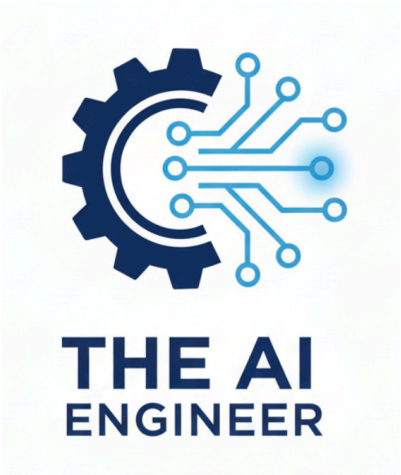
Python Primer for Data Science and Deep Learning

Sample: Preface + One Chapter

Dr. Yves J. Hilpisch with GPT-5

October 3, 2025



Preface

This sample introduces the approach and style of the primer: small surface area, fast payoff, and repeatable habits. Each concept is taught with short, runnable examples so you can apply ideas immediately in your own projects.

How to Read

Keep your editor and a terminal open. Type the code, tweak a parameter, re-run, and save a tiny artifact (a script, a figure, a passing assertion). Momentum matters more than coverage.

Core principles

- **Clarity first** — readable, idiomatic Python over clever one-liners.
- **Small steps** — functions and scripts that do one job well and compose.
- **Reproducibility** — local virtual environments, version control, and simple validation tools.

Contents

Part I

Getting Started

1 Introduction

This primer equips readers with the essential Python concepts and engineering tools required for the books *Python & Mathematics for Data Science and Machine Learning*, *Building a Large Language Model from Scratch*, and *Deep Learning Basics with PyTorch*. It is intentionally concise and example-driven.

1.1 Assumed Prerequisites

Basic command-line literacy and high-school mathematics. No prior deep learning background required.

1.2 Why This Primer Matters

Machine learning and deep learning reward clarity: clear data structures, clear code, and clear workflows. Python gives you all three—but only if you use a small set of well-chosen ideas consistently. Think of this book as your runway: a short, smooth takeoff that gets you to cruising altitude fast so you can focus on the interesting parts (models, math, and experiments) rather than wrestling with tooling.

By the end you will:

- write clean, idiomatic Python that other people (and Future You) can read,
- think in arrays and vectorized operations for performance and clarity,
- explore and visualize data reproducibly,
- evaluate simple models with confidence and avoid common traps, and
- organize projects so they are easy to run, test, and share.

Promise & Payoff

We keep the surface area small and the payoff large. Every example is designed to be copy&paste-able into your own work, and every habit (like using `python -m pip`) fights a real, recurring source of pain.

At a Glance

- **Outcomes** — idiomatic Python, arrays & vectorization, clear plots, tidy data, solid baselines.
- **How to read** — code first, copy then tweak, keep a venv per project, save artifacts.
- **Habits** — `python -m pip`, `src/` layout, tiny tests, type hints, logging.
- **Tools** — NumPy, Matplotlib, pandas, scikit-learn, pytest, black, ruff, mypy, pre-

commit, Git.

- **Style** — minimal surface area, repeatable steps, examples you can reuse verbatim.

1.3 Who This Book Is For

You may be a data scientist, engineer, student, or a curious practitioner crossing over from another language. If you have basic command-line comfort and remember high-school math, you have everything you need. If you already use Python daily, treat this as a tune-up: you will still pick up patterns that make teams move faster with fewer surprises.

1.4 How to Use This Book

This is a hands-on primer. Read with your editor open and a terminal nearby. Type the code, tweak a parameter, re-run. Aim for short sessions that end with a tiny artifact (a working script, a saved plot, a green test).

Suggestions:

- **Make a sandbox repo** and keep all examples there. Commit often.
- **Use a venv** per project (we show how) and pin versions when your work stabilizes.
- **Prefer scripts over notebooks** for steps you will repeat; prefer notebooks to explore.
- **Write one test** for anything you will run twice. Start tiny.

1.5 Google Colab (Optional)

If you prefer a zero-install start, use Google Colab in your browser: open `https://colab.research.google.com`, click **New Notebook**, and run the cells below. Colab already ships with NumPy and Matplotlib.

Open a blank notebook in one click: Open in Colab

Tip: In notebooks, shell commands start with `!` (e.g., `!pip install seaborn`). Save via **File → Save a copy in Drive**. Sessions reset after inactivity, so keep your first cell a "setup" cell and re-run it when needed.

```
import numpy as np
import matplotlib.pyplot as plt

# array math
x = np.array([1, 2, 3, 4])
print("mean:", x.mean())

# a tiny plot
t = np.linspace(0.0, 2.0 * np.pi, 200)
plt.plot(t, np.sin(t))
plt.title("Hello, Colab")
plt.show()
```

To save outputs to Google Drive (persistent across sessions), mount your Drive and write files under `/content/drive/MyDrive/`.

```python
from google.colab import drive
drive.mount('/content/drive')  # authorize once per session

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.0, 2.0 * np.pi, 200)
plt.plot(x, np.sin(x))
plt.savefig('/content/drive/MyDrive/colab/hello_line.png', dpi=150)
```

1.6 What Is Inside: A Guided Tour

The primer is organized in four parts plus a short wrap-up. Each chapter is self-contained and practical.

Part I — Getting Started

- **1. Introduction** (this chapter) sets expectations and the learning playbook.
- **2. Python Environment** shows a clean, repeatable setup with `python`, `pip`, and `venv`—no drama, no vendor lock-in.

Part II — Core Python Concepts

- **3. Data Types and Structures** covers numbers, strings, lists, tuples, sets, and dicts—and how to slice, index, and combine them.
- **4. Control Flow** teaches decisions, loops, comprehensions, and Pythonic iteration.
- **5. Functions and Modules** shows how to design tiny, composable functions and package them.
- **6. Object-Oriented Basics** gives you lightweight classes when data + behavior belong together.
- **7. Idiomatic Python** distills patterns like context managers, decorators, and EAFP into daily use.

Part III — Scientific Python Stack

- **8. NumPy Essentials** builds your mental model for arrays, broadcasting, and linear algebra.
- **9. Matplotlib** turns numbers into clear, reproducible figures.
- **10. pandas Basics** works with labeled tabular data: selection, groupby, joins, and tidy reshaping.
- **11. scikit-learn Mini Primer** gets you to solid baselines and sensible evaluation quickly.

Part IV — Engineering Tools

- **12. Version Control with Git** gives you safe experiments, collaboration, and a time machine.
- **13. Software Craftsmanship** ties it together: layout, tests, typing, formatting, packaging, CLI, logging.

Next Steps

- **14. Bridging to the Main Books** maps how this primer feeds your ML/DL journey.
- **15. Further Resources** points to docs you will actually use.

1.7 Learning Principles We Use

- **Small steps stick** — tiny examples reduce friction and build momentum.
- **Names matter** — we choose explicit names and clear signatures over clever code.
- **Reproducibility by default** — every plot and model is created by code you can re-run.
- **Pragmatic minimalism** — only the tools you need, used the same way every time.

1.8 Exercises

1. **Check your Python**

   Verify your interpreter and pip, then print the Python path from inside the REPL.

   Hint:

   ```
   python3 --version
   python3 -m pip --version
   ```

   ```
   import sys
   sys.executable
   ```

2. **Create a venv**

   Make a project folder, create `.venv`, activate it, and upgrade pip.

   Hint:

   ```
   python3 -m venv .venv
   source .venv/bin/activate
   python -m pip install --upgrade pip
   ```

3. **Try Google Colab**

   Open Colab and create a new Python 3 notebook. Copy the code from `Google Colab (Optional)` above, run it, and save a copy to Drive. Then mount Google Drive and `savefig` a plot to `/content/drive/MyDrive/colab/`.

   Hint: Use Open in Colab to start quickly. Mount Drive, then save files under `/content/drive/MyDrive/`:

   ```
   from google.colab import drive
   drive.mount('/content/drive')
   ```

4. **Hello, arrays**

   Install NumPy in your venv and compute the mean of `[1,2,3,4]`.

   Hint:

```
python -m pip install numpy
```

```
import numpy as np
np.mean([1, 2, 3, 4])
```

5. **First plot**

   Install Matplotlib and save a sine curve to `figures/hello_line.png`.

   Hint:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 6.28, 200)
y = np.sin(x)
```

```
plt.plot(x, y)
plt.savefig('figures/hello_line.png', dpi=150)
```

6. **Start a sandbox repo**

   Initialize Git in your project, add a `README.md`, and make your first commit.

   Hint:

```
git init
printf "# Sandbox\n" > README.md
git add .
git commit -m "init"
```