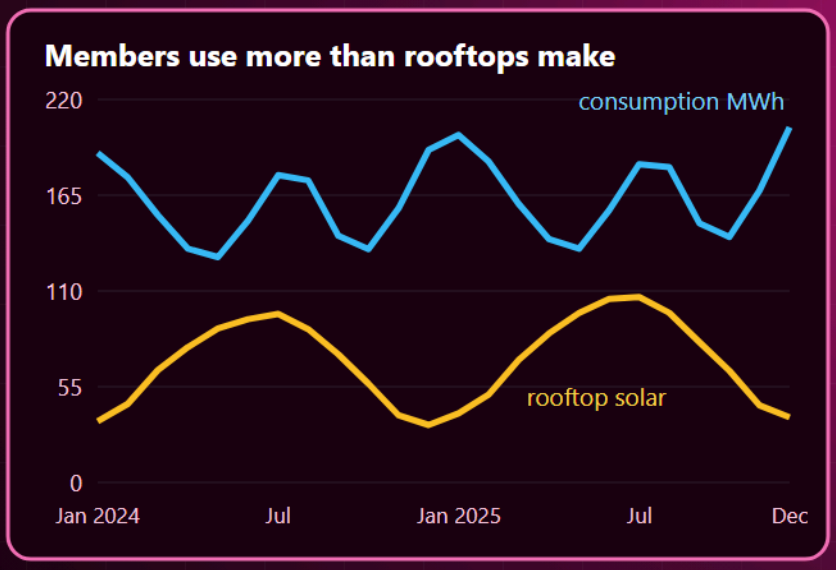
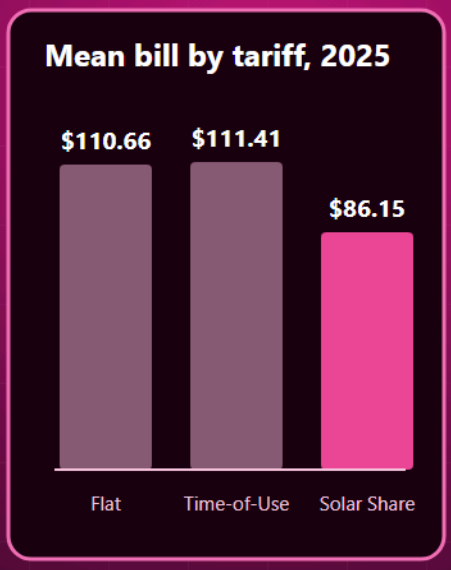
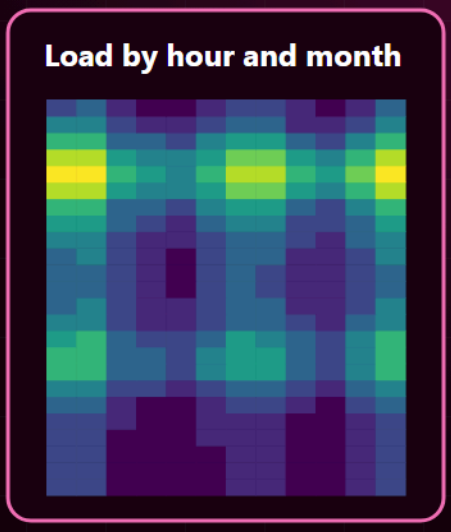







```
fig, ax = plt.subplots()
plot_coop_monthly(ax=ax)
ax.set_ylim(0, None) # honest baseline
save_figure(fig, "coop-monthly")
```

Before you share

✓ bars start at zero ✓ units on axes
✓ works in grayscale ✓ alt text written

CLEAR · HONEST · REPRODUCIBLE

Data Visualization with Python

Clear, honest charts with Matplotlib, Seaborn, and Plotly

Chart choice · Perception · Color & accessibility · Layouts
Time series · Statistical graphics · Interactive HTML · Automation

PRINCIPLES · MATPLOTLIB · SEABORN · PLOTLY · PRODUCTION

THENEXBOOKS

Data Visualization with Python

Clear, Honest Charts with Matplotlib, Seaborn, and Plotly

From choosing the right chart to automated, reproducible chart packs — built step by step

First Edition

THENEXBOOKS

Data Visualization with Python
Clear, Honest Charts with Matplotlib, Seaborn, and Plotly

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. Code examples may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Python and its libraries change over time; function names, defaults, and output can differ between versions. The examples were tested with the versions listed in the Preface; always consult the official documentation for the versions you use.

Python is a trademark of the Python Software Foundation. Matplotlib, seaborn, pandas, NumPy, SciPy, Pillow, Polars, pytest, and Plotly are names of their respective projects or owners. Microsoft, Windows, Edge, Excel, and Power BI are trademarks of the Microsoft group of companies. Other product names are trademarks of their respective owners. Their use in this book is for identification only and does not imply endorsement. This book is independent and is not affiliated with, sponsored by, or endorsed by any of these projects or companies.

Meadowlark Energy Co-op, the community energy cooperative used as the running example in this book, is fictional, as are the town of Meadowlark, its members, solar arrays, tariffs, survey answers, and all sample data. The data is generated by a seeded program listed in Appendix B, so every number and chart in this book can be reproduced exactly.

Preface

A chart is often the only part of an analysis that people actually look at. A board member skims the report, sees one line going up, and makes a decision. A manager forwards a single bar chart to the whole team. If that chart is cluttered, it wastes their attention; if it is misleading, it changes their minds for the wrong reasons; if it cannot be read by someone with color-vision deficiency, it quietly excludes part of the audience. Good visualization is not decoration added at the end. It is how analysis reaches people.

Python is an excellent place to make charts. Matplotlib gives you complete control over every line and label and produces print-quality output. The seaborn library turns tidy data into statistical graphics with very little code. Plotly creates interactive charts that anyone can open in a browser. Together they cover almost every chart a working analyst or developer needs to produce — and because charts are written as code, they can be reviewed, versioned, tested, and regenerated every month with one command.

This book teaches both halves of the craft. You will learn the principles that make charts clear and honest — how people perceive position, length, area, and color; how to choose a chart for a question; how baselines, scales, and missing data can mislead — and you will learn the three libraries deeply enough to apply those principles without fighting the tools. Every chart in the book is produced by code printed in the book and run on the same dataset, so you can reproduce each figure exactly and then adapt it to your own data.

Who This Book Is For

- **Data analysts** who already use pandas or spreadsheets and want charts that explain, not just display.
- **Developers and data engineers** who need to generate reliable charts for reports, dashboards, and documentation from code.
- **BI and report developers** moving from point-and-click tools to reproducible chart scripts.
- **Students, researchers, and scientists** who want figures that are accurate, accessible, and ready for print or the web.

You should know basic Python — variables, lists, dictionaries, functions, and running a script — and a little pandas: reading a CSV file, selecting columns, and grouping. No design background or statistics beyond averages is assumed.

What You Will Learn

- How to choose the right chart for a question, and how human perception decides which charts are easy to read.

- How to keep charts honest: baselines, scales, aspect ratios, dual axes, normalization, smoothing, and missing data.
- Matplotlib from the inside: figures, axes, and artists, the object-oriented API, line, bar, scatter, histogram, and box plots, annotations, and layouts.
- Color and accessibility: palettes and colormaps, highlighting, color-vision deficiency, contrast ratios, text size, and alt text.
- A reusable house style with rcParams and style sheets, small multiples, and storytelling with a sequence of charts.
- Time series, heatmaps and calendars, map-free geographic layouts, stacked bars, waterfalls, slope charts, dumbbells, and Likert charts.
- Statistical graphics with seaborn: distributions, relationships, regression, uncertainty, and faceting.
- Interactive charts with Plotly Express and graph objects, and self-contained HTML dashboards that need no server.
- Exporting for print and web, reproducible chart scripts, tests, and an automated monthly chart pack.

How This Book Is Organized

Part I — Principles of Clear Charts (Chapters 1–5) explains why we visualize, sets up a reproducible workspace, and covers chart choice, perception, and honesty. **Part II — Matplotlib Fundamentals** (Chapters 6–10) teaches the object model and the core chart types, labels and annotations, and multi-panel layouts. **Part III — Design, Color, and Accessibility** (Chapters 11–15) covers color, accessible charts, a house style, small multiples, and storytelling. **Part IV — Time, Place, and Parts of a Whole** (Chapters 16–19) covers time series, heatmaps and calendars, geography without GIS software, and charts of parts, ranks, and change. **Part V — Statistical Graphics with seaborn** (Chapters 20–23) covers seaborn's model, distributions, relationships, and uncertainty. **Part VI — Interactive Charts and Production** (Chapters 24–28) covers Plotly, HTML dashboards, exporting, reproducibility, and a complete automated chart pipeline. The appendices provide a cheat sheet, full reference listings, a chart review checklist, and a glossary.

Each chapter ends with exercises and a **Key Takeaways** box for quick review.

The Running Example: Meadowlark Energy Co-op

Throughout the book we chart the data of **Meadowlark Energy Co-op**, a fictional community energy cooperative in a small Midwest town. Its 240 member households live in four districts — Riverside, Hillcrest, Old Town, and Meadow Park — and pay for electricity on one of three tariffs: Flat, Time-of-Use, or Solar Share. Some members have rooftop solar panels, batteries, or electric vehicles, and the co-op owns three community solar arrays: Barn Roof, North Field, and School Canopy. We

have two full years of data, 2024 and 2025: monthly bills and energy for every member, daily weather and demand, hourly load and solar output, a member survey, and a few events worth annotating, such as a new array coming online and a week-long inverter fault.

The data is realistic enough to raise real questions — Is Solar Share good value? When does the co-op need the most power? Which district's bills grew fastest? — and small enough to check. It is generated by a short, seeded Python program, so you get exactly the same numbers and charts as this book. Chapter by chapter you build a small package, `meadowlark`, with data loaders, a house style, reusable chart functions, interactive figures, an HTML dashboard, tests, and finally a one-command monthly chart pack.

Versions and Conventions

Examples use **Python 3.12** and were tested with Matplotlib 3.11, seaborn 0.13, Plotly 7, pandas 3, and NumPy 2. Most of the code also runs on somewhat older releases; where a behavior is newer or changed recently, a Note says so. Commands are shown for Windows (`py -3.12 -m scripts.ch07_bill_bars`); on macOS and Linux use `python3` instead of `py -3.12`. Scripts render charts to files with Matplotlib's non-interactive Agg backend, which works the same on every system, including servers. Dates and numbers in charts are formatted in US English (`Jul 8, 2025`, `$1,234`) regardless of your computer's regional settings.

> **TIP**
> A recommended practice or a technique that makes your work easier.

> **NOTE**
> Background information, version differences, or a useful aside.

> **WARNING**
> A common mistake or something that can mislead readers, break a chart, or lose work.

Code appears in `monospace` with the file it belongs to shown above it, such as `meadowlark/charts/monthly.py`. Program output appears in a separate block labeled **Output** and is reproduced exactly as the code printed it. Long lines are wrapped to fit the page. Figures are numbered by chapter (Figure 7.1, 7.2, ...) and every figure was produced by the code printed next to it.

Contents

Preface

PART I — PRINCIPLES OF CLEAR CHARTS

1 Why Visualize: Charts as Honest Arguments

2 A Reproducible Chart Workspace: Python 3.12, Libraries, and the Meadowlark Project

3 Choosing the Right Chart: From Question to Form

4 How People Read Charts: Perception, Encodings, and Visual Hierarchy

5 Honest Charts: Baselines, Scales, Aspect Ratios, and Misleading Patterns

PART II — MATPLOTLIB FUNDAMENTALS

6 Inside Matplotlib: Figures, Axes, Artists, and the Object-Oriented API

7 Line and Bar Charts Done Right

8 Scatter Plots, Histograms, and Box Plots

9 Titles, Labels, Annotations, and Reference Lines

10 Subplots and Layouts: Grids, Mosaics, and Constrained Layout

PART III — DESIGN, COLOR, AND ACCESSIBILITY

11 Color with Purpose: Palettes, Colormaps, and Highlighting

12 Accessible Charts: Color Vision, Contrast, Text, and Alt Text

13 Styles and Themes: rcParams, Style Sheets, and a House Style

14 Small Multiples: Comparing Many Groups Without Spaghetti

15 Storytelling with Charts: Message Titles, Emphasis, and Chart Sequences

PART IV — TIME, PLACE, AND PARTS OF A WHOLE

16 Time Series Charts: Dates, Resampling, Rolling Windows, and Gaps

17 Heatmaps and Calendars: Hour-by-Day and Day-by-Month Patterns

18 Geography Without GIS: Tile Maps, Schematic Maps, and Bubble Layouts

19 Parts, Ranks, and Change: Stacked Bars, Waterfalls, Slopes, Dumbbells, and Likert Charts

PART V — STATISTICAL GRAPHICS WITH SEABORN

20 seaborn Essentials: Tidy Data, Figure-Level and Axes-Level Functions, and Themes

21 Distributions: Histograms, KDEs, ECDFs, Violins, and Strip Plots

22 Relationships: Scatter Plots, Regression Lines, Pair Plots, and Correlation Heatmaps

23 Categories and Uncertainty: Error Bars, Confidence Intervals, and Faceted Comparisons

PART VI — INTERACTIVE CHARTS AND PRODUCTION

24 Interactive Charts with Plotly Express

25 Plotly Graph Objects: Traces, Layouts, Hover, Subplots, and Controls

26 Dashboards-Lite: Self-Contained HTML Reports

27 Exporting for Print and Web: Formats, Sizes, Fonts, and Reproducible Scripts

28 Capstone: Automating the Meadowlark Monthly Chart Pack

APPENDICES

A Chart and API Cheat Sheet

B Templates and Reference Listings

C Chart Review and Production Checklist

D Glossary

PART I

Principles of Clear Charts

Why we visualize, a reproducible workspace for chart code, choosing the right chart for a question, how people perceive position, length, area, and color, and the rules that keep charts honest.

1 · Why Visualize
2 · A Reproducible Chart Workspace
3 · Choosing the Right Chart
4 · How People Read Charts
5 · Honest Charts

CHAPTER 1 · PART I — PRINCIPLES OF CLEAR CHARTS

Why Visualize: Charts as Honest Arguments

A chart is not decoration on top of a report; it is a claim backed by evidence, drawn so the reader can check it. In this chapter we look at what charts are for, meet the Meadowlark Energy Co-op whose data runs through the whole book, and draw our first chart in about ten lines of Python.

A Table of 24 Numbers and the Picture Inside It

Imagine you sit on the board of a small community energy cooperative. Every month the staff send you a spreadsheet. One sheet lists, for each of the last 24 months, how much electricity the members used and how much their rooftop solar panels produced. Here is that table exactly as Python prints it from the book's data. Do not study it closely yet; just notice how it feels to read it.

Output

month	consumption_kwh	solar_kwh	members
2024-01-01	189288.7	35089.9	216
2024-02-01	175255.2	45131.8	216
2024-03-01	153547.7	64547.1	217
2024-04-01	134296.1	77628.9	219
2024-05-01	129407.7	88413.9	219
2024-06-01	150476.0	93806.6	220
2024-07-01	176483.1	96815.2	221
2024-08-01	173512.5	88050.5	221
2024-09-01	141770.7	73640.2	221
2024-10-01	134077.2	56755.6	223
2024-11-01	157479.1	38603.7	223
2024-12-01	191044.7	32964.4	226
2025-01-01	199648.4	39702.5	227
2025-02-01	184314.4	50540.8	227
2025-03-01	159871.2	70456.1	227
2025-04-01	139740.2	85591.4	228
2025-05-01	134276.2	97360.3	230
2025-06-01	156287.2	105458.0	231
2025-07-01	182846.7	106539.3	231
2025-08-01	181118.6	97448.3	231
2025-09-01	148936.7	80602.4	233
2025-10-01	141041.7	64174.9	235
2025-11-01	167611.9	44278.0	236
2025-12-01	204036.1	37367.1	238

Everything you might want to know is in there. Which month used the most energy? When does solar peak? Is consumption rising from one year to the next? You can answer each question, but only by scanning down the column, holding numbers in your head, and comparing six-digit values one pair at a time. Reading a table is serial work: one cell, then the next, then the next.

Now look at the same two columns drawn as lines in Figure 1.1.

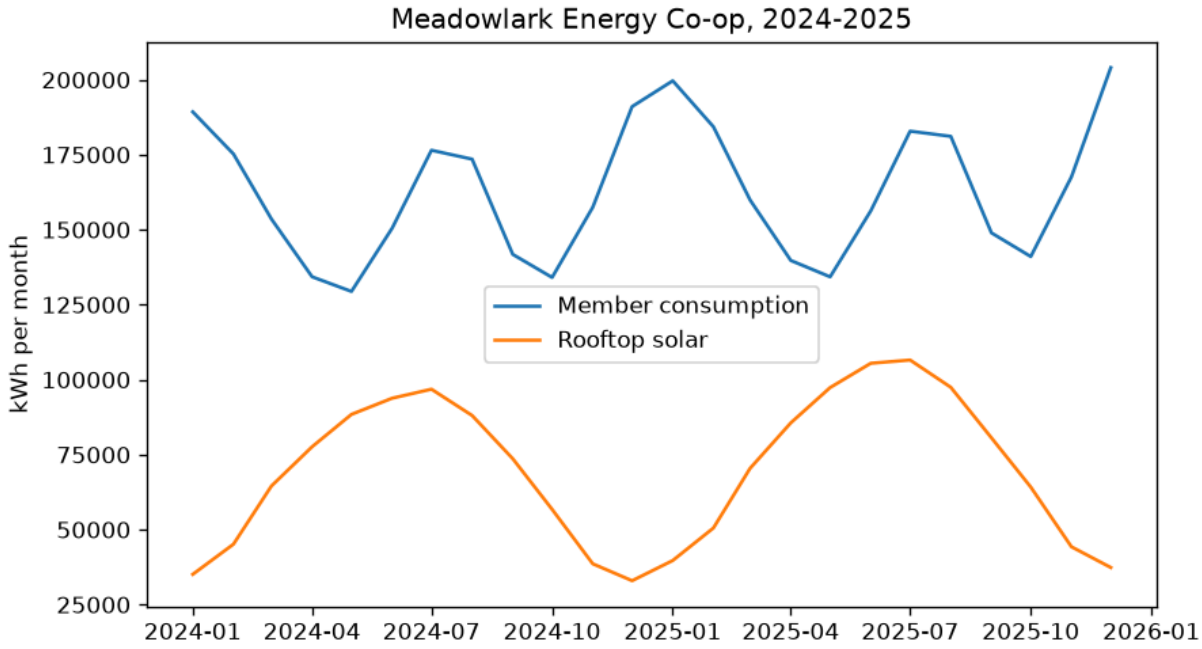


Figure 1.1 — The same 24 months as a default Matplotlib line chart: seasonality is visible at a glance.

In less than a second you see things the table hid. Consumption has two peaks a year, a big one in winter and a smaller one in midsummer, with valleys in spring and autumn. Rooftop solar rises and falls once a year, peaking in July. The two lines never cross: in every month, members used more electricity than their rooftops made. And the second winter peak is a little higher than the first. None of these observations needed arithmetic. Your visual system did the comparisons in parallel, because position, slope, and shape are things human eyes read almost instantly.

That is the core reason we visualize data: a good chart moves work from slow, conscious reading to fast, automatic seeing.

Figure 1.1 is also far from perfect. The y-axis starts near 25,000 instead of zero, so the gap between the lines looks smaller than it is. The tick labels are raw numbers without thousands separators. The legend sits in the middle of the data. The title names the data set but says nothing about what it shows. The x-axis ends with a tick labeled for a month that has no data. We will fix all of that before the chapter ends, and the rest of the book teaches why each fix matters.

> **NOTE**
>
> Every number in this book comes from the book's own data and code, and every output block is pasted from a real run. When you run the same code on the same data you get the same numbers, down to the last decimal. If yours differ, check your library versions against Chapter 2 before you check anything else.

What Charts Are For: Exploring and Explaining

People make charts for two quite different reasons, and confusing them is the source of many bad charts.

Exploration is when *you* are the audience. You have data and questions, and you do not know the answers yet. You draw many quick charts, look at them for a few seconds, throw most of them away, and slowly build an understanding. Exploratory charts can be ugly. They can use default colors, unlabeled axes, and cramped tick labels, because you know what is on them. What matters is speed: the time from "I wonder whether..." to a picture that answers the question should be as short as possible.

Explanation is when *someone else* is the audience. You already know what the data says, and your job is to show it to a reader who has less time, less context, and less patience than you do. Explanatory charts must stand on their own: a clear title, labeled axes with units, a source line, readable text at the size it will be printed, and nothing that distracts from the point.

Both modes are legitimate. The mistake is to send an exploratory draft to a reader who needs an explanation, or to spend an hour polishing a chart that only you will ever see.

> **TIP**
>
> Before you start polishing a chart, ask one question: "Who will look at this, and what should they know after ten seconds?" If the answer is "only me, right now," stop polishing. If the answer names a reader, write the takeaway in one sentence before you write any more code. That sentence often becomes the chart's title.

A Chart Is an Argument

When a chart is meant for someone else, it makes an argument. It says "this is how things are," and it offers the plotted data as evidence. That framing is useful because it gives you a test for every design decision: does this element help the reader evaluate the claim, or does it get in the way?

An argument has three parts, and a good explanatory chart has all three:

1. **The claim.** What the chart wants you to believe. Careful chart makers often put the claim directly into the title: "Members use more energy than their rooftops make, every month" instead of "Monthly energy, 2024–2025." We call these *message titles*, and Chapter 9 and Chapter 15 work

on them in detail.

2. **The evidence.** The data, drawn with encodings that let the reader judge the claim accurately. For the claim above, two lines on the same axis with a zero baseline let the reader see the gap between them in every month.
3. **The warrant.** Why the evidence supports the claim: the axis labels and units, the source, the sample size, notes about what was excluded or smoothed. Without these, the reader has to take the chart on trust.

Thinking in terms of argument also explains why a chart can be dishonest without containing a single wrong number. Every value can be correct while the picture misleads: a bar chart whose y-axis starts at $100 can make a 3.5 percent increase look like a 3.5-fold jump; a chart that shows only April to July can hide the winter; a dual y-axis can make two unrelated series appear to move together. In each case the data is real but the argument is rigged. Chapter 5 catalogs these patterns with Meadowlark examples, and the honesty rules from that chapter apply to every chart in the book.

The flip side is also true. An honest chart lets a skeptical reader *disagree* with you. If the claim is "Solar Share members pay less," an honest chart shows the bills of all three tariffs on the same scale, says how many members are in each group, and mentions that Solar Share members are more likely to have rooftop panels of their own. A reader can then decide whether the tariff, the panels, or something else explains the difference. A chart that cannot be argued with is usually a chart that hides something.

> **WARNING**
> "The numbers are correct" is not the same as "the chart is honest." Baselines, scales, date ranges, aggregation, and color choices all change what a reader concludes. When you review a chart, read the picture as a skeptical stranger would, not as the person who wrote the query.

Same Summary, Different Pictures

There is a second reason to visualize, and it matters most in exploration: summary statistics can hide the shape of data. A mean or a correlation coefficient compresses hundreds of rows into one number, and sometimes the part it throws away is the part that matters.

Statisticians have known this for a long time. In 1973 Francis Anscombe published four small data sets, now called **Anscombe's quartet**, that share almost the same means, variances, correlation, and fitted regression line, yet look completely different when plotted: one is a reasonable straight-line cloud, one is a clean curve, one is a straight line with a single outlier, and one is a vertical stack of points plus one far-away point.

In 2017 Justin Matejka and George Fitzmaurice made the same point more playfully with the **Datasaurus Dozen**: a set of scatter plots, including one shaped like a dinosaur, that all share nearly identical summary statistics.

We do not reproduce either data set in this book. You do not need them, because the Meadowlark data contains its own version of the lesson. The correlation between daily average temperature and daily member demand over 2024–2025 is −0.317, a weak negative number that seems to say "warmer days use slightly less energy." But a scatter plot shows a U shape: demand is high on the coldest days (heating), low on mild days, and high again on the hottest days (cooling). A single coefficient that measures straight-line association cannot describe a U. Chapter 22 draws that scatter plot and explains it in detail; for now, keep the rule in mind: *look at the picture before you believe the summary*.

Meet the Meadowlark Energy Co-op

Every chart in this book uses one running example: the **Meadowlark Energy Co-op**. It is fictional, and so is the town of Meadowlark, a small place with a Midwest US climate: cold winters, hot summers, and temperatures in degrees Fahrenheit. The co-op's data is synthetic, generated by a short Python script (`make_data.py`) with a fixed random seed so that every reader gets exactly the same rows. Chapter 2 runs that script and walks through the files.

Members and homes

The co-op has 240 member households across four districts: Riverside, Hillcrest, Old Town, and Meadow Park. Homes are apartments, townhouses, or detached houses. 114 members have rooftop solar panels (723.5 kW installed in total), 35 have home batteries, and 38 have electric vehicles. Most members joined years ago, but 24 joined during 2024 and 2025, so the number of active members grows from 216 in January 2024 to 238 in December 2025. That growth matters: when total consumption rises, part of the rise is simply more members, and an honest chart says so.

Tariffs

Each member is on one of three tariffs, the pricing plans that decide their monthly bill:

Tariff	Members	How it charges (2025 rates)
Flat	100	$12.00 a month plus $0.180 per kWh
Time-of-Use	83	$12.00 a month; $0.120 off-peak and $0.280 at peak
Solar Share	57	$15.00 a month plus $0.165 per kWh; higher export credit

Members with panels earn a credit for the energy they export to the grid, and a bill never drops below the fixed monthly charge. Rates rose at the start of 2025. These rules produce realistic quirks that charts will reveal later, such as a cluster of bills sitting exactly at $12 or $15.

Community solar and events

Besides the panels on members' roofs, the co-op owns three community solar arrays: **Barn Roof** (120 kW), **North Field** (450 kW), and **School Canopy** (200 kW), which came online on September 16, 2024. In July 2025 an inverter fault took North Field offline for seven days, and those days are genuinely missing from the data. A short events file records these dates along with a battery rebate (April 1, 2024) and the 2025 tariff change, so that charts can annotate them.

The data files

The data comes as nine CSV files, from member records and monthly bills to hourly load and a member survey. Chapter 2 lists every file and column.

The questions the board asks

The Meadowlark board asks the kind of questions any small organization asks of its numbers, and the book's charts answer them:

- How does energy use change through the year, and is it growing? (Chapters 1, 7, and 16)
- What does a typical member pay, and how much do bills vary? (Chapters 8 and 21)
- Is the Solar Share tariff good value, or do its members simply have more panels? (Chapters 15 and 23)
- How are the community arrays performing, and what did the North Field outage cost? (Chapters 16 and 25)
- When during the day and year is demand highest? (Chapter 17)
- How do the four districts compare? (Chapters 14, 18, and 19)
- What do members think of the co-op? (Chapter 19)
- Can the monthly chart pack for the board be produced with one command? (Chapters 26–28)

Notice that every question names a comparison, a change, a distribution, or a relationship. Chapter 3 turns that observation into a method for picking a chart form from the question.

> **NOTE**
> Meadowlark, its members, and all of its numbers are invented for teaching. The patterns are designed to be plausible for a community energy co-op in a cold-winter climate, but nothing in the data describes a real organization, and you should not use the numbers to draw conclusions about real energy systems.

Three Libraries and When Each Fits

Python has many plotting libraries. This book teaches three in depth, because together they cover almost everything an analyst or developer needs, and because they build on each other.

Library	Best at	Output	In this book
Matplotlib	Full control of every element; print-quality static charts	PNG, SVG, PDF	Chapters 6–19
seaborn	Statistical graphics from tidy data with little code	Matplotlib figures	Chapters 20–23
Plotly	Interactive charts with hover, zoom, and toggles	HTML files	Chapters 24–26

Matplotlib is the foundation. It is Python's long-established plotting library, seaborn is built on it, and it gives you control over every line, label, and pixel at the price of more code. We teach it first because design principles are easiest to see when you set each one explicitly, and because you will finish seaborn charts with Matplotlib calls.

seaborn sits on top of Matplotlib and specializes in statistical graphics. Given a table in long ("tidy") format, one function call can draw a histogram per group, a box plot with every category in the right order, or a grid of scatter plots with regression lines and confidence bands. Its results are ordinary Matplotlib figures.

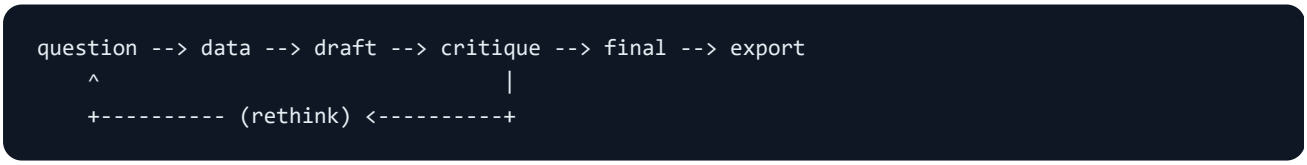
Plotly is different. It draws charts in a web browser using JavaScript, so readers can hover over points to see exact values, zoom into a date range, and click legend entries to hide series. A Plotly chart is usually saved as an HTML file. Hover does not exist in print, and interactivity has accessibility limits we discuss in Chapter 12.

A simple rule: use Matplotlib when the chart is going into a document, slide, or PDF and you care about every detail; use seaborn when you are exploring statistical structure or need a quick comparison across groups; use Plotly when the reader will look at the chart on a screen and benefits from interacting with it.

The book uses these versions: Python 3.12 or newer, Matplotlib 3.11 (examples tested with 3.11.2), seaborn 0.13, Plotly 7 (tested with 7.1), and pandas 3 for loading and reshaping data. Chapter 2 shows how to install exactly those versions in an isolated environment so that your output matches ours.

The Workflow: From Question to Exported Chart

Good charts rarely come out right on the first try. Across the book we follow the same six-step workflow, and it is worth seeing it once before we use it:



1. **Question.** Write the question in plain words: "Do members use more energy than their rooftops make?" A vague question ("show me the energy data") produces a vague chart.

2. **Data.** Load and shape exactly the data the question needs. For our question, that is one row per month with total consumption and total rooftop solar. Most chart problems that look like plotting problems are really data-shaping problems.
3. **Draft.** Draw the simplest chart that could answer the question, usually with the library's defaults. Do not polish yet.
4. **Critique.** Look at the draft as a stranger would. Does it answer the question? What is the first thing your eye lands on, and is that the important thing? Is anything misleading? Sometimes the critique sends you back to the question.
5. **Final.** Fix what the critique found: baselines, labels, units, title, color, emphasis. Remove what does not help.
6. **Export.** Save the chart in the right format and size for where it will appear, from a script that you can run again next month and get the same result.

The last step deserves emphasis. A chart made by clicking and pasting is a dead end: next month someone has to click again, and nobody can check how it was made. A chart made by a script can be rerun on new data, reviewed, and tested. Chapters 27 and 28 turn that idea into an automated monthly chart pack.

Your First Chart in About Ten Lines

Let us draw Figure 1.1 ourselves. If you have not set up the project yet, read this section now and run it after Chapter 2; nothing in the code will surprise you then. The book's project lives in a folder called `meadowlark-charts`, and every script runs from that folder. Here is the whole script:

scripts/ch01_first_chart.py

```
"""Chapter 1: a first chart in about ten lines."""
from pathlib import Path

import matplotlib
matplotlib.use("Agg")  # draw to files, never open a window
import matplotlib.pyplot as plt

from meadowlark.data import coop_monthly

df = coop_monthly()
fig, ax = plt.subplots(figsize=(8, 4.5))
ax.plot(df.index, df["consumption_kwh"], label="Member consumption")
ax.plot(df.index, df["solar_kwh"], label="Rooftop solar")
ax.set(title="Meadowlark Energy Co-op, 2024-2025", ylabel="kWh per month")
ax.legend()
Path("out").mkdir(exist_ok=True)
fig.savefig("out/ch01-first-chart.png", dpi=150)
print("saved out/ch01-first-chart.png")
plt.close(fig)
```

Run it as a module from the project folder. On Windows the command is shown below; on macOS and Linux use `python3 -m scripts.ch01_first_chart` instead (inside an activated virtual environment, plain `python` works everywhere).

Terminal

```
py -3.12 -m scripts.ch01_first_chart
```

Output

```
saved out/ch01-first-chart.png
```

Open `out/ch01-first-chart.png` and you will see Figure 1.1. Let us read the script line by line, because almost every chart in the book follows the same shape.

- `matplotlib.use("Agg")` tells Matplotlib to draw into memory and files instead of opening a window. It must come before `import matplotlib.pyplot`. Chapter 2 explains backends; for now, know that every script in this book starts this way so it behaves identically on a laptop, a server, or a scheduled job.
- `coop_monthly()` comes from the project's own `meadowlark/data.py` module. It reads the monthly member records and returns one row per month with co-op totals. The table at the start of this chapter shows three of its columns.
- `fig, ax = plt.subplots(figsize=(8, 4.5))` creates a *Figure* (the whole image, 8 by 4.5 inches) and one *Axes* (the plotting area inside it). We draw on the Axes by calling its methods. This is Matplotlib's object-oriented style, which Chapter 6 explains and the whole book uses.
- Two calls to `ax.plot` draw two lines. Matplotlib picks the colors automatically from its default color cycle: blue first, orange second.
- `ax.set(...)` sets several properties at once, here the title and the y-axis label. `ax.legend()` builds a legend from the `label=` arguments.
- `fig.savefig(...)` writes the PNG. With `dpi=150` an 8-inch-wide figure becomes 1,200 pixels wide. `plt.close(fig)` frees the memory, which matters when a script draws dozens of charts in a loop.

That is the entire pattern: load data, create a figure and axes, draw, label, save, close. Everything else in the Matplotlib part of this book is a refinement of one of those steps.

> **TIP**
>
> Keep scripts that produce charts short and boring. When a script grows past a screen, move the drawing code into a function in the `meadowlark` package (we start doing that in Chapter 6) and keep the script as a thin wrapper that calls the function and saves the result.

Critique: From a Weak Chart to an Argument

Defaults are a starting point. To see how much design matters, we will look at two more versions of the same data: one deliberately weak, and one built as an argument. First the weak one. Every choice in it is something we have seen in real reports.

scripts/ch01_weak_chart.py

```
"""Chapter 1: a deliberately weak chart of the same monthly data."""
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import numpy as np

from meadowlark.data import coop_monthly

df = coop_monthly()
x = np.arange(len(df))
fig, ax = plt.subplots(figsize=(8, 4.5), layout="constrained")
ax.bar(x - 0.2, df["consumption_kwh"], width=0.4, label="consumption_kwh")
ax.bar(x + 0.2, df["solar_kwh"], width=0.4, label="solar_kwh")
ax.set_xticks(x, [str(m) for m in df.index], rotation=90)
ax.set_ylim(30_000, 210_000)        # truncated: bars no longer start at zero
ax.grid(True, linewidth=1.2)
ax.legend(loc="center", fontsize=12)
ax.set_title("Monthly data")
fig.savefig("out/ch01-weak-chart.png", dpi=150, bbox_inches="tight")
plt.close(fig)
```

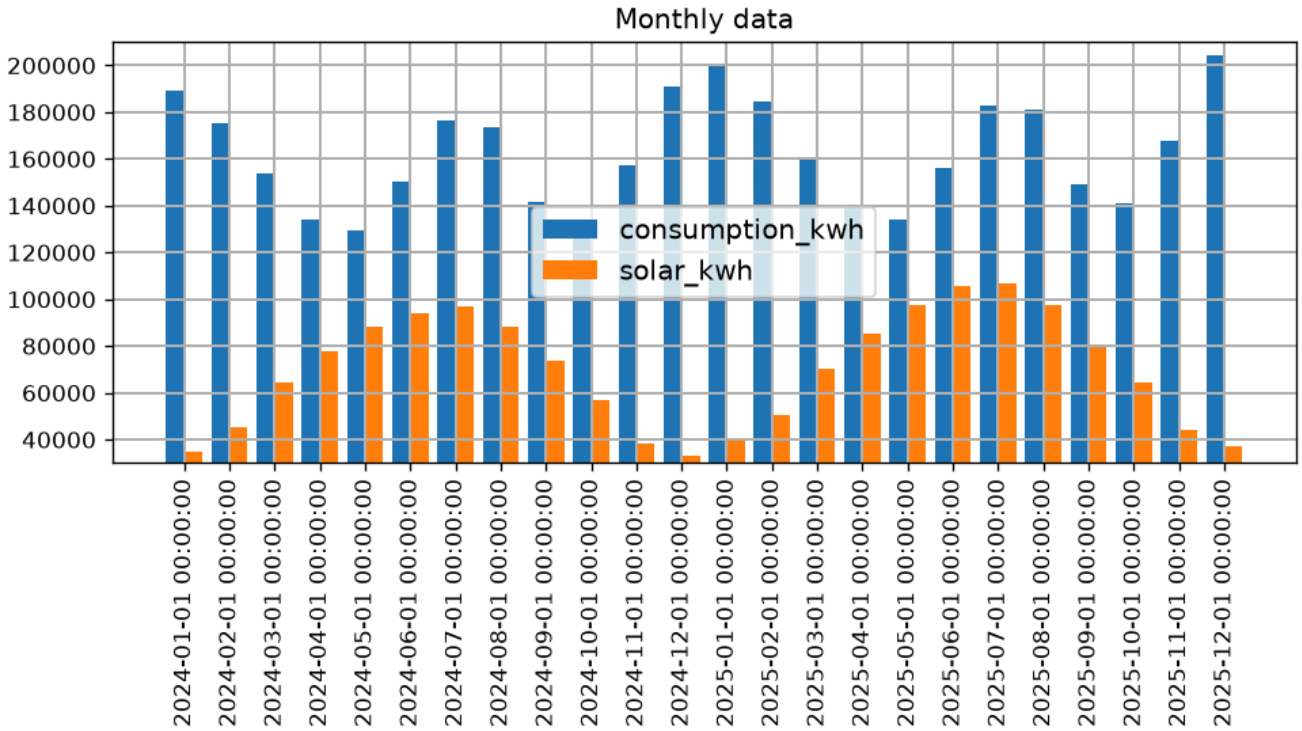


Figure 1.2 — A deliberately weak version: truncated bars, timestamp labels, heavy grid, and a legend on top of the data.

Figure 1.2 is not wrong in its numbers, but it fails the reader in at least six ways:

1. **The y-axis starts at 30,000.** A bar's length is its message. When the baseline is not zero, the December 2024 solar bar is less than a fiftieth the height of the consumption bar beside it, while the real ratio is about one sixth. Bars must start at zero; Chapter 5 shows how badly this can mislead.
2. **Forty-eight bars for a time series.** Bars make you compare individual months. The question was about the pattern through the year, which a line shows with far less ink.
3. **Timestamp labels.** "2024-01-01 00:00:00", rotated ninety degrees, forces the reader to tilt their head to read a time of day that carries no information.
4. **Column names as labels.** `consumption_kwh` is a name for code, not for people. There is no unit on the y-axis, and the numbers have no thousands separators.
5. **Heavy grid and a legend on the data.** The thick grid lines compete with the bars, and the legend hides several of them.
6. **A title that says nothing.** "Monthly data" names the topic, not the finding.

Now the version built as an argument. It uses two small helpers from the project's style module, `month_axis` and `mwh_formatter`, which write English month names and convert kWh to MWh on the axis. We build both in Chapter 6; here we simply use them.

scripts/ch01_better_chart.py

```
"""Chapter 1: the same data, redrawn as an argument."""
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

from meadowlark.data import coop_monthly
from meadowlark.style import month_axis, mwh_formatter  # explained in Chapter 6

df = coop_monthly()
fig, ax = plt.subplots(figsize=(8, 4.5), layout="constrained")
ax.plot(df.index, df["consumption_kwh"], color="#0072B2", linewidth=2.2)
ax.plot(df.index, df["solar_kwh"], color="#E69F00", linewidth=2.2)
ax.set_ylim(0, 230_000)                       # zero baseline
ax.margins(x=0.01)                            # no empty Jan 2026 tick
ax.yaxis.set_major_formatter(mwh_formatter)   # 200,000 kWh -> "200"
ax.set_ylabel("MWh per month")
month_axis(ax, interval=3)
ax.spines[["top", "right"]].set_visible(False)
ax.grid(axis="y", color="#E6E6E6")
ax.set_axisbelow(True)
ax.text(df.index[12], 212_000, "Member consumption", color="#0072B2",
        ha="center", fontweight="bold")
ax.text(df.index[18], 117_000, "Rooftop solar", color="#222222",
        ha="center", fontweight="bold")
ax.set_title("Members use more energy than their rooftops make, every month",
             loc="left", fontweight="bold")
fig.savefig("out/ch01-better-chart.png", dpi=150, bbox_inches="tight")
plt.close(fig)
```