**What is encapsulation and why is it important?**

Encapsulation is like driving a car: you interact with simple controls (like pedals) while the complex engine parts are hidden and protected. In programming, it means grouping related data (variables) and actions (functions) together inside classes, while controlling how they’re accessed—just like a car’s engine is shielded under the hood.

**Key benefits:**

* Security: Prevents accidental data corruption (e.g., no direct access to critical values).
* Organization: Keeps code clean and logical.
* Teamwork: Makes collaboration easier in large projects.

**Real-world uses:**

* Banking apps: Protect account balances and transaction rules.
* Smart thermostats: Hide sensor complexity behind simple temperature controls.
* Video games: Safeguard character stats and physics engines.

**In my water flow program**, I applied encapsulation by:

1. Protecting constants: Like WATER\_DENSITY (no direct changes allowed).
2. Hiding calculations: Pressure formulas are locked inside functions (e.g., pressure\_loss\_from\_pipe())

# Encapsulation: Constants and functions control access to data

WATER\_DENSITY = 998.2 # Hidden from direct changes

def pressure\_loss\_from\_pipe(pipe\_diameter, pipe\_length, friction\_factor, fluid\_velocity):

# Calculation encapsulated here; external code can't alter the formula

return (-friction\_factor \* pipe\_length \* WATER\_DENSITY \* fluid\_velocity\*\*2) / (2000 \* pipe\_diameter)