



Turn your Python scripts
into beautiful **data apps**

Let's build some **apps!**



Randy Zwitch

Head of Developer Relations

GitHub: randyzwitch

Twitter: @randyzwitch

Part 1

**What is
Streamlit?**

Part 2

**Install and
play now!**

Part 3

**Let's build
a data app**



Part 1

**What is
Streamlit?**

Part 2

**Install and
play now!**

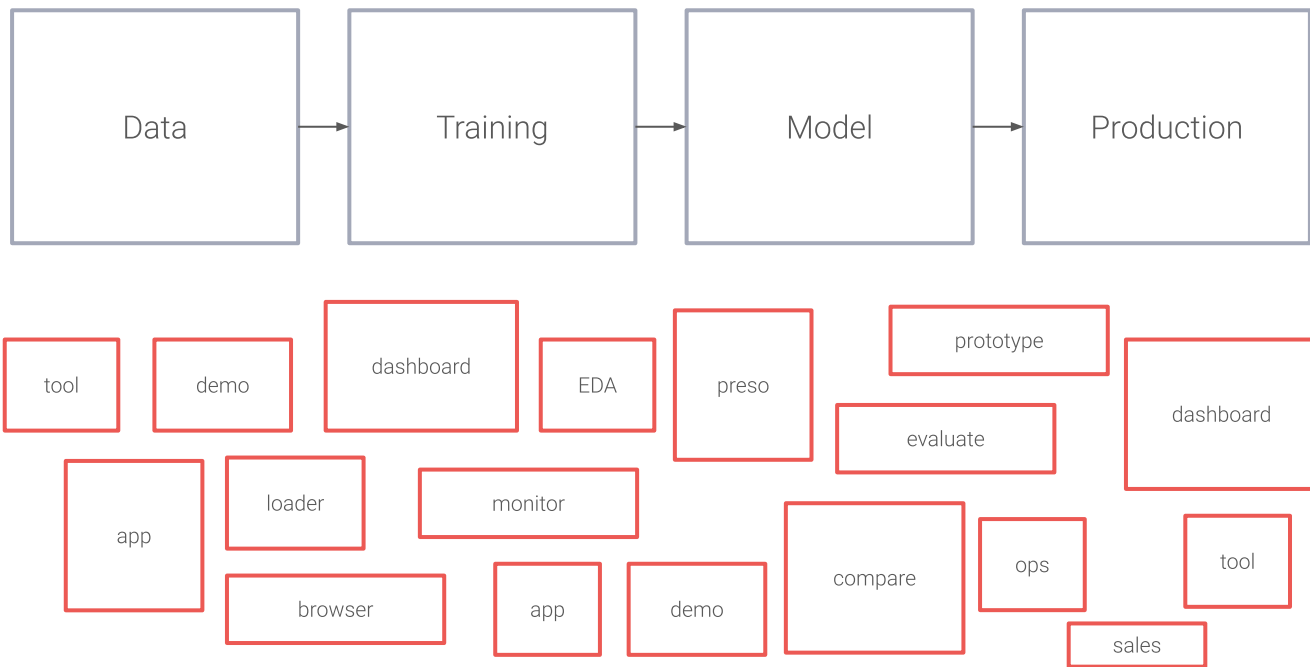
Part 3

**Let's build
a data app**



What **apps** do data scientists need?

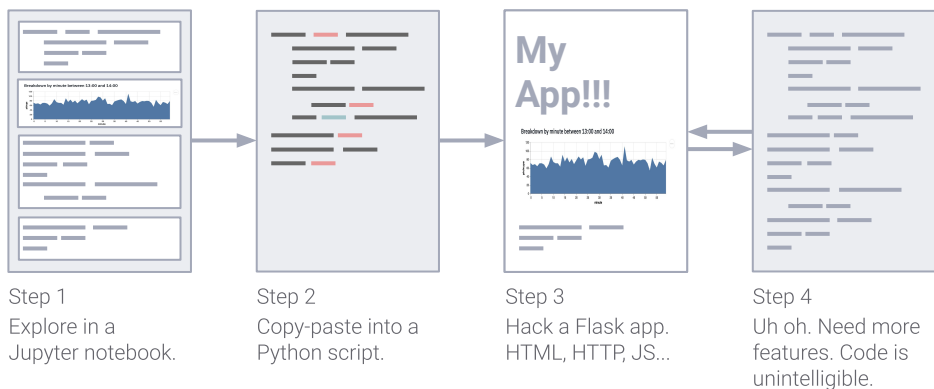
Lots! You're always making apps for your work or to communicate your results to others.



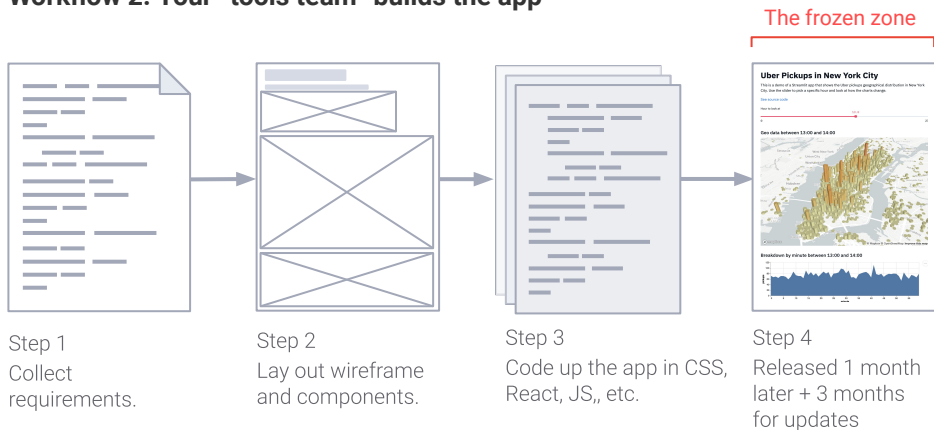
Building data apps is slow and expensive

Building even simple data apps today requires weeks or months of investment, distracts from core work, and often yields an unmaintainable product. And because it's so costly **only a fraction of needed apps and tools are created**.

Workflow 1: You build the app



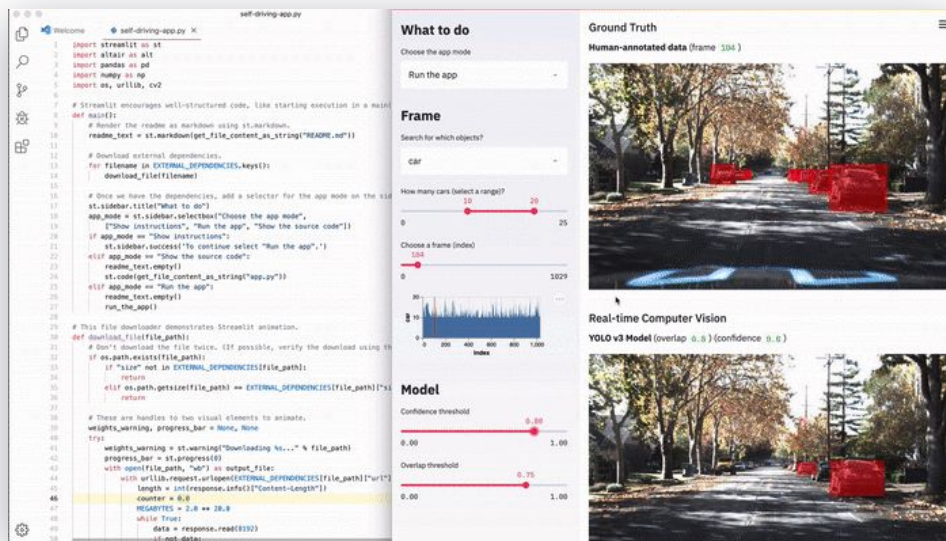
Workflow 2: Your “tools team” builds the app



Streamlit is the **fastest** **and easiest** way to create data apps

Quickly create your own elegant data apps for visualization, debugging, comparing models and presenting data - all in Python.

Streamlit's open-source app framework is built specifically for data scientists to rapidly create beautiful, performant apps in only a few hours!



Streamlit works on **3 simple principles**



Embrace
Python scripting

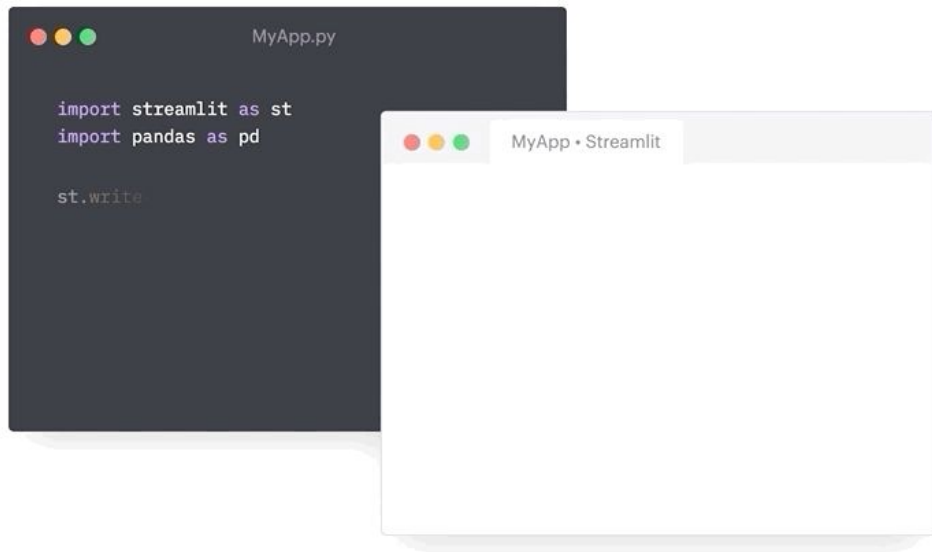


Treat widgets
as variables.

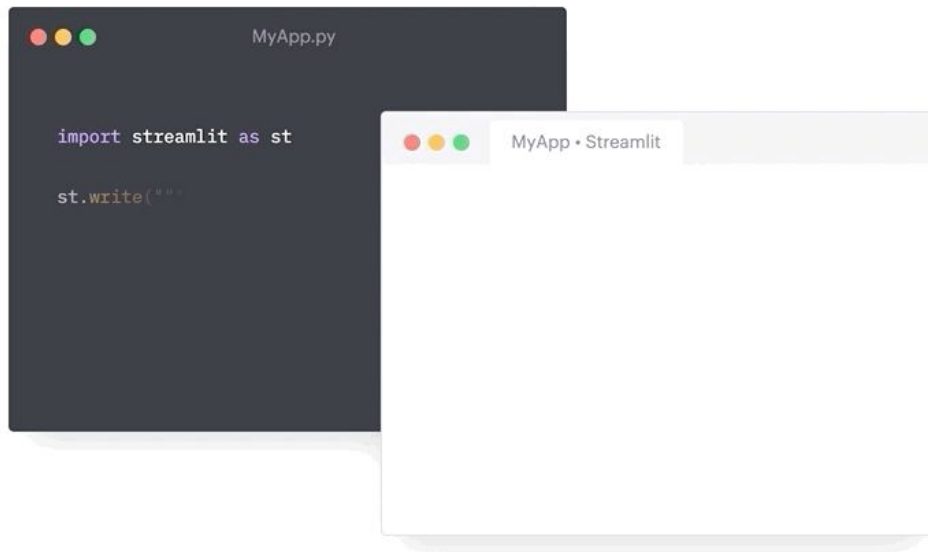


Reuse data and
computation.

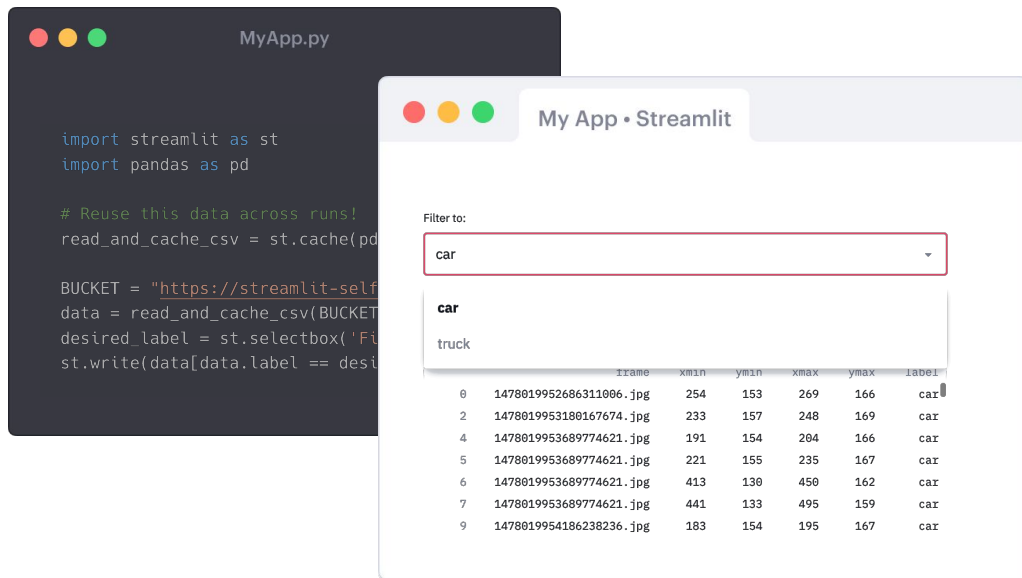
Embrace Python scripting



Treat **widgets** as variables



Reuse data and computation



The image shows a code editor window titled 'MyApp.py' and a Streamlit application window titled 'My App • Streamlit'.

Code Editor (MyApp.py):

```
import streamlit as st
import pandas as pd

# Reuse this data across runs!
read_and_cache_csv = st.cache(pd.read_csv)

BUCKET = "https://streamlit-self/"
data = read_and_cache_csv(BUCKET)
desired_label = st.selectbox('Filter to:', data['label'].unique())
st.write(data[data.label == desired_label])
```

Streamlit Application (My App • Streamlit):

Filter to:

car

car

truck

	irame	xmin	ymin	xmax	ymax	label
0	1478019952686311006.jpg	254	153	269	166	car
2	1478019953180167674.jpg	233	157	248	169	car
4	1478019953689774621.jpg	191	154	204	166	car
5	1478019953689774621.jpg	221	155	235	167	car
6	1478019953689774621.jpg	413	130	450	162	car
7	1478019953689774621.jpg	441	133	495	159	car
9	1478019954186238236.jpg	183	154	195	167	car

Part 1

**What is
Streamlit?**

Part 2

**Install and
play now!**

Part 3

**Let's build
a data app**



Part 1

**What is
Streamlit?**

Part 2

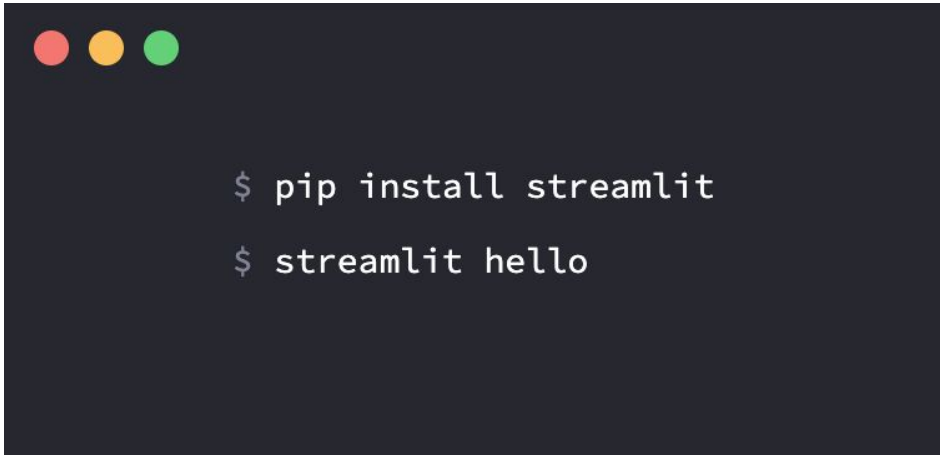
**Install and
play now!**

Part 3

**Let's build
a data app**

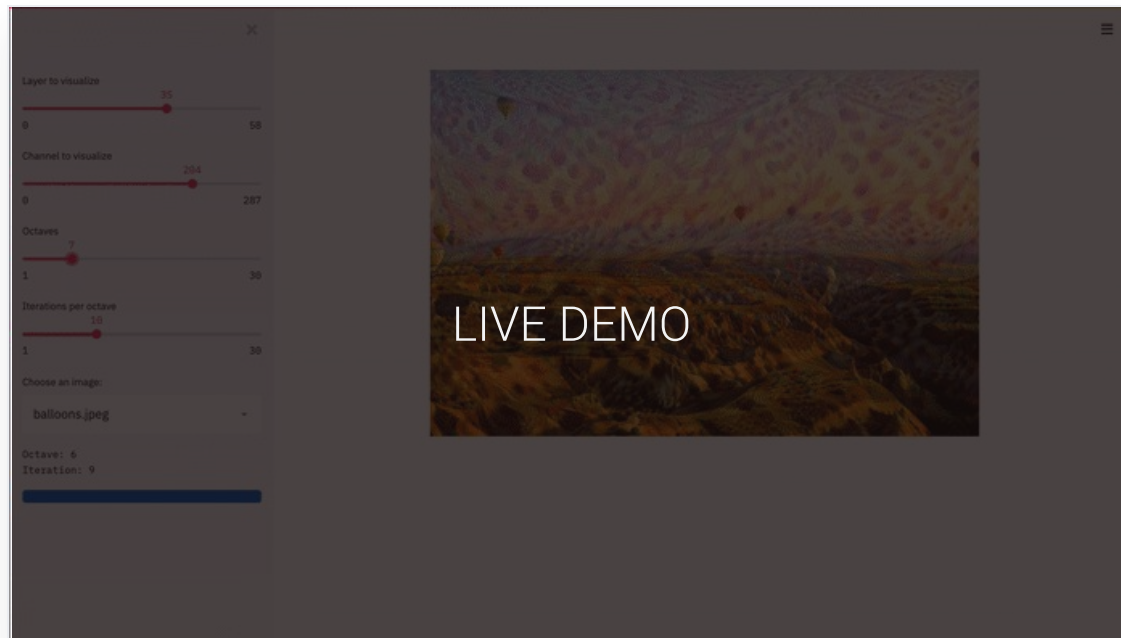


Get started **instantly**



```
$ pip install streamlit  
$ streamlit hello
```

Let's take a look at streamlit hello



Part 1

**What is
Streamlit?**

Part 2

**Install and
play now!**

Part 3

**Let's build
a data app**



Part 1

**What is
Streamlit?**

Part 2

**Install and
play now!**

Part 3

**Let's build
a data app**



Time for some **live coding!**

Let's take a look at some common methods for interactivity in Streamlit

SideBar slider

1100

SideBar slider value is 1

SF Big Analytics - AICamp

Source code

```
import streamlit as st

# markdown supported inline as a string. Streamlit "magic" interprets the script to detect
# ...

"""# 1. Slider example"""

# including a slider as simple as giving it a header label and optional bounds
slider1 = st.slider("Pick a number 1 to 100", 1, 100)

# ""The number you selected was {slider1}""

# moving sliders to the sidebar done by adding "sidebar" to the method name
slider2 = st.sidebar.slider("Sidebar slider", 1, 100)

# can pass one slider output to another as default (though, not sure why you might!)
# slider2 = st.sidebar.slider("Sidebar slider", slider1, 100, slider1)

st.sidebar.markdown(f"Sidebar slider value is {slider2}")

# ...

"""# 2. Displaying data example"""

# common data science libraries part of "magic"
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(100, 3))
"""pandas dataframes are auto-recognized just like this markdown statement"""
df

"""# Why are this tables above changing each time anything changes? Because Streamlit is
# ...

# st.cache works based on memoization...function with the same inputs should always return
# return value from cache instead of re-calculating (potentially expensive) function
@st.cache()
def gm_rand_data(cols):
    return pd.DataFrame(np.random.randn(100, cols))

df2 = gm_rand_data(slider2)

"""# 3. Caching example """
"""only changing the slider in the sidebar will change this table"""
df2
```

streamlit-folium

<https://github.com/randyzwitch/streamlit-folium>

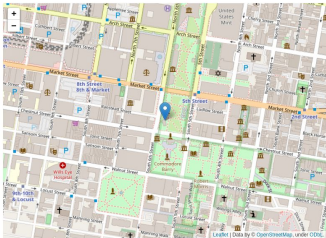
This app highlights the components capabilities in Streamlit 0.63+ Users can extend Streamlit to work with their favorite 3rd-party libraries

```
import streamlit as st
from streamlit_folium import folium_static
import folium

# center on Liberty Bell
m = folium.Map(location=[39.449619, -75.150282], zoom_start=16)

# add marker for Liberty Bell
tooltip = "Liberty Bell"
folium.Marker(
    [39.449619, -75.150282], popup=tooltip, tooltip=tooltip
).add_to(m)

# call to render Folium map in Streamlit
folium_static(m)
```



Try it out and let us know what you think

Visit our website to find more examples, docs, and our community forum.

<https://streamlit.io>

<https://discuss.streamlit.io>

