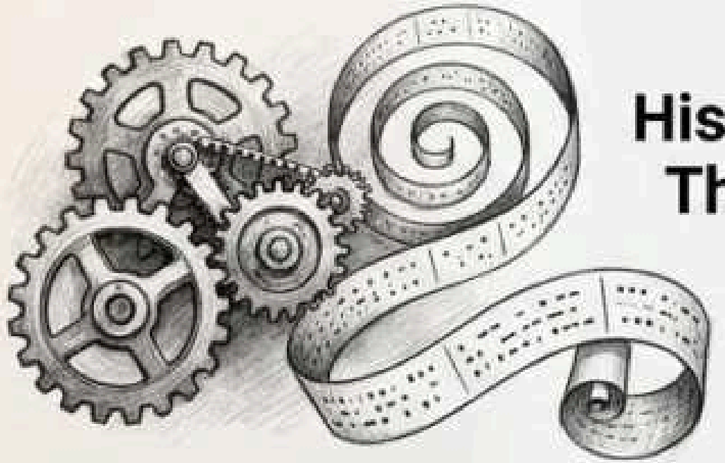
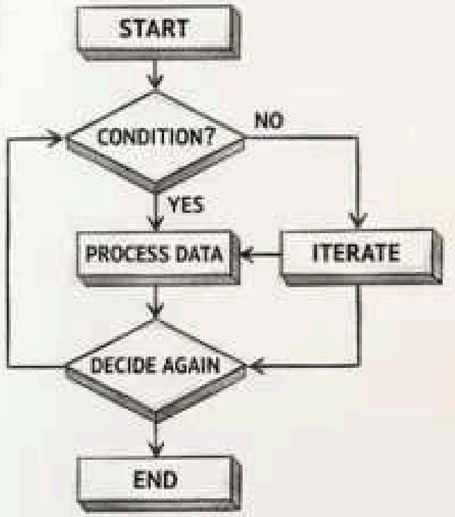
# The Foundations of Computation



History & Theory

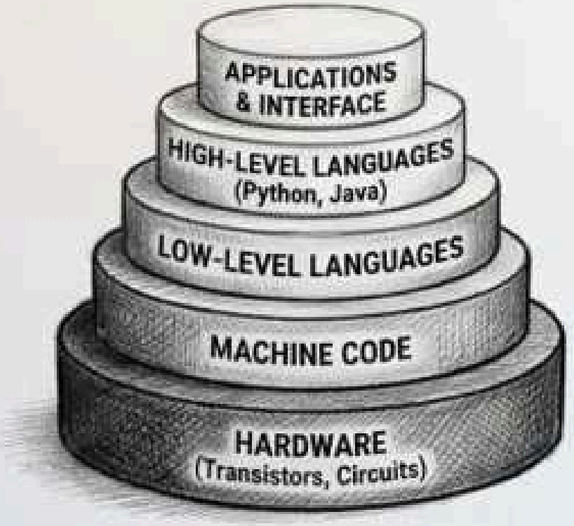
Algorithms

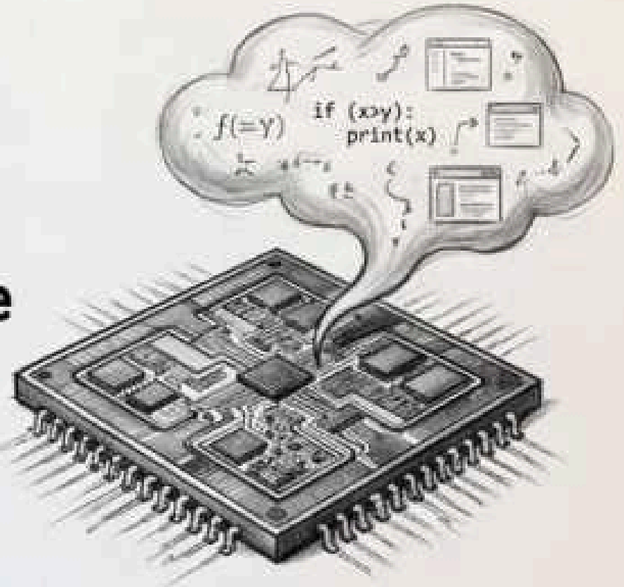



## What Computers Do

Abstraction & Decomposition

Hardware vs. Software



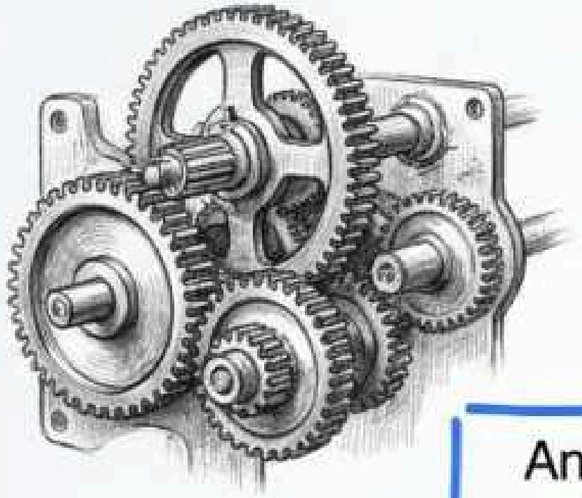



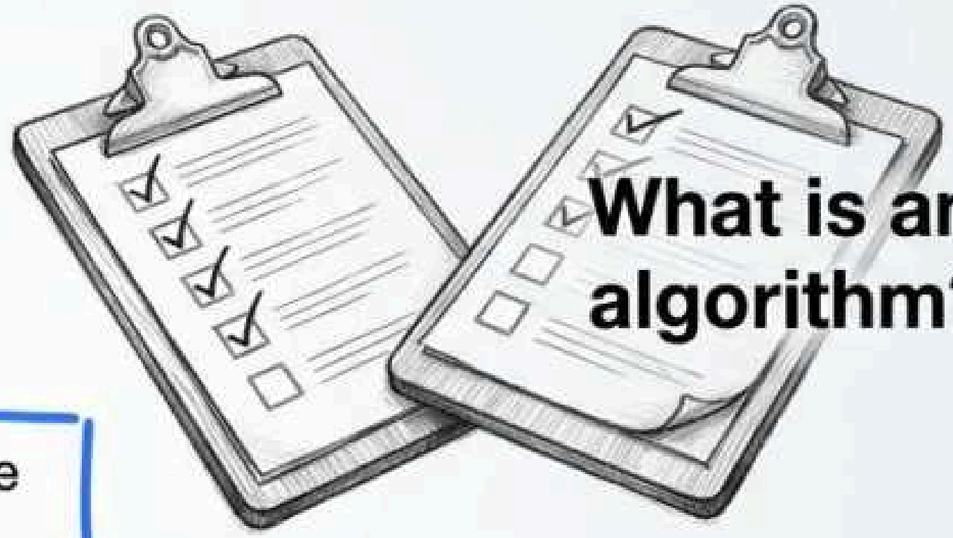

*At a glance — Welcome to Lecture 1.*

# Four Questions Define the Subject

¶ 1.2

**What is a computer?**





**What is an algorithm?**

Answering these gives you the vocabulary for everything that follows.

**How do we describe procedures exactly?**



**What is abstraction?**



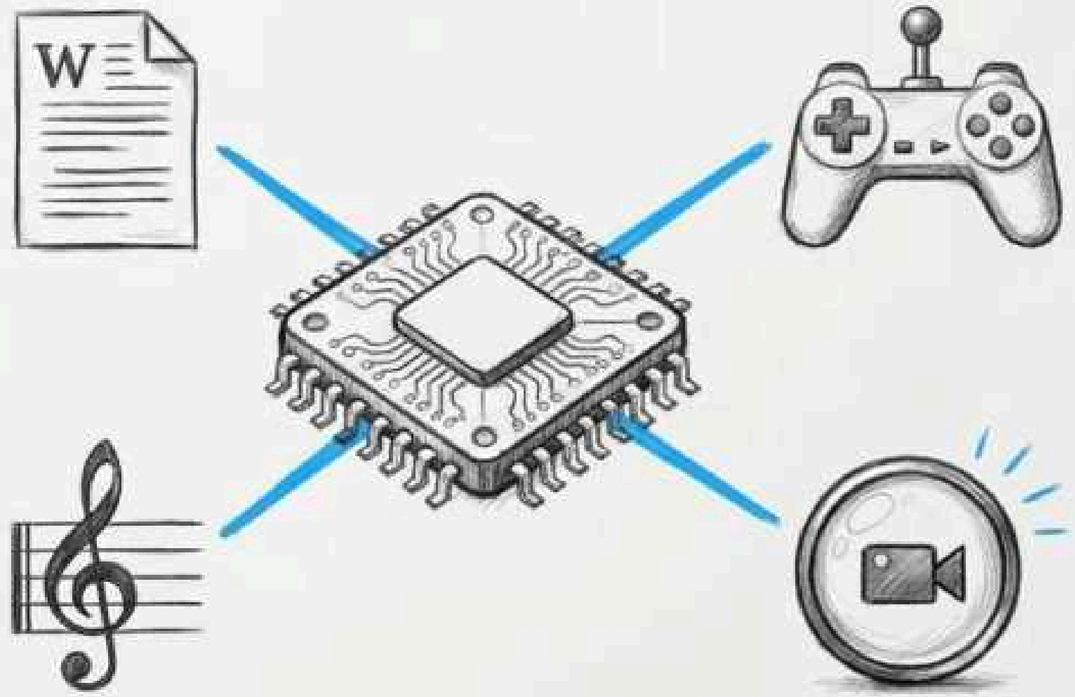
# The General Purpose Machine

## Specific Purpose




Does arithmetic and nothing else.
Hardwired.

## General Purpose




The physical machine stays exactly the same.
Only the instructions—the program—change.

¶ 1.4

# The Mechanical Pioneers





1830s

1843

**Charles Babbage designs the Analytical Engine.**

- A mechanical machine with memory and branching steps. (Never finished).

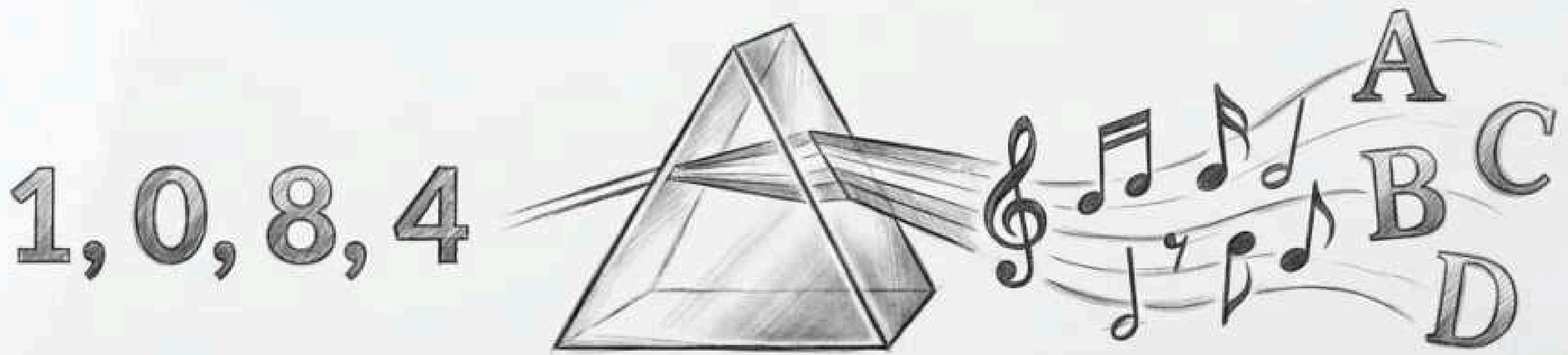
**Ada Lovelace publishes notes.**

- Creates a method for calculating Bernoulli numbers—the first program.

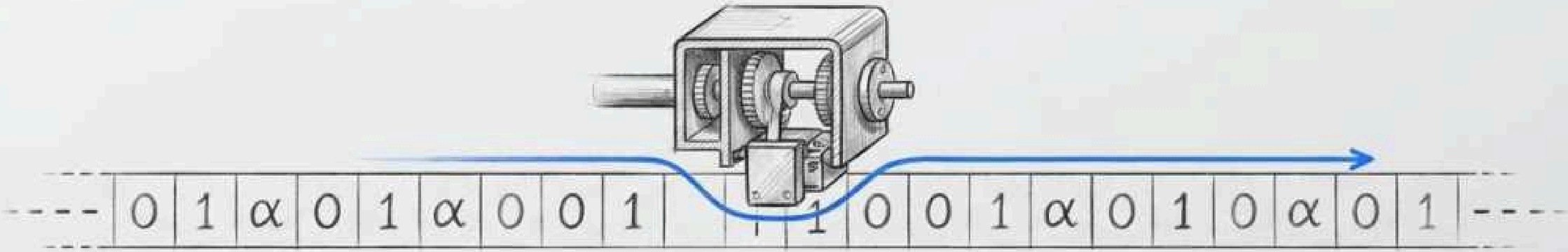
*Before electronics, a "computer" was a person who calculated by hand.*

# The Universal Machine

¶ 1.5

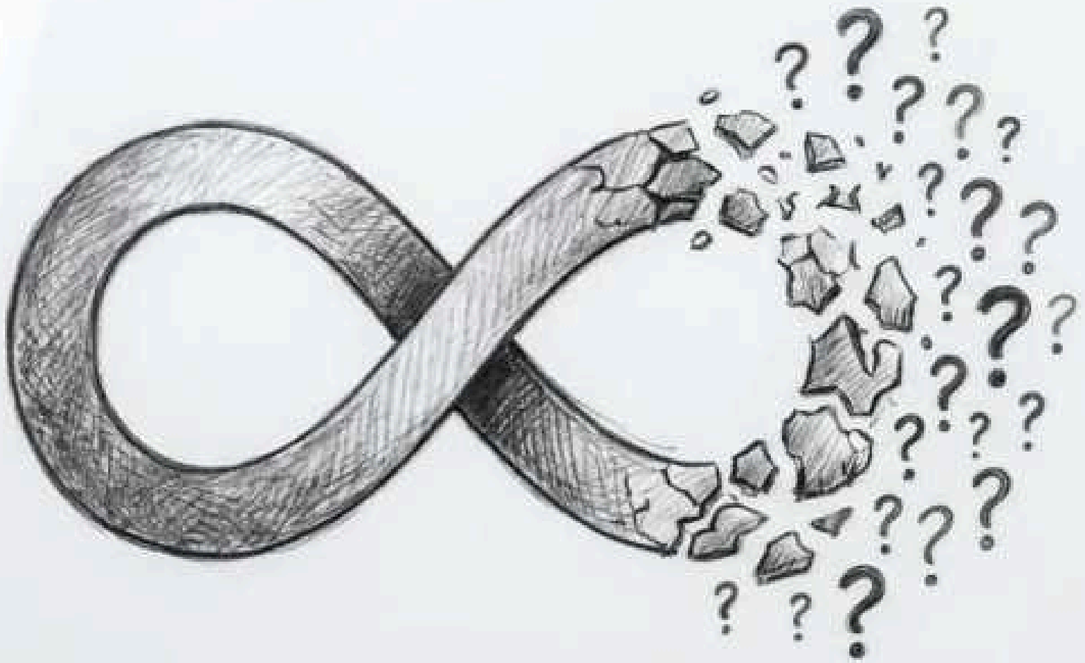



**Lovelace:** Machines can work on any symbols, not just numbers.

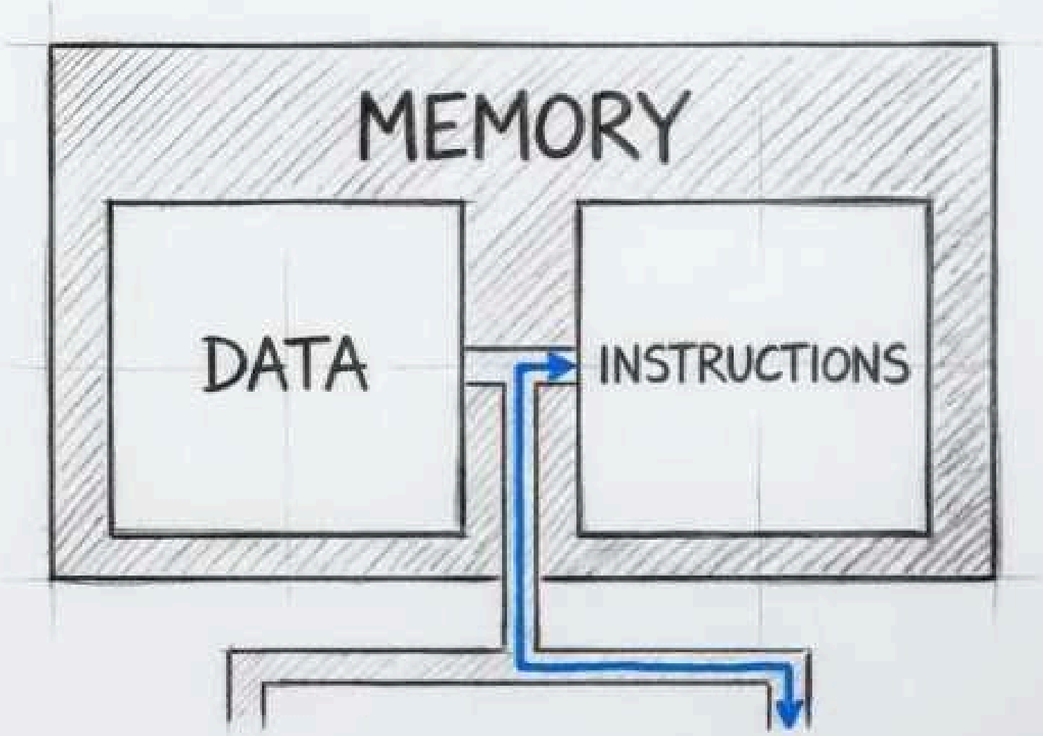



**Alan Turing:** One universal abstract machine can imitate any other.

# Limits and Architecture



**Turing's Limit:** Some questions can never be answered by any program.




**von Neumann Design:** Instructions live in the same memory as data. Almost every computer follows this.

¶ 1.7

# Algorithms Leave Nothing to Judgment

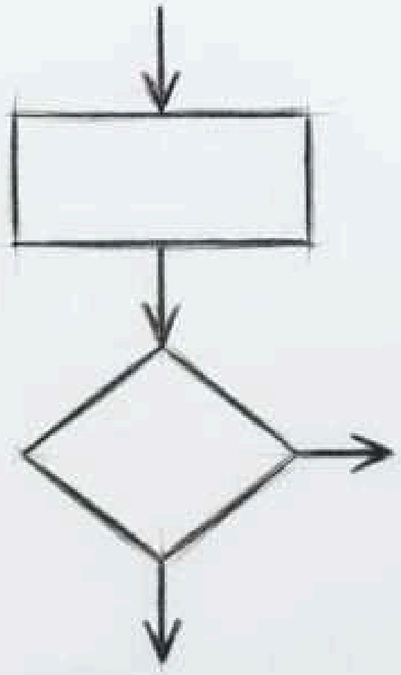
## Recipe



~~Add a pinch of salt.~~

Requires human judgment. A machine cannot judge a pinch.

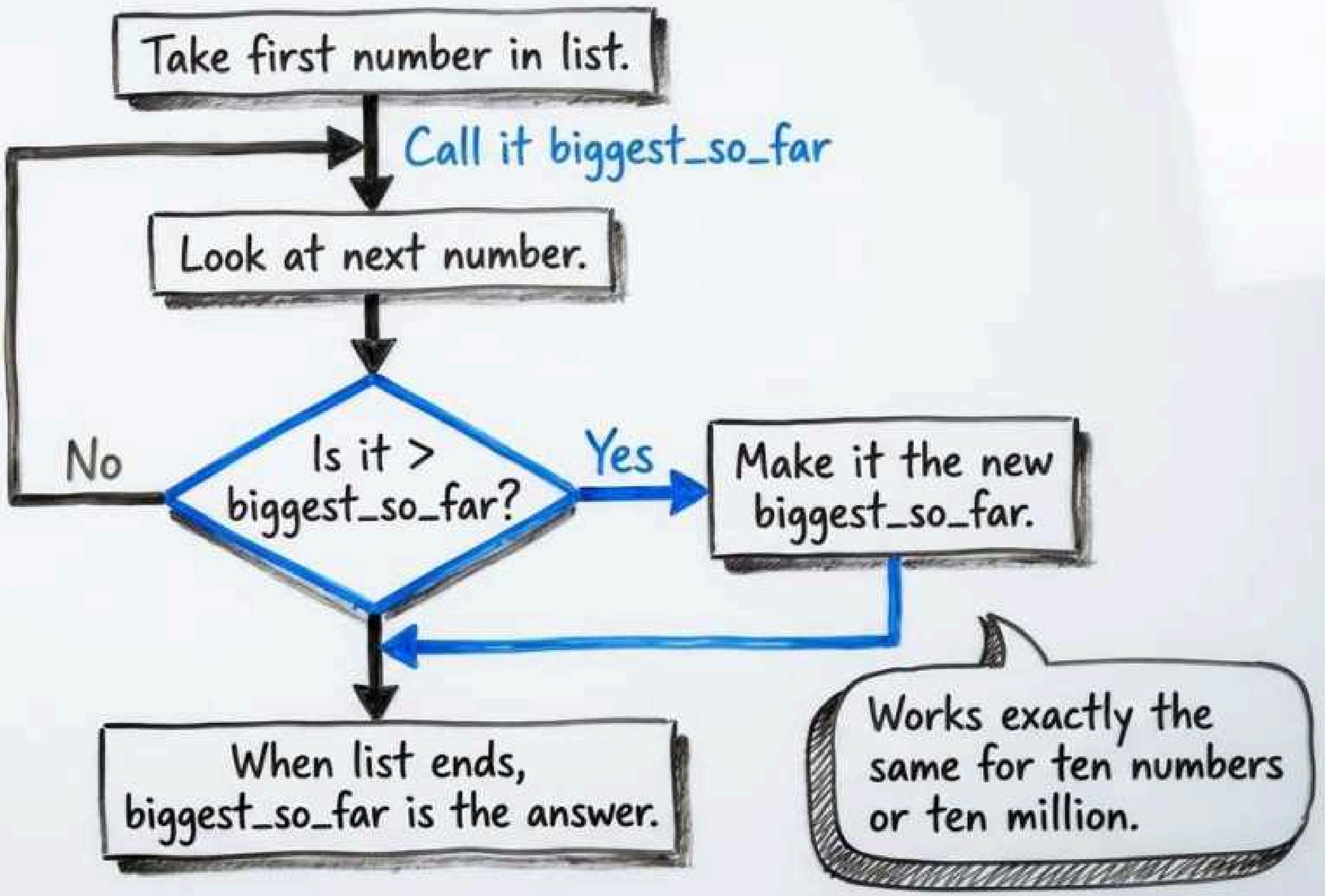
## Algorithm



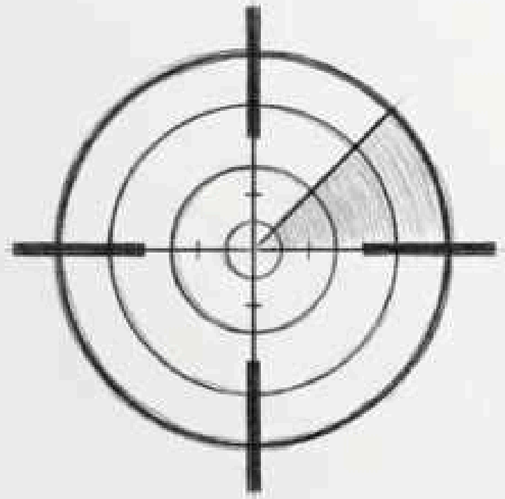
A precise, step-by-step procedure that always finishes and produces an answer.

Origin: al Khwarizmi, 9th-century Baghdad mathematician.

# An Algorithm in Action



# The Four Properties of a Good Algorithm



## Precise

Every step is unambiguous.



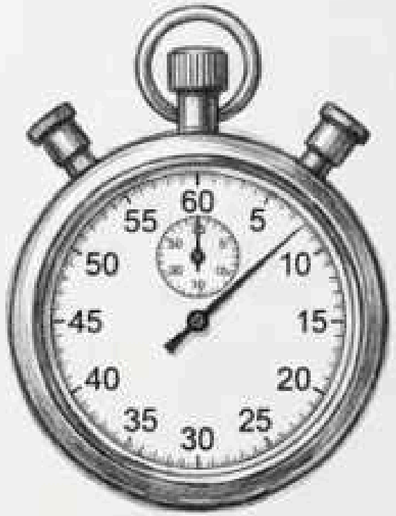
## Finite

It ends.



## Correct

Gives the right answer for every valid input, not just the ones tested.

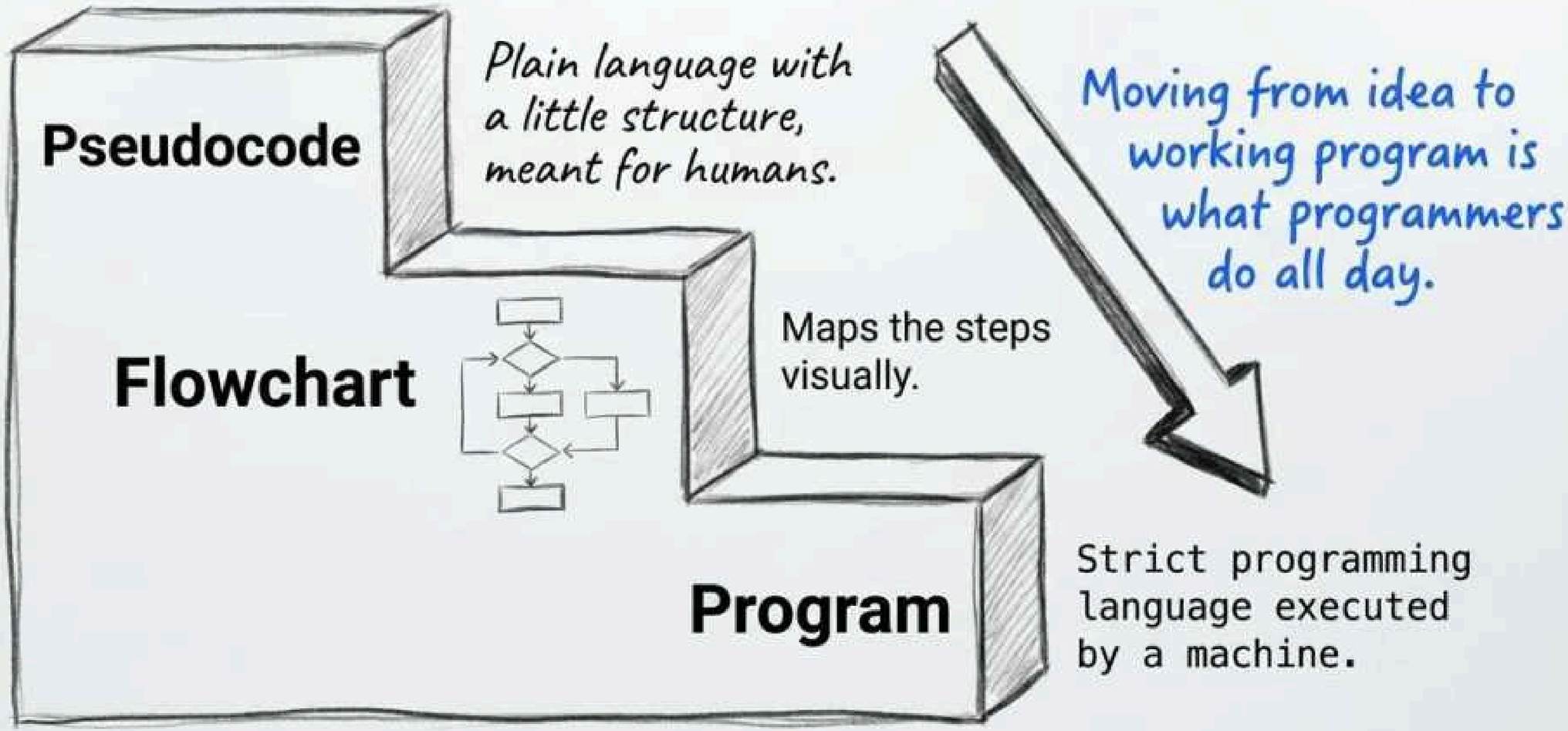



## Efficient

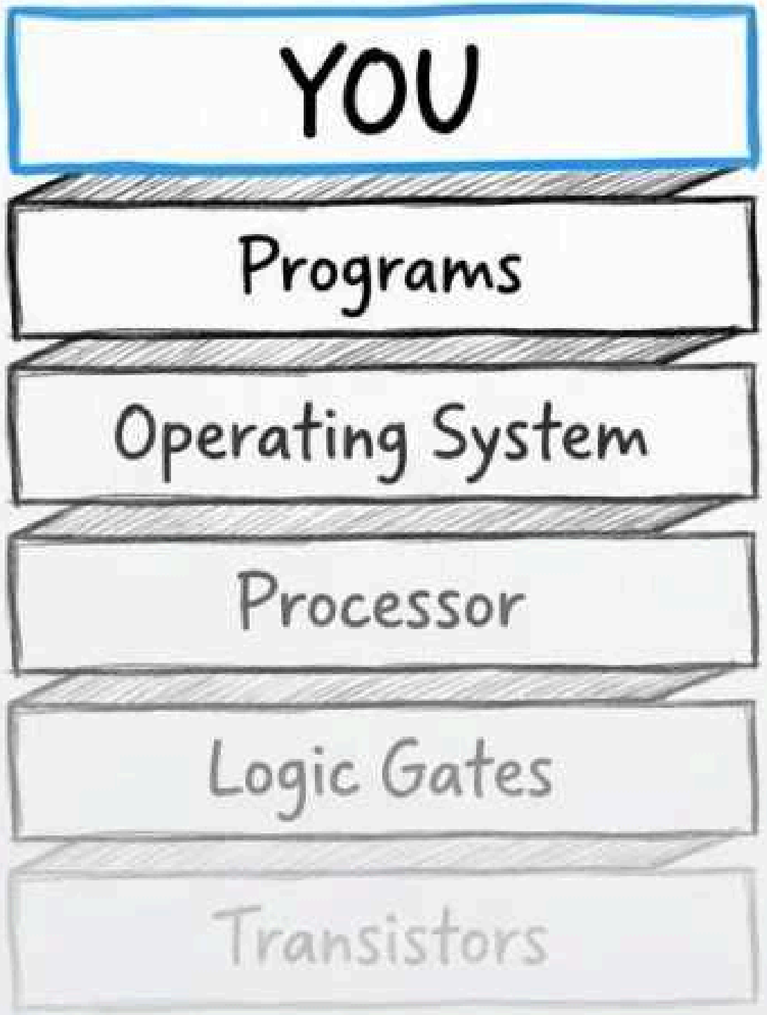
Does not waste time or memory.

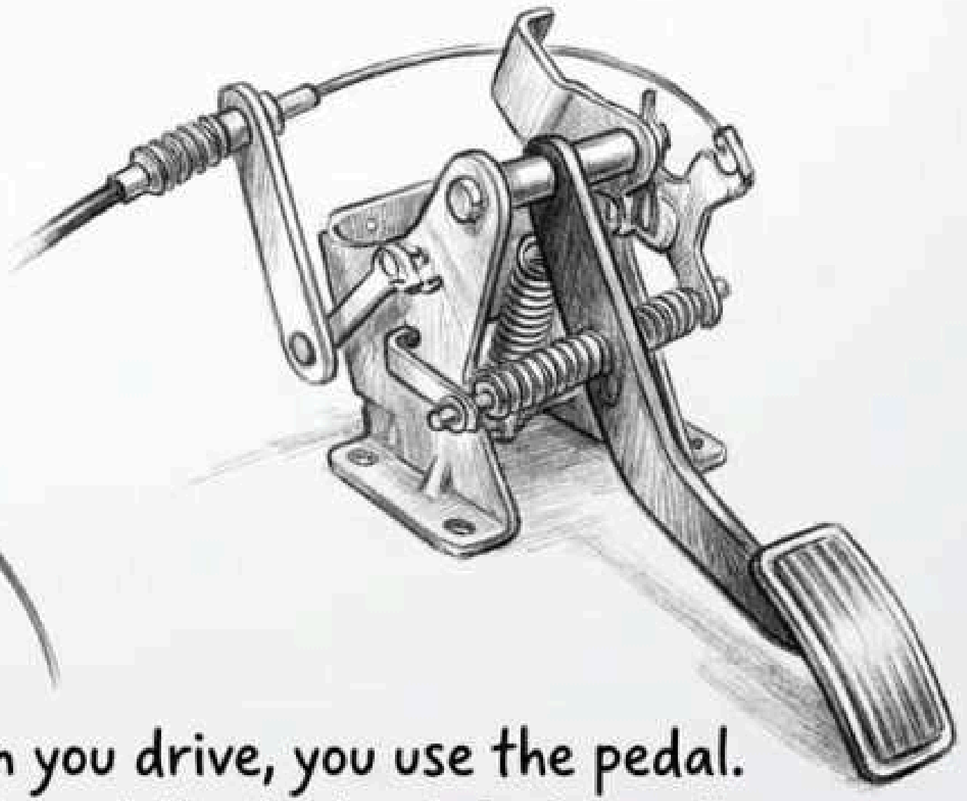
Beginners focus on correctness first. Mastery requires efficiency.

# Translating Thought to Machine



# The Abstraction Layer Cake



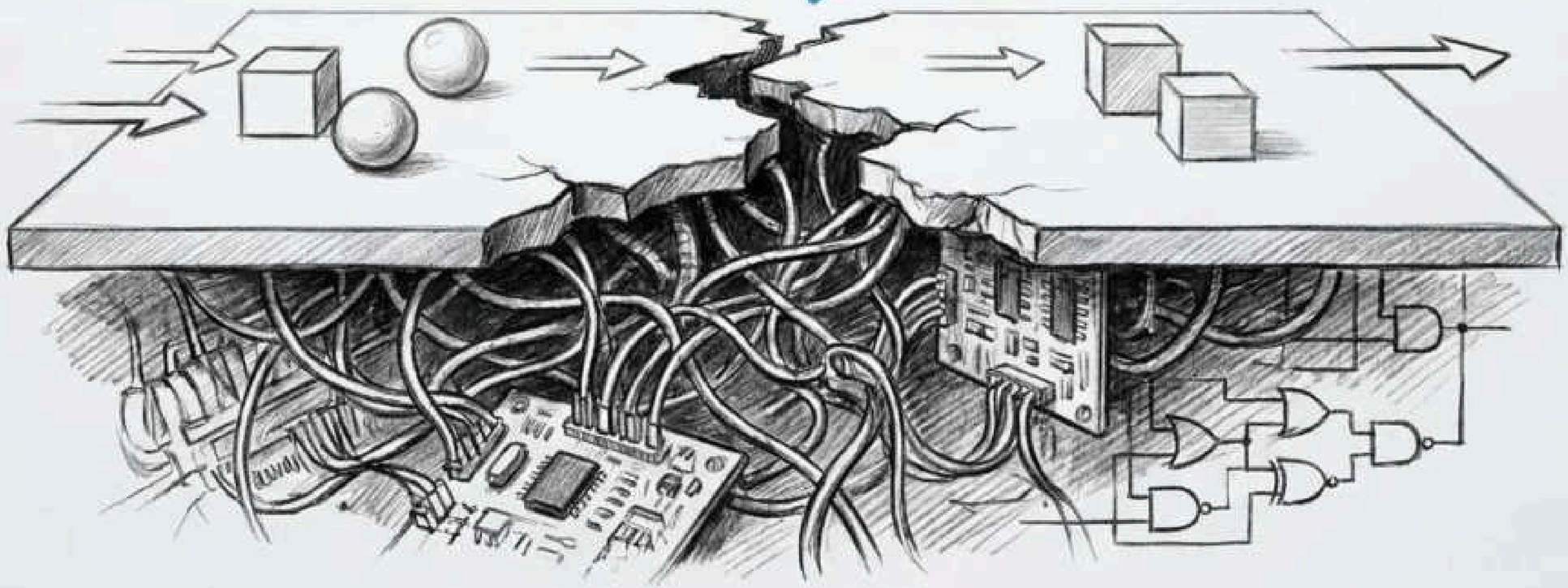



When you drive, you use the pedal.
You do not think about fuel injection.
To abstract is to hide detail behind
a simple description.

# The Limits of Abstraction

No human can hold a whole modern system in mind. We trust the layers below.

Abstractions leak. When a system breaks, you must look beneath the surface to see why.

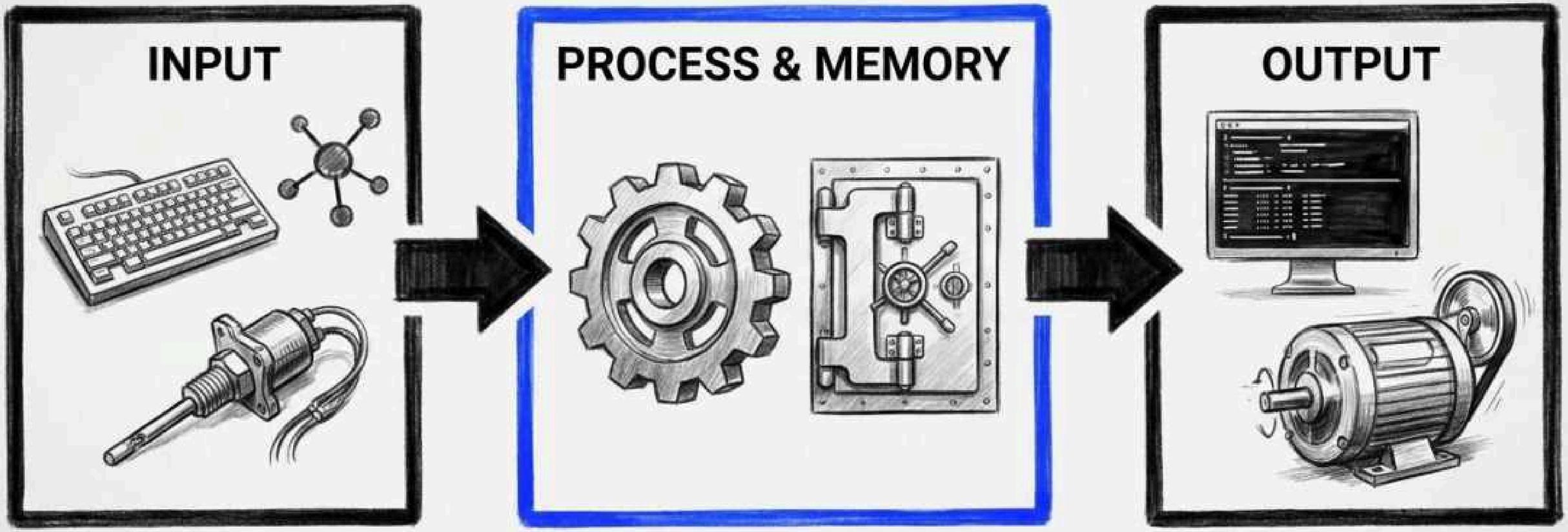


# Decomposition: Conquering Complexity




Break large problems into smaller ones. Solve and test each piece on its own, then join them together.

# The Universal Pattern: I.P.O.




When you meet a new program, ask:
What goes in? What happens? What comes out?

# Hardware, Software, and the Mediator

**Software**

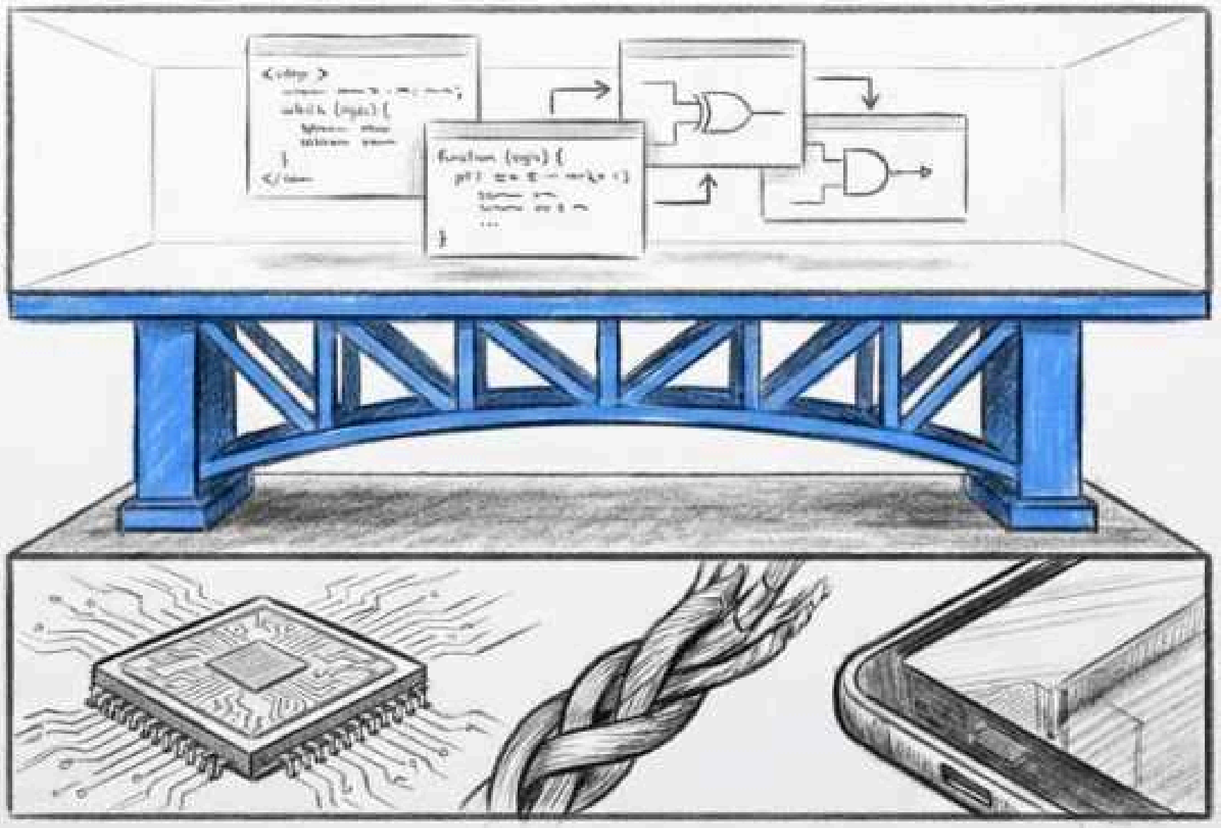
Instructions telling hardware what to do.

**Operating System**

Manages the machine and shares it among programs.

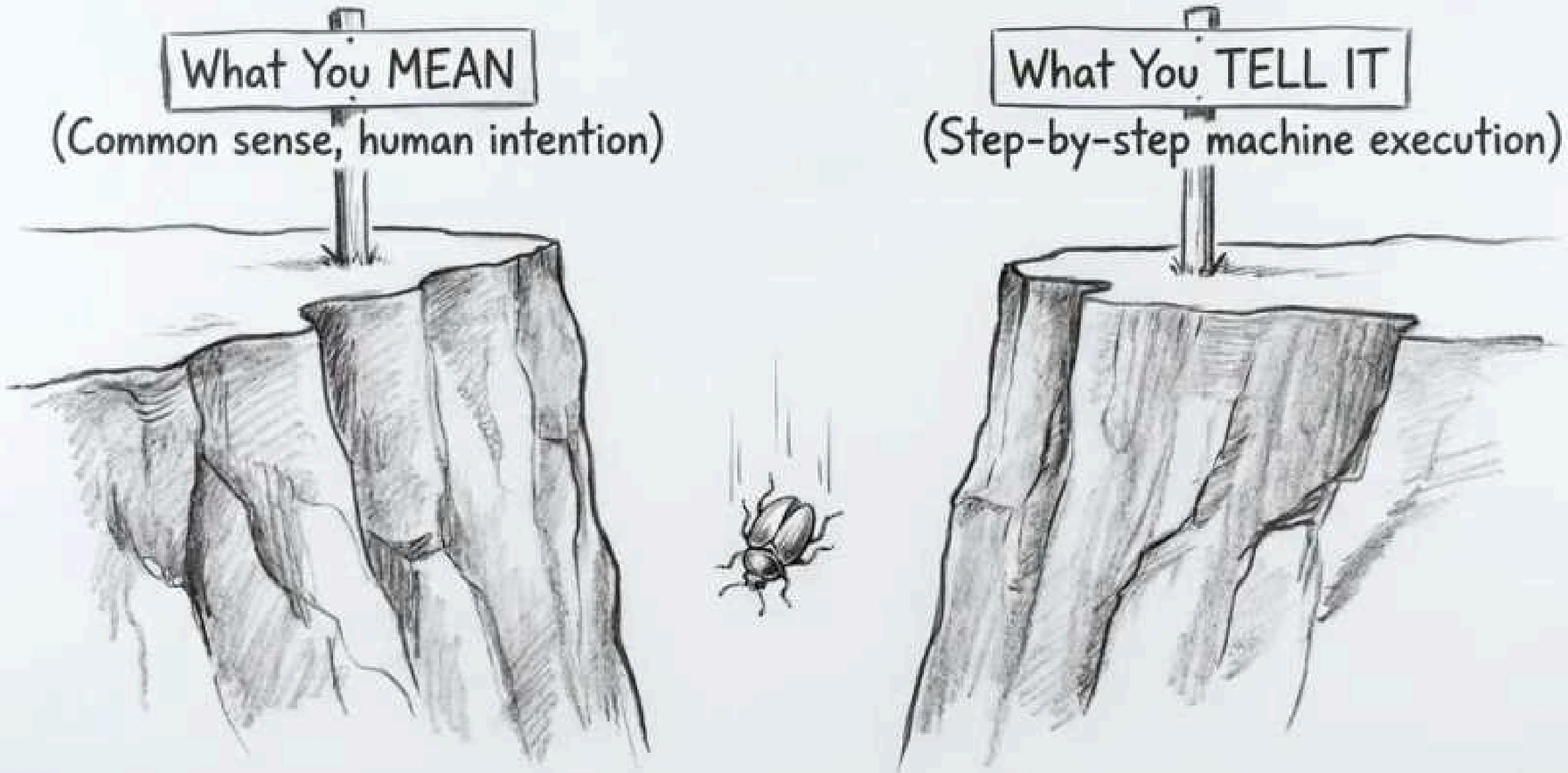
**Hardware**

The physical machine.



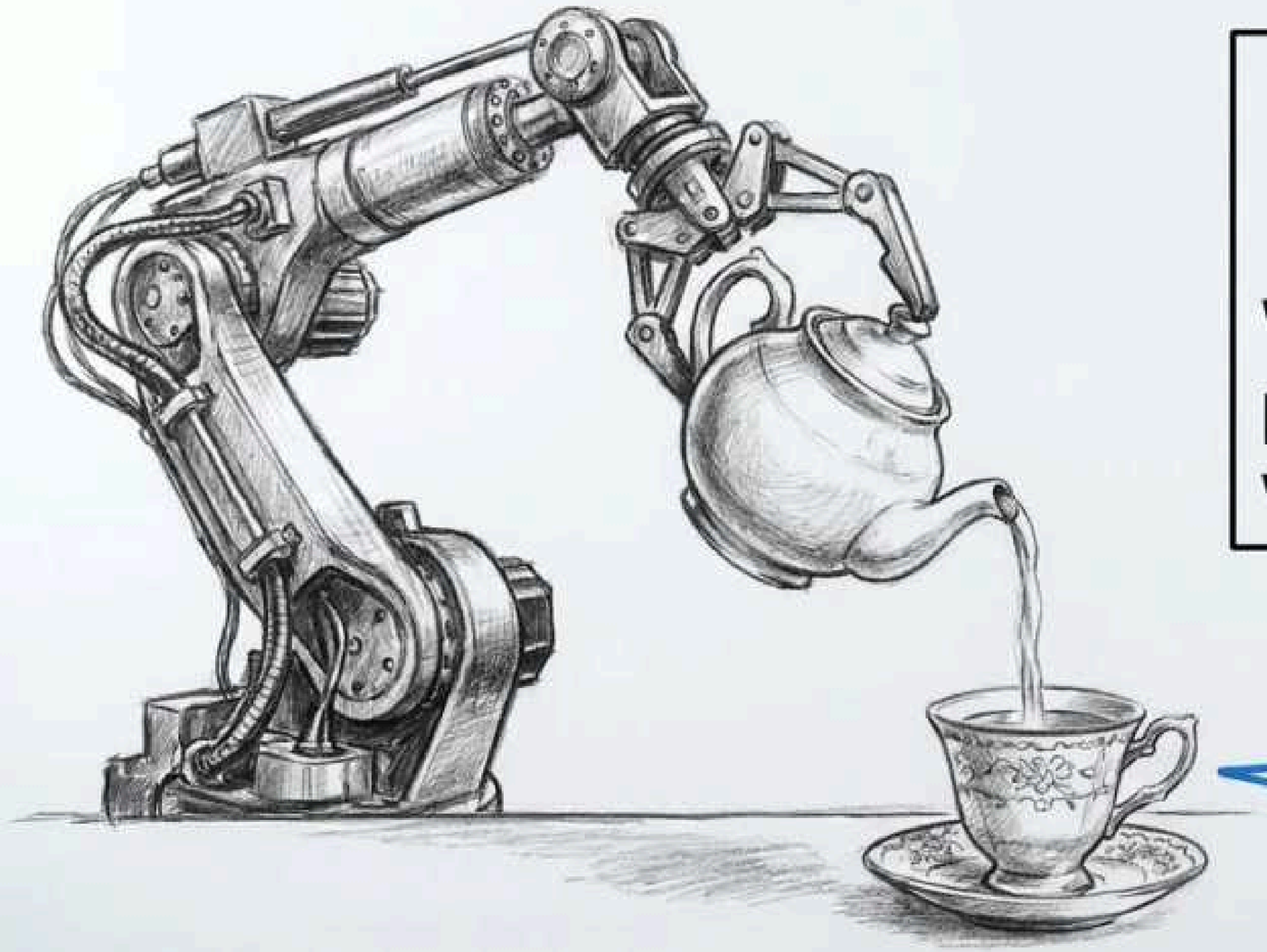
Phones and laptops differ in physical hardware, but they share these exact same underlying ideas.

# The Human-Machine Gap




A computer does exactly what you tell it. Programming is the discipline of saying exactly what you mean, with nothing left to common sense.

# Exercise: The Senseless Robot

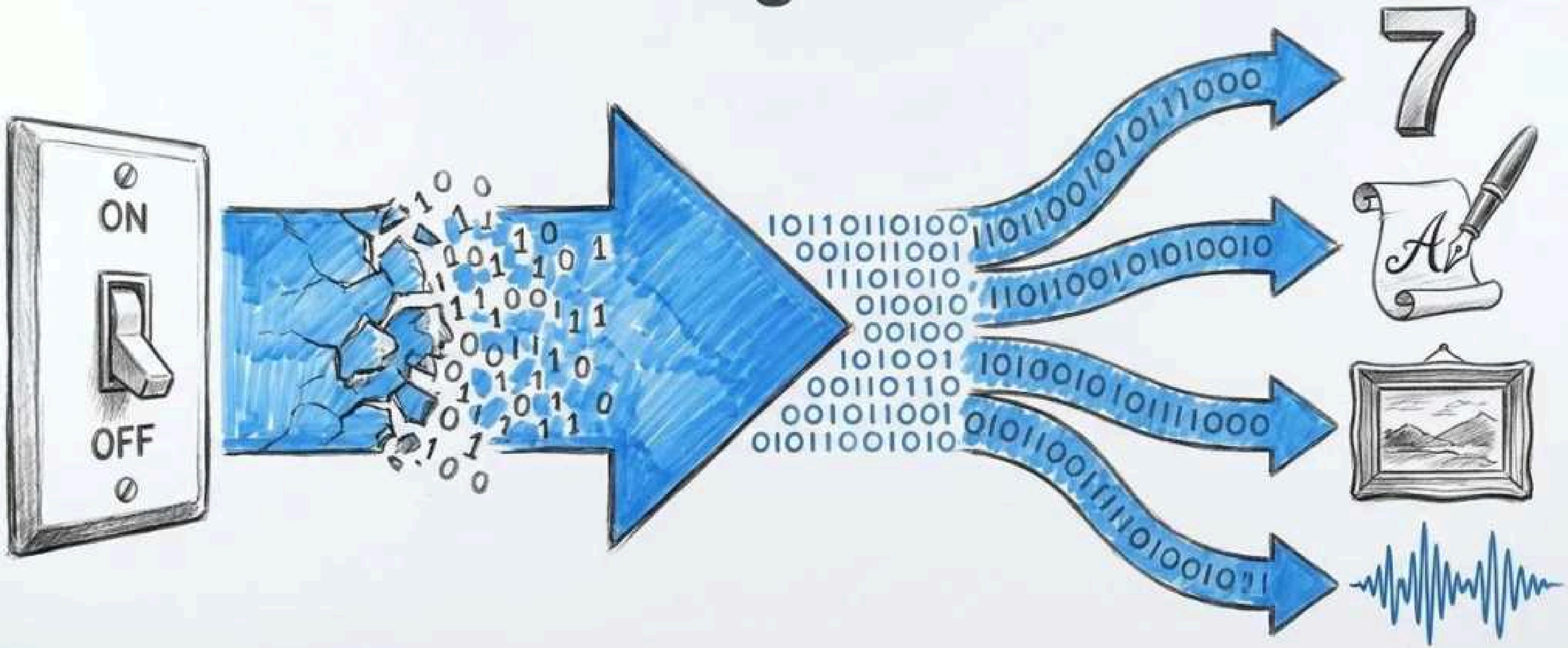


Think of a small daily routine (e.g., making tea).

Write it as a list of steps precise enough for a robot with zero common sense.

Where did you rely on something a machine wouldn't know? How do you define "full"?

# Looking Ahead




**Lecture Two**: How a machine made of simple switches can store numbers, words, pictures, and sound.