


# GenAI Companion for The Data Scientist

GenAI Tools and Techniques for Data Science

Dr. Yves J. Hilpisch 

April 23, 2026



---

<sup>1</sup>Get in touch: [linktr.ee](https://linktr.ee/yhilpisch). Web page: [thedatascientist.dev](https://thedatascientist.dev). Research, structuring, drafting, and visualizations were assisted by GPT 5.x as a co-writing tool under human direction. Comments and feedback are welcome.

# Preface

This companion volume sits beside the core *The Data Scientist* path. It is intentionally parallel: the core books remain the primary source of technical content and progression, while this companion shows how to use modern GenAI tooling in disciplined browser and terminal workflows at each module.

The aim is practical acceleration with quality control. Delegates should become faster at planning, drafting, debugging, and reviewing without outsourcing judgment. Every generated output must still be tested, validated, interpreted, and documented.

## Positioning

This is a companion track, not a replacement track. Fundamentals, statistics, and reproducible engineering habits from the core path stay non-negotiable.

Tooling choice needs one explicit note. This book focuses on OpenAI tooling (ChatGPT in the browser and Codex in the CLI) because that reflects the author's own preference and production experience. There are credible alternatives, including Claude + Claude Code and Gemini + Gemini CLI. They are comparable to a meaningful extent, but each has relative strengths and weaknesses depending on task type, context window behavior, tool integration, and coding workflow style.

Delegates should compare options directly before committing to a daily setup, with special attention to pricing under expected workload. A light usage profile and a heavy code-assistance profile can have very different cost behavior.

## Cost and Tooling Reality

Do not choose a stack by brand alone. Compare capability, reliability in your actual tasks, and monthly cost under realistic usage volume.

This companion is organized in the same sequence as the program:

- primer setup and operating model,
- Module 1 companion (Python foundations),
- Module 2 companion (data work and storytelling),
- Module 3 companion (statistics and machine learning),
- Module 4 companion (reproducible projects and delivery).

The practical loop remains constant across all chapters:

1. define the task clearly,
2. generate a first draft,

3. verify with checks and tests,
4. revise and document decisions,
5. keep artifacts reproducible.

That loop is where GenAI becomes useful rather than noisy. The goal is not to produce more text or more code. The goal is to produce clearer, safer, and more explainable work.

#### Companion Asset Sync

Chapter examples point to files in `code/`, `notebooks/`, and `artifacts/`. Some are fully populated starter assets, while others are intentionally lightweight placeholders for delegate work.

If you are unsure whether a referenced file is already provided, check `artifacts/asset_manifest.md` first. Treat that file as the single source of truth for current availability and next planned additions.

# Contents

<b>Preface</b>	<b>i</b>
<b>1 GenAI Setup and Study Workflow</b>	<b>1</b>
1.1 Operating Model for the Companion Track	1
1.2 Accounts, Plans, and Access	2
1.2.1 ChatGPT Account and Plan	2
1.2.2 API Platform Access	3
1.3 System Prerequisites	3
1.4 Codex CLI Installation	3
1.4.1 macOS	3
1.4.2 Linux	4
1.4.3 Windows	4
1.5 First Run and Authentication	5
1.6 API Key Configuration (Detailed)	5
1.6.1 Create and store the key	5
1.6.2 macOS/Linux (bash/zsh)	5
1.6.3 Windows PowerShell	5
1.7 Security Baseline for the Primer	6
1.8 Privacy-First Local Model Fallback	6
1.9 Primer Workflow Templates	6
1.9.1 Prompt template (browser)	6
1.9.2 Task template (CLI)	7
1.9.3 Mock-Data Prompt Template	7
1.10 Three Primer Companion Exercises	7
1.11 First Companion Mini Project (Primer)	8
1.12 Definition of Done for Primer Completion	9
<b>2 Python Foundations</b>	<b>10</b>
2.1 Module 1 Companion Operating Pattern	10
2.2 Prompting Patterns for Module 1	11
2.2.1 Browser Prompt Template	11
2.2.2 CLI Task Template	11
2.3 Workflow Example 1: Function Draft and Repair	12
2.3.1 Task	12
2.3.2 Draft in Browser, Implement in CLI	12
2.3.3 Local Verification	12
2.3.4 Review Checklist	13
2.4 Workflow Example 2: Strings and Files Script	13
2.4.1 Task	13
2.4.2 Reference Script Shape	13
2.4.3 Implementation Loop	13
2.5 Workflow Example 3: Algorithm Comparison	14

2.5.1	Task	14
2.6	Verification Protocol for Module 1	14
2.7	Three Module 1 Companion Exercises	14
2.8	Module 1 Companion Project	16
2.8.1	Project Title	16
2.8.2	Project Objective	16
2.8.3	Required Deliverables	16
2.8.4	Project Rubric (Companion Layer)	16
2.8.5	Definition of Done	16
2.9	Code Comprehension and Legacy Audit Workflow	17
2.10	Bridge to Module 2	17
<b>3</b>	<b>Data Work and Storytelling</b>	<b>18</b>
3.1	Module 2 Companion Operating Pattern	19
3.2	Prompting Patterns for Tabular Work	19
3.2.1	Browser Template: Analysis Planning	19
3.2.2	CLI Template: Implementation Task	19
3.3	Workflow Example 1: Cleaning Pipeline with Validation	20
3.3.1	Task	20
3.3.2	CLI Implementation	20
3.3.3	Run and Verify	20
3.4	Workflow Example 2: EDA and Chart Narrative Critique	20
3.4.1	Task	21
3.4.2	Browser to CLI Split	21
3.4.3	Narrative Review Checklist	21
3.5	Workflow Example 3: SQL and pandas Reconciliation	21
3.5.1	Task	21
3.5.2	Reference SQL Query	21
3.5.3	CLI Reconciliation Task	22
3.6	Module 2 Verification Protocol	22
3.6.1	Quick Validation Snippets	22
3.7	Three Module 2 Companion Exercises	23
3.8	Module 2 Companion Project	24
3.8.1	Project Title	24
3.8.2	Project Objective	24
3.8.3	Required Deliverables	24
3.8.4	Project Rubric (Companion Layer)	24
3.8.5	Definition of Done	25
3.9	Synthetic Data for Pipeline Stress Tests	25
3.10	Bridge to Module 3	25
<b>4</b>	<b>Statistics and ML Foundations</b>	<b>26</b>
4.1	Module 3 Companion Operating Pattern	27
4.2	Prompting Patterns for Statistics and ML	27
4.2.1	Browser Template: Statistical Framing	27
4.2.2	CLI Template: Modeling Implementation	27
4.3	Workflow Example 1: Baseline and Supervised Pipeline	28
4.3.1	Task	28
4.3.2	CLI Implementation	28
4.3.3	Run and Verify	28
4.4	Workflow Example 2: Metric Integrity and Threshold Review	28
4.4.1	Task	29

4.4.2	CLI Refinement Task	29
4.4.3	Narrative Guardrails	29
4.5	Workflow Example 3: Feature Thinking and Leakage Checks	29
4.5.1	Task	29
4.5.2	Feature Review Prompt	29
4.5.3	CLI Implementation and Checks	30
4.6	Module 3 Verification Protocol	30
4.6.1	Quick Validation Snippets	30
4.7	Three Module 3 Companion Exercises	30
4.8	Module 3 Companion Project	32
4.8.1	Project Title	32
4.8.2	Project Objective	32
4.8.3	Required Deliverables	32
4.8.4	Project Rubric (Companion Layer)	32
4.8.5	Definition of Done	32
4.9	Advanced Prompting for Statistical Reasoning	33
4.10	Bridge to Module 4	33
<b>5</b>	<b>Reproducibility and Delivery</b>	<b>34</b>
5.1	Module 4 Companion Operating Pattern	35
5.2	Prompting Patterns for Engineering Delivery	35
5.2.1	Browser Template: Delivery Planning	35
5.2.2	CLI Template: Repo Hardening Task	35
5.3	Workflow Example 1: Repository Structure Hardening	36
5.3.1	Task	36
5.3.2	CLI Implementation	36
5.3.3	Verification and Review	36
5.4	Workflow Example 2: Test and Reliability Loop	36
5.4.1	Task	36
5.4.2	CLI Reliability Pass	37
5.4.3	Failure-Triage Pattern	37
5.5	Workflow Example 3: Environment and Delivery Readiness	37
5.5.1	Task	37
5.5.2	CLI Environment Task	37
5.5.3	Readiness Checks	37
5.6	Module 4 Verification Protocol	38
5.6.1	Quick Delivery Snippets	38
5.7	Three Module 4 Companion Exercises	38
5.8	Module 4 Companion Project	39
5.8.1	Project Title	40
5.8.2	Project Objective	40
5.8.3	Required Deliverables	40
5.8.4	Project Rubric (Companion Layer)	40
5.8.5	Definition of Done	40
5.9	Context Window Management for Repository Work	40
5.10	Program-Level Close	41
	<b>Epilogue</b>	<b>42</b>
	<b>Glossary</b>	<b>43</b>

# List of Figures

1.1 The companion workflow loop used across all stages. . . . .	2
2.1 Browser and CLI assistants feed one artifact stream with explicit verification. . .	11

# Bibliography

- [1] Hilpisch, Yves J. (2026): *Python Primer for Data Science. The Data Scientist.*
- [2] Hilpisch, Yves J. (2026): *Python Programming Foundations for Data Science. The Data Scientist.*
- [3] Hilpisch, Yves J. (2026): *Python Data Analysis, Visualization, and Storytelling. The Data Scientist.*
- [4] Hilpisch, Yves J. (2026): *Statistics, Machine Learning, and Model Evaluation. The Data Scientist.*
- [5] Hilpisch, Yves J. (2026): *Software Engineering, Reproducibility, and Deployment Basics. The Data Scientist.*
- [6] OpenAI (2026): *CLI – Codex Documentation.* [developers.openai.com](https://developers.openai.com).
- [7] OpenAI (2026): *openai/codex README.* [github.com](https://github.com).
- [8] OpenAI (2026): *openai/codex installation guide.* [github.com](https://github.com).
- [9] OpenAI (2026): *OpenAI API Authentication and API Keys.* [platform.openai.com](https://platform.openai.com).
- [10] OpenAI Help Center (2026): *Where do I find my OpenAI API Key?.* [help.openai.com](https://help.openai.com).
- [11] OpenAI (2026): *ChatGPT Pricing.* [openai.com](https://openai.com).

# Contact

GenAI Companion for The Data Scientist  
GenAI Tools and Techniques for Data Science



Get in touch:

[linktr.ee](https://linktr.ee)

[thedata scientist.dev](https://thedata scientist.dev)

© 2026 Dr. Yves J. Hilpisch — All rights reserved.