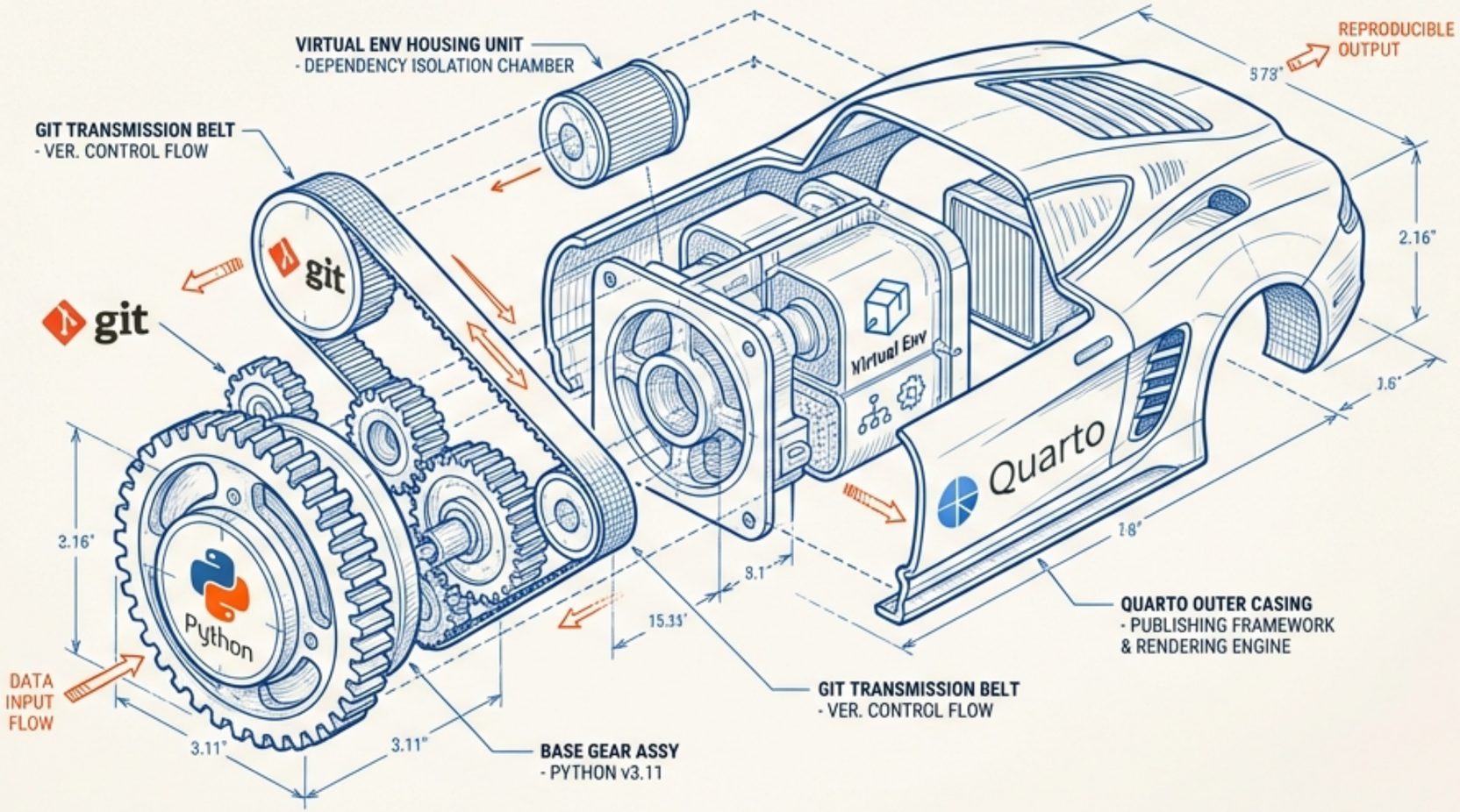
The Modern Analyst's Toolkit

From Chaos to Clarity: The Architecture of Reproducible Research



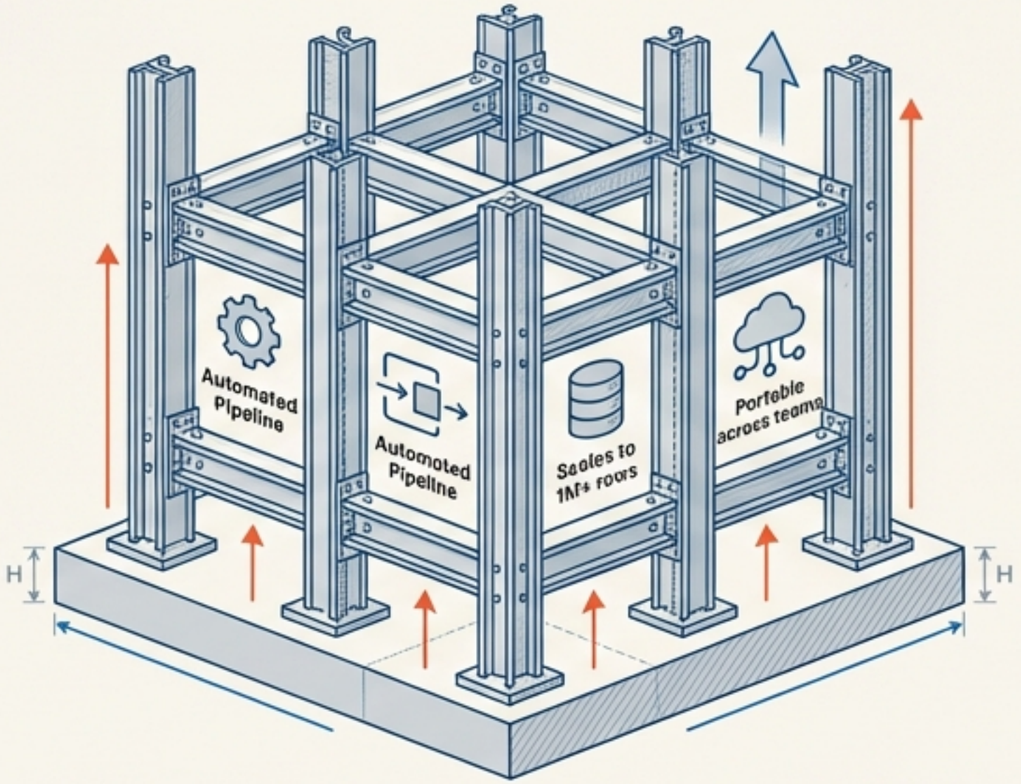
Understanding the *Why* Behind the Workflow: Quarto | Python | Environmental Architecture

Moving Beyond “It Works on My Machine”

The Ad-Hoc Method

- **Fragile** links
- Manual updates
- Unscalable
- **Zero reproducibility**

Reproducible Research

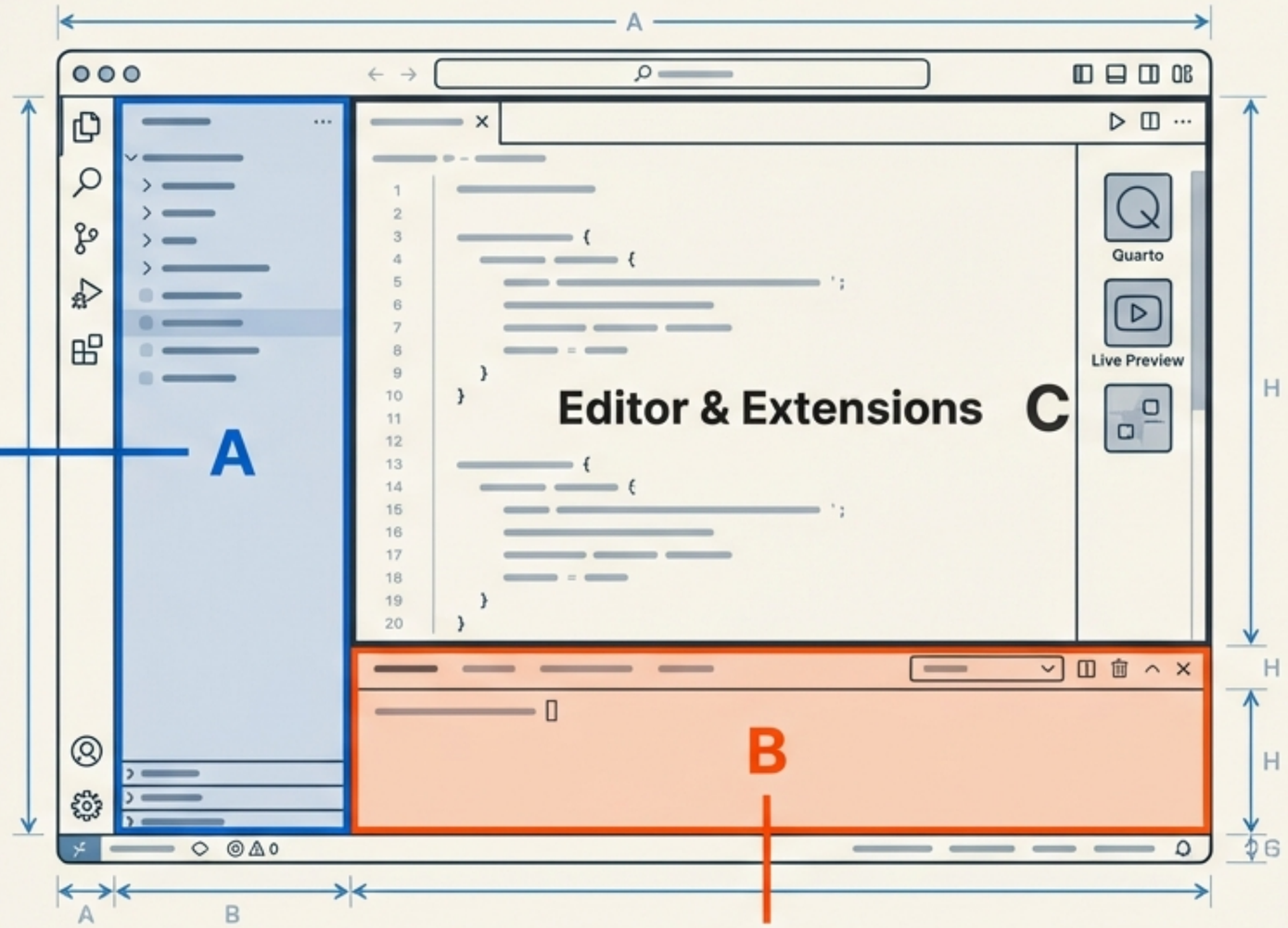


- **Version Controlled**
- **Automated Pipeline**
- **Scales to 1M+ rows**
- **Portable** across teams

Core Concept: We build systems, not just spreadsheets.

The Cockpit: Your IDE (Cursor AI)

File Explorer: Visualizing project structure

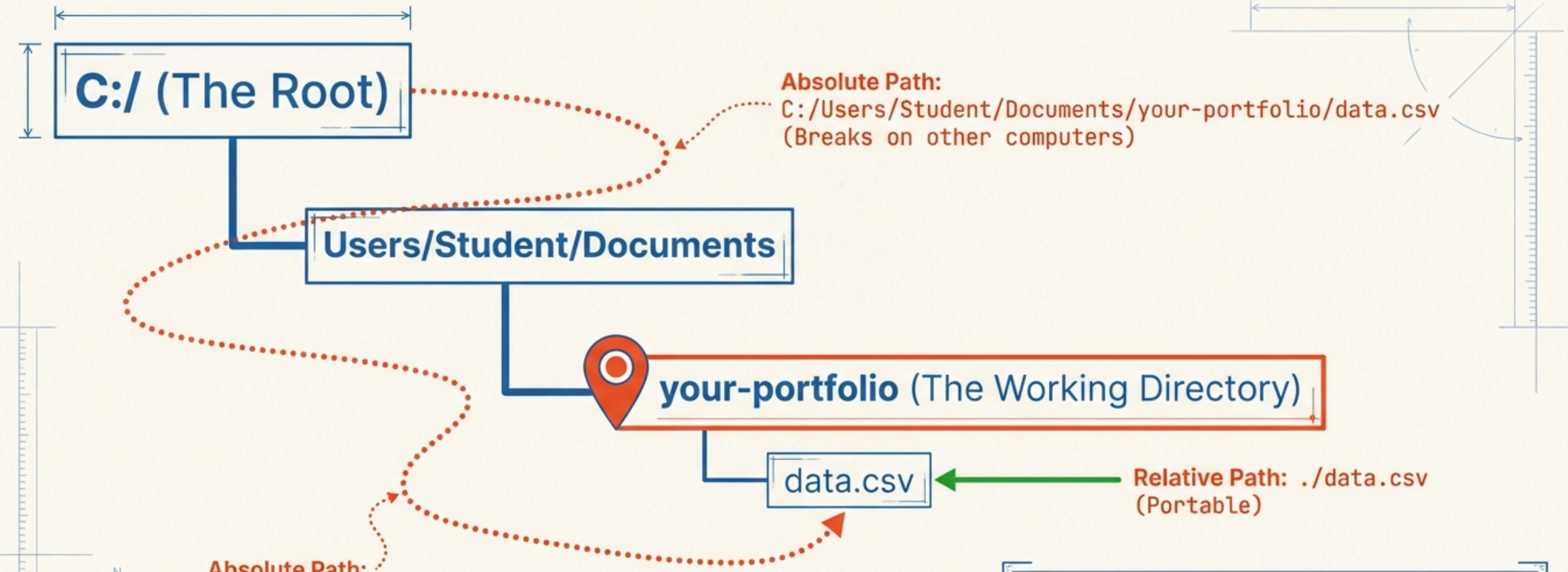


The Terminal: Direct OS communication

Why use an IDE?

1. **Unified Interface:** Edit code and manage files in one view.
2. **Extensions:** Tools like 'Quarto' for syntax highlighting and 'Live Preview' for real-time feedback.
3. **Analogy:** If Python is the engine, the IDE is the dashboard and steering wheel.

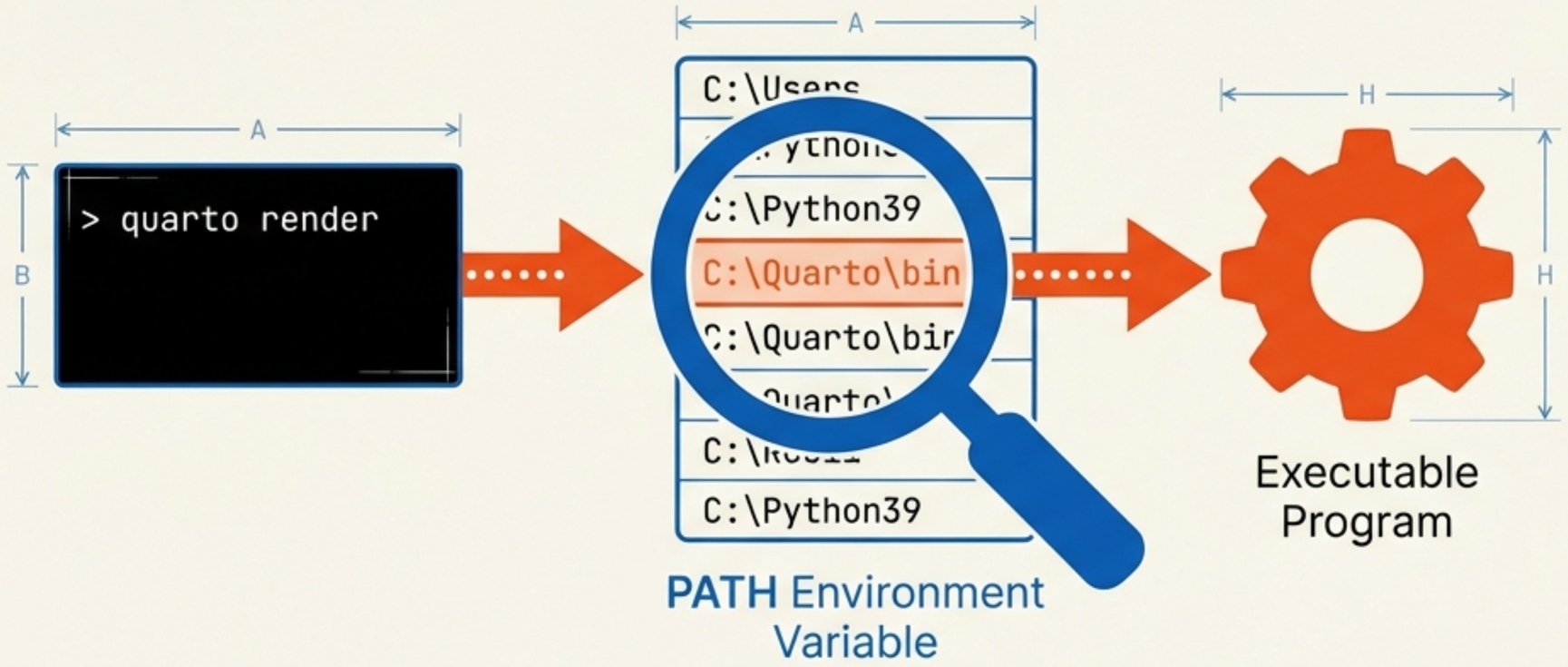
Ground Zero: Navigating the File System



Key Insight Box

The 'Working Directory' is where your code is currently standing. Always use relative paths so your code works on any machine.

The Interface: Speaking the Computer's Language



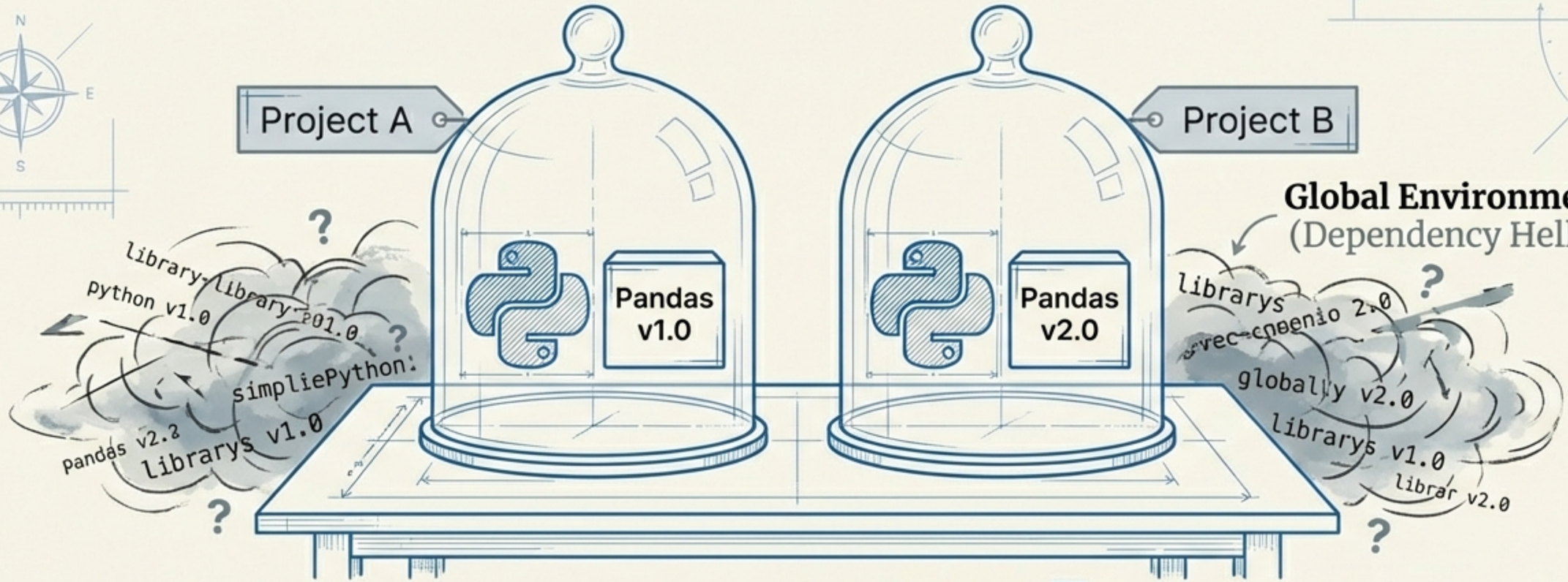
The Terminal is not magic. It is a text-based method to execute programs.

The PATH
An address book your computer uses to find programs. If you type 'quarto' and it fails, the computer doesn't know where to look.

Common Commands

- `"ls / dir"`: List files
- `"cd"`: Change directory

The Quarantine: Python Virtual Environments



The Problem:

- Different projects need different library versions.
- Installing globally creates conflicts.

The Solution:

- 'venv'.
- A self-contained sandbox sandbox for each project.

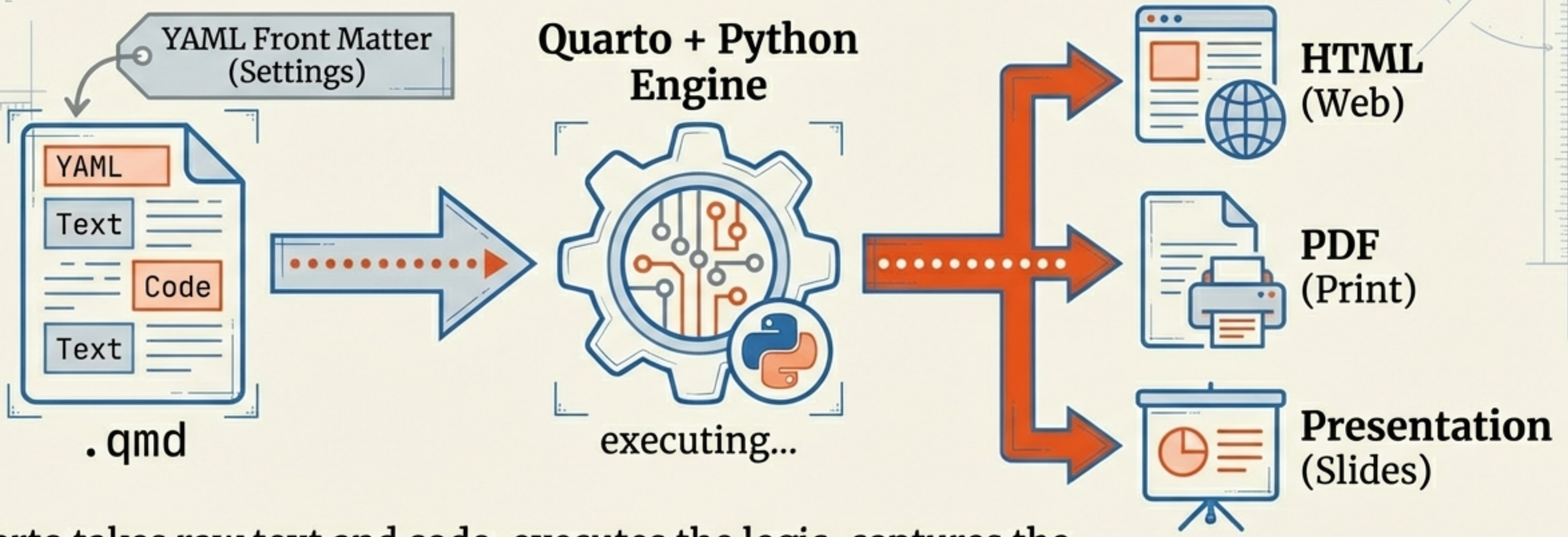
The Critical Step: ACTIVATION.

```
source venv/bin/activate
```

Tells the terminal to look inside the jar, not outside.

The Publisher: What is Quarto?

The Universal Document Converter



Quarto takes raw text and code, executes the logic, captures the results, and weaves them into professional documents.

YAML: The control panel at the top of your file that defines the output style.

The Artist: Data Visualization as Code

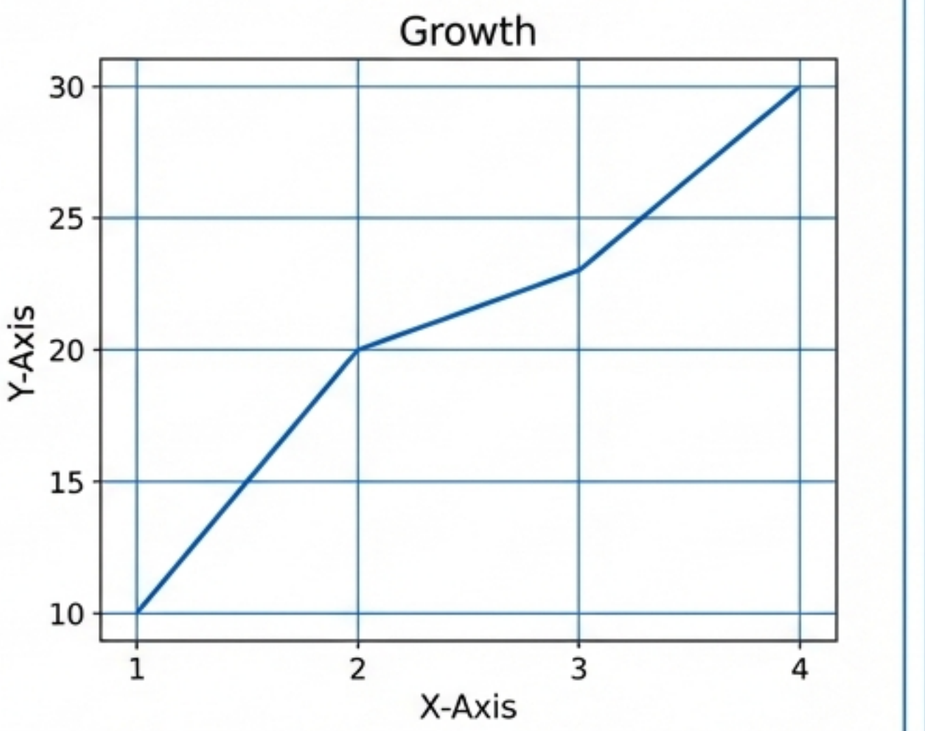
```
import matplotlib.pyplot as plt
```


```
import matplotlib.pyplot as plt
```


```
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x, y)
plt.title('Growth')
plt.show()
```

Rendered by Quarto

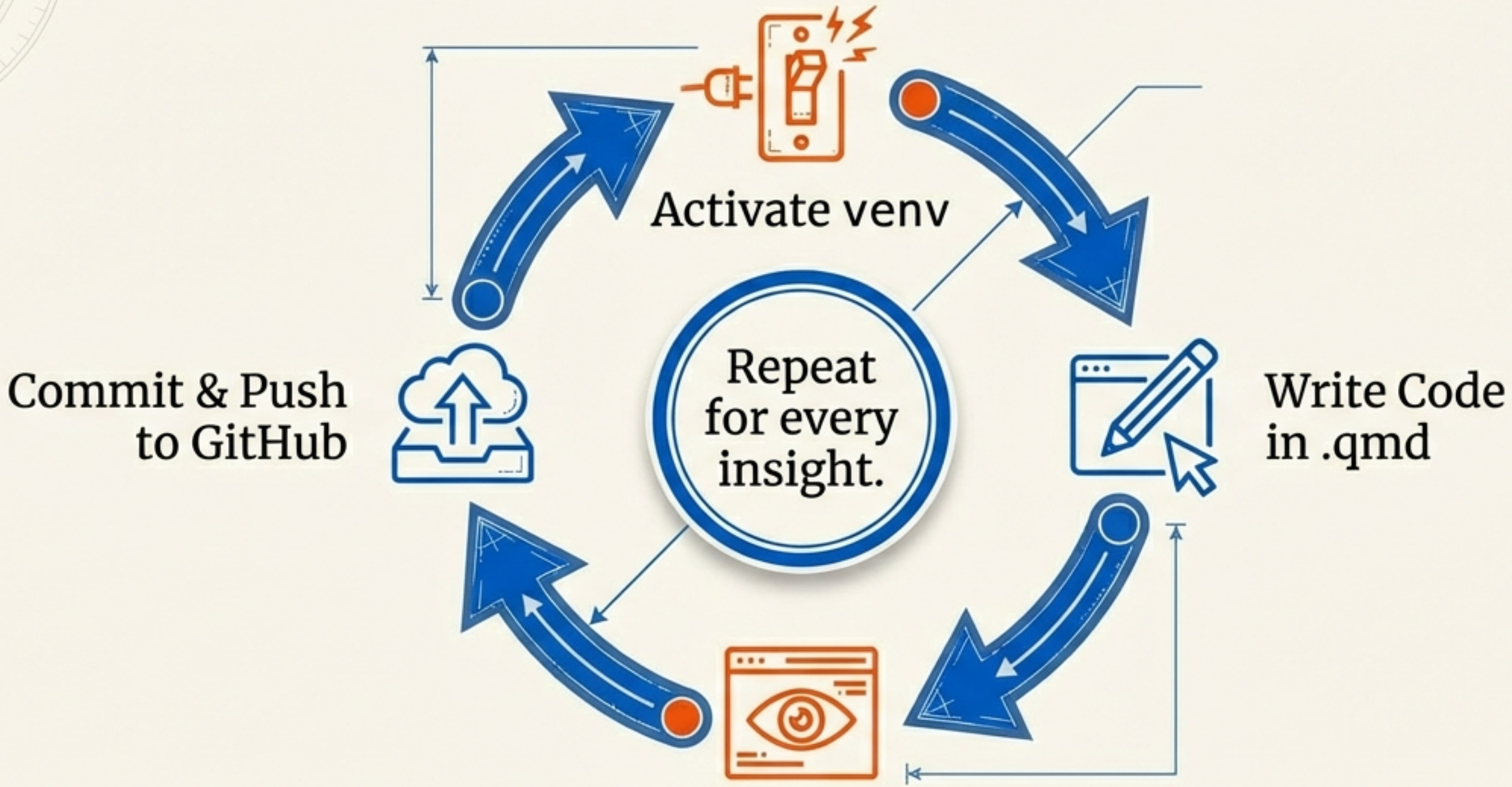


Matplotlib gives you granular control over every pixel. Unlike Excel, the chart is generated programmatically, making it reproducible instantly with new data.

The Time Machine: Git & GitHub

- **`Git`**: The tool that tracks changes and creates save points.
- **`GitHub`**: The platform that hosts the backup and enables collaboration.
- **`Commit`**: Saving a snapshot locally.

The Cycle of Development



The Product: GitHub Pages

A repository is just code. GitHub Pages turns that code into a live website.

Value: Instant credibility. A live URL is a product; a file attachment is just homework.

Configuration: `Settings > Pages > Source: Deploy from Main Branch / Root.`

Troubleshooting Checklist: When Things Break

- [ ] **Error: “Module not found”**
  Fix: Did you activate the venv? Is your IDE interpreter set to `.\venv\Scripts\python.exe`?
- [ ] **Error: “Quarto command not found”**
  Fix: Is Quarto in your PATH variable? Restart the terminal.
- [ ] **Error: “Script disabled” (PowerShell)**
  Fix: Run `Set-ExecutionPolicy RemoteSigned`.
- [ ] **Error: “Changes not visible online”**
  Fix: Did you Commit AND Push? (Committing is local only).

The setup is the hardest part. Errors are part of the process.

You Are Now an Architect

You haven’t just installed software. You have built a professional analytics infrastructure.

The Inventory

- **Repository Management** (GitHub)
- **Isolated Environments** (Venv)
- **Automated Reporting** (Quarto)
- **Programmatic Visualization** (Matplotlib)

Trust the process. The infrastructure is built. Now, go build the analysis.