NUNO BISPO

PRACTICAL PYDANTIC

THE MISSING GUIDE TO DATA VALIDATION IN PYTHON

UNLOCK THE FULL POWER OF PYTHON DATA VALIDATION WITH PYDANTIC — THE LIBRARY TRUSTED BY FASTAPI, SQLMODEL, AND MODERN PYTHON PROJECTS WORLDWIDE

Practical Pydantic

The Missing Guide to Data Validation in Python

Nuno Bispo

This book is available at https://leanpub.com/practical-pydantic

This version was published on 2025-10-14

Tweet This Book!

Please help Nuno Bispo by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just got Practical Pydantic and I'm ready to write safer, cleaner, and more reliable Python code! 🚀🐍 #Python #Pydantic #PracticalPydantic

The suggested hashtag for this book is #PracticalPydantic.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#PracticalPydantic

Also By Nuno Bispo

Contents

Preface

Why This Book Matters

Python is powerful, flexible, and easy to learn, but it comes with a trade-off. Its dynamic typing makes it simple to write code quickly, yet fragile when dealing with messy, unpredictable data. Type hints have improved the situation by allowing developers to describe intent, but they still don't enforce correctness at runtime.

This is where **Pydantic** steps in. By combining Python's type hints with robust runtime validation, Pydantic closes the gap between *what you say your code does* and *what it actually does in production.*

I wrote this book because Pydantic has become one of the most important tools in the modern Python ecosystem. Whether you're building APIs with FastAPI, ingesting data for analytics, or managing complex configuration across environments, Pydantic makes your applications safer, more predictable, and easier to maintain. Yet, despite its growing popularity, there hasn't been a single, practical, end-to-end resource that shows developers how to go from beginner to production-ready with confidence. That's what this book aims to be.

* * *

Who This Book Is For

This book is written for:

- **Python developers** who want to write more reliable and maintainable applications.
- **FastAPI users** who rely on Pydantic for request/response validation and want to unlock its full potential.

- **Data engineers** who wrangle with unstructured or messy data and need tools to validate, normalize, and transform it before downstream use.
- Anyone who wants to make their Python code **safer, cleaner, and more future-proof**.

You don't need to be an expert in type hints or validation libraries to benefit from this book. A working knowledge of Python and curiosity about writing better code is all that's required.

* * *

How to Use This Book

The book is designed to be **practical first**. Each chapter introduces key concepts, followed by **code examples** you can run and adapt. At the end of most chapters, you'll find **exercises** to reinforce what you've learned and encourage experimentation.

You can read the book cover to cover, but it's also structured so you can **jump straight to the parts that matter most to you**. For example:

- New to Pydantic? Start at the beginning with the foundations.
- Already using it with FastAPI? Skip to the API chapters.
- Exploring alternatives? Head straight to Part III.

Use it as both a learning resource and a **daily reference**.

* * *

A Personal Note from the Author

When I first started working with Python in production, I loved the speed and flexibility, but I also ran into countless bugs caused by bad data. API payloads didn't match the schema I expected, configuration files had typos, and messy input data broke entire workflows.

Discovering Pydantic was a turning point. Suddenly, type hints became more than just annotations for my editor; they became **guards at the gate** of my applications. My code became more resilient, my APIs more predictable, and my confidence in shipping software grew.

This book is my way of sharing that discovery with you. My goal is not just to teach you the mechanics of Pydantic, but to help you experience the same shift in mindset: to treat data validation as a **first-class part of your codebase**, not an afterthought.

Thank you for picking up this book. I'm excited to take this journey with you and I hope by the end, you'll see Pydantic not as just another library, but as an essential tool in your Python toolkit.

– Nuno Bispo
(a.k.a Developer Service)

Part I – Foundations of Pydantic

Chapter 1: Why Data Validation Matters in Python

Before diving into Pydantic itself, it's important to understand the problem it solves. Python gives us incredible flexibility, but that flexibility comes at a cost: **data safety is not guaranteed**. Let's explore why.

The Problem with Dynamic Typing

Python is dynamically typed, which means you don't declare the type of a variable when you create it. The interpreter doesn't care whether a variable is an integer, string, or dictionary - it only finds out when that variable is actually used at runtime.

This makes Python very productive for quick scripting and prototyping, but it also leads to subtle (and sometimes catastrophic) bugs:

```python
def calculate_discount(price, discount):
    return price - (price * discount)

# Works fine:
print(calculate_discount(100, 0.2))  # 80.0

# Unexpected behavior:
print(calculate_discount("100", 0.2))
# TypeError: can't multiply sequence by non-int of type 'float'
```

Here, Python happily lets you pass a string as the `price`. You don't discover the problem until runtime, and possibly in production, when users are depending on your code.

Dynamic typing means that **every piece of incoming data is a potential risk**:

- A JSON payload from an API might have the wrong types.
- A configuration file might include a typo.

- A database query result might not match your expectations.

The language won't protect you. It's up to you as the developer to check and validate everything, unless you use a tool like Pydantic.

Type Hints vs. Runtime Validation

Starting with Python 3.5, **type hints** (also called type annotations) were introduced. They allow you to specify the expected type of variables, function arguments, and return values.

```python
def calculate_discount(price: float, discount: float) -> float:
    return price - (price * discount)
```

This gives tools and IDEs a mechanism that can warn you if you misuse the function. But there's a catch:

- Type hints are **not enforced at runtime**.
- They are mainly for static analysis and readability.

```python
print(calculate_discount("100", 0.2))
# Still raises TypeError at runtime
```

Python ignores the type hints when actually running the code. That means type hints alone won't save you from invalid input in real-world applications.

This gap between **what you say your code expects** (type hints) and **what your code actually enforces** (nothing at runtime) is exactly where Pydantic comes in.

Where Data Goes Wrong in Real-World Projects

Let's consider a few scenarios where bad or inconsistent data can cause major issues:

- **APIs and Web Applications**

    - A user sends a POST request with `"age": "twenty-five"` instead of an integer.
    - Without validation, your application may throw an error or silently misbehave.
- **Configuration Management**
    - An environment variable meant to be a boolean is accidentally set to `"False"` (string).
    - Your app might interpret it as truthy and behave incorrectly in production.
- **Data Pipelines and ETL**
    - You ingest CSV files from external sources. Some rows have `NaN`, some have strings instead of numbers.
    - Without validation, downstream calculations may break or produce misleading results.
- **Machine Learning Workflows**
    - A model expects numeric features, but the dataset includes mixed types.
    - Training fails hours into the process, wasting time and resources.

In all these cases, the root issue is the same: **Python trusts whatever data it's given, and you pay the price later**.

Why Validation Matters

Good data validation gives you:

- **Early error detection**: Catch problems at the boundary of your system.
- **Clearer contracts**: Define exactly what kind of data your functions, APIs, or models expect.
- **Cleaner code**: No more scattered `if` checks, validation is centralized.
- **Confidence in production**: Your code behaves predictably, even with messy input.

This is why Pydantic is such a powerful library. It doesn't just annotate your data, it **enforces** types at runtime, automatically converts inputs when possible, and produces helpful error messages when validation fails.

Summary

In this chapter, you learned how to:

- Recognize the **limitations of Python's dynamic typing** when working with untrusted data.
- Understand the difference between **type hints** (static safety) and **runtime validation** (actual safety).
- Identify **common failure points** in real-world projects, such as APIs, config files, and CSV ingestion.
- See why **runtime validation is essential** for building reliable Python applications.

* * *

Chapter 2: Getting Started with Pydantic

Now that we've seen why validation matters, let's start working with Pydantic. This chapter will guide you through installation, your first `BaseModel`, and one of Pydantic's most powerful features: automatic type conversion.

Installation & Setup

Pydantic works with Python **3.8 and above**. To install the latest version (v2 at the time of writing), simply run:

```
pip install pydantic
```

If you're planning to use Pydantic with **FastAPI**, you can install both at once:

```
pip install "fastapi[all]" pydantic
```

For projects that use configuration files (`.env` files, environment variables), you'll also want the **settings extras**:

```
pip install pydantic-settings
```

✅ **Tip**: Always check your installed version with:

```
python -m pip show pydantic
```

The output from my system:

```
Name: pydantic
Version: 2.11.7
Summary: Data validation using Python type hints
Home-page: https://github.com/pydantic/pydantic
...
```

This ensures you're using the latest release, as syntax and features differ between **v1** and **v2**.

Defining Your First `BaseModel`

The heart of Pydantic is the `BaseModel`. It looks similar to a Python `dataclass`, but with one major difference: **it enforces types at runtime**.

Here's a simple example:

```python
from pydantic import BaseModel

class User(BaseModel):
    id: int
    name: str
    is_active: bool = True

# Creating an instance
user = User(id=1, name="Alice")

print(user)
# id=1 name='Alice' is_active=True
```

A few things to note:

- Fields are declared with **Python type hints**.
- Default values are supported (`is_active` defaults to `True`).

- Instantiating the model automatically validates input data.

Unlike regular Python classes or dataclasses, if you pass the wrong types, Pydantic will catch it immediately.

```
User(id="one", name="Alice")
# pydantic_core._pydantic_core.ValidationError: 1 validation error for User
# id
#   Input should be a valid integer, unable to parse string as an integer
↪   [type=int_parsing, input_value='one', input_type=str]
```

This makes `BaseModel` a **trustworthy contract** for your data.

Automatic Type Conversion in Action

One of Pydantic's most impressive features is its ability to **coerce types** whenever possible. Instead of just throwing errors, it intelligently converts values to match your model.

```
user = User(id="42", name="Bob", is_active="true")

print(user)
# id=42 name='Bob' is_active=True
```

What happened here?

- `"42"` (a string) was converted to `42` (an integer).
- `"true"` (a string) was converted to `True` (a boolean).

This makes Pydantic extremely useful when dealing with **unpredictable external data** (like JSON payloads, config files, or API responses).

Of course, if the conversion is impossible, you'll get a **clear validation error**:

```python
User(id="hello", name="Charlie")
# pydantic_core._pydantic_core.ValidationError: 1 validation error for User
# id
#   Input should be a valid integer, unable to parse string as an integer
↪  [type=int_parsing, input_value='hello', input_type=str]
```

Why This Matters

With just a few lines of code, you now have:

- A **data model** with clear type expectations.
- Automatic runtime validation and conversion.
- Meaningful error messages when something goes wrong.

This foundation will scale to much more complex use cases, nested models, API requests, and configuration management, all of which we'll explore in later chapters.

Quick Exercise

Try this yourself:

- Define a `Product` model with the following fields:
  - `id: int`
  - `name: str`
  - `price: float` (default: `0.0`)
  - `in_stock: bool` (default: `True`)
- Create a product with the following data (notice the types are "wrong" on purpose):

```python
data = {
    "id": "101",
    "name": "Keyboard",
    "price": "49.99",
    "in_stock": "false"
}
```

- Print the model. What happens?

👉 This simple exercise will show you just how much Pydantic does behind the scenes.

Summary

In this chapter, you learned how to:

- Install and set up **Pydantic** in your Python environment.
- Define your first **`BaseModel`** and see how it enforces type safety.
- Leverage **automatic type conversion** to handle messy input gracefully.
- Validate data at runtime, catching errors early before they cause bigger problems.

* * *

Chapter 3: Core Concepts You Must Know

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Fields, Defaults, and Optional Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Nested Models and Lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Working with Enums and Constrained Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Enums

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Constrained Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Handling Validation Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Quick Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 4: Serialization and Transformation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

`model_dump()` and JSON Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Parsing Input from APIs, CSV, and Dicts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

From Dicts (e.g., API payloads)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

From JSON (e.g., webhooks)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

From CSV Rows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Round-Tripping Data Safely

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Quick Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 5: Pydantic v1 vs v2 – What Changed and Why It Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

The Rust-Powered `pydantic-core`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Performance Improvements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Migrating from v1 to v2

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Validation Entry Points

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Field Validators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Config

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Migration Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Quick Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Part II – Pydantic in the Real World

Chapter 6: Building APIs with FastAPI and Pydantic

Request and Response Models

Response Model

Automatic OpenAPI Docs

Reusable and Nested Schemas

Validation Best Practices for APIs

Quick Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 7: Data Engineering with Pydantic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Ingesting Messy CSV and JSON Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

CSV Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

JSON Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Custom Validators for Cleaning Datasets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Normalizing Strings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Handling Common "Messy" Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Structuring ETL Pipelines with Pydantic Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Simple ETL Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Quick Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 8: Configuration and Settings Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Using `BaseSettings` for Environment Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Basic Settings

Loading `.env` Files and Secrets Securely

`.env` File Example

Loading with Pydantic

Multi-Environment Setups (Dev, Staging, Prod)

Strategy: Multiple `.env` Files

Example:

Quick Exercise

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 9: Advanced Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Custom Validation with `@field_validator`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Password Strength

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Complex Constraints (Regex, Ranges, Dates)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Regex Constraint

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Numeric Ranges

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Date Constraints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Model Inheritance, Composition, and Dynamic Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Inheritance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Composition (Nested Models)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Dynamic Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Performance Tuning for Large-Scale Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Tips for Scaling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: ETL Pipeline Benchmark

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Quick Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 10: Case Study – An End-to-End Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

The Project: Product Feedback API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Step 1: Defining the Application Settings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Step 2: Defining the SQLite Database

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Step 3: Pydantic Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Step 4: Defining the API Endpoints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Step 5: Putting all Together

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Lessons Learned from Real Production Scenarios

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Part III – Beyond Pydantic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 11: Marshmallow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Strengths of Marshmallow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Weaknesses of Marshmallow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Marshmallow in Action

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Defining a Schema

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Where Marshmallow Might Be a Better Fit

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 12: attrs and Dataclasses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

When Lightweight Models Are Enough

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

How attrs Compares to Pydantic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: Defining a Model with attrs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Writing Custom Validation with Dataclasses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example: A Simple Dataclass

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Adding Custom Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

attrs vs Dataclasses vs Pydantic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 13: Other Contenders (Cerberus, TypedDict, Voluptuous)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Cerberus

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Pros

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Cons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

TypedDict and Python Typing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Pros

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Cons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Voluptuous

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Pros

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Cons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Why Pydantic Still Wins

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Chapter 14: Choosing the Right Tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Decision Matrix

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Recommendations by Use Case

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Data Pipelines (ETL, data ingestion, cleaning)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Configurations & Environment Settings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Machine Learning Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

The 80/20 Rule

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Final Thoughts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

A Personal Note

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/practical-pydantic.

Appendix: Next Steps & Resources

GitHub Repository

Official Documentation

Suggested Projects

Final Advice