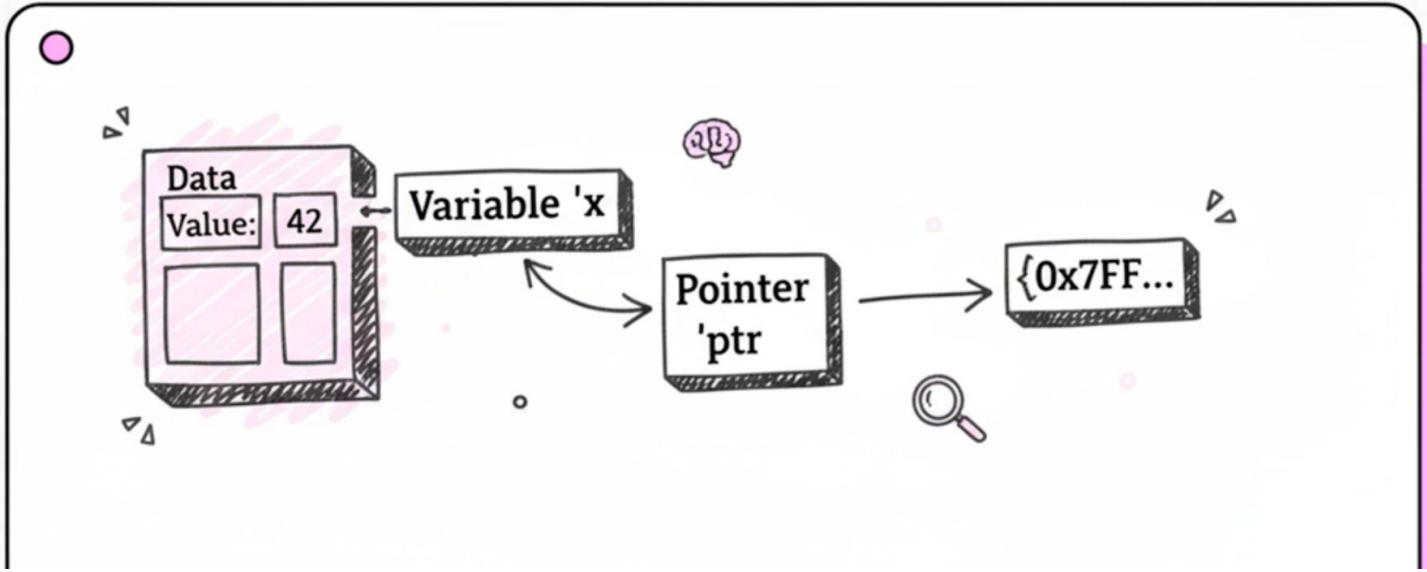
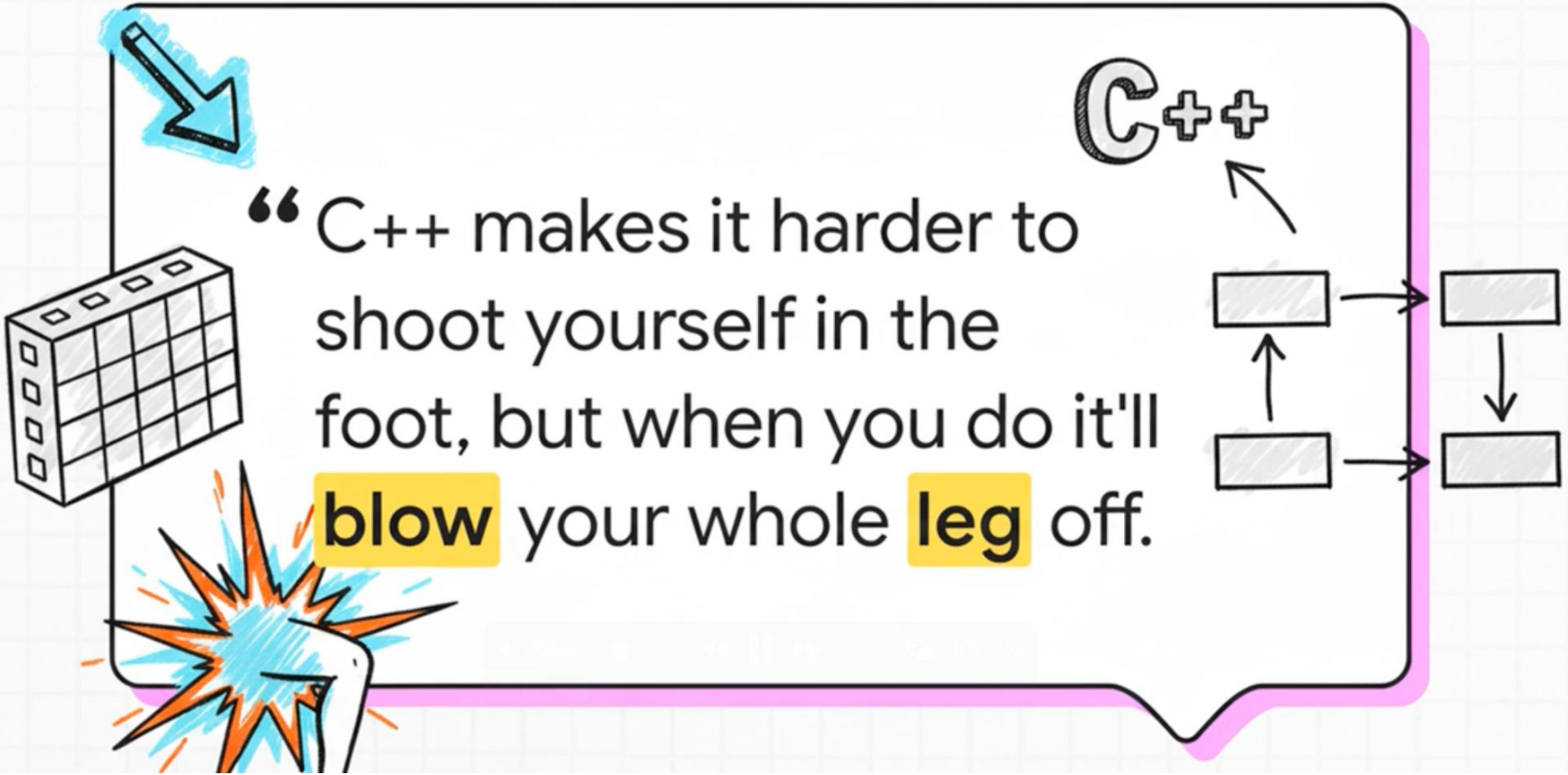
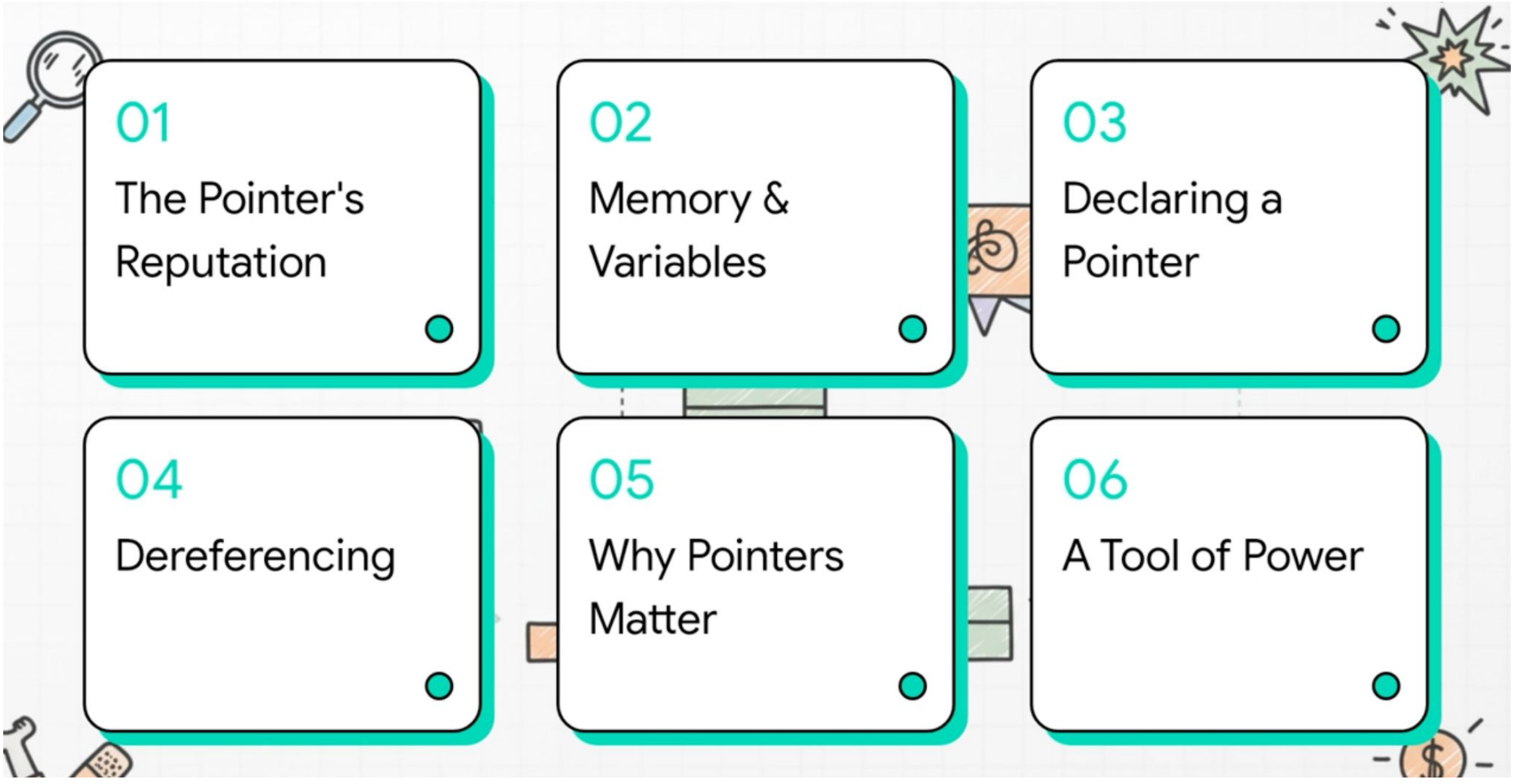
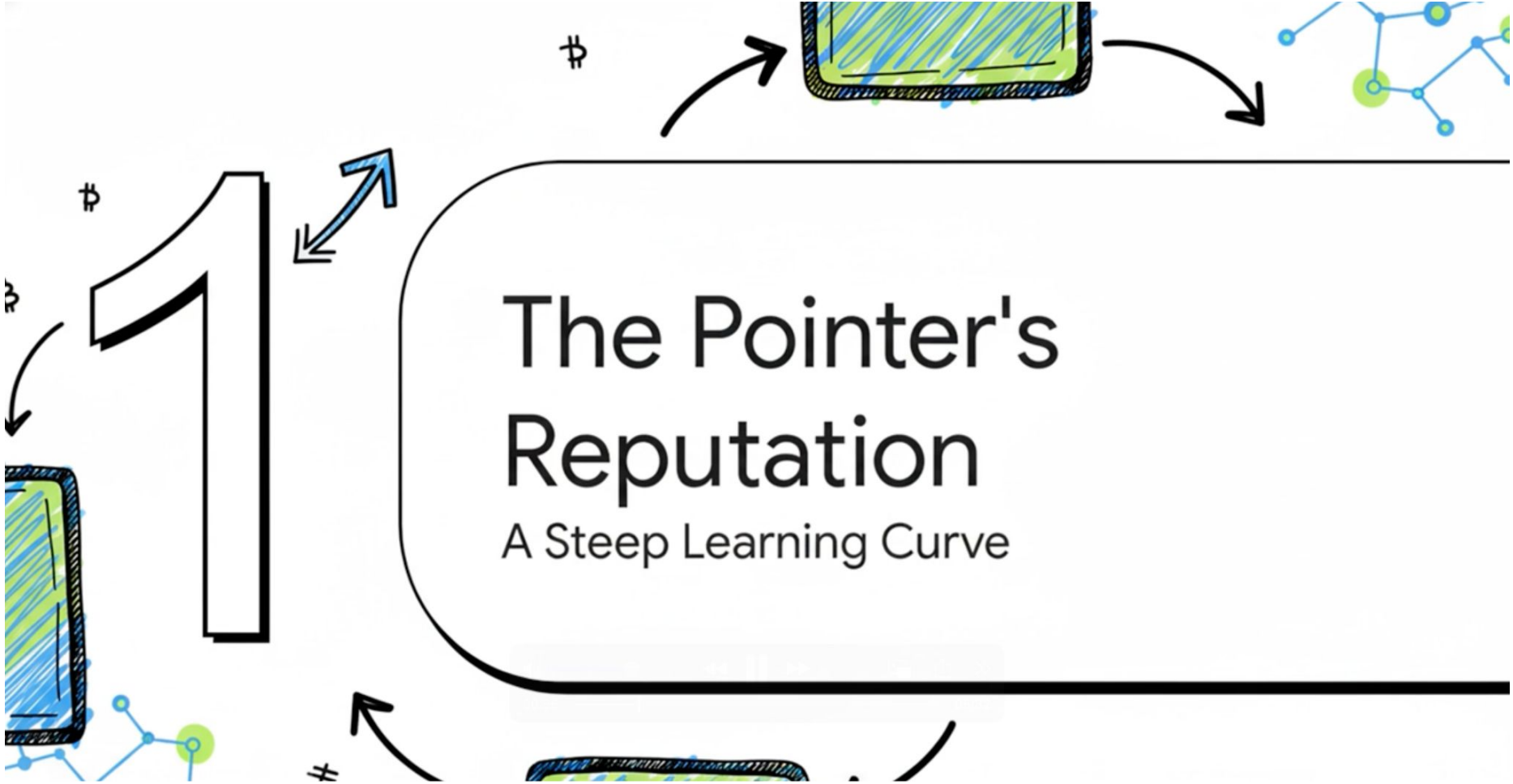
Week 9

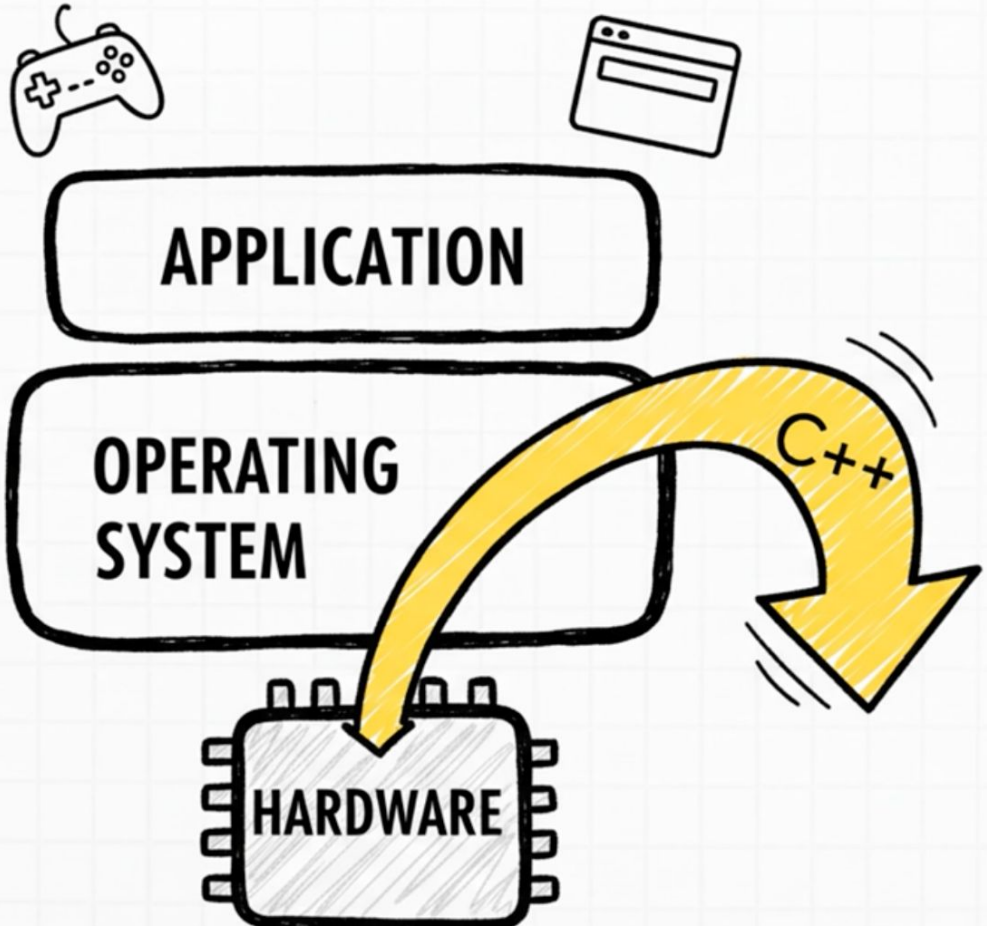
Demystifying C++ Pointers

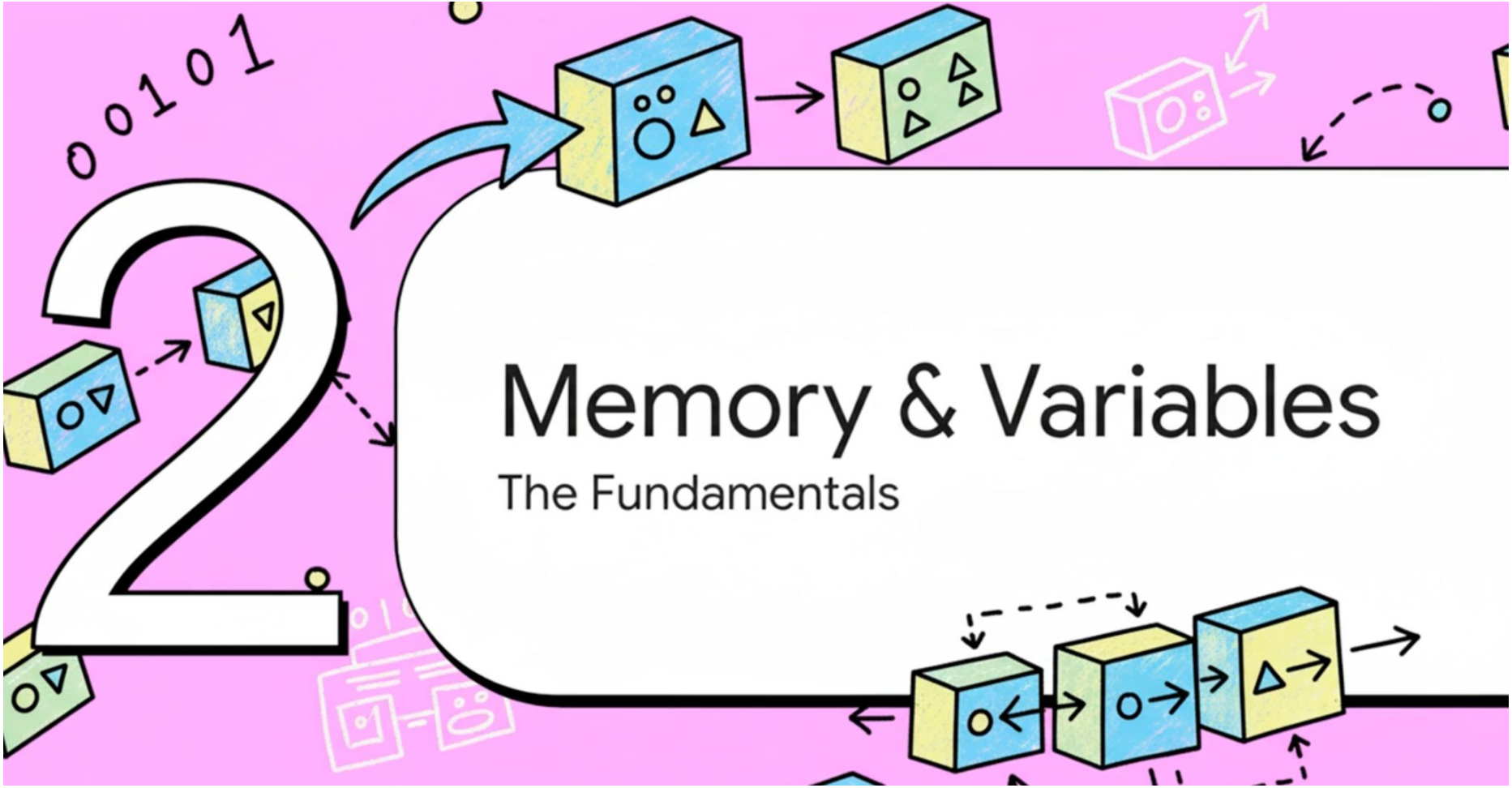


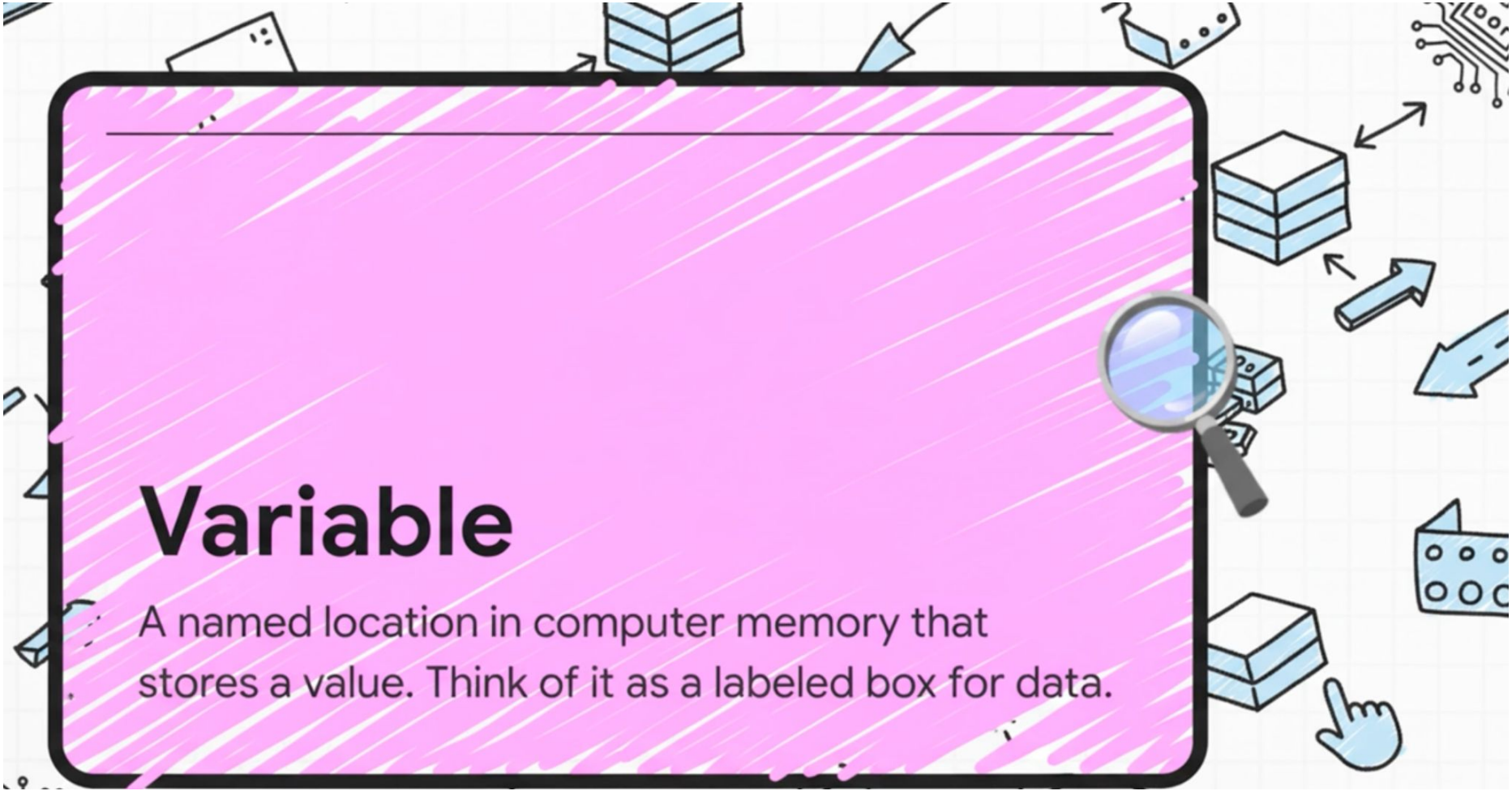

C++
“C++ makes it harder to shoot yourself in the foot, but when you do it'll blow your whole leg off.

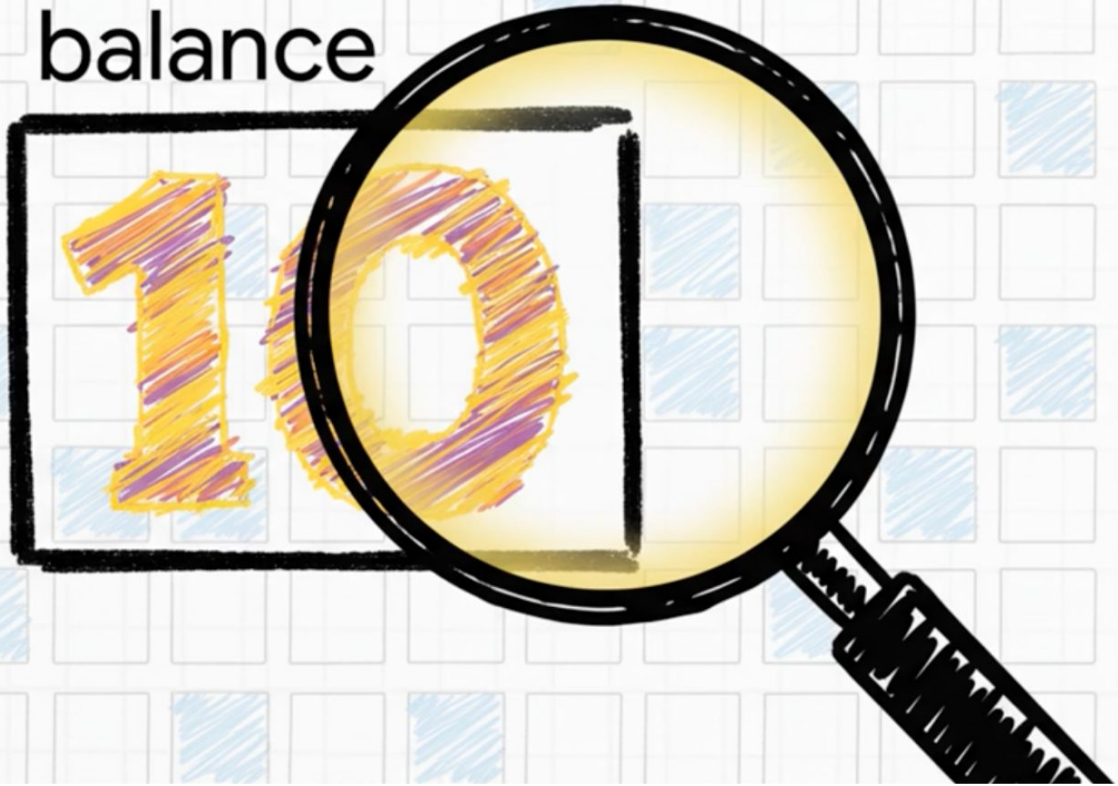

01
The Pointer's Reputation
02
Memory & Variables
03
Declaring a Pointer
04
Dereferencing
05
Why Pointers Matter
06
A Tool of Power

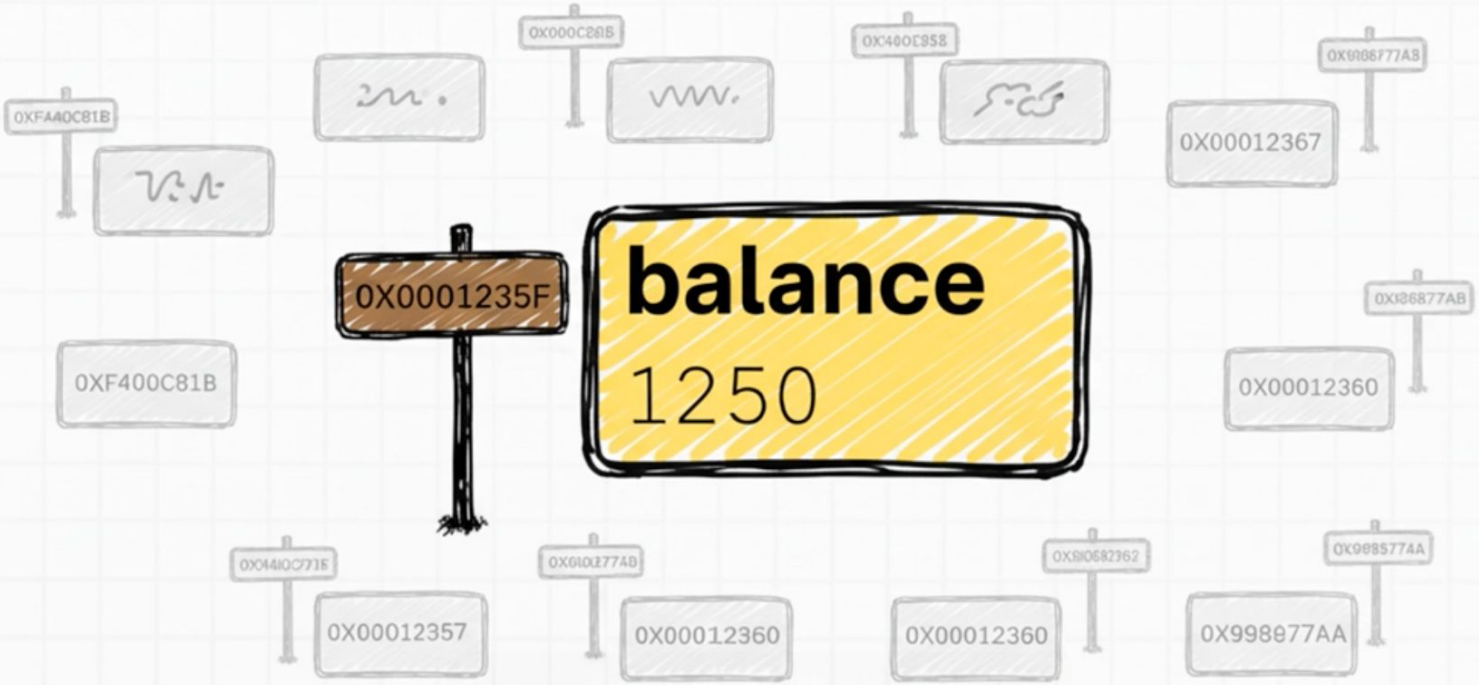

1
The Pointer's Reputation
A Steep Learning Curve

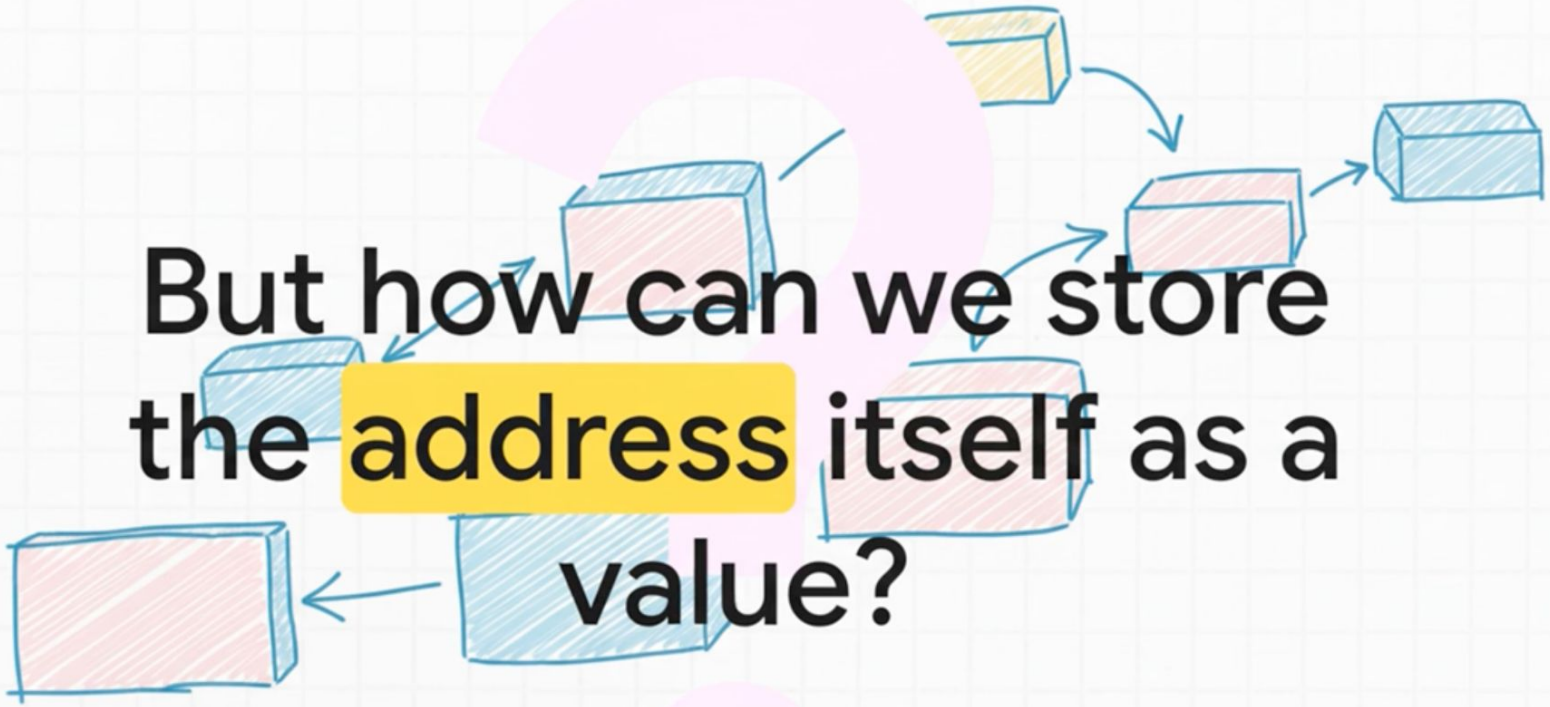

APPLICATION
OPERATING SYSTEM
C++
HARDWARE


00101
2
Memory & Variables
The Fundamentals


Variable
A named location in computer memory that stores a value. Think of it as a labeled box for data.


balance
10


0X0001235F
balance
1250
0XF400C81B
0X00012367
0X00012360
0X00012357
0X00012360
0X00012360
0X998077AA


But how can we store
the address itself as a
value?

1. Declaring a Pointer Variable

To declare a pointer, you use the data type followed by the asterisk (`*`) and the pointer name [2].

General Syntax:

```
datatype * Name; [2]
```

Example Declarations:

- `Int *P;` [2]
- `int* pBal;` (for an integer pointer) [3]
- `double* pTest;` (for a double pointer) [3]

2. Obtaining the Address of a Variable

To get the memory address (reference) of a regular variable, you use the **address-of operator** (`&`) before the variable name [2]

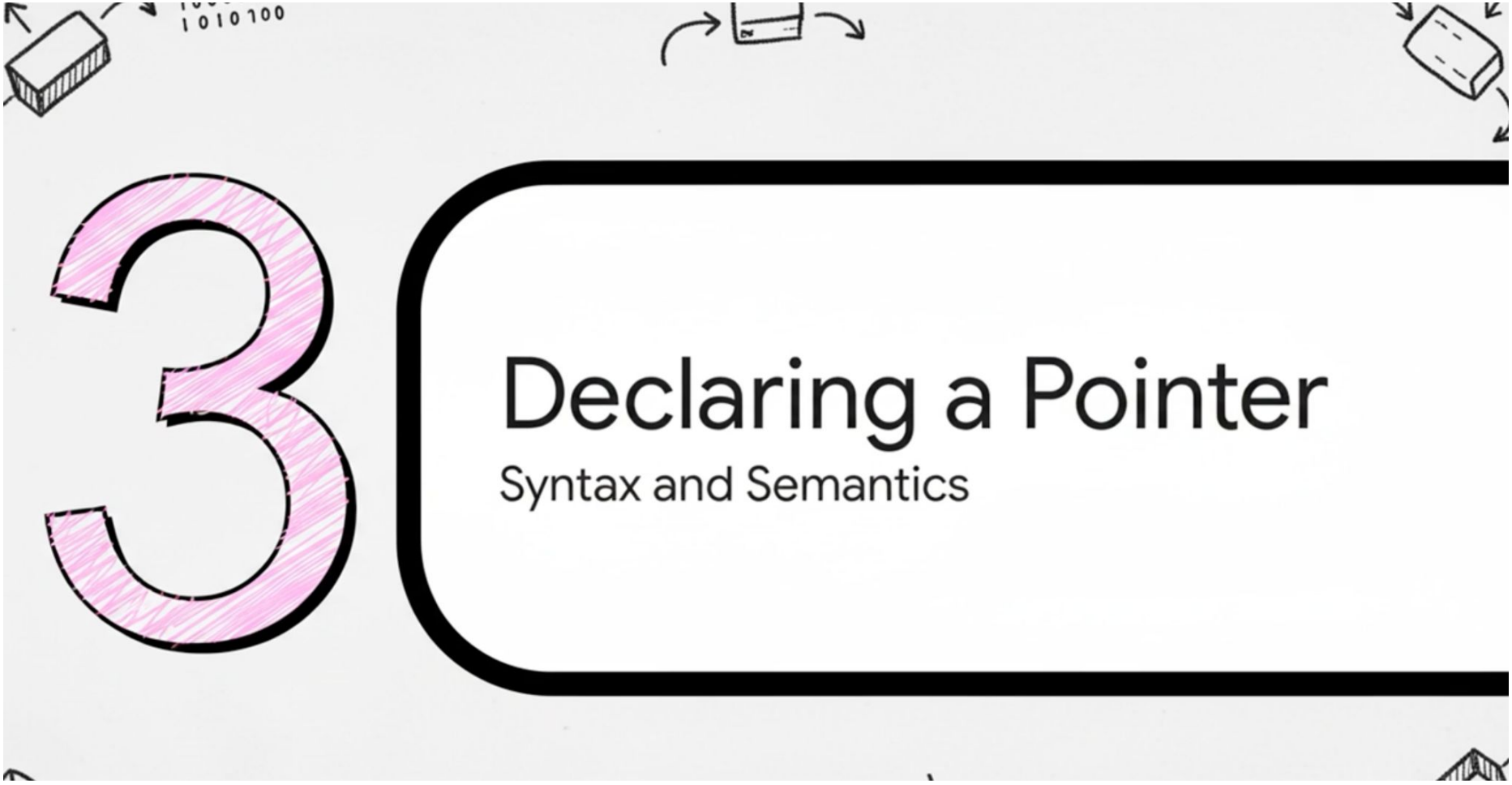
Example Declarations:

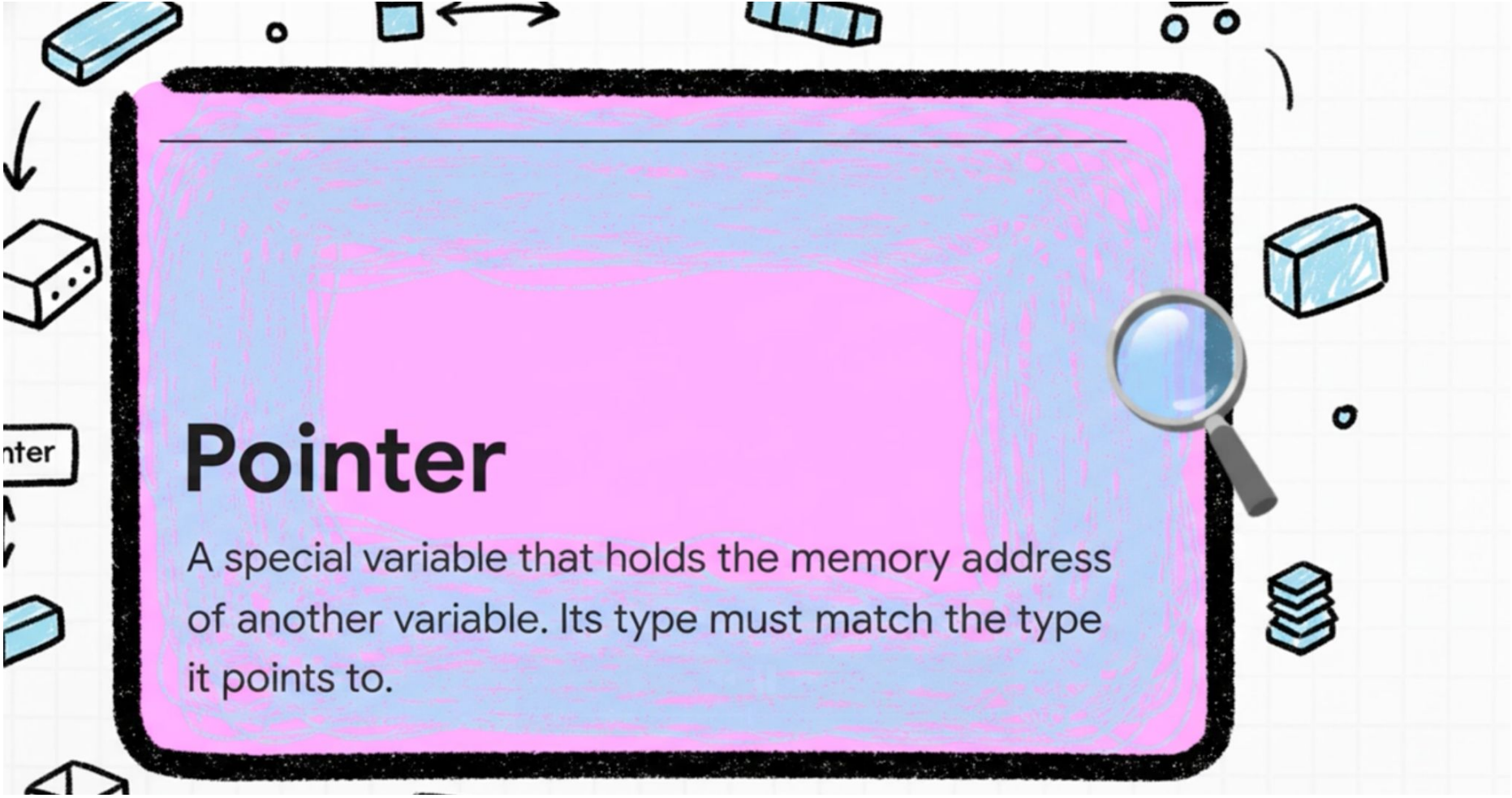
- `Int *P;` 2
- `int* pBal;` (for an integer pointer) 3
- `double* pTest;` (for a double pointer) 3

2. Obtaining the Address of a Variable

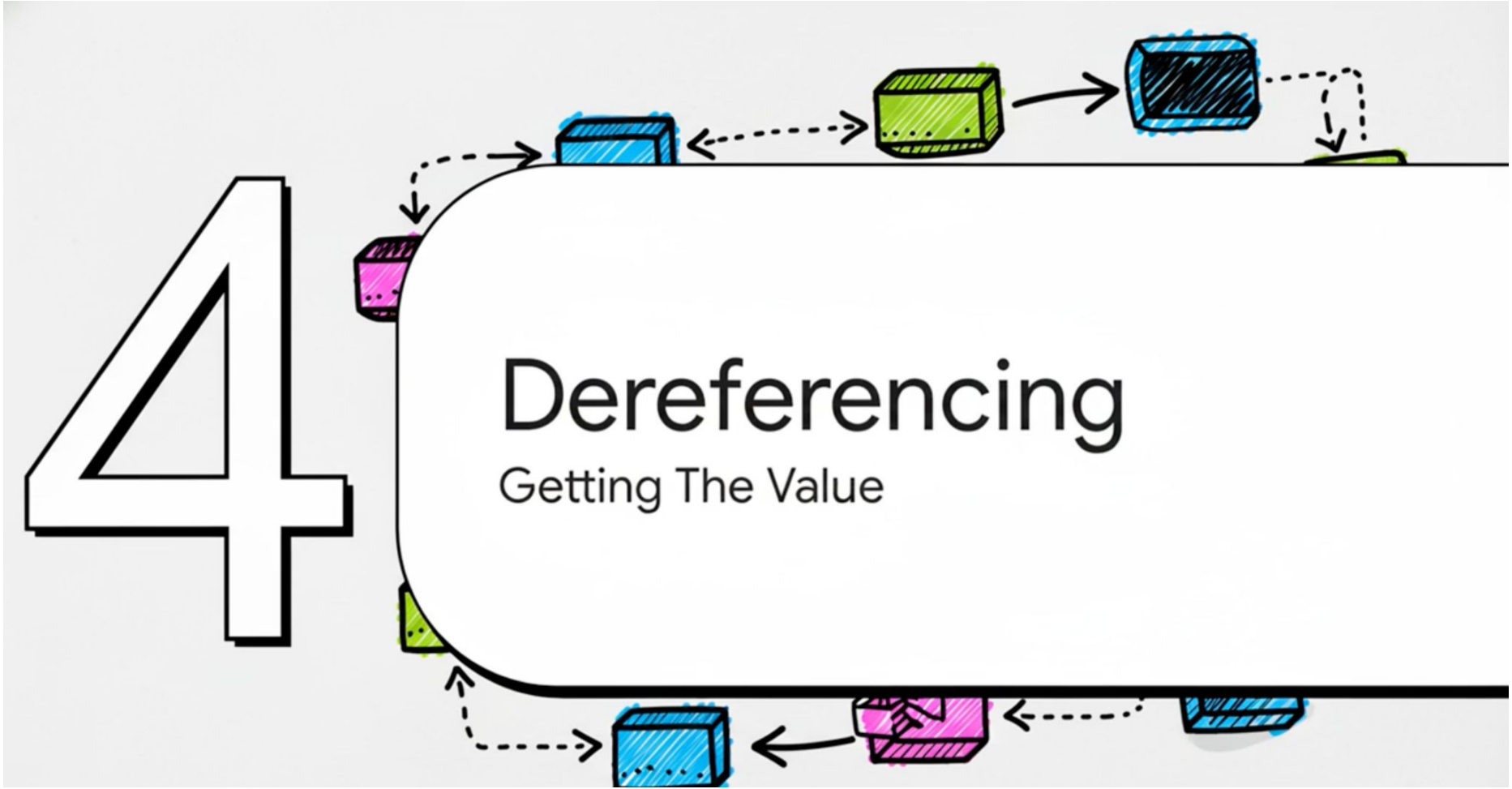
To get the memory address (reference) of a regular variable, you use the **address-of operator** (`&`) before the variable name 2

.

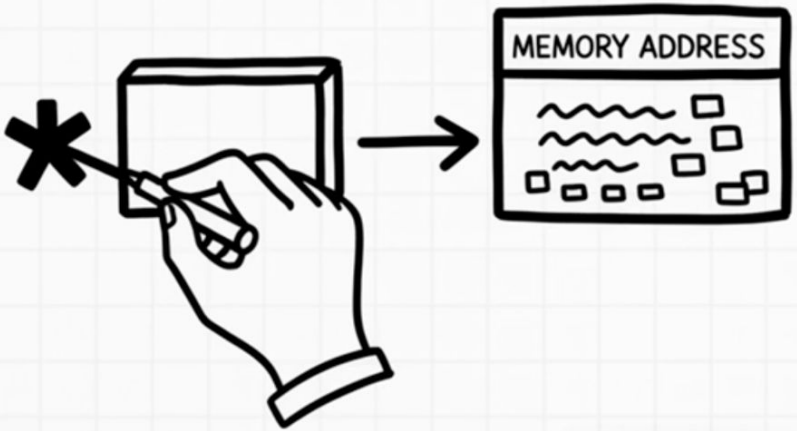

3
Declaring a Pointer
Syntax and Semantics


Pointer
A special variable that holds the memory address of another variable. Its type must match the type it points to.

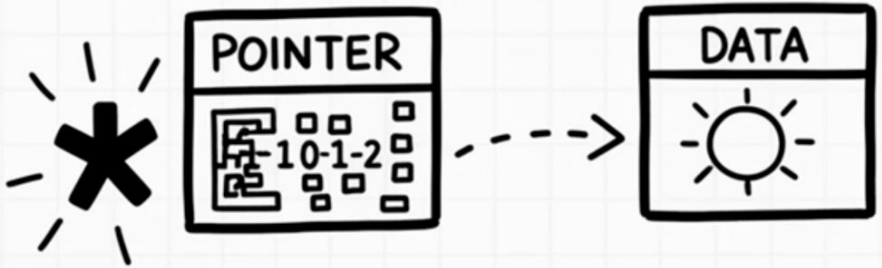
Pointer Declaration


4
Dereferencing
Getting The Value

DECLARATION



DEREFERENCING



General Syntax:

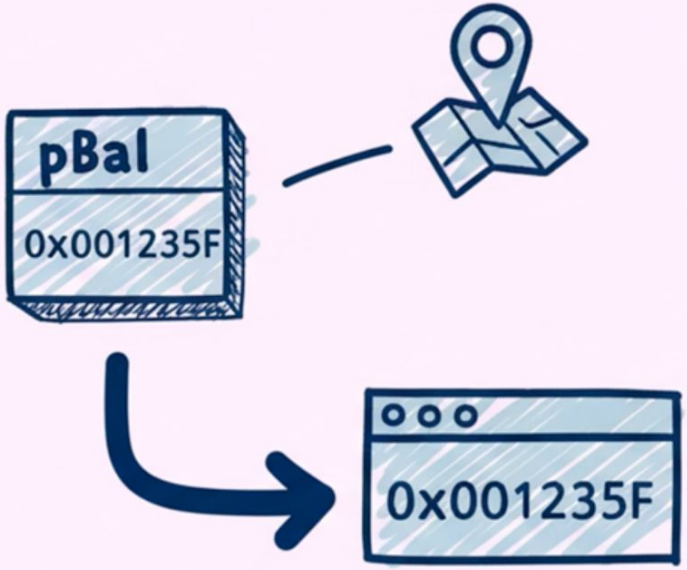
```
&variable_name [2]
```

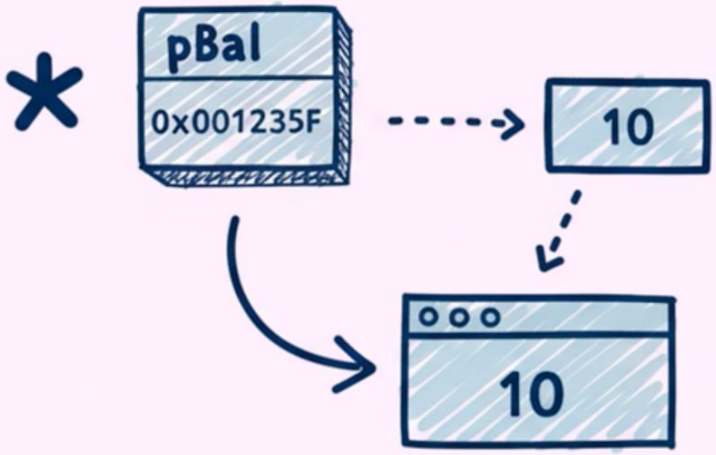
3. Initializing and Assigning Pointers

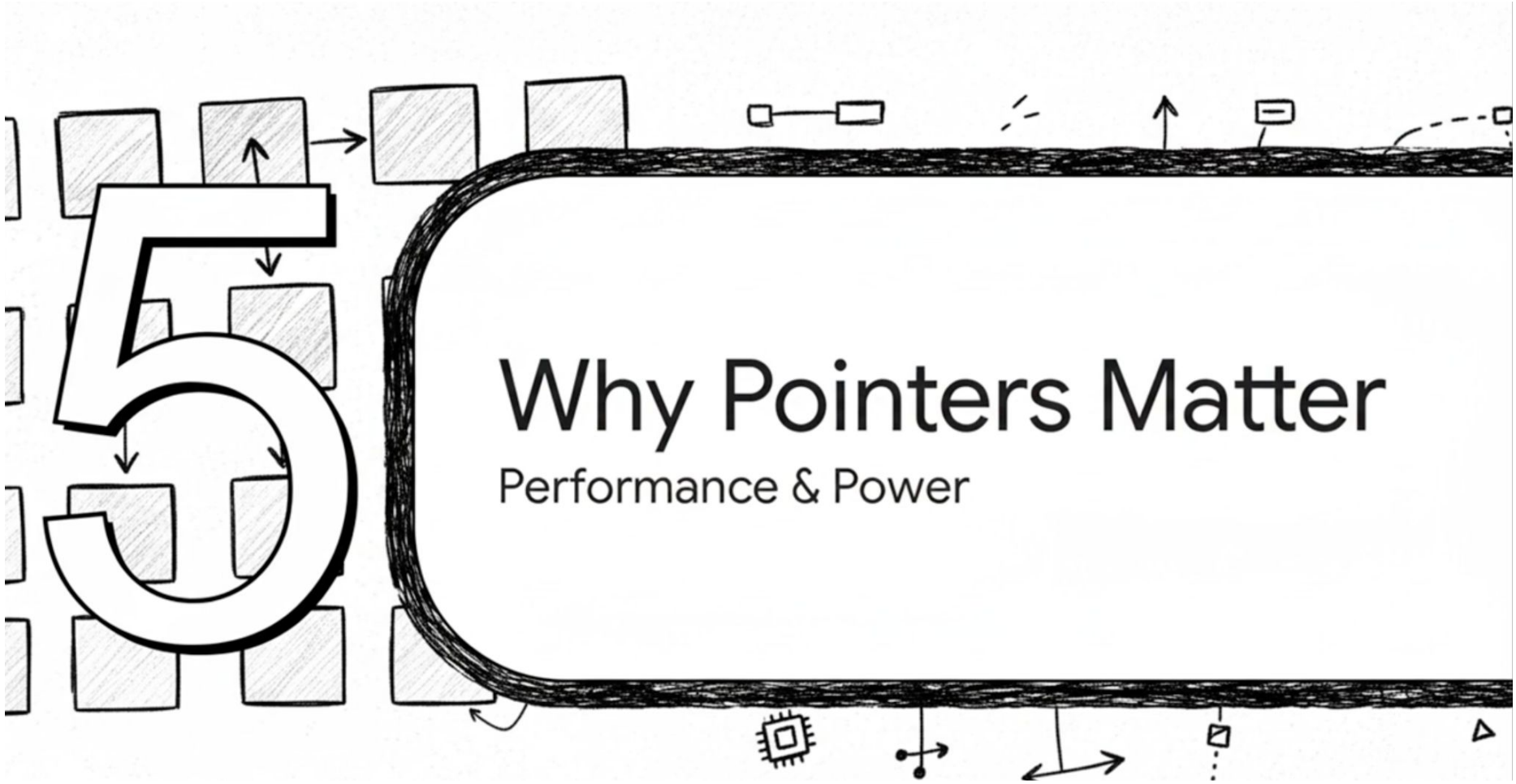
You assign the address of a variable to a pointer variable 2 .

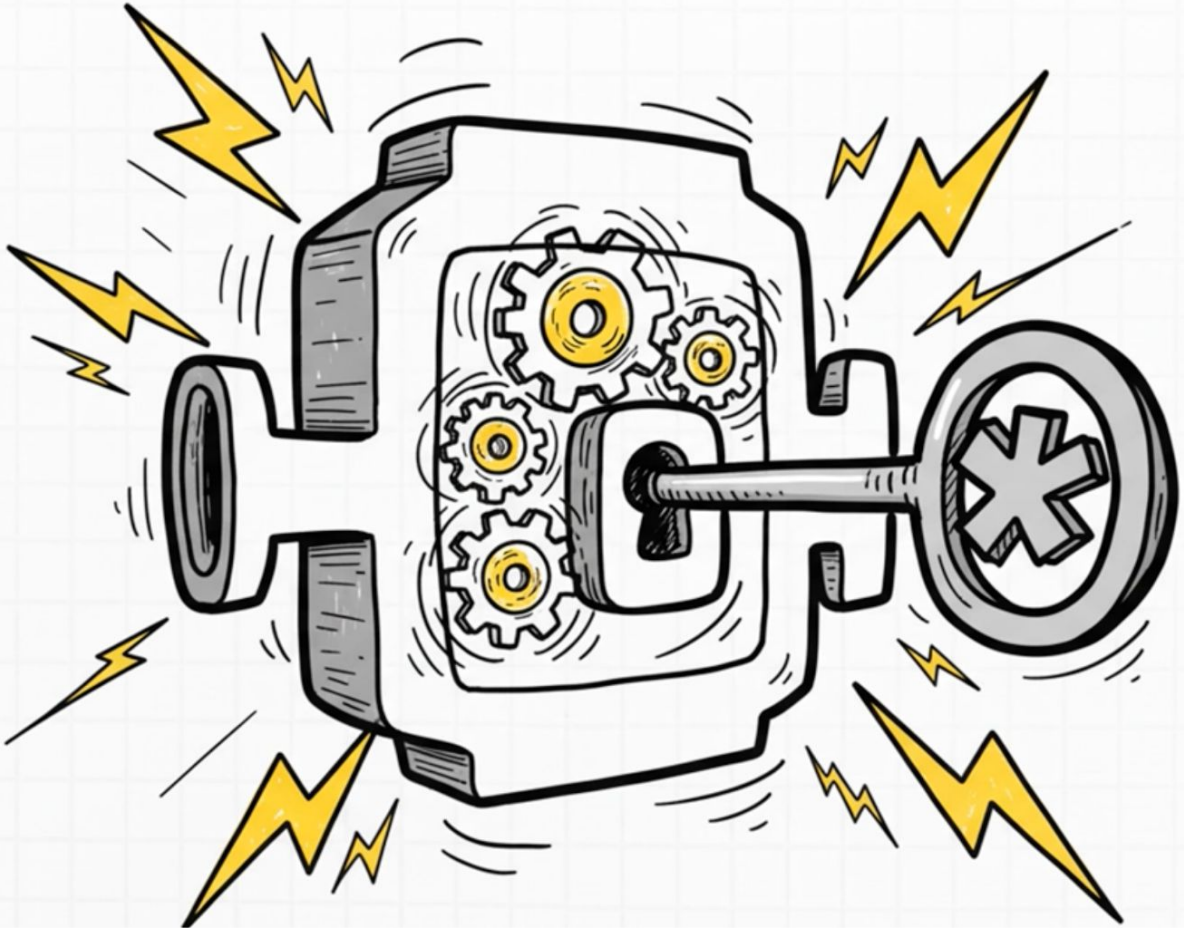
Example Initialization:

```
Int balance;
Int *pBal = &balance; [2]
```

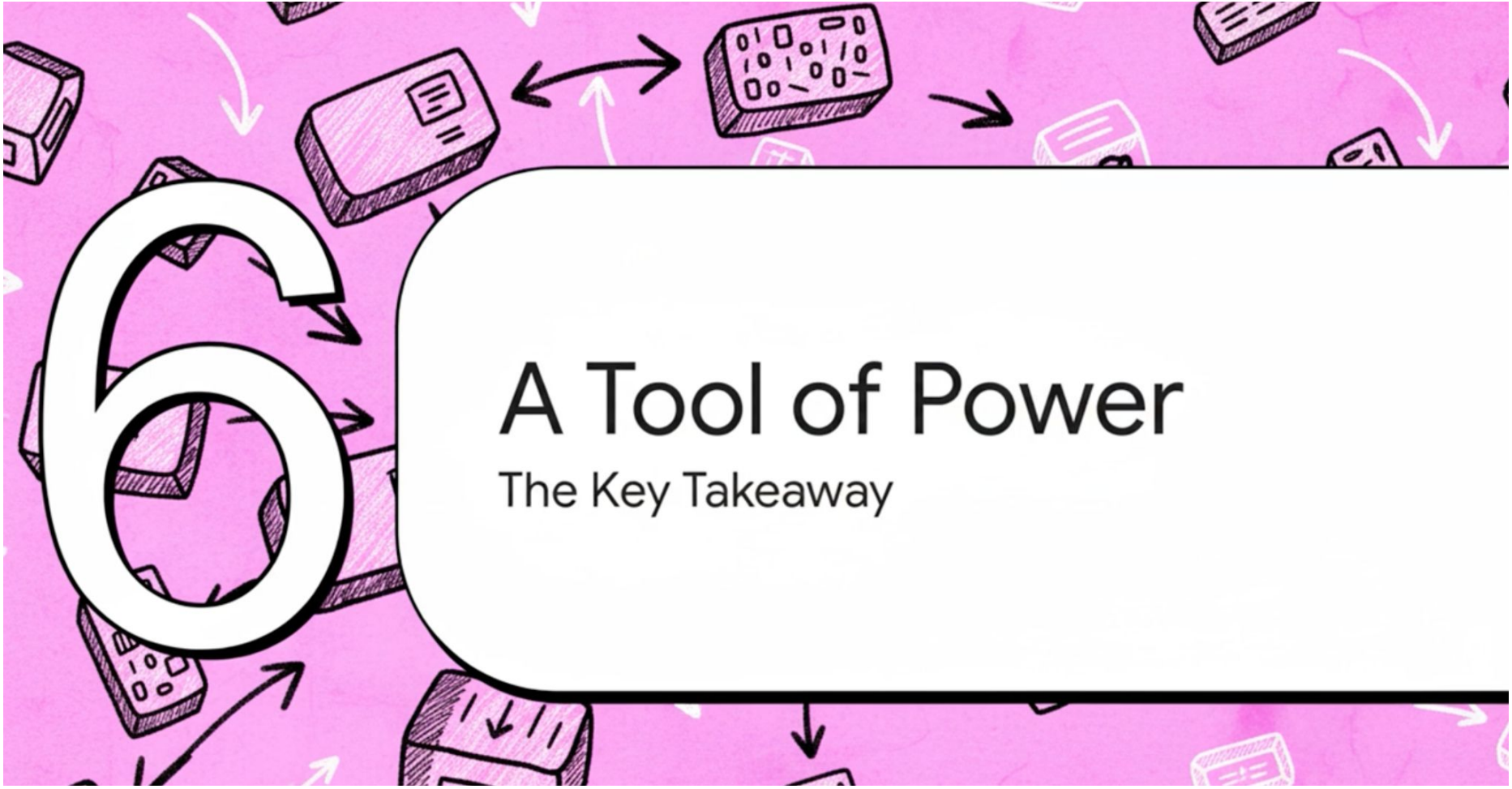

pBal
0x001235F
0x001235F


*
pBal
0x001235F
10
10


5
Why Pointers Matter
Performance & Power

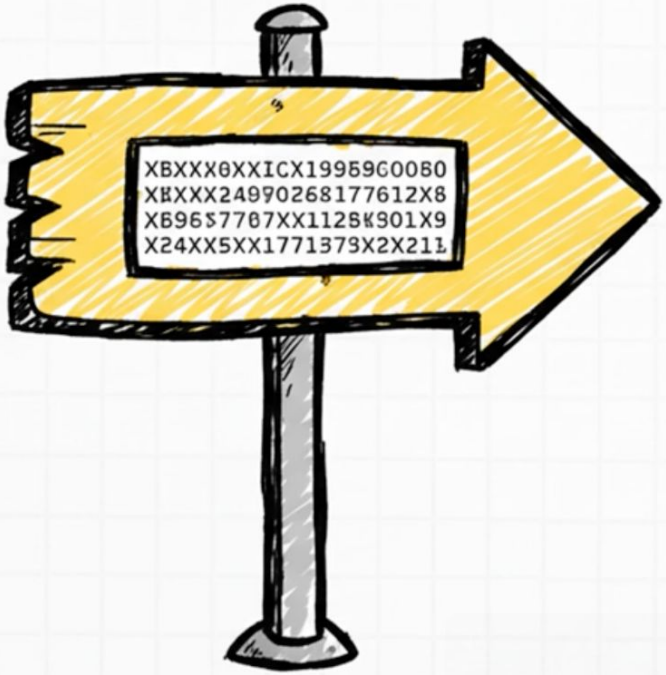


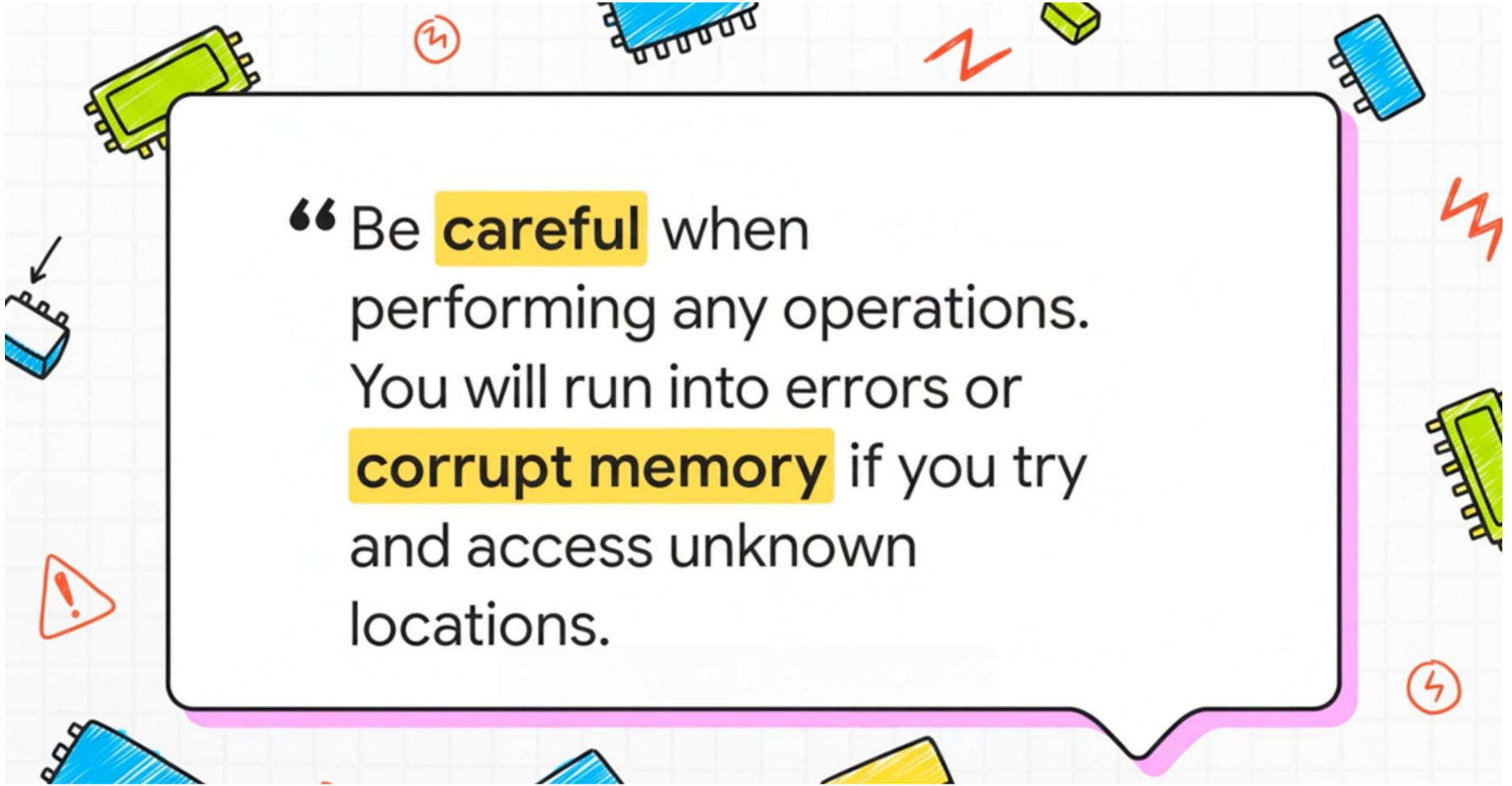
Key Pointer Applications

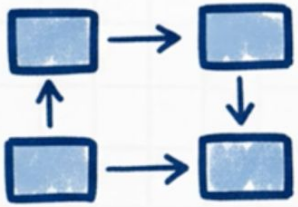


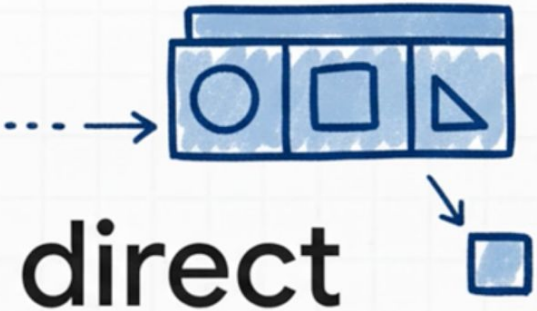
A Tool of Power

The Key Takeaway

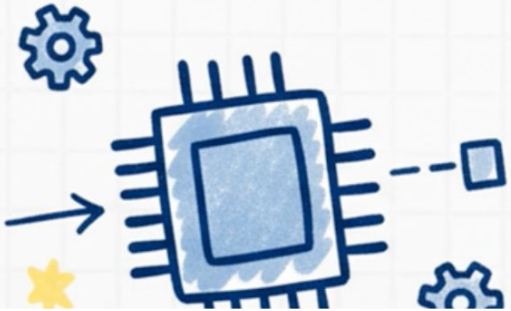

XBXXX0XXICX19959C0080
XHXXX24970268177612X8
XB9657707XX1126K901X9
X24XX5XX1771379X2X211

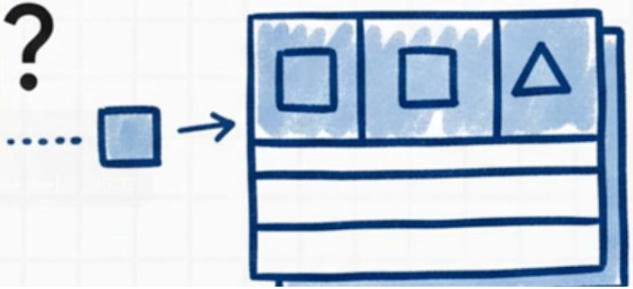

“ Be careful when performing any operations. You will run into errors or corrupt memory if you try and access unknown locations.





Pointers give you direct control over memory. How will you use that power?





Lab Assignment: MTA Train Stops

```
list
list mta_stops {

    Grand Central
    Times Sq-42 St,
    Times Sq- St,
    Penn Station
    W 4 St-Wash Sq,
    W 4 St-Wash Sq,

for {
    print(stops'stop))
}
```

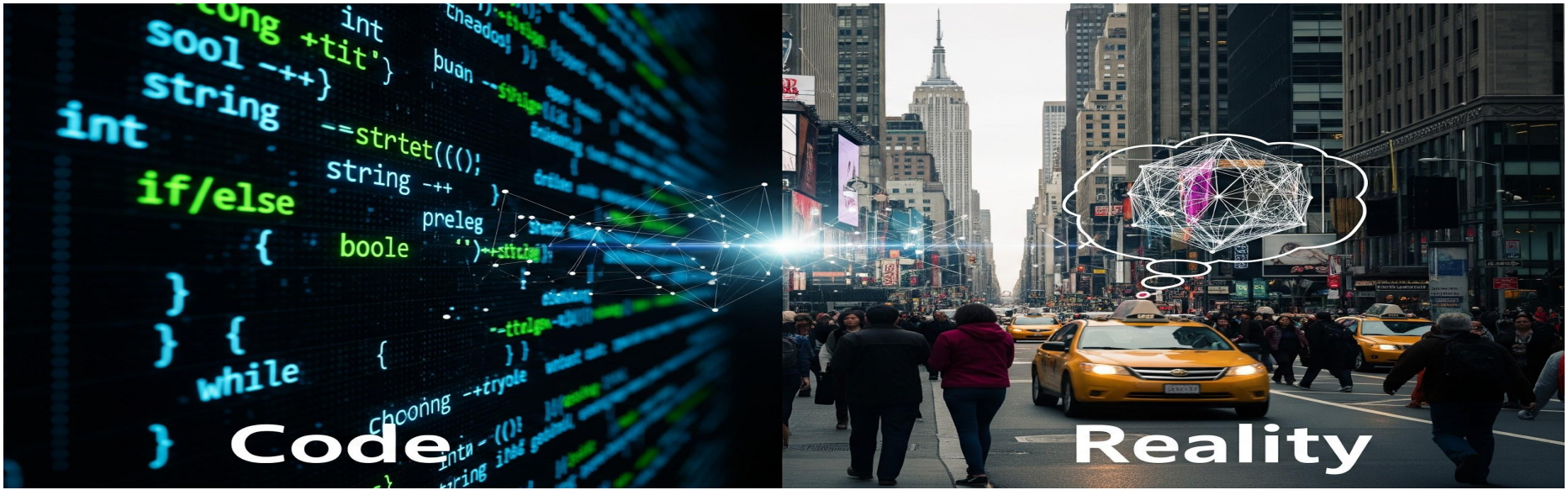
Create an array (list) of train stops frour favorite MTA line and through each stop in a “for” loop

Research Project: CS 101: The NYC Tech Design Challenge

The Big Idea

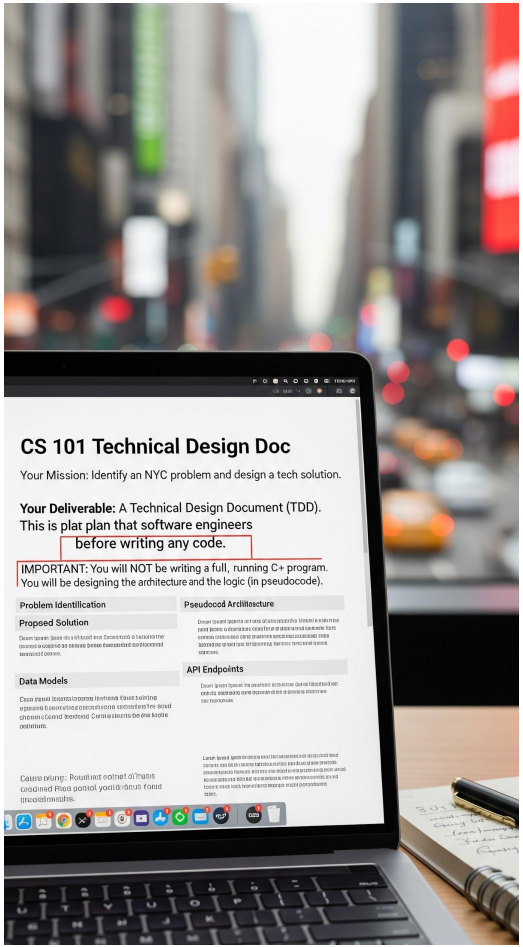
From Code to Reality

- **You're learning:** `int`, `string`, `bool`, `if/else`, `while`...
- **The Goal:** Programmers don't just write code. They solve problems.
- **Our Challenge:** How can we use the *logic* of C++ to solve a real problem in NYC?



The "CS 101 Technical Design Doc"

- **Your Mission:** Identify an NYC problem and design a tech solution.
- **Your Deliverable:** A **Technical Design Document (TDD)**. This is a *plan* that software engineers create before writing any code.
- **IMPORTANT:** You will **NOT** be writing a full, running C++ program. You will be designing the *architecture* and the *logic* (in pseudocode).



The TDD (Part 1): The Problem

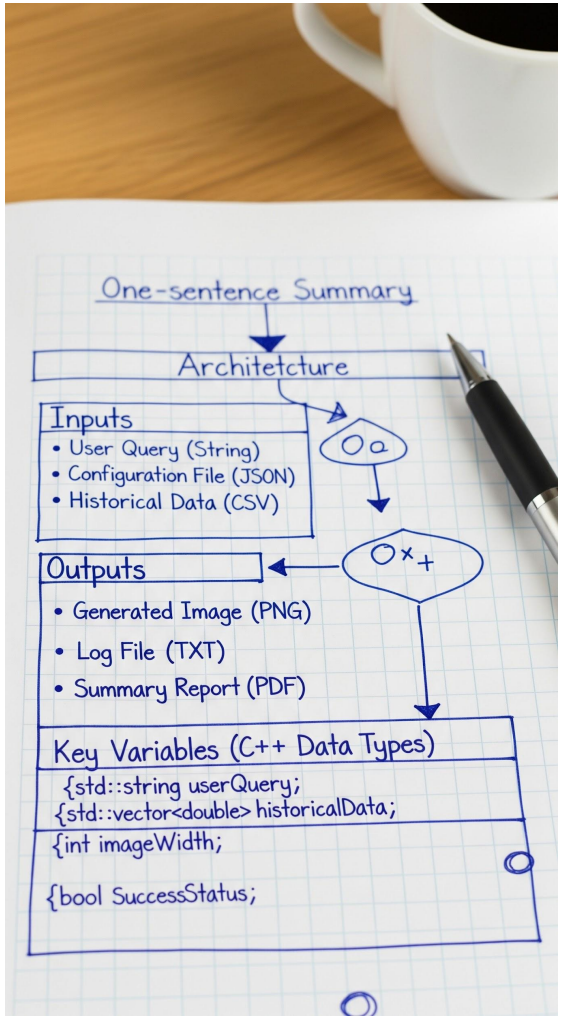
Step 1: Define the Problem

- **Project Title:** Give it a catchy name.
- **Problem Statement:** What is the *specific* problem?
  - *Bad Example:* "Trash is bad."
  - *Good Example:* "The L train platform at Bedford Ave is dangerously overcrowded between 8:00 AM and 9:30 AM."
- **Who It Affects:** Who experiences this?
  - e.g., Commuters, small business owners, sanitation workers.

The TDD (Part 2): The Solution

Step 2: Describe Your Solution (High-Level)

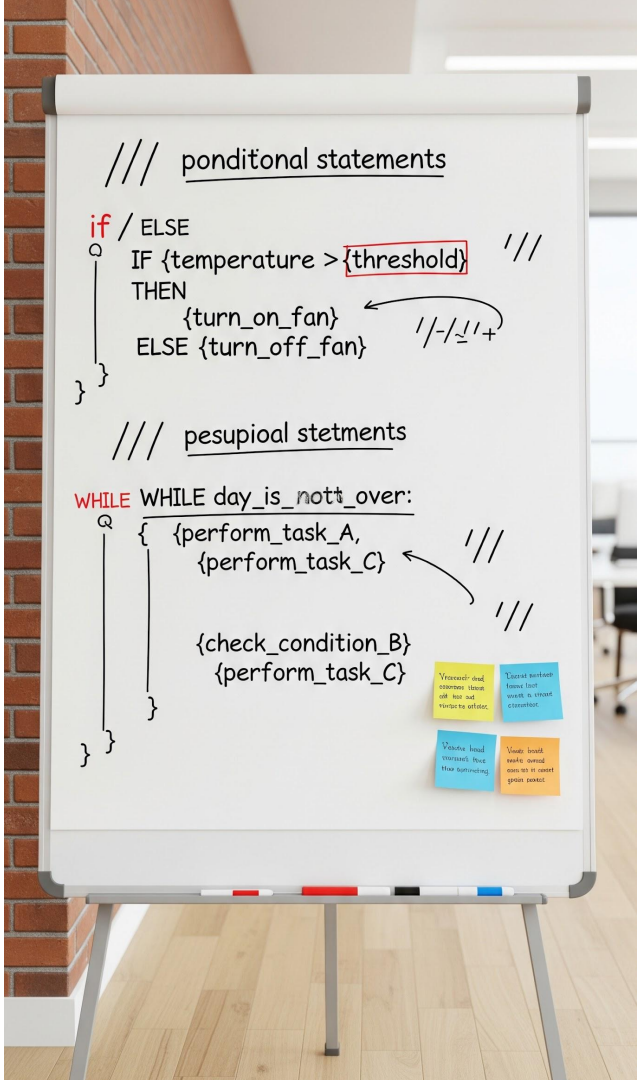
- **Solution Summary:** A one-sentence pitch.
  - e.g., "A smart sensor system that tells commuters the *actual* crowd level before they enter the station."
- **Simple "Architecture":**
  - **INPUTS:** What information does your program need?
    - (e.g., Camera data, time of day, user reports)
  - **OUTPUTS:** What does your program *do*?
    - (e.g., Display a message, send a command to a motor)
  - **DATA:** What C++ variables would you need?
    - (e.g., `int passengerCount;`, `bool isCrowded;`)



The TDD (Part 3): The Core Logic

Step 3: Write the Pseudocode

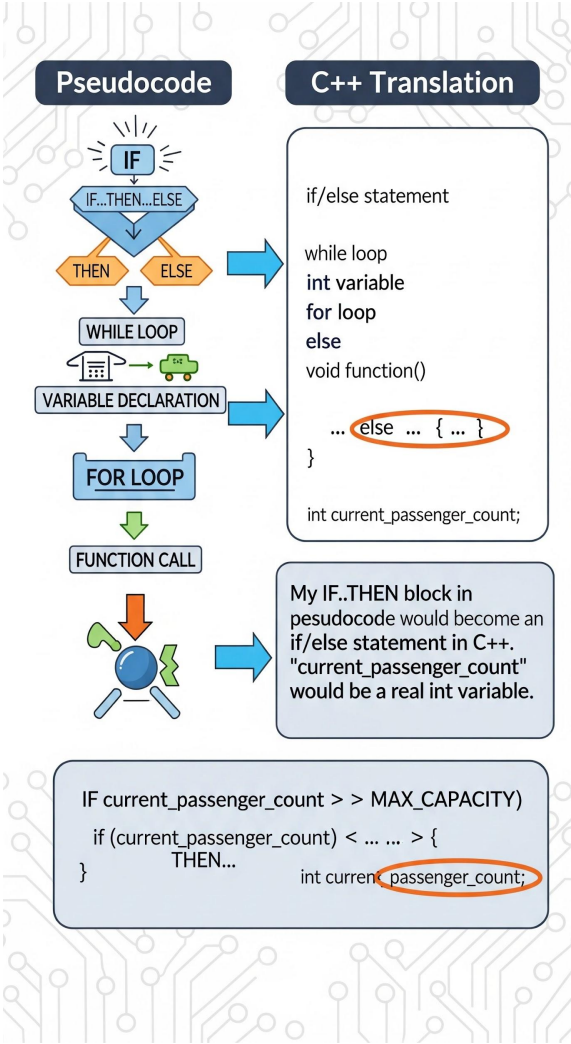
- This is the **heart** of the assignment.
- **Pseudocode** = Structured English. It's logic, without the perfect C++ syntax.
- Focus on the *decisions* your program has to make.
- Use `IF / ELSE IF / ELSE`.
- Use `WHILE` or `FOR` loops.
- Get your `INPUTS` and send your `OUTPUTS`.



The TDD (Part 4): The Connection

Step 4: The C++ Connection

- Finally, explain how your pseudocode maps to C++.
- **Example:**
  - "My `WHILE (robot_is_on) DO`... would be a `while(true)` loop in C++."
  - "My `IF (battery < 10)`... would be an `if` statement checking a `float` variable."
  - "The `SEND command_to_motor`... would be a C++ function call."



Logistics & Grading

Assignment: Fill out the `CS101_TDD_Template.md` & Slide Deck Pitch file with Supplementary files from the project.

Submission: Submit the completed file on GitHub.

Due Date: Nov 25th End of Class

Grading:

- Problem Clarity (25%)
- Solution Creativity (25%)
- **Pseudocode Logic (30%)**
- C++ Connection (20%)

Attendance