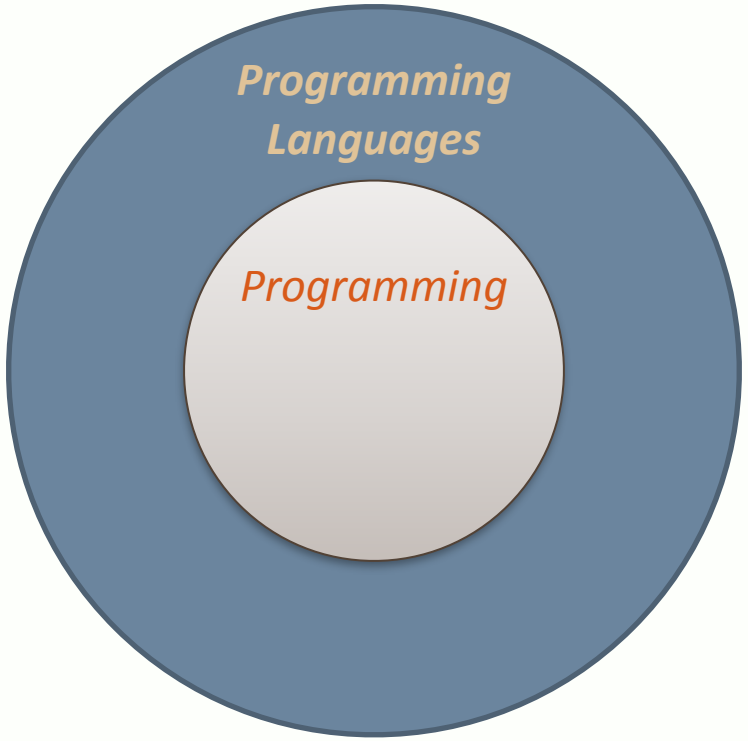
- *by example*
- *English*
- *visual rules*



Central role of programming languages in computer science

Program: a description of the plan to carry out and the representation it transforms

Programming language: a language for written programs, i.e. describing computation



Programming languages support both aspects of computer science:

- 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

Why study programming languages?

- Languages are at the heart of computer science
- Good languages really matter
- Language design can have a hug impact

How to study programming languages

Course logistics

Why good languages matter

The languages we use …

- influence our **perceptions**

  What problems do we see?

- guide and support our **reasoning**

  How do we reason about and discuss them?

- enable and shape our **communication**

  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, "Notations as a Tool of Thought"

Example: positional numbering system

13th century European number representations:

$$\mathrm{MMCDXXXI} \div \mathrm{XVII} = ???\ \text{:-0}$$

…even basic arithmetic is hard!

Fibonacci popularized the Hindu-Arabic notation:

- not only made mathematics more convenient…
- completely changed the way people thought of numbers, and revolutionized European mathematics

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

Oregon State UNIVERSITY

Example: symbolic logic

For **over 2000 years** Europeans focused logic around syllogisms:

Every philosopher is mortal.

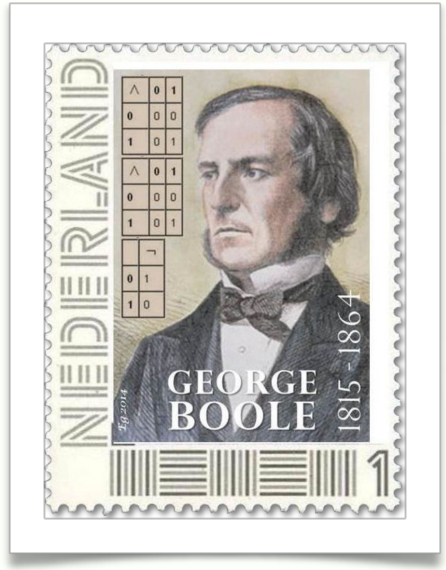
Aristotle is a philosopher.

Therefore, Aristotle is mortal.

Only 256 possible forms…field solved!

Until some 19th century **notational** innovations:

- George Boole — Boolean algebra
- Gottlob Frege — *Begriffsschrift* (symbolic predicate logic)



Oregon State UNIVERSITY

Example: Feynman diagrams

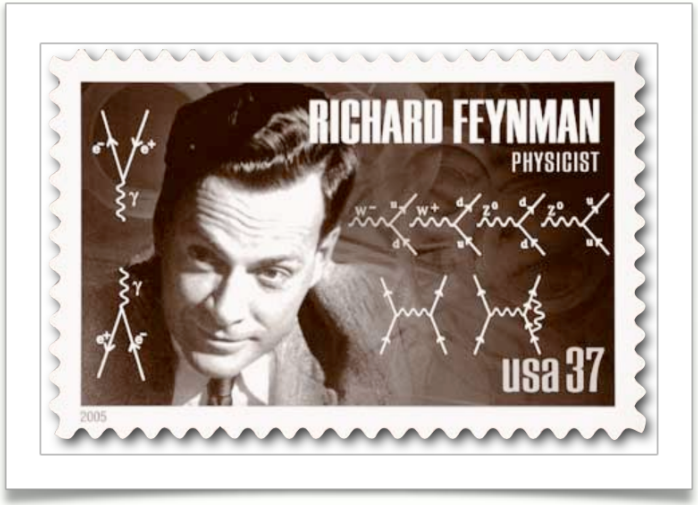
Subatomic particle interactions:

- large brain-melting equations
- reasoning about interactions requires complex math
- high overhead for communicating problems *and* solutions

Only a few people in the world can do this!

In 1948, Richard Feynman introduces a **visual language**:

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



Oregon State UNIVERSITY

The languages we use matter...

Because this does matter so much, you better know how to **choose** the right one!

...or if all else fails, how to **create** the right one!

Outline

Why study programming languages?

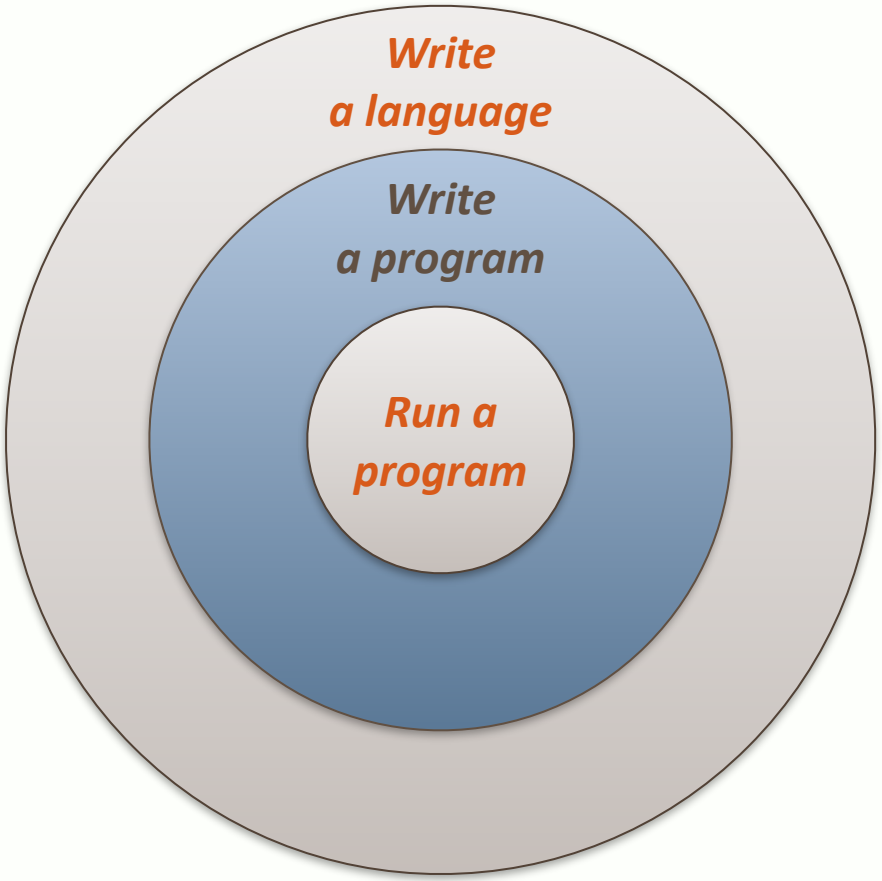
- Languages are at the heart of computer science
- Good languages really matter
- Language design can have a hug impact

How to study programming languages

Course logistics

Impact of programs and languages

If you can…



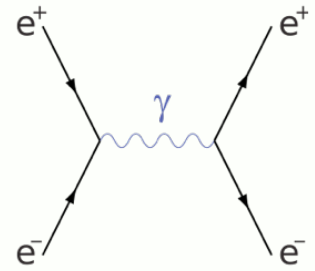
then you can…

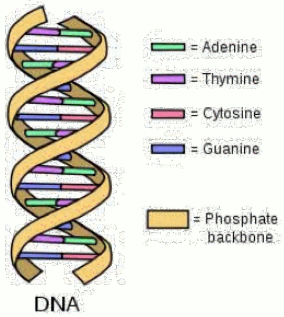
- do something faster and more reliably
- empower others to do new things
- empower others to write new and better programs

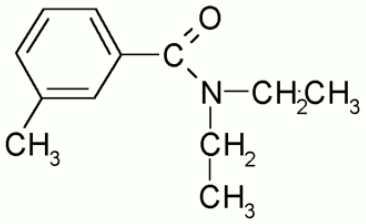
Each level is a **multiplier** for impact!

Domain-specific languages

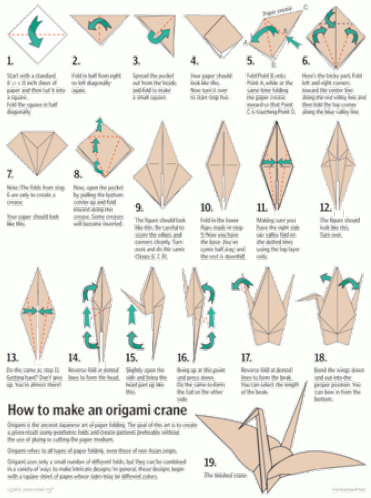
$$F = ma$$
$$E = mc^2$$

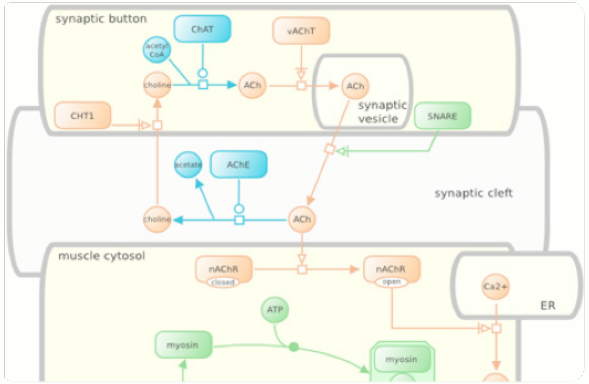


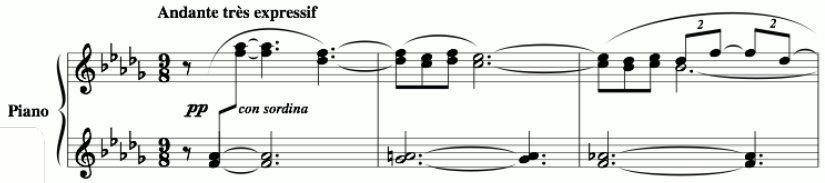


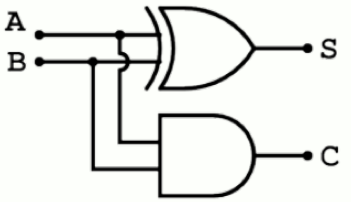


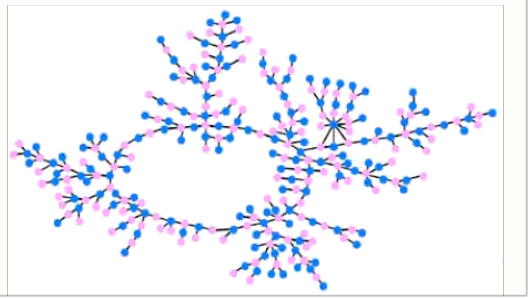
$$2H_2 + O_2 \rightarrow 2H_2O$$











Oregon State University

Outline

Why study programming languages?

- Languages are at the heart of computer science
- Good languages really matter
- Language design can have a hug impact

How to study programming languages

Course logistics

Brute force

Analogy: choosing or building a vehicle

Analogy: choosing or building a vehicle

Features: components of a vehicle, define what it can do

Aspects: views/interpretations of a vehicle

Descriptions: how the features and aspects are described

Features

- engine
- transmission
- chassis
- safety system
- entertainment system
- …

Aspects

- form, color, style
- performance (speed, economy, load)
- how to operate
- usage profile (sedan, SUV, sport)
- …

Descriptions

- diagrams
- mathematics
- English
- …

Language concept landscape

Language features: components of a language, define what programs written in the language can do

Aspects of a language: how to understand/define a language

Descriptions (metalanguages): mathematical and programming tools used to define the various aspects of a language's features — a language for describing languages!

Features

- values
- operations
- types
- states
- …

Aspects

- syntax (structure)
- semantics (meaning)
- type system (consistency)
- paradigm (feature sets)
- …

Descriptions (metalanguages)

- grammars
- rule systems
- Haskell
- English
- …

On the other hand…

The only way to **really** learn how to drive a bulldozer…

…is to **drive a bulldozer**!

Approach and tools

Focus mostly on programming language **concepts**

1. define **abstract syntax** of languages
2. define **semantics** of languages
   - scoping
   - parameter passing
   - exceptions
3. define **type systems** for languages

Introduce two new **programming paradigms**

1. functional programming (Haskell) — lots of practice
2. logic programming (Prolog) — toward end of term

Outline

Why study programming languages?

- Languages are at the heart of computer science
- Good languages really matter
- Language design can have a hug impact

How to study programming languages

Course logistics