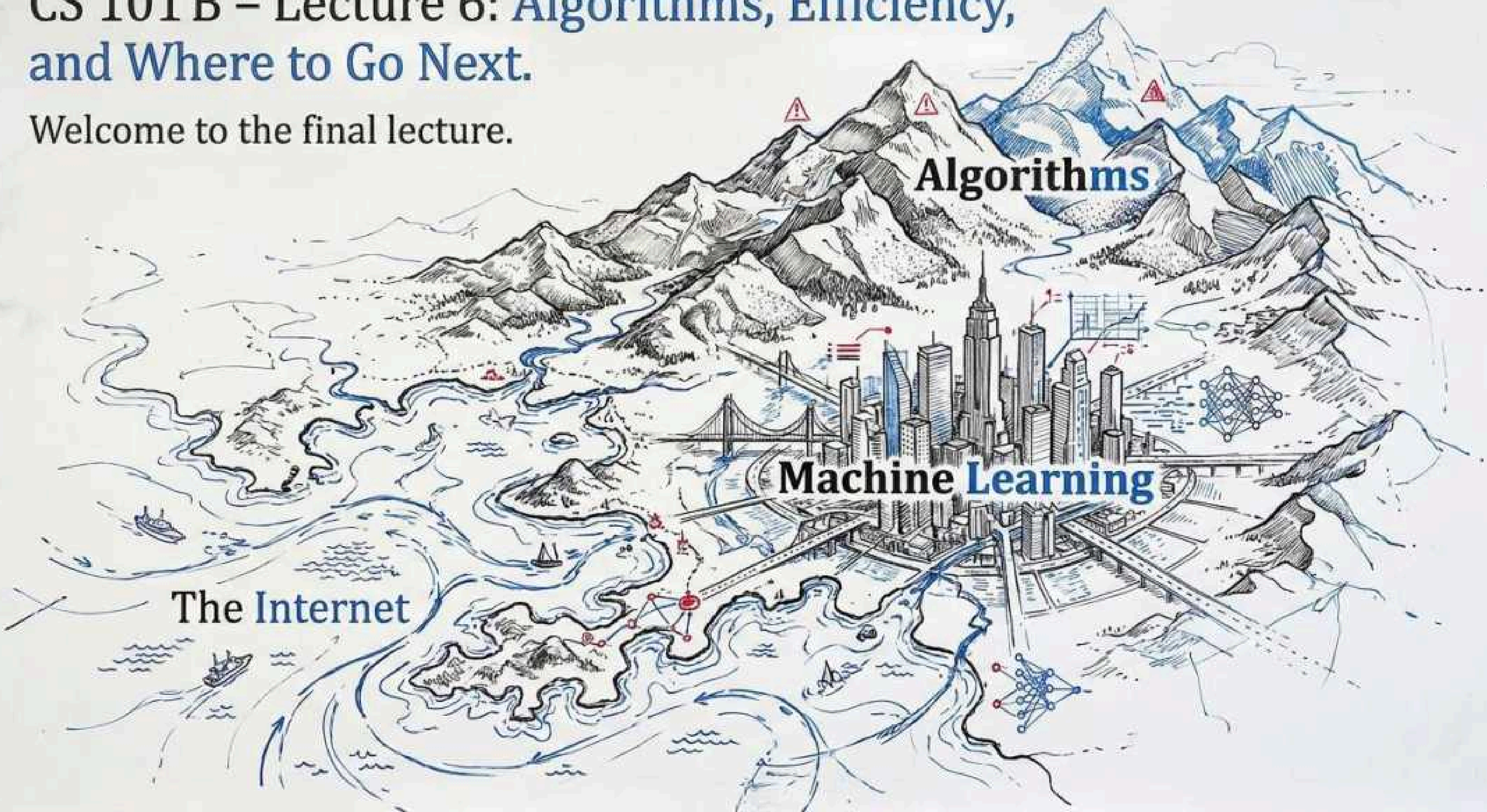
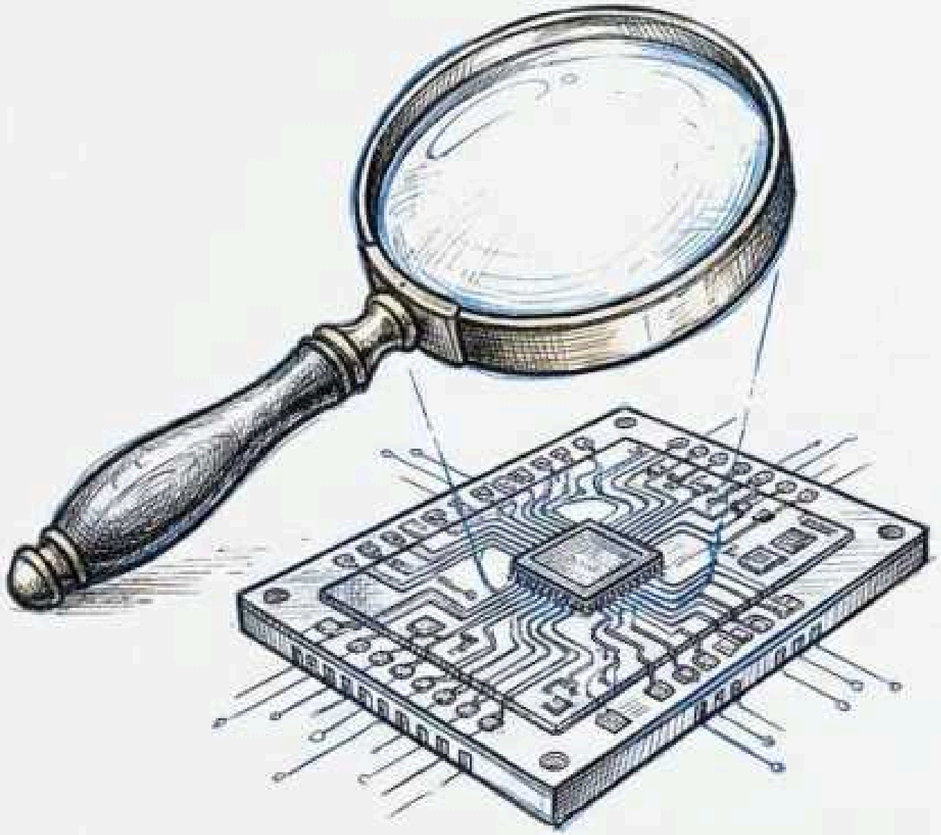
CS 101 B – Lecture 6: Algorithms, Efficiency, and Where to Go Next.

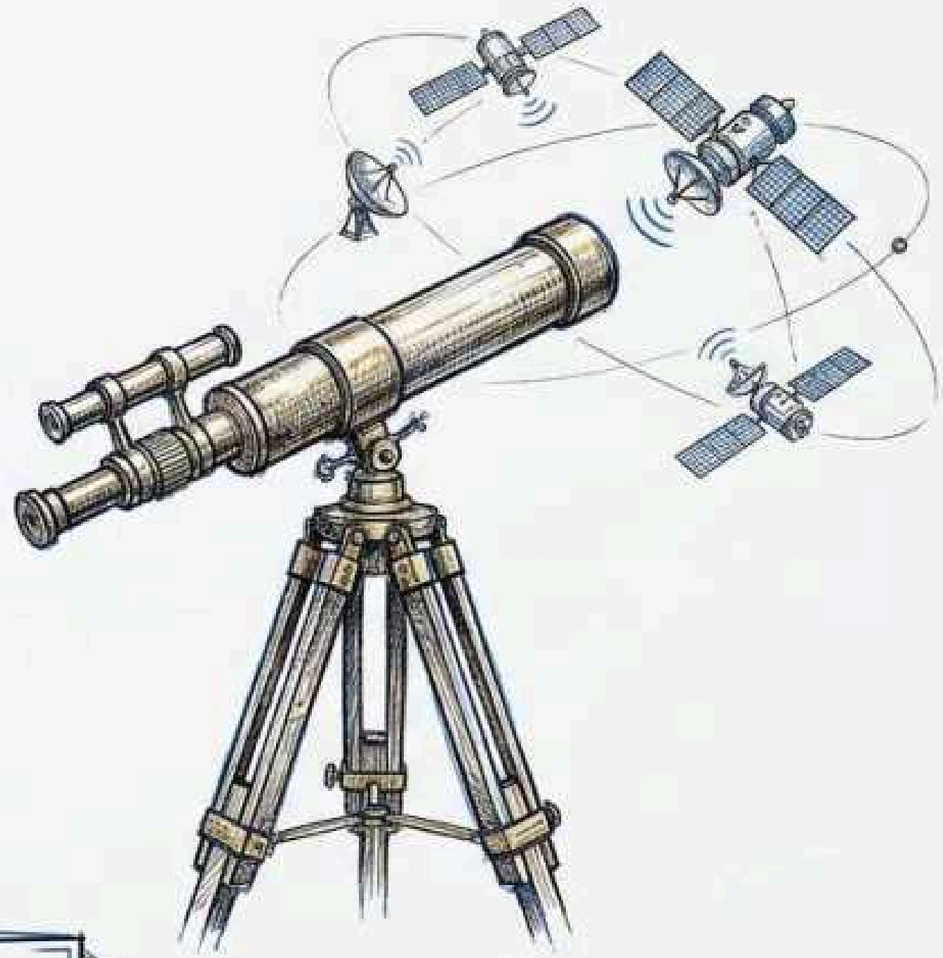
Welcome to the final lecture.





The Micro: Comparing algorithms, searching, sorting, and recursion.

Think of it as a map of the country you have just entered.

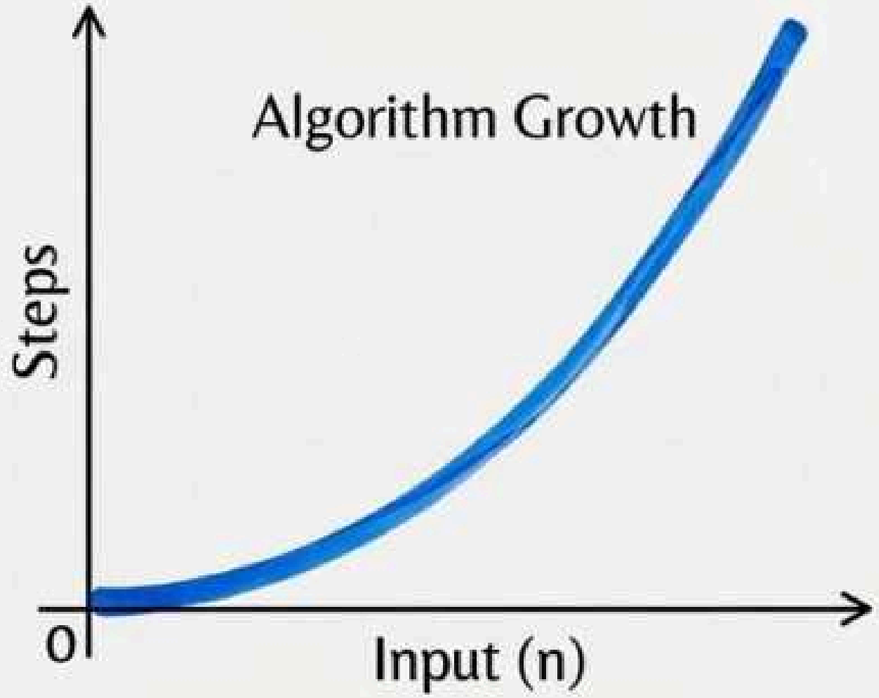


The Macro: Networks, security, artificial intelligence, and ethics.

Measuring Algorithms Fairly



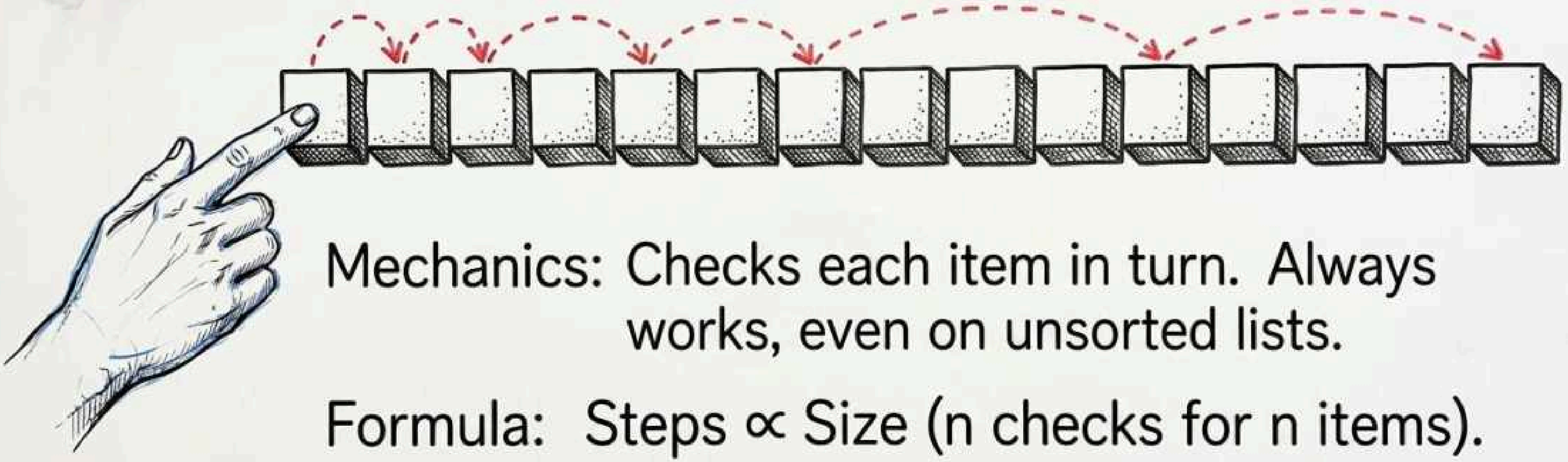
We do not time algorithms in seconds (which depends on the machine's hardware).



We measure how the number of steps grows as the input (n) grows.

Work that doubles vs. work that quadruples when the data doubles.

Linear Search

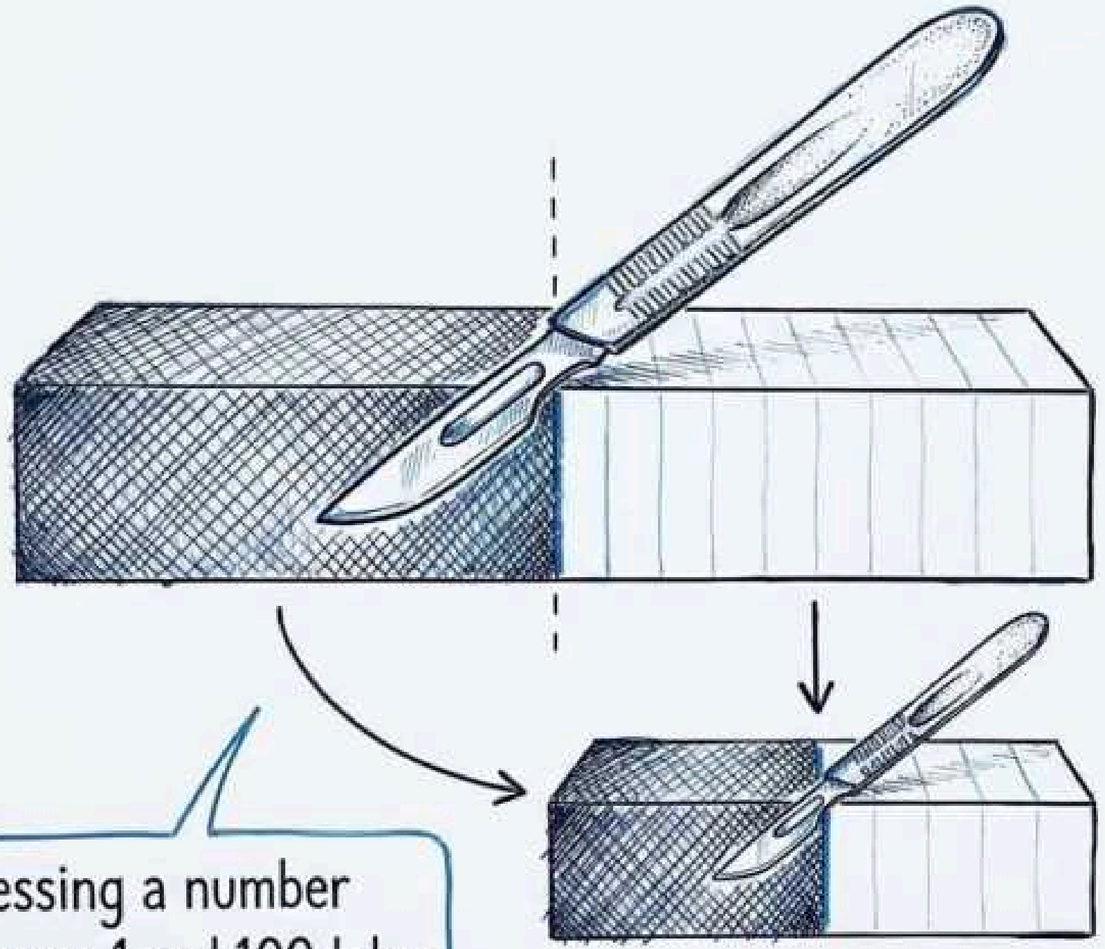


Mechanics: Checks each item in turn. Always works, even on unsorted lists.

Formula: Steps ∝ Size (n checks for n items).

The Catch: Costly at scale (e.g., 1 million items = 1 million checks).

Binary Search (Requires Sorted List)

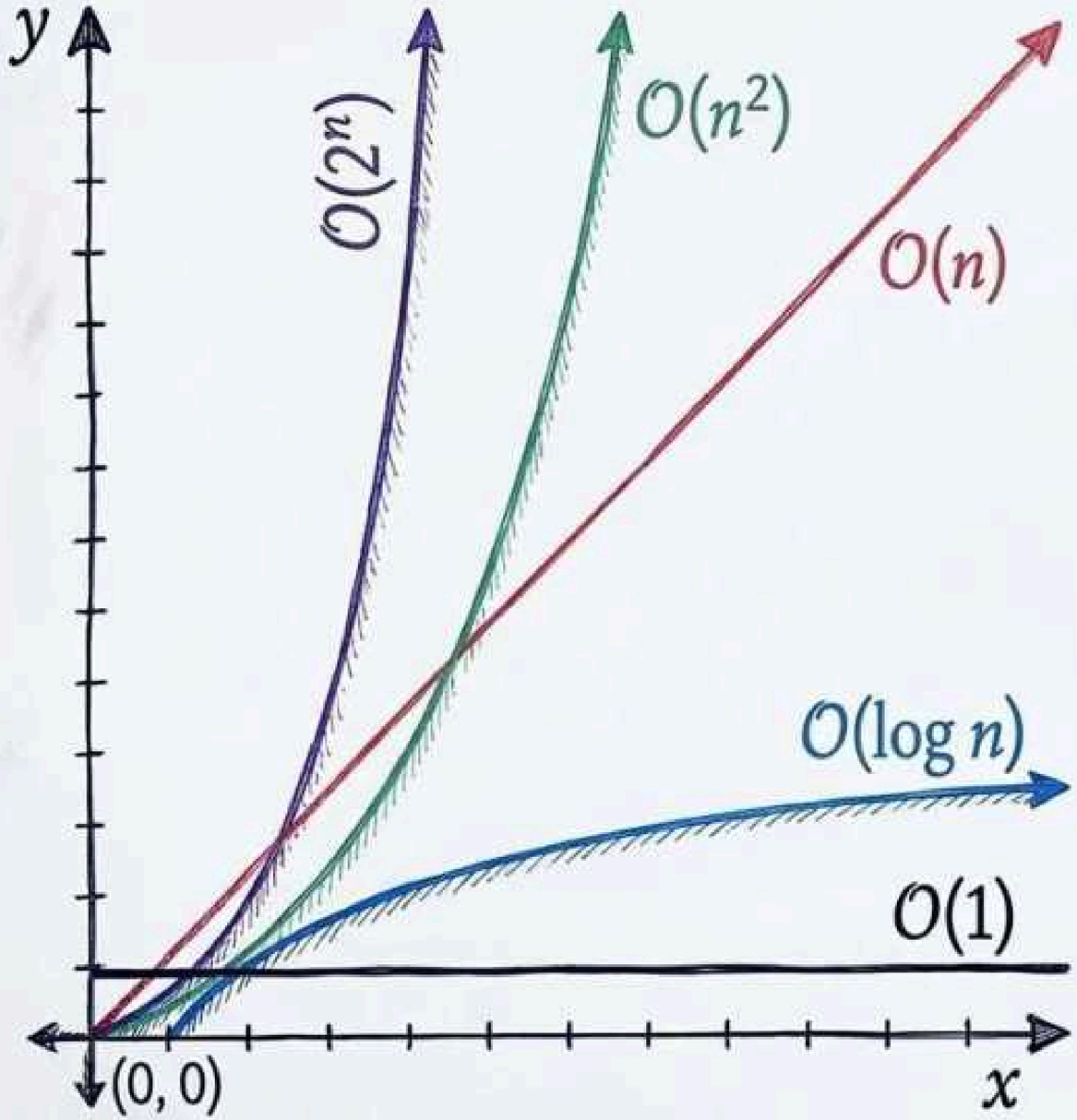


Guessing a number between 1 and 100 takes at most 7 guesses.

Mechanics:

Look at the middle → discard the irrelevant half → repeat.

The Math:	
1,000 items	≈ 10 steps.
1,000,000 items	≈ 20 steps.
1,000,000,000 items	≈ 30 steps.



$O(1)$ Constant:
Work does not grow at all.

$O(\log n)$ Logarithmic (e.g., Binary Search):
Adds one step when data doubles.

$O(n)$ Linear:
Grows in direct proportion.

$O(n^2)$ Quadratic (e.g., Nested Loops):
Grows with the square.

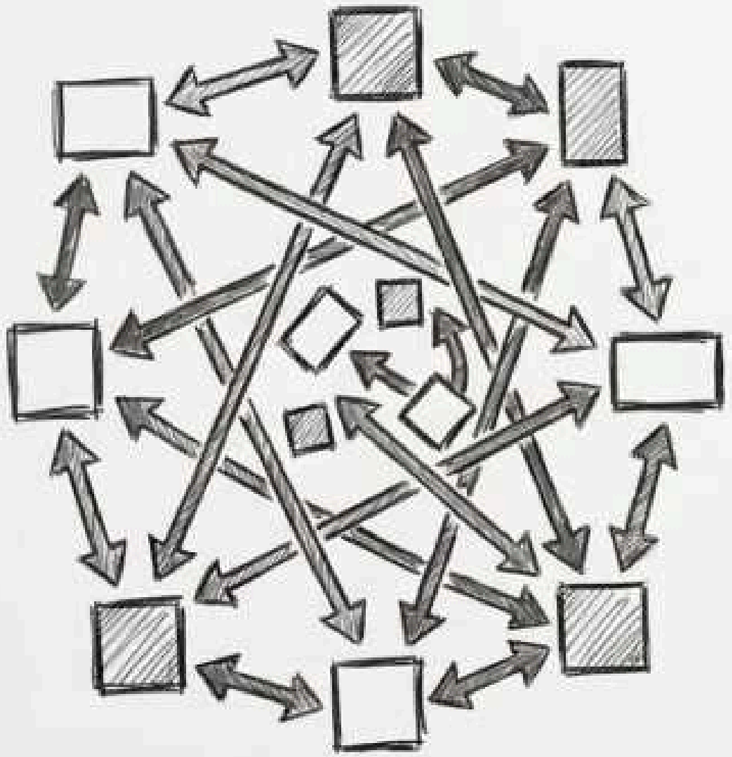
$O(2^n)$ Exponential: Explodes instantly;
modest inputs become impossible.

Algorithm Type	1,000 Items	1,000,000 Items
Logarithmic	~10 steps	~20 steps
Linear	1,000 steps	1,000,000 steps
Quadratic	1,000,000 steps	1 Trillion steps (Takes hours or days)

Choosing a good algorithm often beats buying a faster computer.

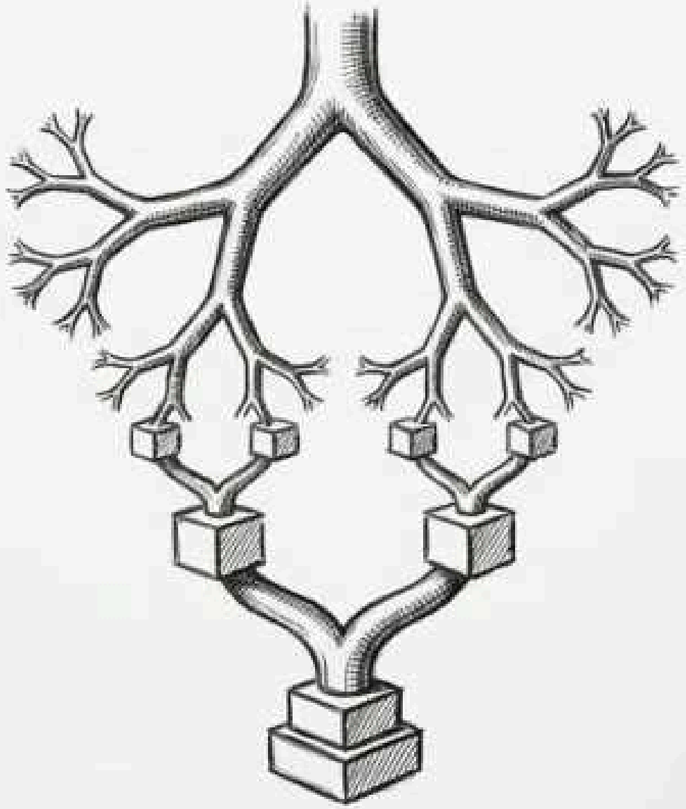
Simple (Quadratic):

Bubble & Selection sort compare repeatedly. Slow on large lists.



Clever (Log-Linear):

Merge sort splits, sorts, and merges. Far fewer steps.

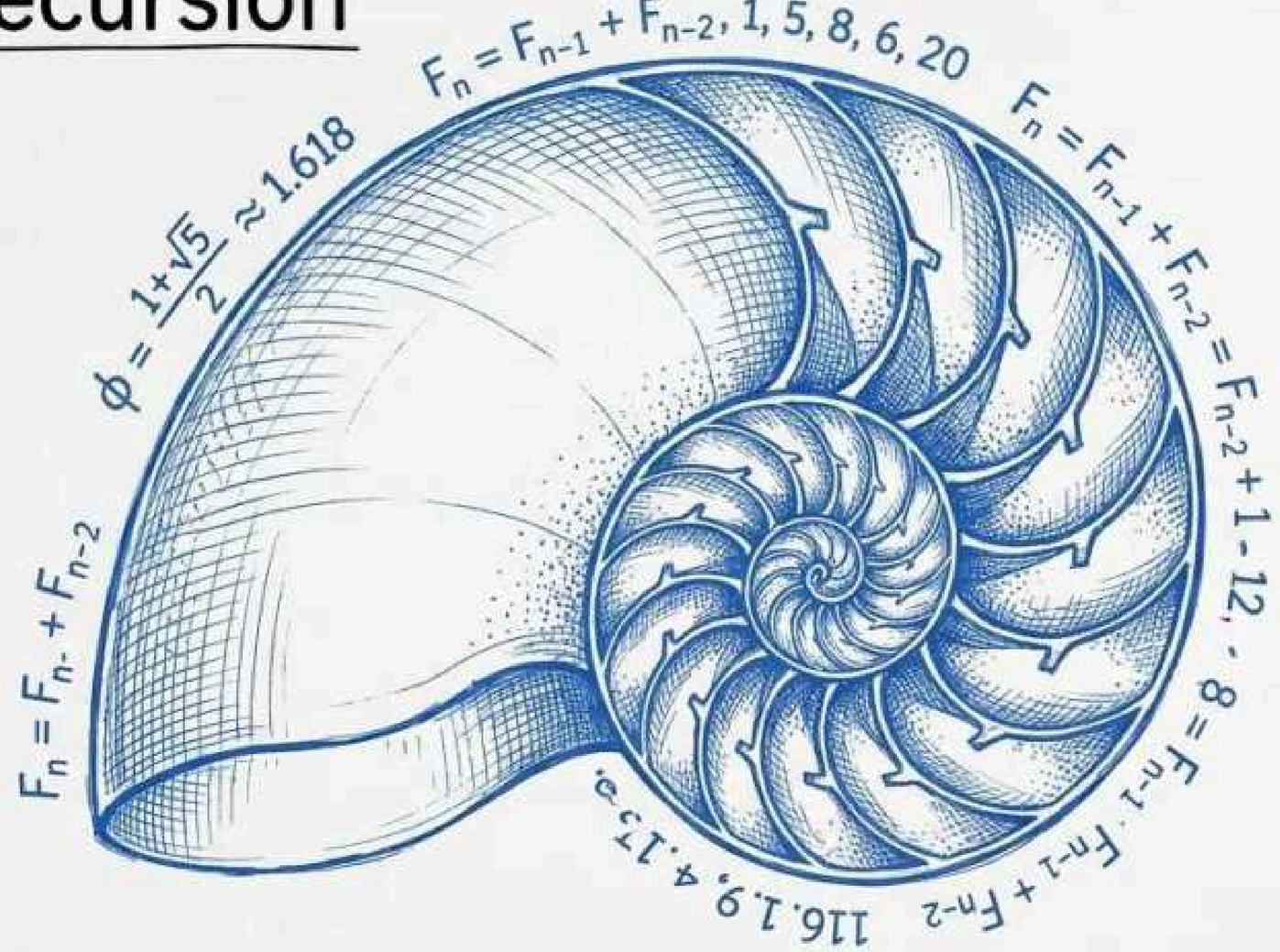


Practical:

Use Python's built-in `sort()` — it is fast and well tested.



Recursion



Concept:

A function solves a problem by calling itself on a smaller version.

The Formula:

$n! = n \times (n-1)!$

(with base case $0! = 1$)

The Golden Rule: Every recursive function needs a base case to stop the process. Without it, the calls never stop and the program crashes.



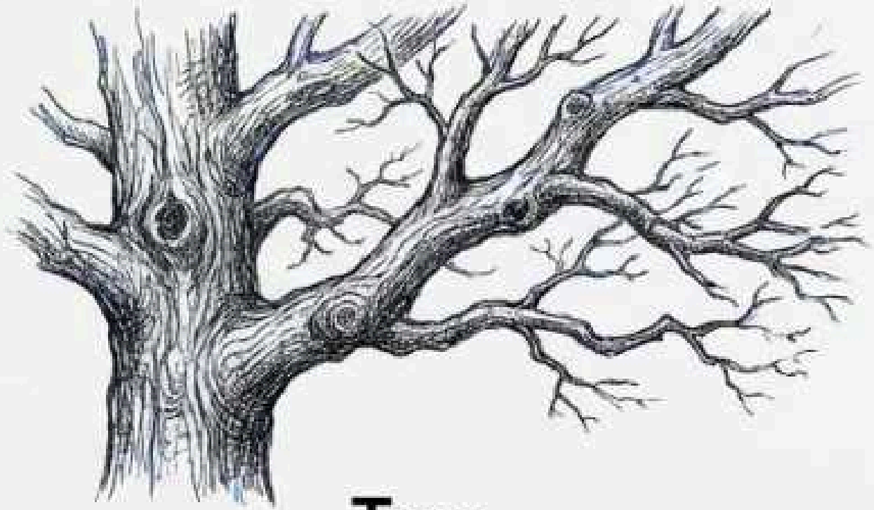
Stack (LIFO)

Last in, first out. Powers the Undo button.



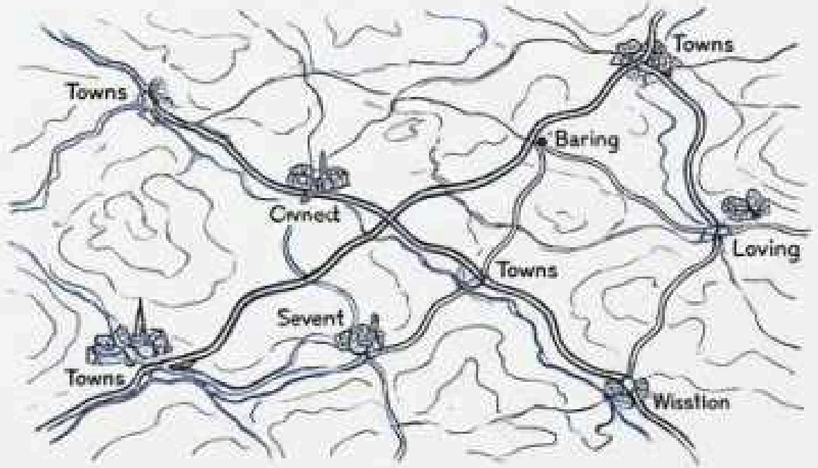
Queue (FIFO)

First in, first out. Manages print jobs.



Tree

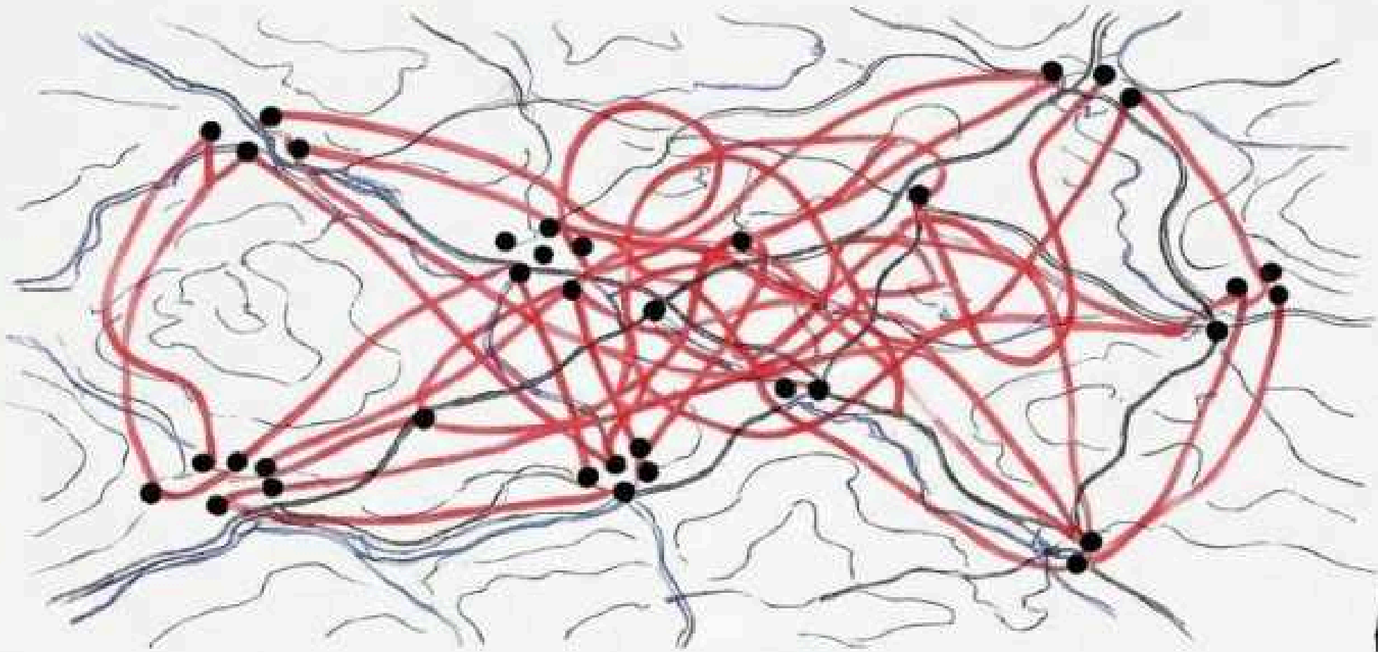
Arranges items in branches. Manages file folders.



Graph

Links items by connections. Maps web pages and links.

Hard and Impossible Problems

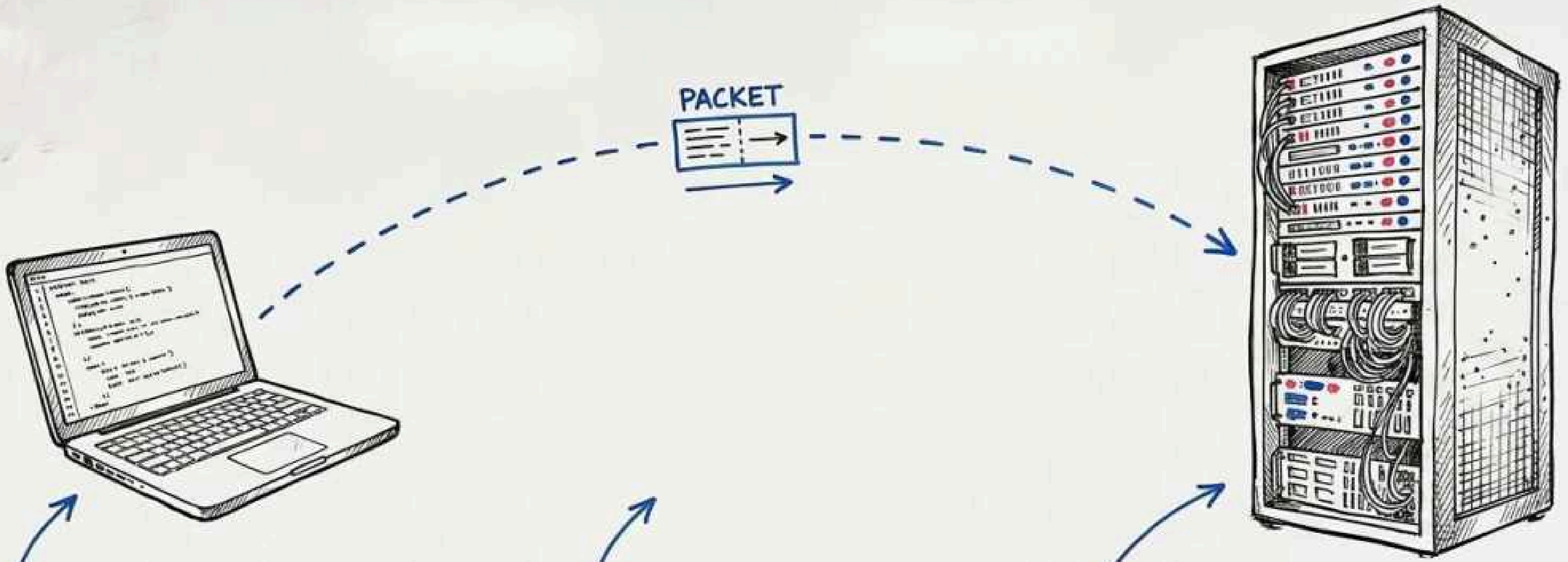


Travelling Salesman: What is the shortest route visiting many cities?
No efficient solution is known that always works.

P vs. NP: The Millennium Prize problem regarding efficient solvability.

The Halting Problem: Turing proved that no program can decide,
for every program, whether it will ever stop running.

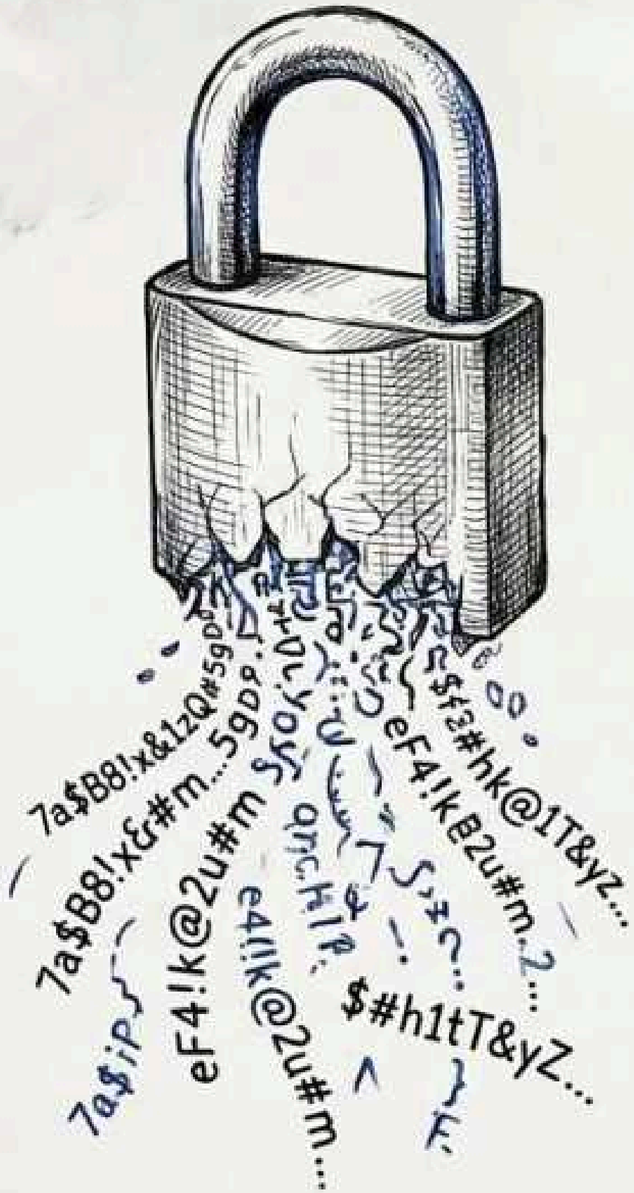




Packets & Protocols:
Data travels in small packets governed by agreed rules (protocols) for addressing and reassembly.

Client-Server Model:
Browsers request pages; servers deliver them.

DNS (Domain Name System):
The internet's address book—turns a name into the numeric address of a machine.



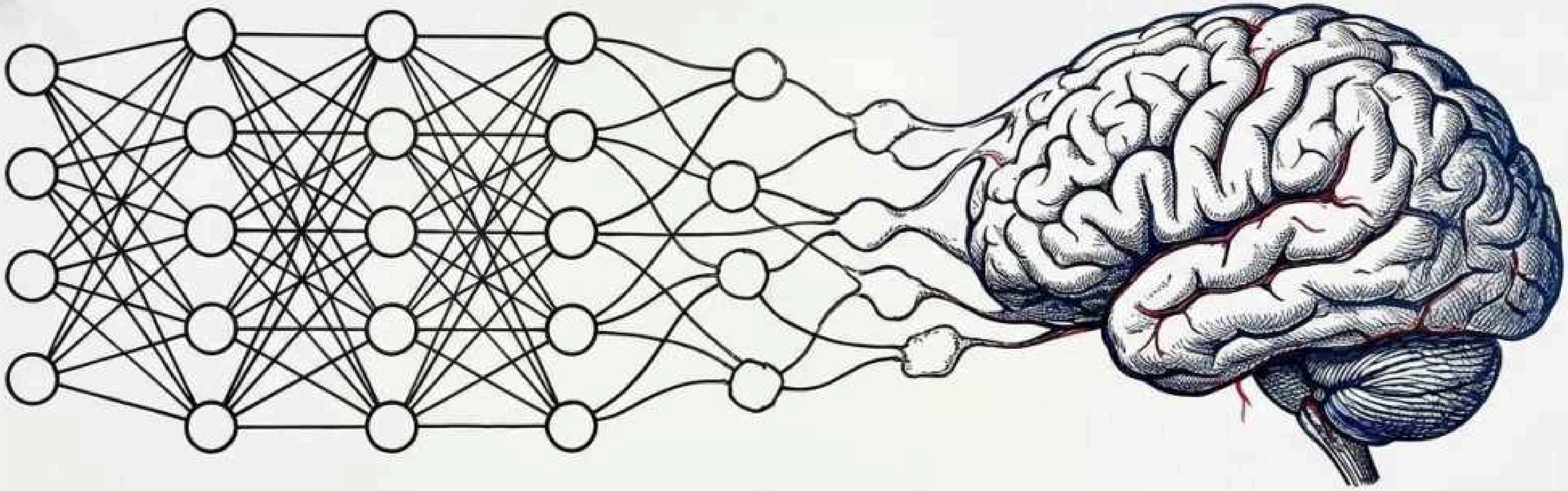
Storage: Never store passwords in plain form; use scrambled hashes.

Transit: Encryption protects messages in transit.

The Weakest Link: People. Phishing messages trick users into surrendering access.

Defense: Use unique passwords for each service, keep software updated, and click cautiously.





Machine Learning: Finding patterns from massive examples rather than writing rules by hand.

Large Language Models (LLMs): Learn to predict the next word to draft text and write code.

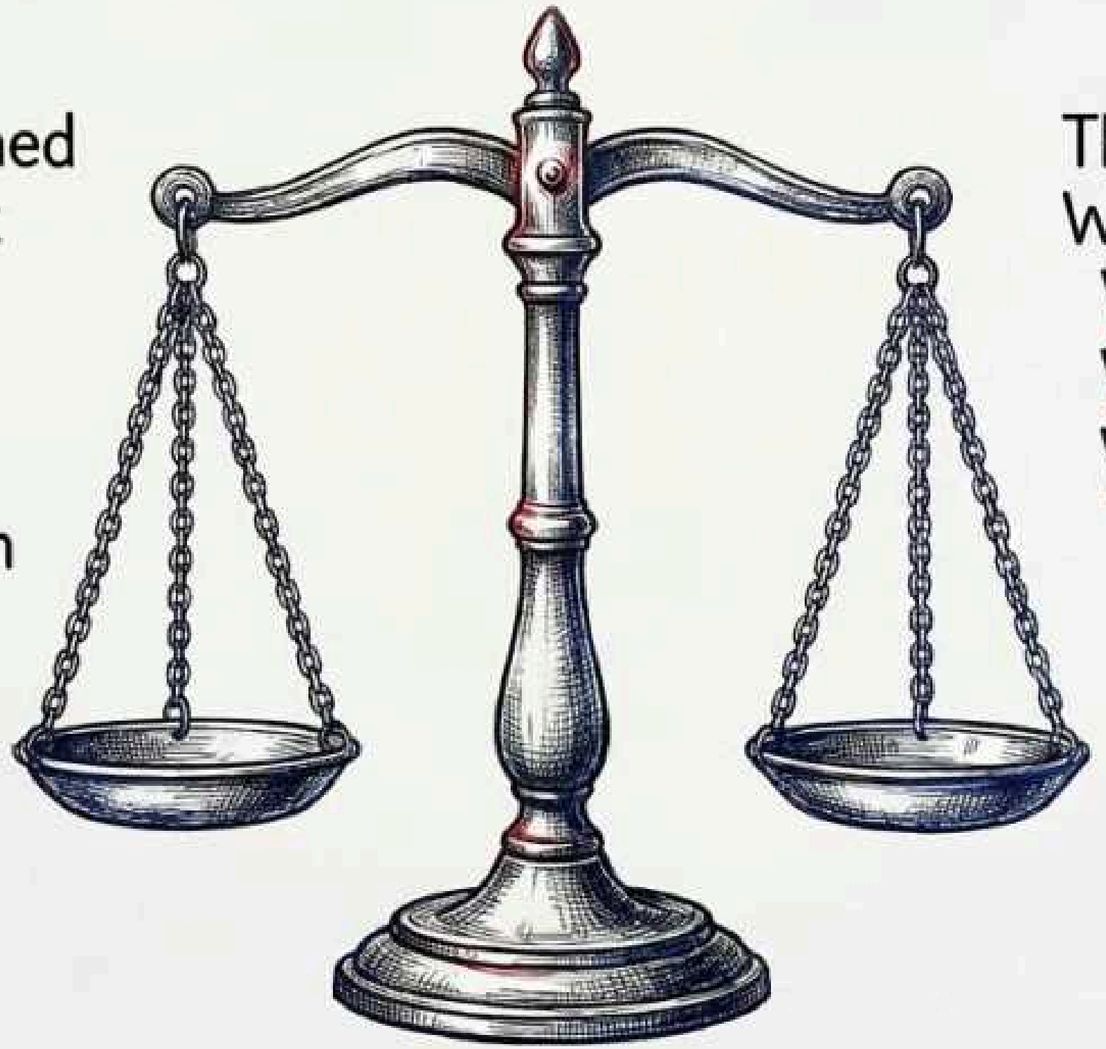
Warning: They make confident mistakes. A human expert must always check their work.

Note: This very course was built with AI assistance!

Software Shapes Lives

Bias: Programs trained on biased data treat people unfairly.

Privacy: Collecting personal information requires strict consent and protection.

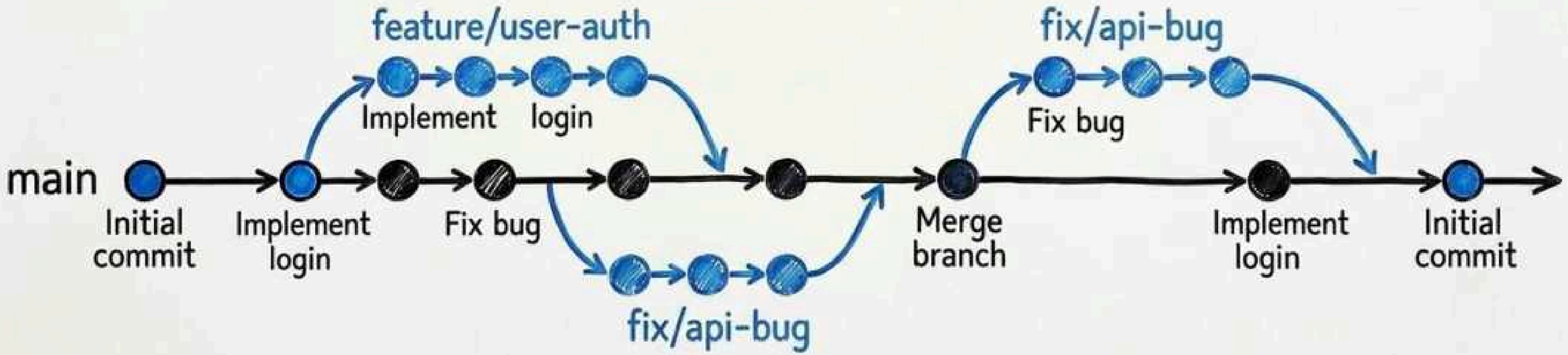


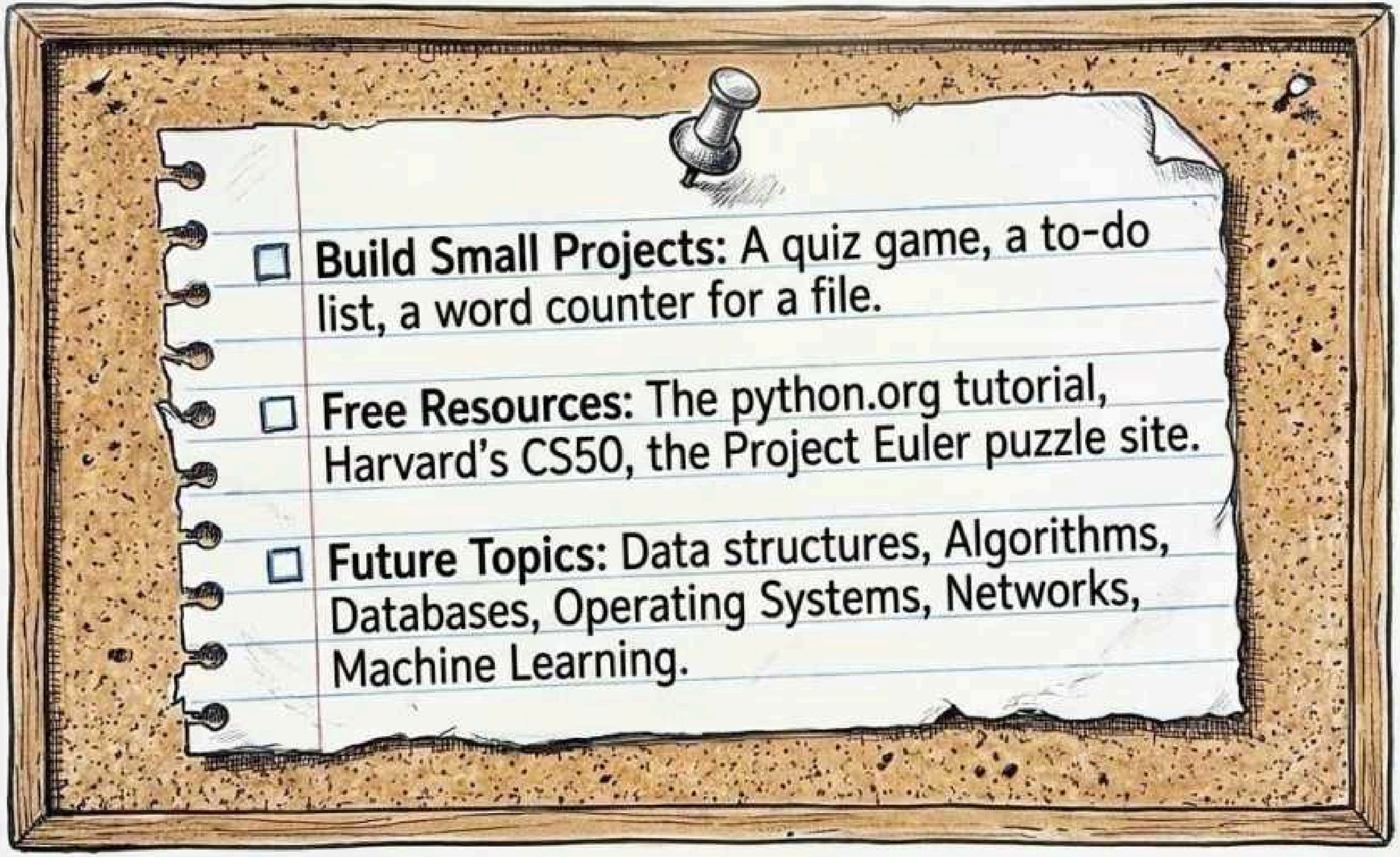
The Developer's Prompt: When building, ask:
Who is affected?
Who might be left out?
What could go wrong?

Accountability: The responsibility lies entirely with those who write, release, and use the software.

- Version Control (git): Records every change and enables teams to collaborate safely.
- Professional Habits: Write tests, document decisions, and actively review each other's code.

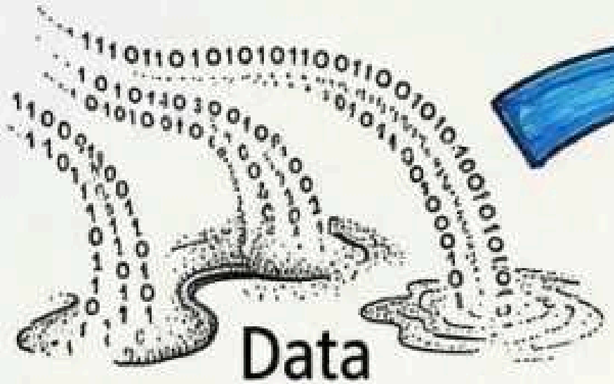
> The Golden Rule: Most real software is maintained for years. Readability and care are worth far more than cleverness.





The Routine: 30 minutes a day beats a long session once a week.

Course Summary



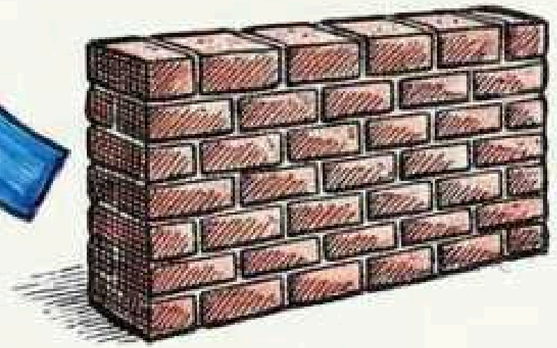
Data

```
for i in range(n):
    print(i)
```

Instructions

You understand what a computer is and how it stores information.

You can give instructions in Python via sequence, selection, repetition, and functions.



Limits

You know the bounds of efficiency, recursion, and computation limits.

The rest is practice.

Your Final Question: Think of one repetitive task in your life or work. Write the precise steps: Input -> Process -> Output. How would you test that it works?

Thank you for taking Computer Science 101 B.

Gary Anderson
garyanderson@mail.grokbot.com
X: @garyanderson

Anderson Academy