Python Visualization Basics

From notebooks → web-ready charts: SVG/HTML outputs, interactive graphics, and modern chart components.

Marc Reyes
Professional Lecturer · marc.reyes@dlsu.edu.ph
DATA101 — De La Salle University

Today's Plan

01 · WEB OUTPUTS

HTML, CSS, SVG

What the browser actually renders.

02 · PYTHON WORKFLOW

Notebook → chart → export

Reproducible visuals you can ship.

03 · MODERN COMPONENTS

Scales, marks, guides

A reusable mental model.

04 · INTERACTIVITY

Selections + tooltips

When interactivity is worth it.

Learning Outcomes

WEB

Explain SVG vs PNG vs HTML

And choose the right export format.

PYTHON

Build a reproducible chart workflow

Data → transforms → plot → export.

DESIGN

Name modern chart components

Scales, axes, marks, guides, interactions.

PRACTICE

Write code that produces legible outputs

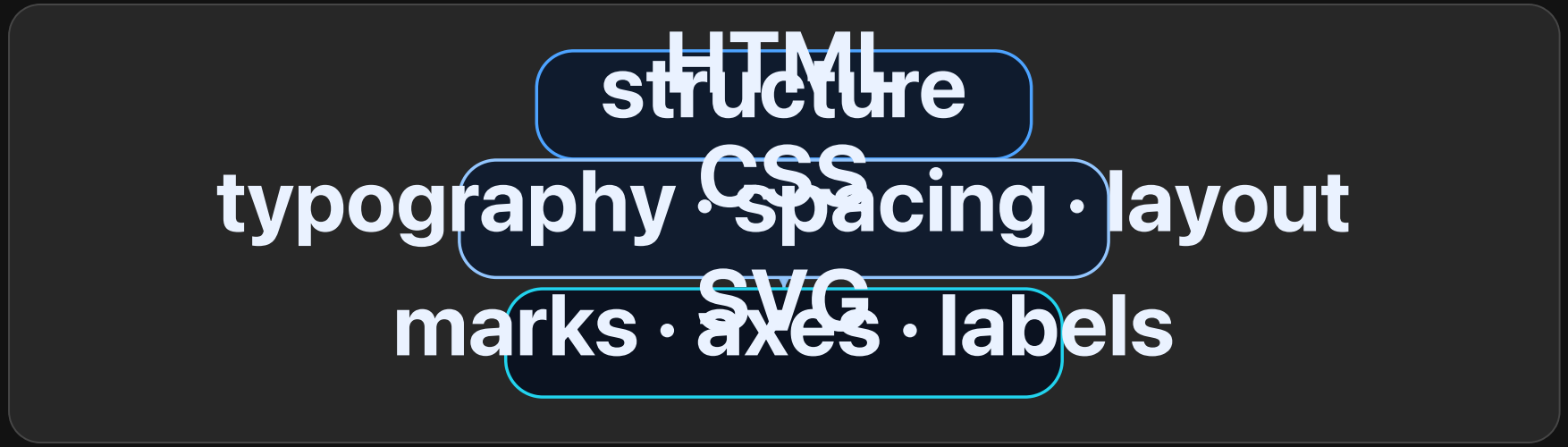
Readable on slides, PDFs, and web pages.

PART 1 · WEB OUTPUTS

What the browser understands

HTML + CSS + SVG as the delivery layer for charts

The Web Output Stack (for Charts)



SVG: The “Native” Format of Many Charts

WHY SVG

- Scales crisply (great for slides and PDFs)
- Text remains selectable and searchable
- Shapes are editable (Illustrator/Figma)

WHEN NOT SVG

- Huge point clouds (file size)
- Complex maps with many paths
- Photographic backgrounds

CSS: The Hidden Part of “Professional”

CSS CONTROLS

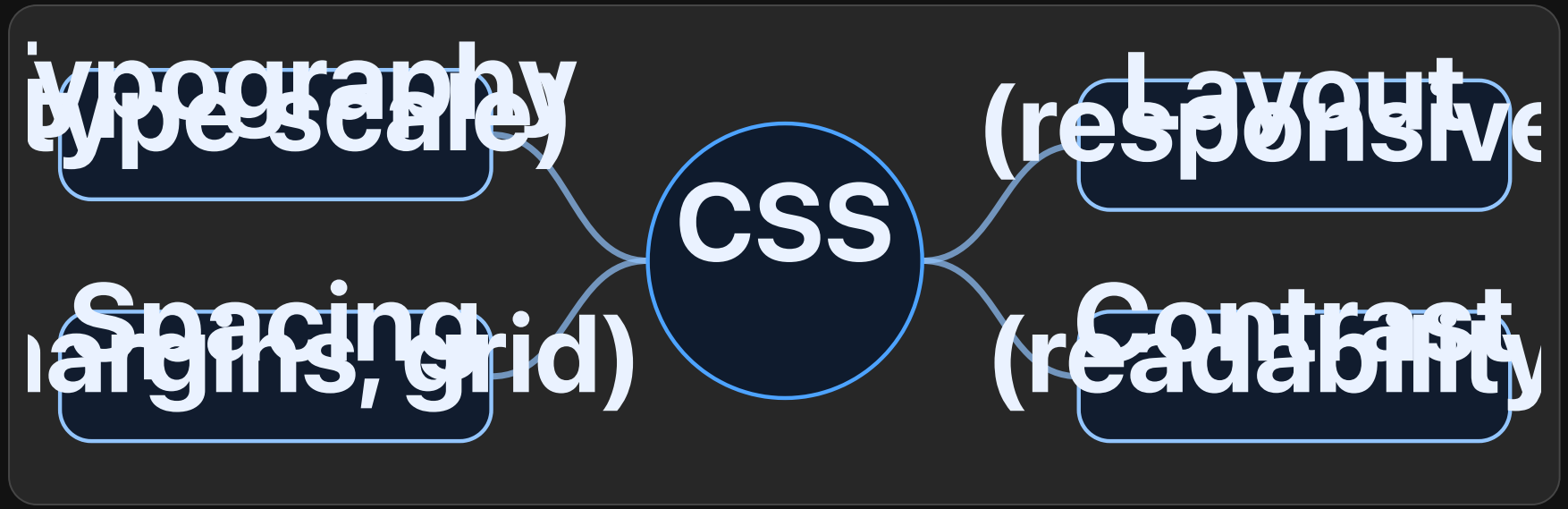
Hierarchy + spacing + readability

Font sizes, line heights, margins, contrast.

CHART IMPLICATION

Your chart lives inside a layout

So it must be responsive and legible.



PART 2 · PYTHON WORKFLOW

Notebook → chart → export

Reproducible visuals you can ship

A Repeatable Python Visualization Pipeline

Pandas as the “Chart Data Engine”

COMMON TRANSFORMS

- `groupby` + aggregate
- derive rates and deltas
- `melt` / tidy reshape
- sort for ranking

WHY IT MATTERS

Most chart bugs are data bugs

Wrong denominator, wrong unit, wrong grain.

Code Demo A: Make a Chart-Ready Table

RUNNABLE PYTHON

Copy into a notebook and run.

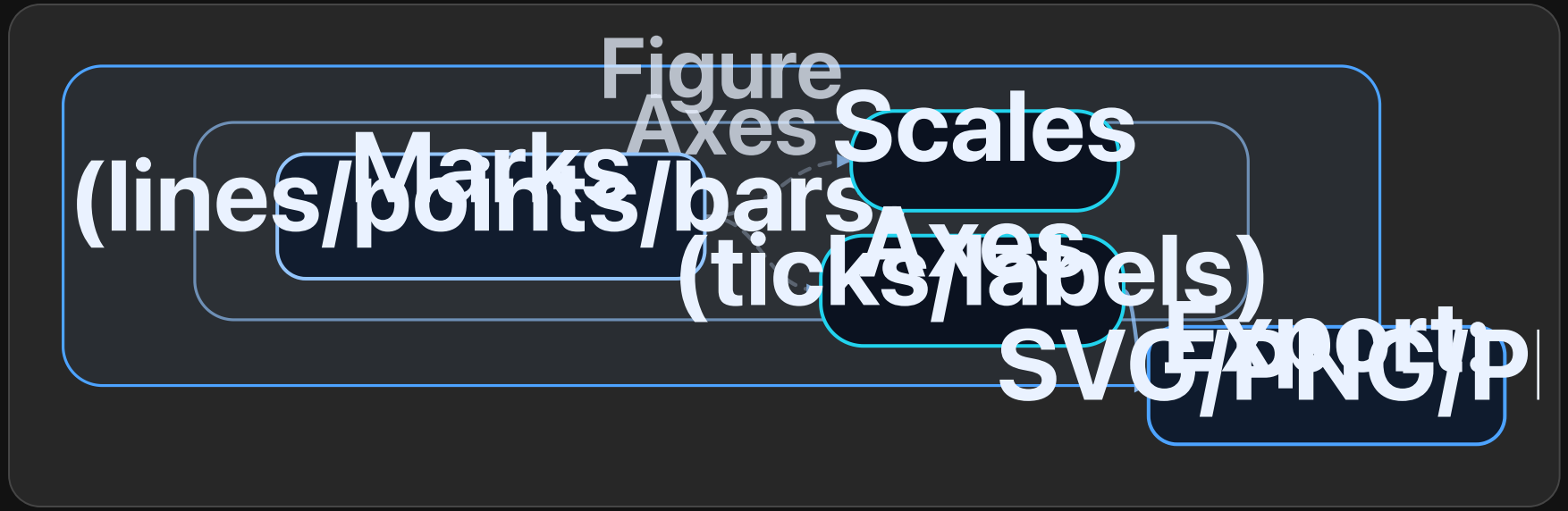
```python
import pandas as pd
df = pd.DataFrame(
    {
        "program": ["A", "A", "B", "B", "C", "C"],
        "week": [1, 2, 1, 2, 1, 2],
        "n_pass": [70, 62, 90, 88, 40, 44],
        "n_students": [100, 100, 120, 120, 50, 50],
    }
)

df["pass_rate"] = df["n_pass"] / df["n_students"]
weekly = (
    df.sort_values(["program", "week"])
    .assign(delta=lambda d: d.groupby("program")["pass_rate
)
weekly
```

OUTPUT YOU SHOULD SEE

A tidy table with a derived rate + delta.

program	week	n_pass	n_students	pass_rate	delta
A	1	70	100	0.7	—
A	2	62	100	0.62	-0.08
B	1	90	120	0.75	—
B	2	88	120	0.733	-0.017
C	1	40	50	0.8	—
C	2	44	50	0.88	0.08

Matplotlib: The “Artist” Model (Mental Map)



Code Demo B: Matplotlib → SVG Export

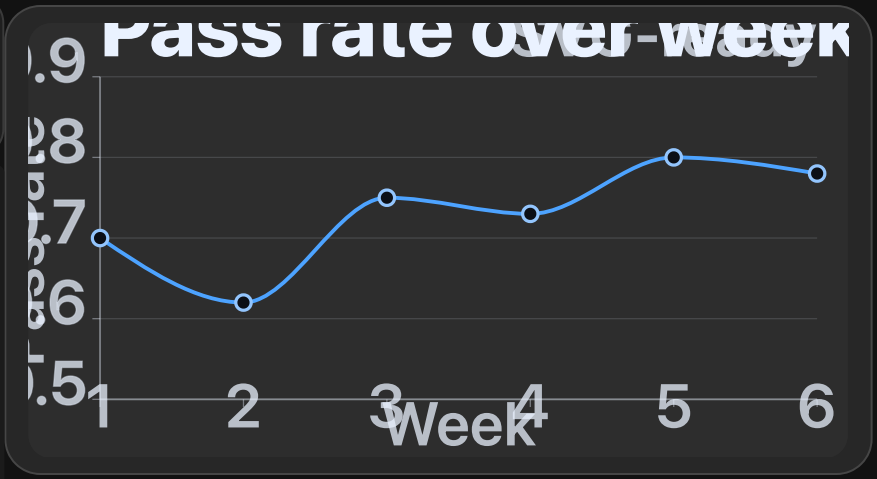
RUNNABLE PYTHON

Produces a line chart and saves an SVG.

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

x = np.arange(1, 7)
y = np.array([0.70, 0.62, 0.75, 0.73, 0.80, 0.78])

fig, ax = plt.subplots(figsize=(7.2, 3.6))
ax.plot(x, y, marker="o", linewidth=2)
ax.set_title("Pass rate over weeks")
ax.set_xlabel("Week")
ax.set_ylabel("Pass rate")
ax.set_ylim(0.5, 0.9)
ax.grid(True, alpha=0.25)

fig.tight_layout()
fig.savefig("pass_rate.svg")
```

RENDERED RESULT (ILLUSTRATION)

Your SVG will scale crisply in slides.



Seaborn: Statistics Defaults + Cleaner Styles

WHAT IT ADDS

- Statistical plots (distributions, categories)
- Reasonable default aesthetics
- Easy small multiples (`FacetGrid`)

COMMON TRAP

Pretty defaults ≠ correct story

Always check units, bins, and baselines.

Code Demo C: Faceted Histograms

RUNNABLE PYTHON

Shows why distribution beats averages.

```python
import numpy as np, pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

rng=np.random.default_rng(7)
program=np.repeat(list("ABC"), 250)
score=np.r_[rng.normal(76, 6, 250), rng.normal(76, 14, 250)
df=pd.DataFrame({"program": program, "score": score})

sns.displot(
    df, x="score", col="program", bins=18,
    facet_kws=dict(sharex=True, sharey=True),
    height=3, aspect=1.1
)
plt.show()
```

WHAT TO LOOK FOR

- Panels share axes (fair comparison)
- Shape shows variance (not just the mean)
- Bins + scale are design decisions

Code Demo C (Output): Small Multiples Reveal Spread

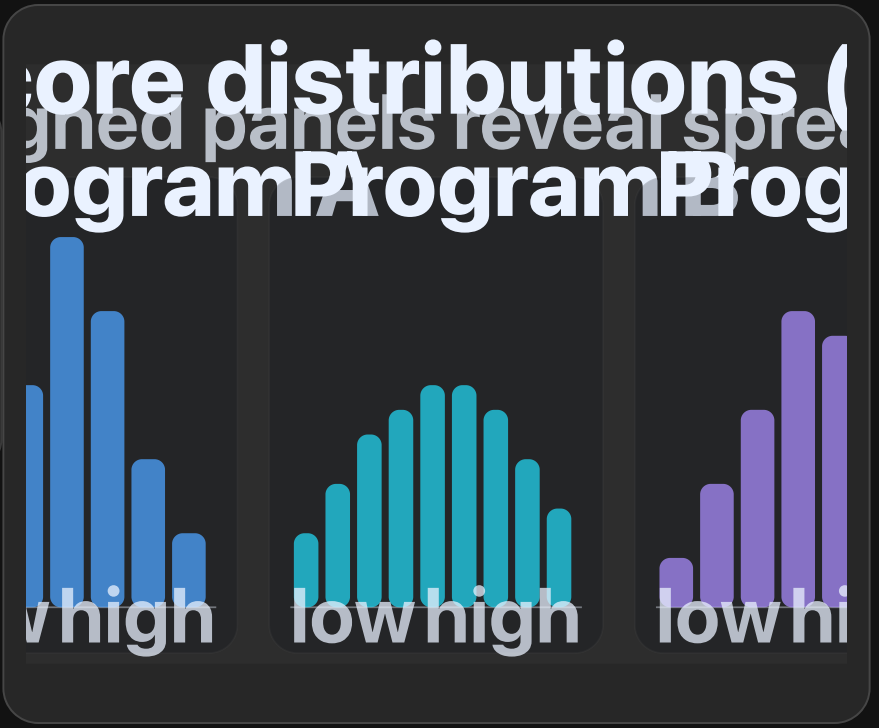
INTERPRETATION

Same mean, different risk

- Wider distributions imply more extreme outcomes
- Aligned panels make differences obvious
- Label bins and units when you publish

RENDERED RESULT (ILLUSTRATION)

This is what your FacetGrid histogram should resemble.

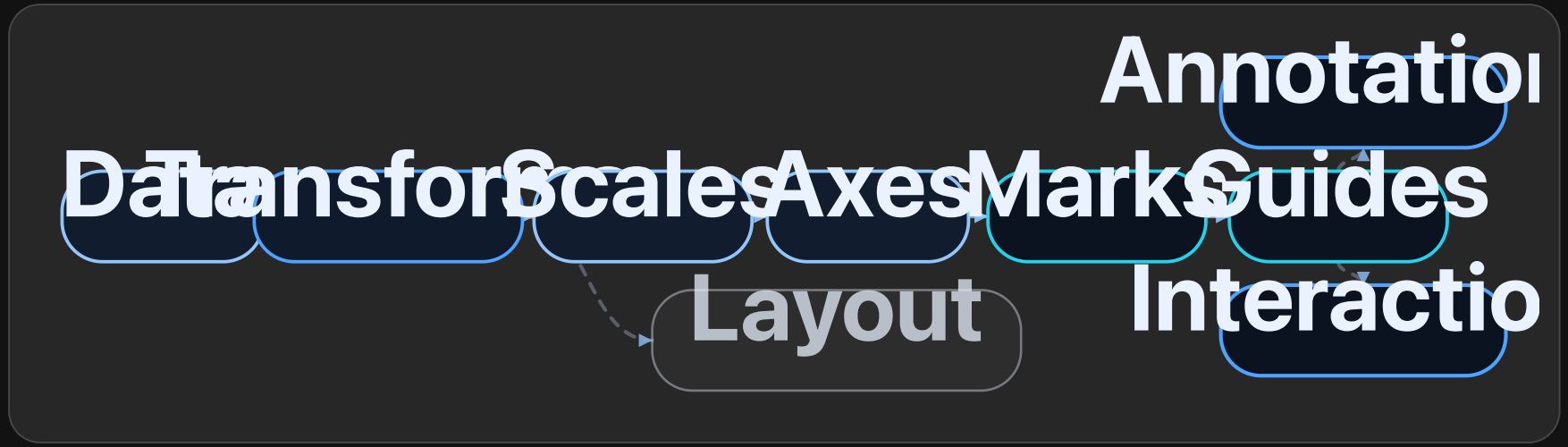


PART 3 · MODERN COMPONENTS

Think in chart parts

Build charts like reusable UI components

The Modern Chart Component Checklist



“D3 Concepts” in a Python World

CORE IDEA

Data ↔ marks mapping

Bind rows to marks; encode columns to channels.

PYTHON ANALYSIS

DataFrame → spec → renderer

Altair/Plotly generate web renderers.

Code Demo D: Altair Encodings

RUNNABLE PYTHON

Data → encoding → interactive tooltip.

```
import pandas as pd
import altair as alt

df=pd.DataFrame({
    "x":[1,2,3,4,5,6],
    "y":[0.70,0.62,0.75,0.73,0.80,0.78],
    "term":["baseline"]*3 + ["current"]*3
})

alt.Chart(df).mark_circle(size=110).encode(
    x="x:Q",
    y=alt.Y("y:Q", scale=alt.Scale(domain=[0.5, 0.9])),
    color="term:N",
    tooltip=["x", "y", "term"],
).properties(width=380, height=260, title="Encoded scatter
```

WHAT TO WATCH

- Domains are explicit (comparison-friendly)
- Color encodes category, not magnitude
- Tooltip adds detail without clutter

Code Demo D (Output): Encoded Scatter + Tooltip

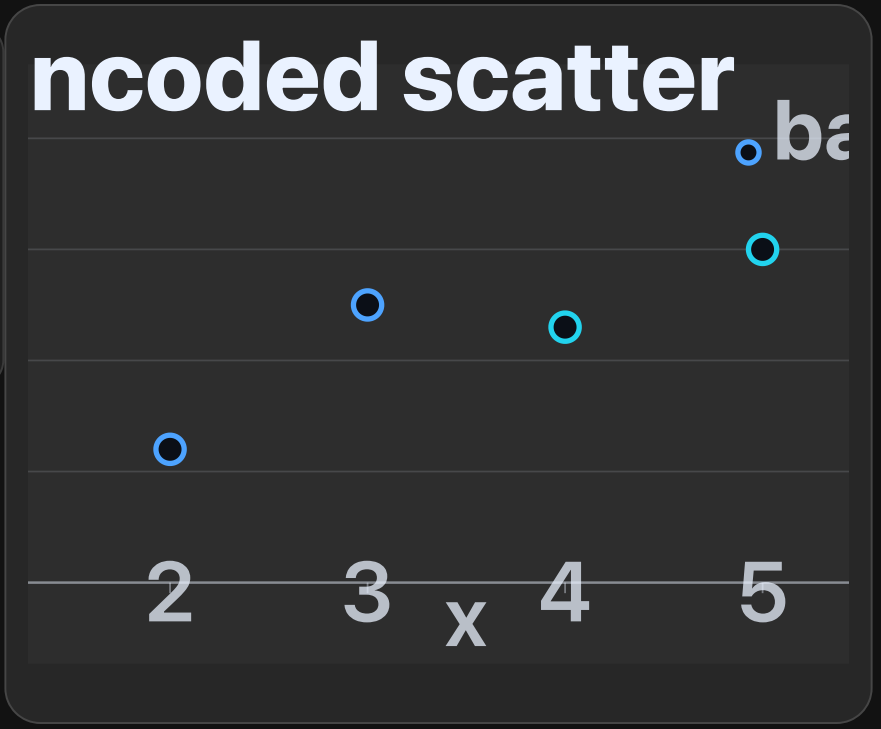
DESIGN NOTE

Encoding is a contract

- Quantitative → position/scale
- Categorical → hue/grouping
- Set domains when comparisons matter

RENDERED RESULT (ILLUSTRATION)

Your Altair chart should match this structure.



PART 4 · INTERACTIVITY

Interactivity with purpose

Tooltips, selection, filtering, and readable dashboards

Tooltips Are “Details on Demand”

GOOD TOOLTIP

Confirms values

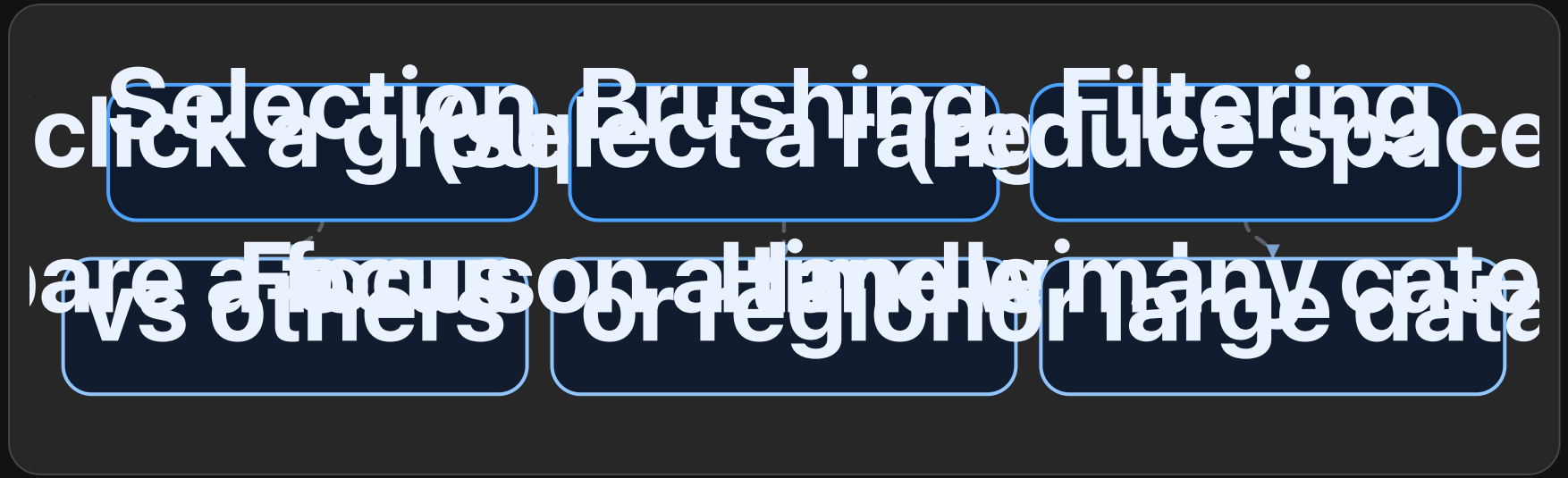
Units, exact numbers, IDs for traceability.

BAD TOOLTIP

Replaces the chart

If you need to hover everything, redesign.

Selections, Brushing, and Filtering (Task-Driven)



Code Demo E: Plotly for Interactive HTML

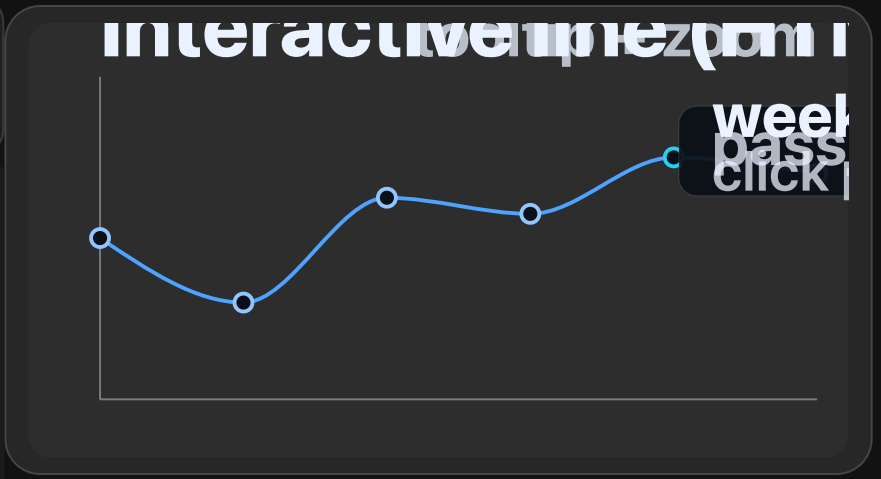
RUNNABLE PYTHON

Exports an interactive chart to HTML.

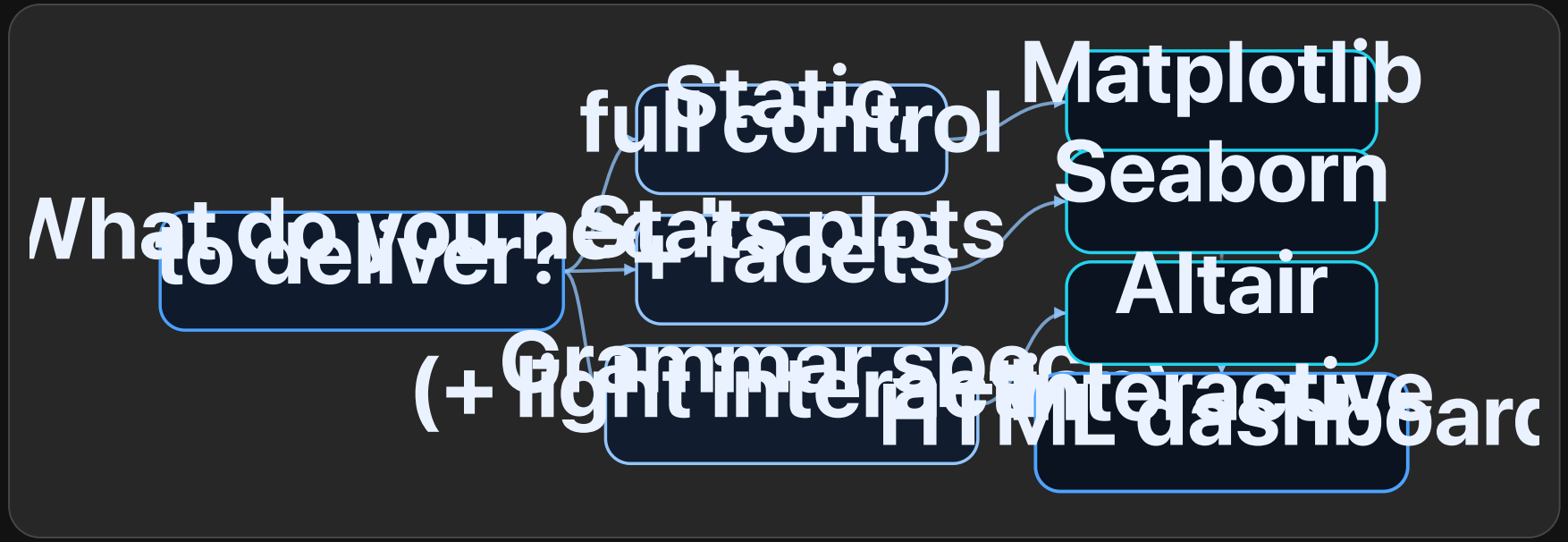
```
import pandas as pd
import plotly.express as px

df = pd.DataFrame(
    {
        "week": [1, 2, 3, 4, 5, 6],
        "pass_rate": [0.70, 0.62, 0.75, 0.73, 0.80, 0.78],
    }
)

fig = px.line(df, x="week", y="pass_rate", markers=True, ti
fig.update_yaxes(range=[0.5, 0.9])
fig.write_html("pass_rate.html", include_plotlyjs="cdn")
fig
```

RENDERED RESULT (ILLUSTRATION)

Hover + zoom are built in.



Choosing the Right Tool (A Practical Heuristic)



Export Formats: What You Hand In (and Why)

SVG

Slides + print

Crisp, editable, searchable.

PNG

Fixed images

Good for dense marks; stable layout.

HTML

Interactive

Tooltips, zoom, filters.

RULE

Choose format based on *use case*, not preference.

Micro-Checklist: “Looks Professional” in Practice

TYPOGRAPHY

- Readable title (claim or task)
- Axis labels with units
- Limited tick density

GRAPHICS

- Aligned scales for comparisons
- Legend only if necessary
- One clear emphasis (not rainbow)

DATA

- Rates vs counts handled
- Missingness shown explicitly
- Aggregation explained

EXPORT

- SVG/PNG/HTML matches use case
- Consistent sizing across figures
- Works on dark backgrounds

Practice (In Class): Make One Chart, Three Exports

TASK

Create one chart and export it as SVG, PNG, and HTML.

Then explain which format you would submit for: slides, a PDF report, and an interactive critique.

- Pick a simple dataset (10–200 rows)
- Include units and a baseline if relevant
- Write 2–3 sentences justifying your export choices

Key Takeaways

- Web outputs matter: SVG/HTML are common “final forms”
- Python workflow: data → transforms → chart → export
- Modern charts are components: scales, marks, axes, guides, interaction
- Interactivity is only “professional” when it supports a task

References (Recommended)

Matplotlib documentation

Figure/Axes model, export formats, styling

https://matplotlib.org/stable/

Altair documentation

Grammar of graphics + interactive selections

https://altair-viz.github.io/

Plotly documentation

Interactive charts + HTML export

https://plotly.com/python/

Wickham (2014)

Tidy data as a foundation for chart-ready tables

https://doi.org/10.18637/jss.v059.i10