Interactive Dashboards and Storytelling with Python

Erich Assuncao

March, 2025

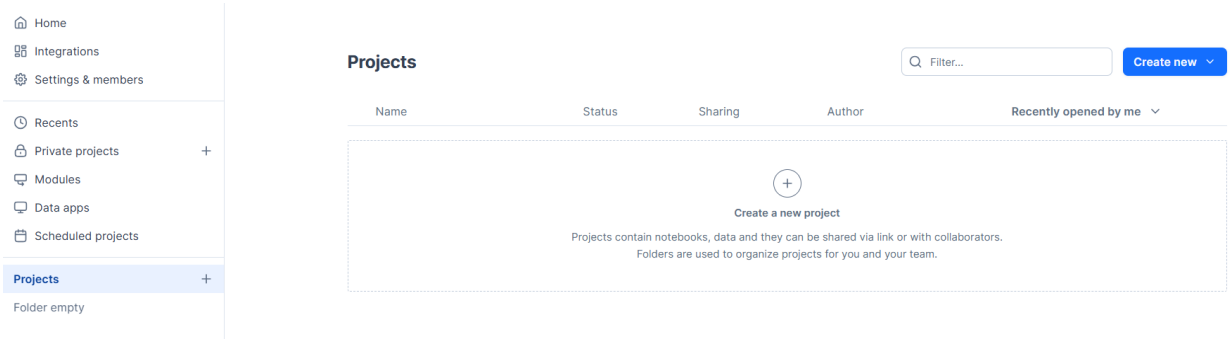
A learning technology portfolio resource introducing Python-based data visualization, interactive dashboards, and data storytelling through hands-on examples using Deepnote, Plotly, Streamlit, pandas, and Altair.

Part I: Python Quick-Start Guide for Data Visualization

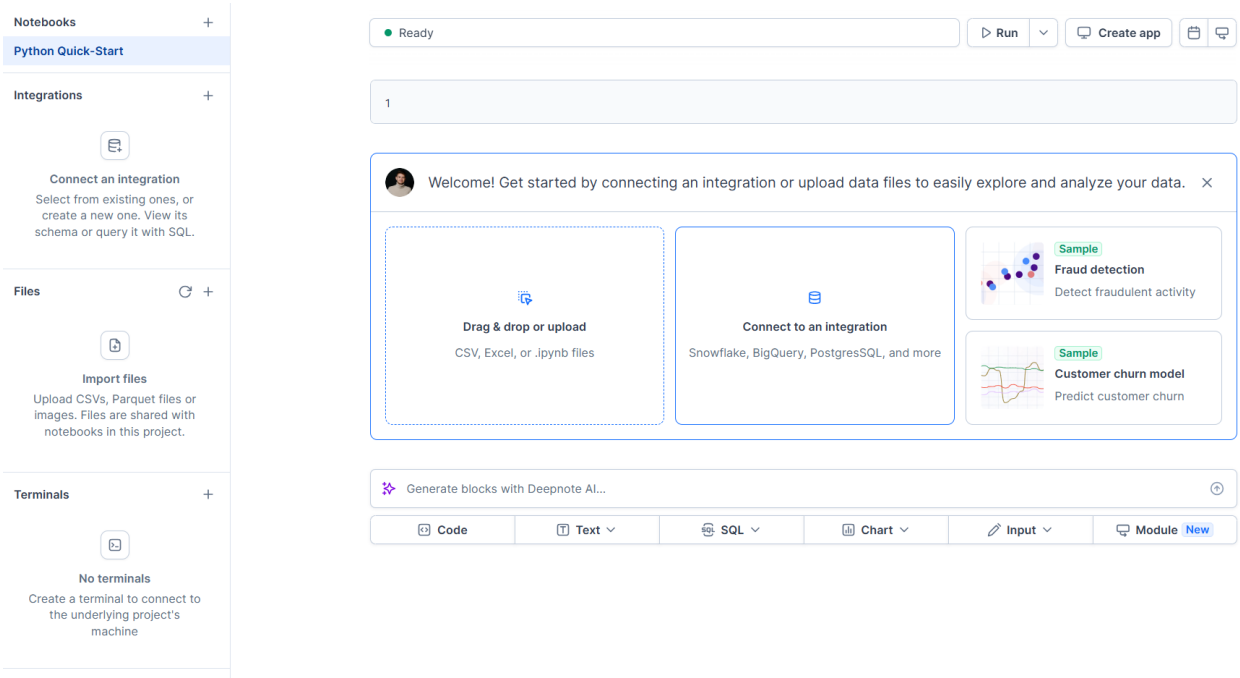
Welcome to the Python Quick-Start Guide! This brief tutorial introduces essential Python syntax and constructs for students transitioning from other Object-Oriented Programming (OOP) languages. We'll use Deepnote for easy practice and execution. If you haven't yet, create your free account at Deepnote.com.

Getting Started with Deepnote

1. **Create a Project:**
    - Once you have created your account and a workspace, create a new project.



2. **Creating a Notebook:**
    - Click on the + sign beside “Notebooks.”
    - Rename the notebook to "Python Quick-Start."



3. **Essential Python Syntax & Concepts**

Below are key elements you'll use frequently in building interactive dashboards and visualizations.

1. Variables and Data Types

Python dynamically infers data types:

```
x = 10                      # integer
y = 3.14                    # float
name = "Alice"              # string
is_active = True            # boolean
```

2. Lists and Dictionaries

Use lists for ordered collections and dictionaries for key-value pairs:

```
# Lists
numbers = [1, 2, 3, 4, 5]
names = ["Alice", "Bob", "Charlie"]

# Dictionaries
student = {"name": "Alice", "grade": 85}
```

3. Functions

Simple function syntax:

```
def greet(name):
    return f"Hello, {name}!"

message = greet("Bob")
print(message)  # Output: Hello, Bob!
```

4. Control Flow

- **Conditional Statements:**

```
score = 75
if score >= 70:
    print("Passed")
else:
    print("Failed")
```

- **Loops:**

```
# For loop
for name in names:
    print(name)

# While loop
count = 0
while count < 5:
    print(count)
    count += 1
```

5. Libraries and Imports

Importing libraries allows you to leverage external Python code:

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

6. Using Pandas for Data Handling

Pandas simplifies data manipulation:

```
# Reading data
 df = pd.read_csv('filename.csv')

# Viewing data
 print(df.head())
```

7. Simple Visualization with Plotly

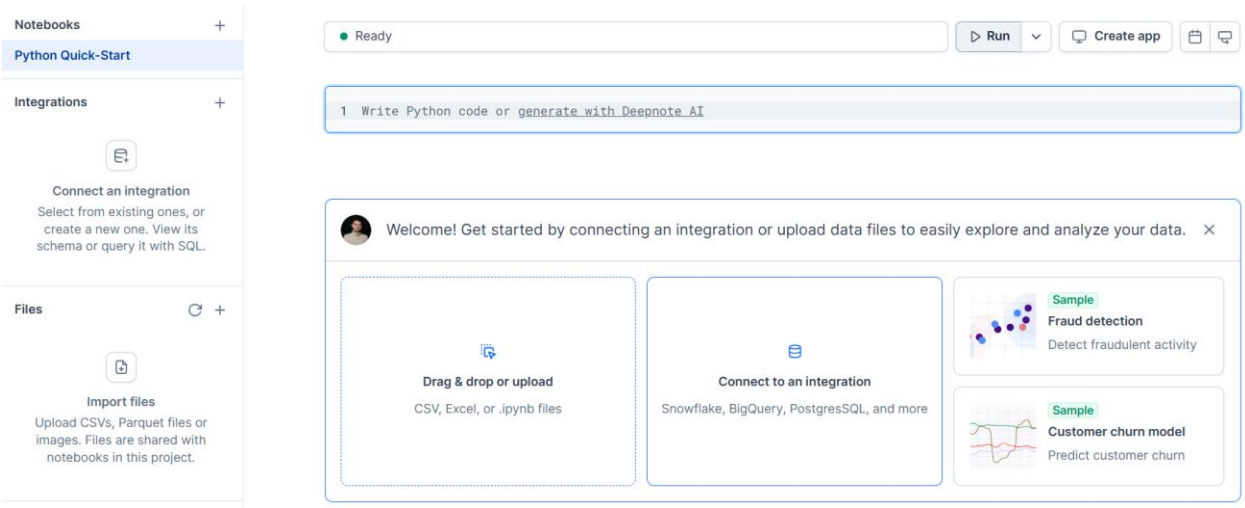
Quickly create interactive charts:

```
import plotly.express as px

# Sample data
data = {'Fruit': ['Apple', 'Banana', 'Cherry'], 'Count': [10, 15, 7]}
df = pd.DataFrame(data)

# Bar chart
fig = px.bar(df, x='Fruit', y='Count', title='Fruit Count')
fig.show()
```

Practice Exercise (Deepnote):

1. In your Deepnote notebook, create a list of your favorite fruits.
2. Write a function to print each fruit with a custom greeting.
3. Create a simple Plotly bar chart displaying fruits and a random quantity (optional).

Hint: You can write and run Python code straight from your notebook on Deepnote.



This quick-start equips you with essential Python basics needed for interactive dashboards and data storytelling. You're now ready to proceed with visualization and dashboard building in Python!

Additional Resources

Python Cheatsheet: https://www.pythoncheatsheet.org/

Official Python Tutorials: https://docs.python.org/3/tutorial/index.html

Part II: Basics of Data Visualization

Conceptual Introduction

Importance of Data Visualization

Data visualization converts complex data into understandable graphical representations, facilitating the extraction of meaningful insights. Effective visualizations:

- Simplify large datasets.
- Highlight trends, relationships, and outliers.
- Enhance comprehension and decision-making.
- Facilitate clear and compelling storytelling.

Selecting Appropriate Visualization Types

Choosing the correct visualization depends on the data type and your communication goals:

- **Bar Charts:** Best for comparing quantities across categories.
- **Line Graphs:** Ideal for showing trends or changes over time.
- **Scatter Plots:** Useful for demonstrating relationships between two numerical variables.

Essential Visual Design Principles

Creating clear and effective visualizations involves:

- **Clarity:**
    - Keep visualizations clean and avoid unnecessary elements.
    - Prioritize simplicity to ensure easy interpretation.
- **Color Usage:**
    - Use colors strategically to highlight or distinguish data points.
    - Avoid excessive or confusing color schemes.
- **Readability:**
    - Ensure all textual elements (titles, labels, legends) are clear and appropriately sized.
    - Organize data logically for intuitive understanding.

Reading Assignment

- **"Doing Better Data Visualization"** – *Eric Hehman and Sally Y. Xie*. *Advances in Methods and Practices in Psychological Science* (2021).
  Summary: This recent peer-reviewed tutorial distills modern data visualization best practices into an accessible guide. It emphasizes communicating patterns in data as clearly as possible and is designed to be usable by both new and experienced R/ggplot2 users. The article discusses overarching design philosophies (like maximizing information without distortion) and decisions

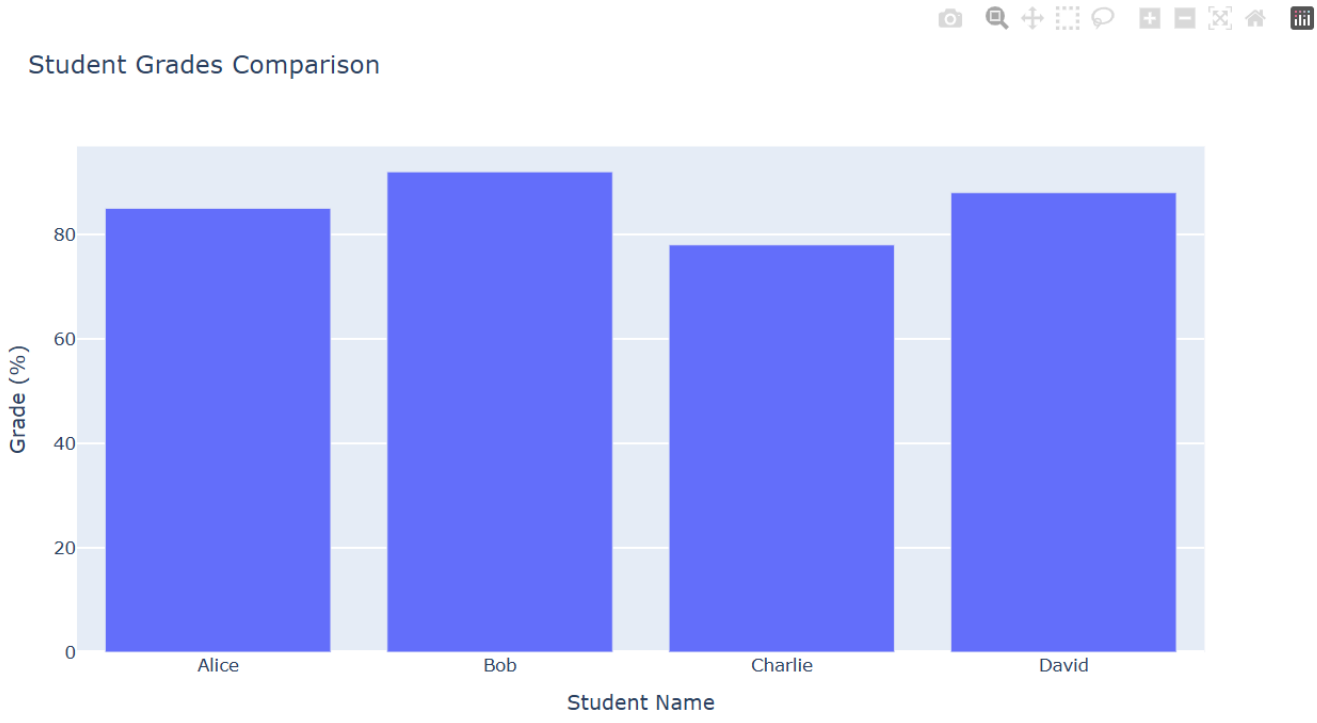
about figure components, then demonstrates how to improve common types of plots (e.g. means, proportions, relationships) with annotated R code. It's highly useful for students because it bridges visualization theory and practice, helping them learn why and how to create better graphs in a scientific context. Retrieved March 26, 2025, from https://journals.sagepub.com/doi/full/10.1177/25152459211045334

Hands-On Examples Using Plotly Express

Example 1: Bar Chart

Bar charts help illustrate categorical comparisons.

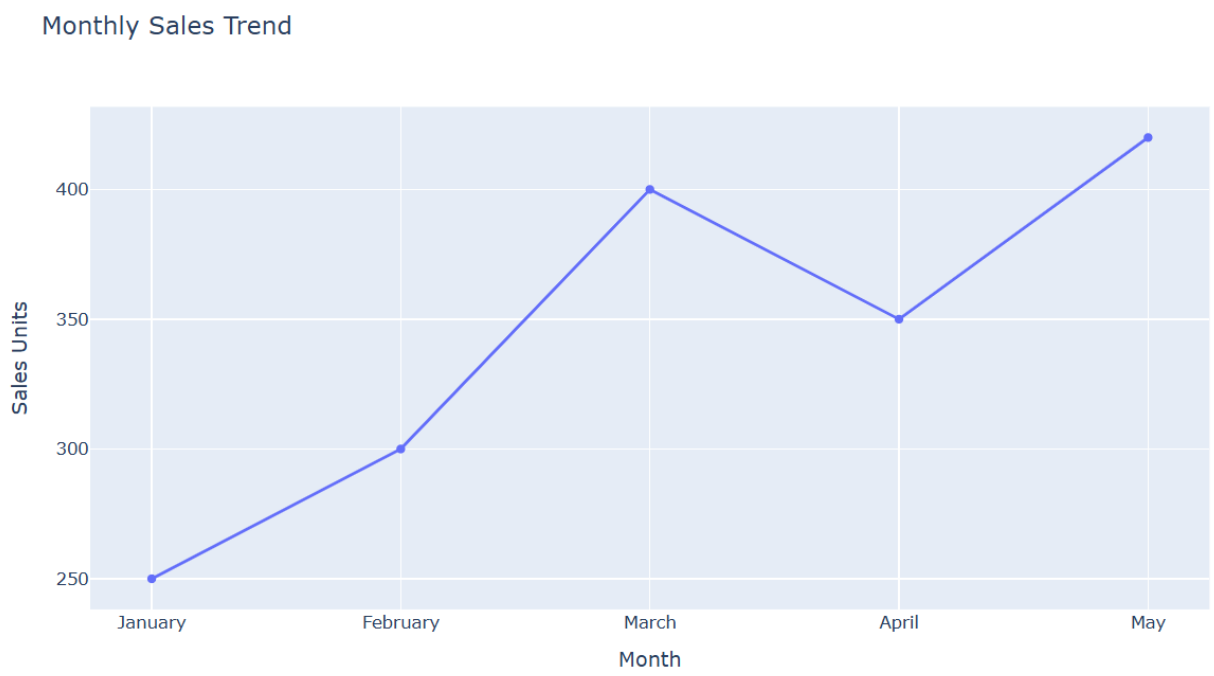
```python
import plotly.express as px
import pandas as pd

# Data
grades_data = {'Student': ['Alice', 'Bob', 'Charlie', 'David'],
               'Grade': [85, 92, 78, 88]}
df_grades = pd.DataFrame(grades_data)

# Visualization
fig = px.bar(df_grades, x='Student', y='Grade', title='Student Grades Comparison',
             labels={'Grade': 'Grade (%)', 'Student': 'Student Name'})
fig.show()
```



Example 2: Line Graph

Line graphs visualize data trends over intervals or periods.

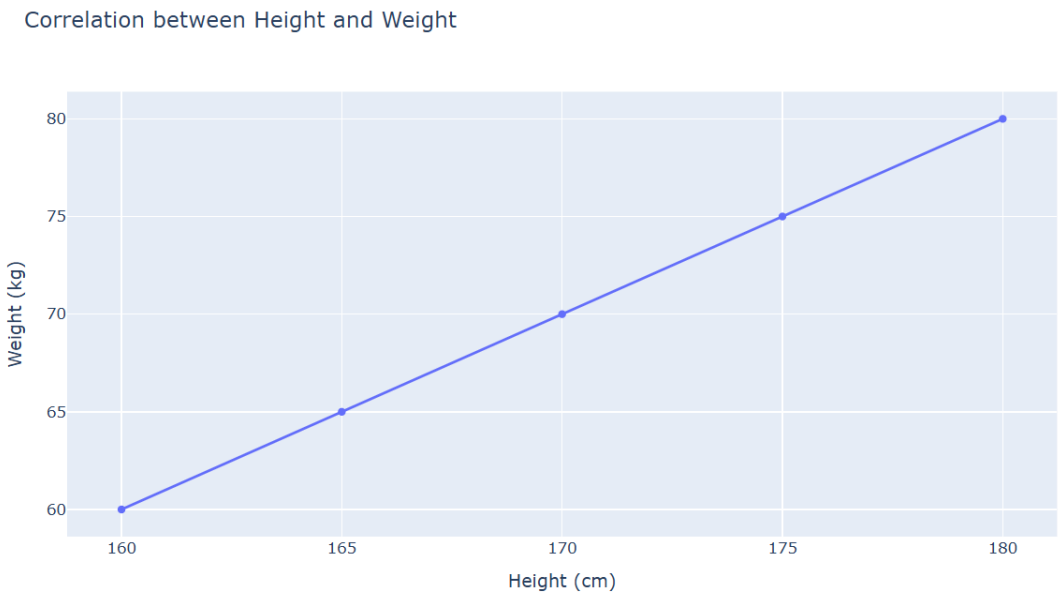
```
sales_data = {'Month': ['January', 'February', 'March', 'April', 'May'],
              'Sales': [250, 300, 400, 350, 420]}
df_sales = pd.DataFrame(sales_data)

fig = px.line(df_sales, x='Month', y='Sales', title='Monthly Sales Trend',
              markers=True, labels={'Sales': 'Sales Units', 'Month': 'Month'})
fig.show()
```



Example 3: Scatter Plot

Scatter plots illustrate the correlation between numerical variables.

```
height_weight_data = {'Height_cm': [160, 165, 170, 175, 180],
                      'Weight_kg': [60, 65, 70, 75, 80]}
df_hw = pd.DataFrame(height_weight_data)

fig = px.scatter(df_hw, x='Height_cm', y='Weight_kg',
                 title='Correlation between Height and Weight',
                 labels={'Height_cm': 'Height (cm)', 'Weight_kg': 'Weight (kg)'},
                 trendline='ols')
fig.show()
```



Activity

Using Deepnote, complete the following practice activity:

1. **Bar Chart:** Create a bar chart comparing the number of hours you spend weekly on various activities (e.g., study, exercise, hobbies).
2. **Line Graph:** Illustrate a line graph showing your daily water intake over a week.
3. **Scatter Plot:** Generate a scatter plot examining the relationship between hours of sleep and daily productivity ratings for a week.

Ensure each visualization includes clear titles, axis labels, and appropriate design elements for clarity and readability.

Share the results with your colleagues.

Supplementary Resources

- **"Ten Simple Rules for Better Figures"** – *Nicolas P. Rougier, Michael Droettboom, and Philip E. Bourne*. *PLOS Computational Biology* (2014).
  Summary: This concise, peer-reviewed article presents ten guidelines for creating effective scientific figures. It introduces a basic set of rules to improve figure design and explains common pitfalls (e.g. knowing your audience, using color wisely, avoiding misleading defaults and "chartjunk") to help readers make clear, accurate graphs. The advice is easy to follow and broadly applicable, making it very useful for undergraduate students learning how to present data clearly and ethically. Retrieved March 26, 2025, from https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003833

- ***Fundamentals of Data Visualization*** – *Claus O. Wilke* (2019). *O'Reilly Media* – Open-access web edition.
  Summary: This is a comprehensive introductory book on data visualization principles, available for free online. Written by an experienced scientist, it serves as a guide to making visualizations that accurately reflect data, tell a story, and look professional. The book covers a wide range of basics – from perceptual aspects (how viewers interpret size, color, position) to practical guidance on choosing effective chart types and designing with clarity. It's very useful for undergraduates because it not only teaches *how* to create charts (with examples in R), but also *why* certain design choices make graphs more informative and compelling. Retrieved March 26, 2025, from https://clauswilke.com/dataviz/

- ***Hands-On Data Visualization: Interactive Storytelling from Spreadsheets to Code*** – *Jack Dougherty and Ilya Ilyankou* (2021). *O'Reilly Media* – Open-access web edition.
  Summary: This beginner-friendly guide (available as a free web book) teaches the basics of creating interactive charts and maps using free, easy-to-learn tools. It starts with simple drag-and-drop platforms like Google Sheets, Datawrapper, and Tableau Public, and gradually introduces readers to editing open-source templates (using Chart.js, Highcharts, Leaflet) – all with no coding experience required. Through step-by-step tutorials and real-world examples, the book helps students turn spreadsheet data into engaging visual stories. It's an excellent resource for undergraduates because it provides hands-on practice in designing data visuals and reinforces core concepts of clarity and audience engagement in visualization. Retrieved March 26, 2025, from https://handsondataviz.org/index.html

Part III: Building Interactive Dashboards

Introduction to Dashboard Fundamentals

Interactive dashboards are dynamic visual displays that allow users to explore data actively, rather than passively viewing static graphs. The primary goal of dashboards is to communicate complex information quickly and effectively, enabling users to interact with data to uncover insights.

Types of Interactivity

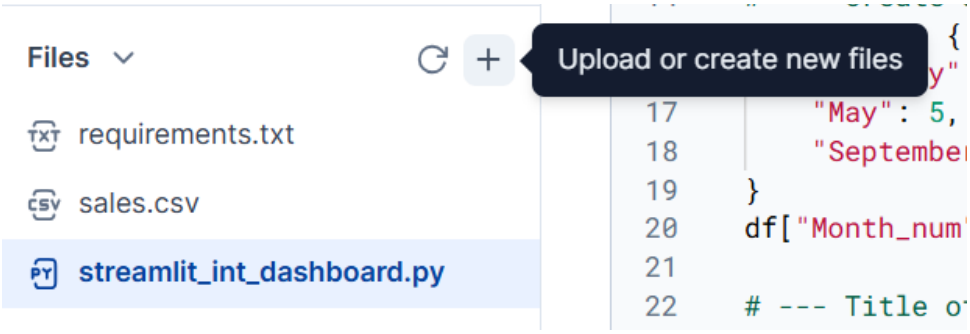
Interactive elements significantly enhance the utility and usability of dashboards by providing users control over how they view data. Key interactive elements include:

- **Buttons:** Allow users to trigger specific actions, such as refreshing data or updating visualizations.
- **Sliders:** Enable users to explore data across a continuous range, such as dates or numerical values.
- **Dropdowns:** Allow users to filter data based on categorical choices or criteria, simplifying navigation and exploration.

Practical Example: Interactive Dashboard with Streamlit

Streamlit simplifies the creation of interactive Python web applications, making it an ideal tool for beginners. The interactive dashboard below demonstrates key interactive features available using Streamlit.

Note: A Python file named streamlit_int_dashboard.py was created on Deepnote, and a sample dataset (sales.csv), available in the resources folder, was uploaded to the project to provide realistic data for visualization.



streamlit_int_dashboard.py

```python
import streamlit as st
import pandas as pd
import altair as alt

# Set page configuration (optional)
st.set_page_config(
    page_title="Sales Dashboard",
    layout="wide"
)

# --- Read data ---
df = pd.read_csv("sales.csv")

# --- Create a numeric month column for filtering ---
month_map = {
    "January": 1, "February": 2, "March": 3, "April": 4,
    "May": 5, "June": 6, "July": 7, "August": 8,
    "September": 9, "October": 10, "November": 11, "December": 12
}
df["Month_num"] = df["Month"].map(month_map)

# --- Title of the app ---
st.title("Sales Dashboard")

# --- Toggle button using session state ---
if "show_data" not in st.session_state:
    st.session_state.show_data = False

# Flip the state when the button is clicked
if st.button("Show/Hide Raw Data"):
    st.session_state.show_data = not st.session_state.show_data

# Conditionally display the data
if st.session_state.show_data:
    st.subheader("Data Table")
    st.dataframe(df.drop(columns=["Month_num"]))

# --- Slider to filter data by month range ---
st.subheader("Filter by Month Range")
start_month, end_month = st.slider(
    "Select a month range (1 = January, 12 = December)",
    min_value=1,
    max_value=12,
    value=(1, 12)  # default range
)

# Filter the dataframe based on the slider selection
filtered_df = df[(df["Month_num"] >= start_month) & (df["Month_num"] <= end_month)]

# --- Product selection (multiselect) ---
product_options = ["Laptop", "Tablet", "Smartphone", "All Products"]
selected_products = st.multiselect(
    "Select product(s) to display",
    product_options,
    default=["Laptop", "Tablet", "Smartphone"]  # default
)

# --- Create line chart for selected products ---
if selected_products:
    st.subheader("Sales Trends (Filtered by Month Range)")
```

```
    # Melt the filtered DataFrame for easier plotting
    melted_df = filtered_df.melt(
        id_vars=["Month", "Month_num"],
        value_vars=selected_products,
        var_name="Product",
        value_name="Sales"
    )

    line_chart = (
        alt.Chart(melted_df)
        .mark_line(point=True)
        .encode(
            x=alt.X("Month:N", sort=None),
            y="Sales:Q",
            color="Product:N",
            tooltip=["Month", "Product", "Sales"]
        )
        .interactive()
    )

    st.altair_chart(line_chart, use_container_width=True)

# --- Bar chart comparing total annual sales by product (for filtered months) ---
st.subheader("Total Sales by Product (Filtered Range)")
# Sum up each product's total within the filtered range
filtered_totals = filtered_df[product_options].sum().reset_index()
filtered_totals.columns = ["Product", "Total Sales"]

bar_chart = (
    alt.Chart(filtered_totals)
    .mark_bar()
    .encode(
        x=alt.X("Product:N", sort=None),
        y="Total Sales:Q",
        color="Product:N",
        tooltip=["Product", "Total Sales"]
    )
)

st.altair_chart(bar_chart, use_container_width=True)

# --- Summary statistics (for filtered range) ---
st.subheader("Summary Statistics (Filtered Range)")
col1, col2, col3, col4 = st.columns(4)
col1.metric("Total Laptops Sold", f"{filtered_df['Laptop'].sum()}")
col2.metric("Total Tablets Sold", f"{filtered_df['Tablet'].sum()}")
col3.metric("Total Smartphones Sold", f"{filtered_df['Smartphone'].sum()}")
col4.metric("Total (All Products)", f"{filtered_df['All Products'].sum()}")

st.write("""
- **Show/Hide Raw Data**: Toggles display of the entire dataset via session state.
- **Month Range Slider**: Filters the data (and charts) to the selected months.
- **Multiselect**: Lets you choose which products to visualize.
""")
```

By clicking on “Create Streamlit app” in the top-right corner of the code file, the dashboard will be displayed. Watch the video “Streamlit_Interactive_Dashboard” available in the “Resources” folder to see the interactive dashboard in action.

Practice Activity: Create an Interactive Streamlit Dashboard

- Use a dataset of your choice or the provided sample dataset.
- Include at least two interactive elements (e.g., dropdown and slider).
- Allow users to filter data dynamically and visualize results.
- Share it with your colleagues.

Part IV: Data Storytelling

An Introduction to Data Storytelling

Based on Knaflic, C. N. (2015). *Storytelling with data: A data visualization guide for business professionals*. Wiley.

In *Storytelling with Data*, Knaflic emphasizes that data storytelling is more than just presenting numbers or creating charts — it is about crafting a narrative that resonates with the audience. She explains that while data provides the foundation for analysis, it often lacks the human connection needed to drive action or decision-making. By integrating narrative elements such as context, structure, and design, data storytelling transforms raw information into meaningful insights. This process involves understanding your audience, defining a clear takeaway message, and using visual design principles to communicate that message effectively.

Knaflic also underlines the importance of simplicity and intentional design in visualizing data. She advocates for removing clutter and focusing on what matters most to support the story being told. Charts and graphs are tools in the storyteller's toolkit, not ends in themselves. The ultimate goal is to guide the audience through the data in a way that is both informative and engaging. By doing so, storytellers can influence business decisions, highlight trends, and uncover opportunities — all by presenting data in a more accessible and emotionally resonant way.

Supplementary Reading

Schröder, K., Eberhardt, W., Belavadi, P., Ajdadilish, B., van Haften, N., Overes, E., Brouns, T., & Calero Valdez, A. (2023, December 2). *Telling stories with data - A systematic review*. arXiv. https://arxiv.org/abs/2312.01164v1

Exercise: Telling a Story with Weather Data

Scenario

The city of Springfield has collected daily weather data for one month (e.g., January). The dataset includes:

- Date
- Average Temperature (°C)
- Precipitation (mm)
- Humidity (%)

Local officials want to understand seasonal weather patterns and identify any unusual trends (like unexpected heatwaves or heavy rain days). Your task is to create an interactive dashboard in Deepnote that communicates these insights clearly.

Task Overview

You will:

1. Upload and load the CSV file (weather_data.csv)
2. Create interactive visualizations to explore:
    - Temperature trends (line chart)
    - Precipitation patterns (bar chart)
    - Temperature versus humidity (scatter plot with a trendline)
3. Add narrative annotations that explain the insights found in the data.

Hint: You can create a new notebook on Deepnote and use the sample weather data file provided in the "Resources" folder. It's best to divide your code and comments into separate cells, testing each piece of functionality as you go. While pandas and plotly express are enough to complete this exercise, you're free to explore other libraries if you prefer. If you run into challenges, there's an answer file available for reference.