Weather Dashboards: From Notebooks to GUIs

Building Interactive Data Visualisations with Python

Michael Borck

Table of contents

What is a Dashboard?

- **Definition:** A visual display of the most important information needed to achieve one or more objectives.
- **Benefits:**

  - Consolidated view of key metrics
  - Faster decision-making
  - Improved communication

- **Types:**

  - Operational (real-time monitoring)
  - Strategic (long-term trends)
  - Analytical (in-depth exploration)

Dashboards in Python

- **Why Python?**

  - Versatile language
  - Rich ecosystem of data science and visualisation libraries
  - Easy to integrate with other tools

- **Popular Libraries:**

  - Matplotlib (basic plotting)
  - Seaborn (statistical plots)
  - Plotly (interactive plots)
  - Bokeh (web-based dashboards)
  - Panel (high-level dashboarding)
  - Tkinter (GUI library)

Jupyter Widgets (ipywidgets)

- **What are they?** Interactive elements for Jupyter Notebooks.
- **Examples:**

  - Sliders
  - Dropdowns
  - Text boxes
  - Buttons

- **Benefits:**

  - Easy to create and use
  - Enable exploration of data within the notebook
  - Great for prototyping dashboard ideas

- **Common Widgets**

- **IntSlider**: `ipywidgets.IntSlider()`
- **FloatSlider**: `ipywidgets.FloatSlider()`
- **IntRangeSlider**: `ipywidgets.IntRangeSlider()`
- **FloatRangeSlider**: `ipywidgets.FloatRangeSlider()`
- **Dropdown**: `ipywidgets.Dropdown(options=['Option 1', 'Option 2'])`
- **Text**: `ipywidgets.Text()`
- **Button**: `ipywidgets.Button(description='Click Me')`

From Notebook to GUI: Why?

- **Limitations of Notebooks:**
  - Not ideal for sharing with non-technical users
  - Limited customisation options
- **Benefits of GUIs:**
  - More user-friendly interface
  - Can be packaged into standalone applications
  - Greater control over the look and feel

Tkinter: A Python GUI Library

- **Introduction:** Standard Python GUI toolkit.
- **Features:**
  - Cross-platform (Windows, macOS, Linux)
  - Relatively easy to learn
  - Good for simple to moderately complex applications
- **Alternatives:**
  - PyQt, wxPython (more powerful, but steeper learning curve)

Essential Tkinter Widgets

- **Label** (`tk.Label`): Displays text or images
- **Button** (`tk.Button`): Triggers actions when clicked
- **Entry** (`tk.Entry`): Single-line text input
- **Text** (`tk.Text`): Multi-line text input
- **Combobox** (`ttk.Combobox`): Selection from a list
- **Frame** (`tk.Frame`): Container for organising widgets

Design Considerations for Your Weather Dashboard

- **Audience:** Who will be using it? (Students, instructors, the public?)
- **Data Sources:** Where will you get the weather data?
- **Key Metrics:** What information is most important to display?
- **Layout:** How will you arrange the elements for optimal usability?
- **Interactivity:** What kind of user controls will you provide?

Live Coding Demo - Jupyter Notebook

- **Walkthrough:** Build a basic weather dashboard in a Jupyter Notebook using ipywidgets.
- **Highlight:** How to create and connect widgets to data visualisations.
- **Keep it Simple:** Focus on the core concepts, not every possible feature.

Migration to Tkinter (Overview)

- **Explain:** The process of converting the notebook code into a Tkinter GUI.
- **Challenges:**
  - Adapting notebook layout to GUI elements
  - Managing event-driven programming
- **Tips:**
  - Plan the GUI layout carefully
  - Use functions to organise code
  - Test frequently

Live Coding Demo - tkinter

- **Walkthrough:** Build a basic weather dashboard in tkinter
- **Highlight:** How to create and connect widgets to data visualisations.
- **Keep it Simple:** Focus on the core concepts, not every possible feature.

Conclusion and Next Steps

- **Summarise:** Key takeaways from the session.
- **Homework/Challenge:**
  - Add more plots to the dashboard, both in the notebook, and GUI
  - Explore additional Tkinter features or alternative GUI libraries