



Lecture Roadmap

Data Domains

Comparing Categories | Relationships | Geospatial | Time |
Part-to-whole | Distributions | Uncertainty | ...

Storytelling

Perception +
Visualization Design

Python + Tools

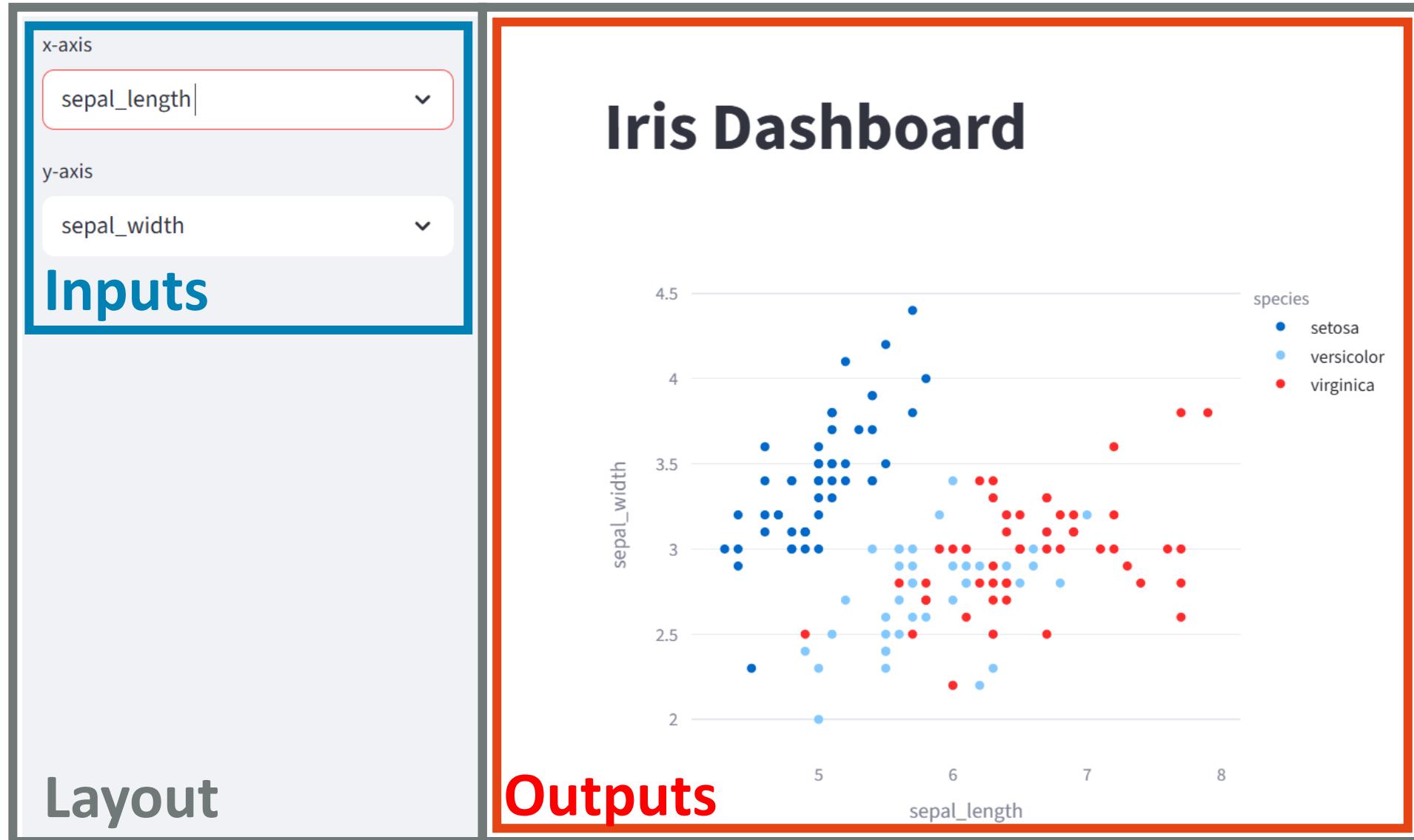
Interactive
Visualization

Streamlit Overview

“A faster way to build and share data apps: Streamlit turns data scripts into shareable web apps in minutes. All in pure Python. No front-end experience required.”

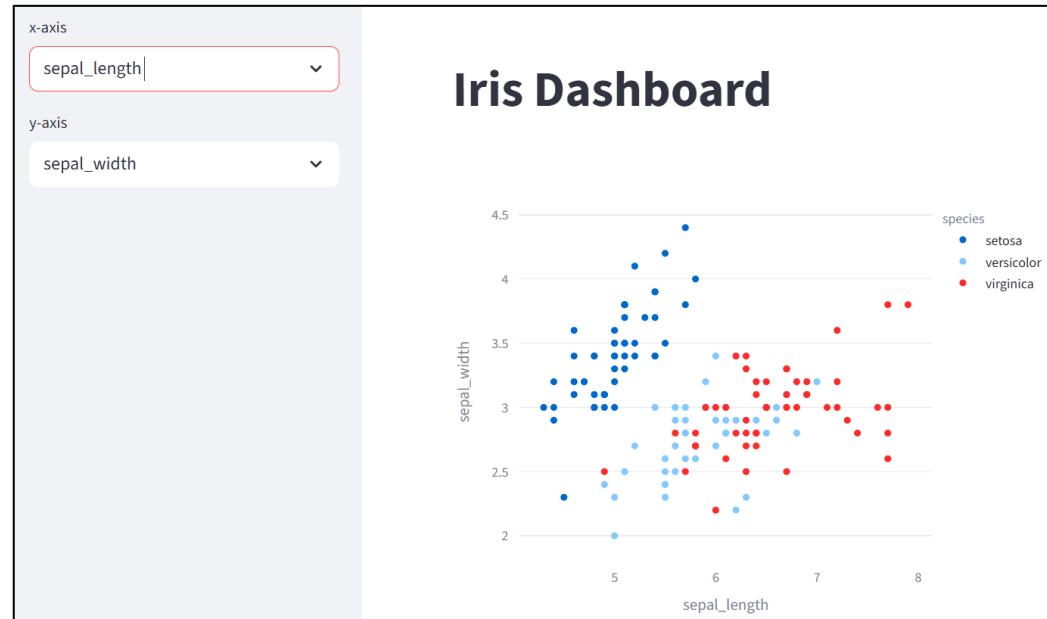
- ▶ **Easy-to-use:** no web development knowledge required
- ▶ **User interaction:** app updates in response to user input
- ▶ **Integration :** Pandas, Plotly, Matplotlib, Altair, Markdown, etc.
- ▶ **Deployment:** host and share via Streamlit Community Cloud

Streamlit Frontend



Streamlit Python Script

Browser Frontend



Backend

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

df = px.data.iris()

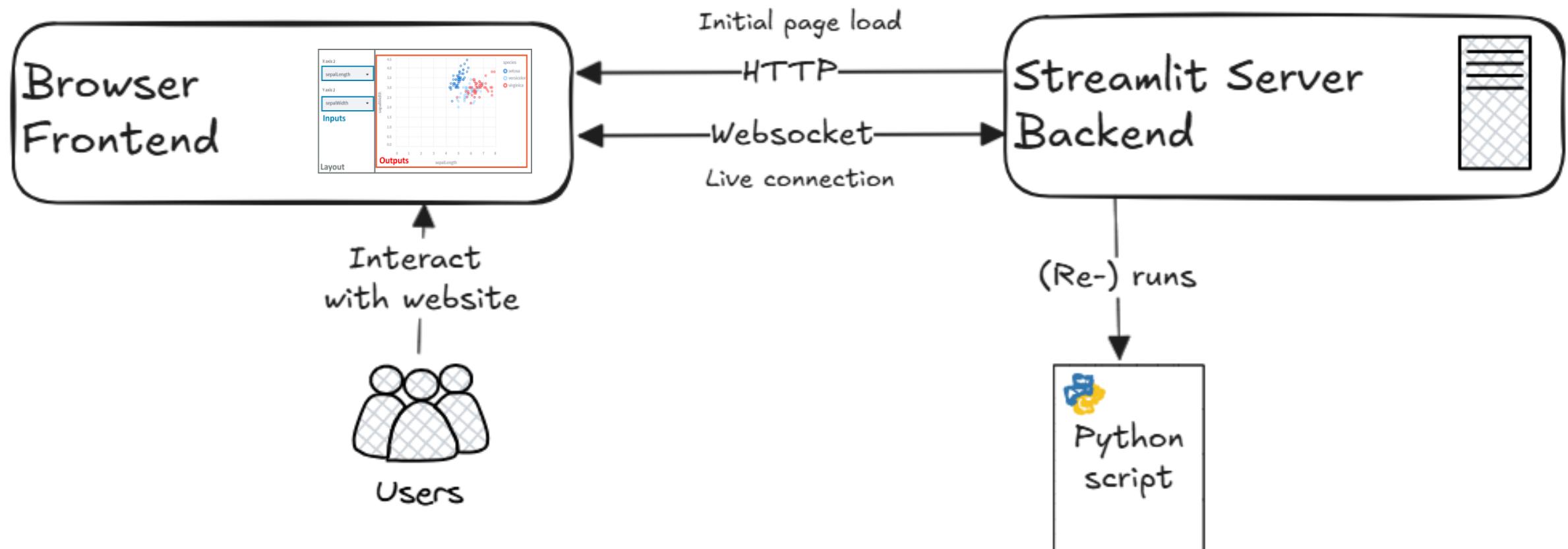
st.title('Iris Dashboard')

cols = ["sepal_length", "sepal_width", "petal_length", "petal_width"]
x = st.sidebar.selectbox(label="x-axis", options=cols, index=0)
y = st.sidebar.selectbox(label="y-axis", options=cols, index=1)

fig = px.scatter(df, x=x, y=y, color='species')
st.plotly_chart(fig)
```

streamlit run app.py

Architecture



App Model

- ▶ **Scripting:** Streamlit apps are Python scripts running from top to bottom
- ▶ **Inputs:** user interacts with input widget → rerun full (!) script
- ▶ **Outputs:** visualizations, images, text, data, ...
- ▶ **Caching:** Scripts can cache results to avoid repeating expensive operations

Inputs

Classify image

- Dog
- Cat
- Goldfish

Radio

Display a radio button widget.

```
choice = st.radio("Pick one",
```

Pick one

A screenshot of a selectbox interface. The input field contains the text "cats". Below it is a dropdown menu with two options: "cats" and "dogs". A cursor arrow is pointing at the "dogs" option.

Selectbox

Display a select widget.

```
choice = st.selectbox("Pick or
```

Visible in image

A screenshot of a multiselect box interface. Two items are selected: "milk" and "bananas". Below the selection area is a dropdown menu with two options: "apples" and "potatoes". A cursor arrow is pointing at the "potatoes" option.

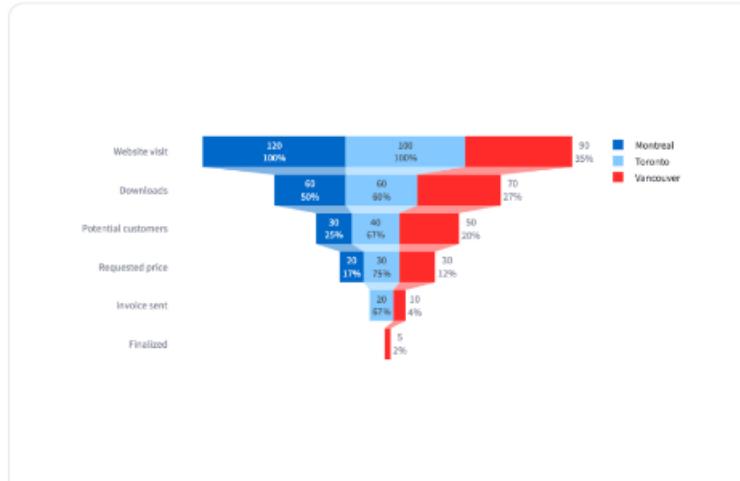
Multiselect

Display a multiselect widget. The multiselect widget starts as empty.

```
choices = st.multiselect("Buy"
```

More inputs: <https://docs.streamlit.io/develop/api-reference/widgets>

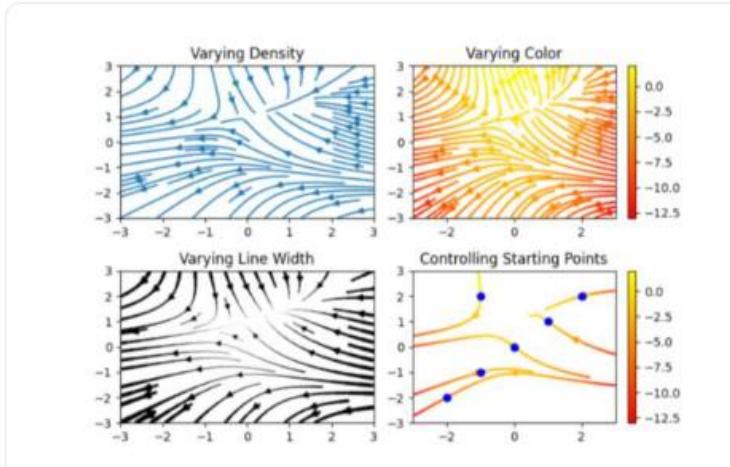
Charts outputs



Plotly

Display an interactive Plotly chart.

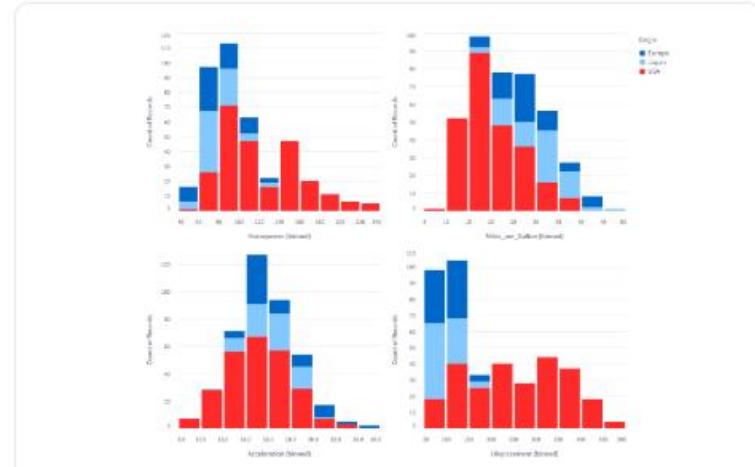
```
st.plotly_chart(my_plotly_chart)
```



Matplotlib

Display a matplotlib.pyplot figure.

```
st.pyplot(my_mpl_figure)
```



Altair

Display a chart using the Altair library.

```
st.altair_chart(my_altair_chart)
```

More chart outputs: <https://docs.streamlit.io/develop/api-reference/charts>

Data outputs

```
st.dataframe(iris)
st.json({'Name':'Till','City':'Kiel'})
st.metric('Temperature', '25°', 2)
```

	sepalLength	sepalWidth	petalLength	petalWidth	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa

```
▼ {
  "Name": "Till"
  "City": "Kiel"
}
```

Temperature

25°

↑ 2

More data outputs: <https://docs.streamlit.io/develop/api-reference/data>

Text outputs

```
st.text('Fixed width text')

st.markdown('_Markdown_') # see *

st.latex(r''' e^{i\pi} + 1 = 0 ''')

st.write('Most objects') # df, err, func, keras!

st.write(['st', 'is <', 3]) # see *

st.title('My title')

st.header('My header')

st.subheader('My sub')

st.code('for i in range(8): foo()')

* optional kwarg unsafe_allow_html = True
```

Fixed width text
Markdown
 $e^{i\pi} + 1 = 0$

Most objects

```
[  
  0 : "st"  
  1 : "is <"  
  2 : 3  
]
```

My title

My header

My sub

```
for i in range(8): foo()
```

More text outputs: <https://docs.streamlit.io/develop/api-reference/text>

Caching

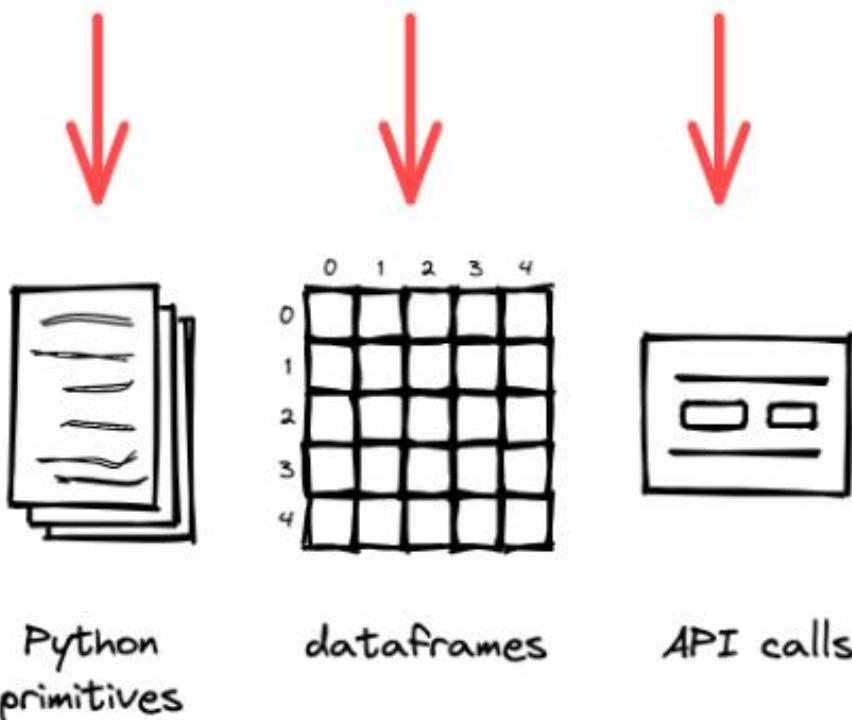
- ▶ Purpose: Avoid long running re-computations of static objects
- ▶ How:
 - ◆ Wrap long-running operation into a function
 - ◆ Prepend cache decorator

```
@st.cache_data
def long_running_function(param1, param2):
    return ...
```

Caching

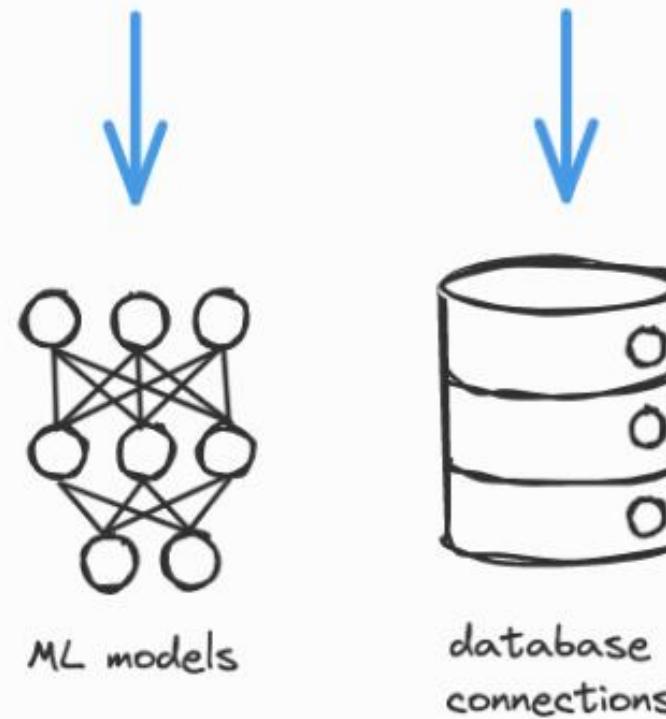
st.cache_data

anything you CAN store in a database



st.cache_resource

anything you CAN'T store in a database



Layout



A sidebar component with a red dashed border. Inside are several red rectangular sections of varying heights, representing scrollable content.

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Duis quis enim lobortis scelerisque fermentum dui [faucibus](#). Pharetra magna ac placerat vestibulum lectus mauris ultrices.

Adipiscing [elit duis tristique](#) euismod, velit aliquet sagittis id consectetur purus et facilisis pulvinar.

Tincidunt lobortis

Feugiat vivamus at augue eget arcu dictum. Sed risus pretium quam vulputate. Curiosus in hac habitasse platis dictur. Aliquam ultrices sagittis ante.

Non diam phasellus vestibulum

Vel quam elementum pulvinar etiam. Blanckt vulputate maecenas vulputate blandit aliquam. Et sit amet facilisis magna etiam tempor arcu et lobortis.

- Ut pharetra sit amet aliquam id.
- Etiam tempor orci et lobortis elementum nibh tellus molestie nunc.
- Sed erint ut sem viverra aliquip eget id.
- Duis at consectetur lorem donec massa sapien faucibus et molestie.

Sans integer vitae justo eget. In agoran erat impendiad sed euismod nisi porta lorem mollis. Sit feugiat pretium nibh ipsum consequat risus vel pretium. Et sit aliquip purus sit amet. Aliquet nibh present tenebris magna id. Sapphos ultrices in tacuisse nunc.

Etiam eu turpis egosim pretium venenatis pharetra. Nunc sed blandit libero volutpat sed etiam ornare arcu.

Sidebar

Display items in a sidebar.

```
st.sidebar.write("This lives in the  
st.sidebar.button("Click me!")")
```



A columns component with a purple dashed border. It contains two columns of different widths, each with a light purple background.

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Duis quis enim lobortis scelerisque fermentum dui [faucibus](#). Pharetra magna ac placerat vestibulum lectus mauris ultrices.

Columns

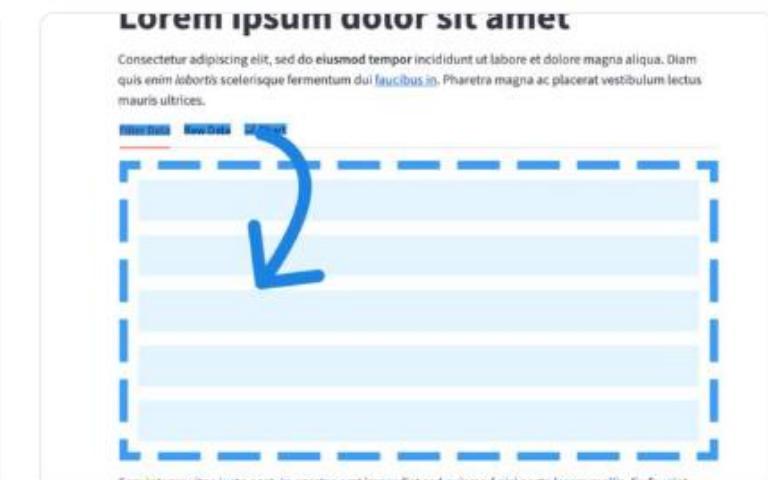
Insert containers laid out as side-by-side columns.

```
col1, col2 = st.columns(2)  
col1.write("this is column 1")  
col2.write("this is column 2")
```

Columns

Insert containers laid out as side-by-side columns.

```
col1, col2 = st.columns(2)  
col1.write("this is column 1")  
col2.write("this is column 2")
```



A tabs component with a blue dashed border. It shows two tabs: 'Tab 1' (selected) and 'Tab 2'. A blue arrow points from the text area to the tab indicator.

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Duis quis enim lobortis scelerisque fermentum dui [faucibus](#). Pharetra magna ac placerat vestibulum lectus mauris ultrices.

Tabs

Insert containers separated into tabs.

```
tab1, tab2 = st.tabs(["Tab 1", "Tab  
tab1.write("this is tab 1")  
tab2.write("this is tab 2")
```

Tabs

Insert containers separated into tabs.

```
tab1, tab2 = st.tabs(["Tab 1", "Tab  
tab1.write("this is tab 1")  
tab2.write("this is tab 2")
```

Multipage Apps

- ▶ Entrypoint file: hello.py
- ▶ Each page is represented as a separate script and placed in the pages subdirectory

The screenshot shows the Streamlit welcome page. On the left, a sidebar lists several demo scripts: "Hello" (selected), "Plotting Demo", "Mapping Demo", "DataFrame Demo", and "Webcam Demo". Below the sidebar, a green button says "Select a demo above.". The main content area features a large heading "Welcome to Streamlit!" with a yellow clapping hands emoji. It includes a descriptive paragraph about Streamlit's purpose and a call to action: "Select a demo from the sidebar to see some examples of what Streamlit can do!". Below this, a section titled "Want to learn more?" lists three items: "Check out [streamlit.io](#)", "Jump into our [documentation](#)", and "Ask a question in our [community forums](#)". Finally, a section titled "See more complex demos" lists two items: "Use a neural net to [analyze the Udacity Self-driving Car Image Dataset](#)" and "Explore a [New York City rideshare dataset](#)".

Welcome to Streamlit! 🙌

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects. ⏵ Select a demo from the sidebar to see some examples of what Streamlit can do!

Want to learn more?

- Check out [streamlit.io](#)
- Jump into our [documentation](#)
- Ask a question in our [community forums](#)

See more complex demos

- Use a neural net to [analyze the Udacity Self-driving Car Image Dataset](#)
- Explore a [New York City rideshare dataset](#)

Page Configuration

```
st.set_page_config(  
    page_title="Ex-stream-ly Cool App",  
    page_icon="🧊",  
    layout="wide",  
    initial_sidebar_state="expanded",  
    menu_items={  
        'Get Help': 'https://www.extremelycoolapp.com/help',  
        'Report a bug': 'https://www.extremelycoolapp.com/bug',  
        'About': "# This is a header. This is an *extremely* cool app!"  
    }  
)
```

App Deployment

Apps can be easily hosted and shared via [Streamlit Community Cloud](#)

 Deploy in one click Your fully hosted app is ready to share in under a minute.	 Keep your code in your repo No changes to your development process. Code stays on GitHub.	 Live updates Your apps update instantly when you push code changes.
 Securely connect to data Connect to all your data sources using secure protocols.	 Restrict access to apps Authenticate viewers with per-app viewer allow-lists.	 Easily manage your apps View, collaborate, and manage all your apps in a single place.

App Deployment Workflow

- ▶ **GitHub Repo:** Streamlit Community Cloud launches apps directly from your GitHub repo, so your app code and dependencies need to be on GitHub before you try to deploy the app.
- ▶ **Requirements:** Include a requirements.txt file for Python package dependencies
- ▶ **Deploy** on Streamlit Community Cloud

Deploy an app

Apps are deployed directly from their GitHub repo. Enter the location of your app below.
Or [click here to fork and deploy a sample app.](#)

Repository Paste GitHub URL

Branch

Main file path

[Advanced settings...](#)

Deploy!

App Settings

App settings

General

Sharing

Secrets

App URL

Pick a custom subdomain for your app's URL. The default URL is based on the app's location in GitHub.

fhkiel-dataviz-wdi .streamlit.app

Save

App settings

General

Sharing

Secrets

Who can manage this app

Anyone with push access to this app's repo can manage this app.

[View or change access in GitHub →](#)

Who can view this app

This app is public and searchable

Invite viewers by email

Enter a comma-separated list of emails.

person1@example.com
person2@example.com

Save

Streamlit Resources

- ▶ **Cheat Sheet:** <https://cheat-sheet.streamlit.app/>
- ▶ **Gallery:** <https://streamlit.io/gallery>
- ▶ **Explore Streamlit Apps:** <https://share.streamlit.io/explore>
- ▶ **Third Party Extensions:** <https://streamlit.io/components>

Challenges of interactive visualization

- ▶ Risk of **overwhelming** the user with too much information
- ▶ **Little control** over what the viewer sees and interprets
- ▶ Bad user experience due to
 - ◆ Long **loading times**
 - ◆ **No data available** due to user choices
 - ◆ **Bad axis representation** due to user choices
 - ◆ ...

Recommendations

- ▶ Focus on **user needs**
- ▶ Use **interactivity** sparingly and with purpose
- ▶ Optimize **performance**
- ▶ Ask **users** to test the application