System-First Learning Template (Printable)

Use this for **any topic** (Errors, File I/O, OOP, APIs, Databases, AI, etc.).

Topic

Name:

Layer 1 — WHY (System Purpose)

Question: Why does this concept exist? What real-world problem forced it to exist?

-

One-sentence summary:

Layer 2 — WHERE (System Placement)

Question: Where does this live inside a system? Choose one or more: - ☐ Control (decision-making) - ☐ State (data that persists) - ☐ Flow (movement of data) - ☐ Boundary (interaction with outside world) - ☐ Responsibility (ownership)

Explain in 1–2 lines:

Layer 3 — WHAT CAN GO WRONG? (Failure-First Thinking)

Question: List the most likely failures and risks. 1. 2. 3.

What should happen when failure occurs?

Layer 4 — HOW (Minimal Syntax & Mechanism)

Question: How does Python (or the tool) express this concept? Only essentials — no memorization. - Key functions / keywords: - Typical pattern:

Small example (optional):

```

```

Layer 5 — WHO (Ownership & Responsibility)

Question: Who should handle this and who should NOT? - Owner: - Not responsible: - Why:

Design Notes (Optional but Powerful)

- Tradeoffs:
- Simplest acceptable design:
- What I would change later:

One-Line Mental Model

(Example: “File I/O is the persistence boundary of a system.”)

END

Reusable Prompt (Copy–Paste)

Use this prompt with any topic name.

```
You are an expert system-design-first teacher.

Topic: <INSERT TOPIC HERE>

Teach this topic using the **5-Layer System-First Learning Framework**:

1. WHY — Explain why this concept exists in the real world (system purpose).
2. WHERE — Place it inside a system (control, state, flow, boundary,
responsibility).
3. WHAT CAN GO WRONG — List common failures and how good systems respond.
4. HOW — Explain minimal Python syntax/mechanism (no memorization, only
essentials).
```

```
5. WHO — Clarify ownership: who should handle this and who should not.

Rules:
- Use very simple language.
- Use concrete examples.
- Avoid unnecessary jargon.
- Focus on thinking, not typing.
- End with a one-line mental model.

Goal:
Help me understand the topic as a **system designer**, not a syntax memorizer.
```