Streamlit

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.

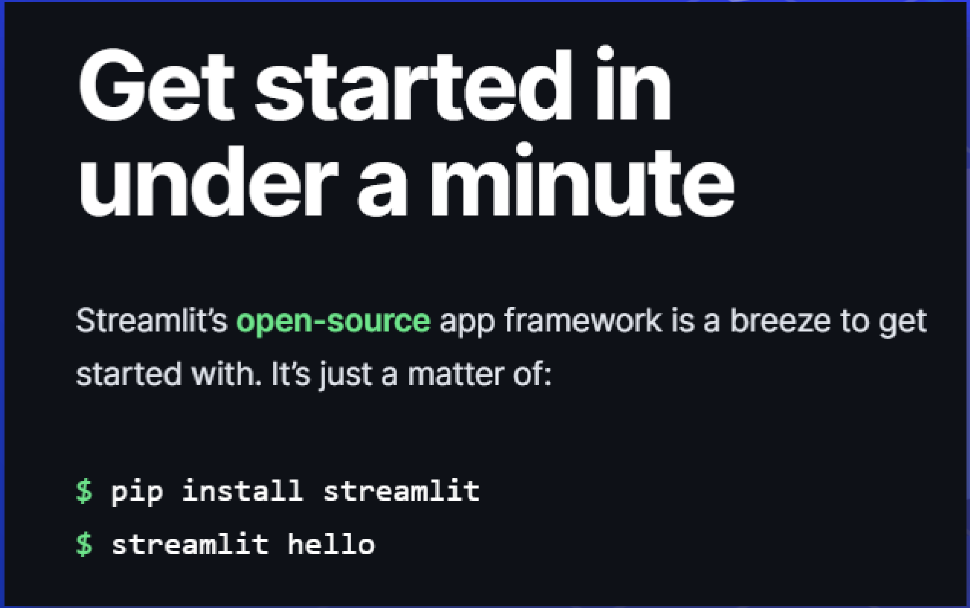
.py

My Favorite
Text Editor

Agenda

- Streamlit Get Started
- Write and Magic
- Text Elements
- Data Display
- Chart Elements
- Input Widgets
- Complex Layouts
- Media Elements
- Display Progress and Status
- Themes
- Multi-Pages
- Deployment

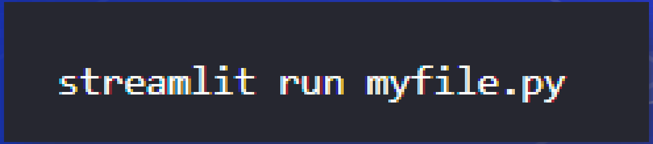
Streamlit Get Started



Get started in under a minute

Streamlit's open-source app framework is a breeze to get started with. It's just a matter of:

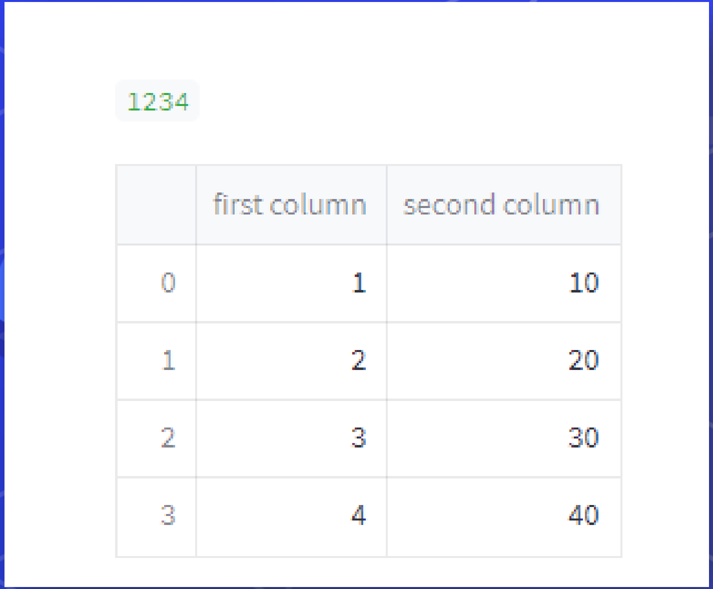
```
$ pip install streamlit
$ streamlit hello
```



```
streamlit run myfile.py
```

Write

```
st.write(1234)
st.write(pd.DataFrame({
     'first column': [1, 2, 3, 4],
     'second column': [10, 20, 30, 40],
 }))
```

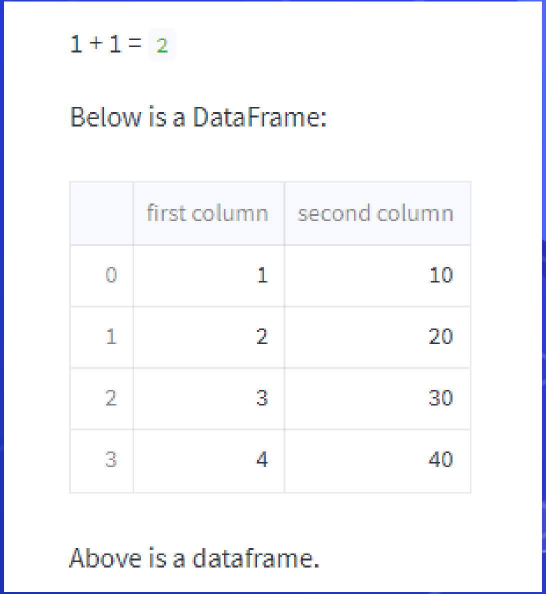


```
1234
```

	first column	second column
0	1	10
1	2	20
2	3	30
3	4	40

Write

```
st.write('1 + 1 = ', 2)
st.write('Below is a DataFrame:', data_frame, 'Above is a dataframe.')
```



1 + 1 = 2

Below is a DataFrame:

	first column	second column
0	1	10
1	2	20
2	3	30
3	4	40

Above is a dataframe.

Write

→ Some Options:

- write(string)
- write(data_frame)
- write(error) : Prints an exception specially.
- write(func) : Displays information about a function.
- write(module) : Displays information about the module.
- write(dict) : Displays dict in an interactive widget.
- write(plotly_fig) : Displays a Plotly figure.

Magic

- Magic commands are a feature in Streamlit that allows you to write almost anything (markdown, data, charts) without having to type an explicit command at all.
- Just put the thing you want to show on its own line of code, and it will appear in your app.
- Any time Streamlit sees either a variable or literal value, it automatically writes that to your app using *st.write.*
- Also, magic is smart enough to ignore docstrings.

Magic

```python
# Draw a title and some text to the app:
'''
# This is the document title

This is some _markdown_.
'''

import pandas as pd
df = pd.DataFrame({'col1': [1,2,3]})
df  # 👈 Draw the dataframe

x = 10
'x', x  # 👈 Draw the string 'x' and then the value of x
```

```python
# Also works with most supported chart types
import matplotlib.pyplot as plt
import numpy as np

arr = np.random.normal(1, 1, size=100)
fig, ax = plt.subplots()
ax.hist(arr, bins=20)

fig  # 👈 Draw a Matplotlib chart
```

Text Elements

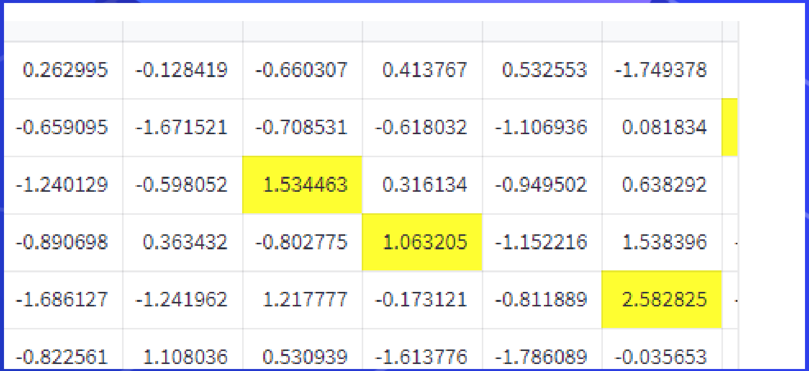
- Streamlit apps usually start with a call to *st.title* to set the app's title.
- After that, there are 2 heading levels you can use: *st.header* and *st.subheader*.
- Pure text is entered with *st.text*, and Markdown with *st.markdown*.
- You can also display a code format using *st.code*

Data Display

- *st.table* : Display a static table.
- To display a dataframe as an interactive table: *st.dataframe(data , width , height)*

```
st.dataframe(df)  # Same as st.write(df)
```

```
df = pd.DataFrame(
    np.random.randn(10, 20),
    columns=('col %d' % i for i in range(20)))

st.dataframe(df.style.highlight_max(axis=0))
```



0.262995	-0.128419	-0.660307	0.413767	0.532553	-1.749378
-0.659095	-1.671521	-0.708531	-0.618032	-1.106936	0.081834
-1.240129	-0.598052	1.534463	0.316134	-0.949502	0.638292
-0.890698	0.363432	-0.802775	1.063205	-1.152216	1.538396
-1.686127	-1.241962	1.217777	-0.173121	-0.811889	2.582825
-0.822561	1.108036	0.530939	-1.613776	-1.786089	-0.035653

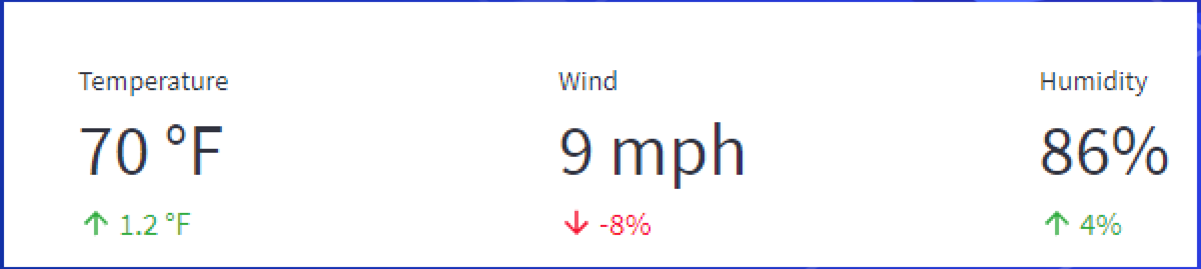
Data Display: Interactivity

- Column sorting.
- Column resizing
- Table (height, width) resizing
- Search: using hotkeys (Ctrl + F)
- Copy to clipboard

Data Display

- *st.metric :* Display a metric in big bold font, with an optional indicator of how the metric changed.

```python
col1, col2, col3 = st.columns(3)
col1.metric("Temperature", "70 °F", "1.2 °F")
col2.metric("Wind", "9 mph", "-8%")
col3.metric("Humidity", "86%", "4%")
```



```
Temperature      Wind       Humidity
70 °F            9 mph      86%
↑ 1.2 °F         ↓ -8%      ↑ 4%
```

Chart Elements

- Streamlit supports several different charting libraries.
- Right now, the most basic library is *Matplotlib*.
- Then there are also interactive charting libraries like *Plotly*.
- And finally, it also provides a few chart types that are *"native"* to Streamlit, like *st.line_chart* and *st.bar_chart*.

Chart Elements

- *st.plotly_chart* :

  Display an interactive

  Plotly chart.

```
# Create distplot with custom bin_size
fig = ff.create_distplot(
         hist_data, group_labels, bin_size=[.1, .25, .5])

# Plot!
st.plotly_chart(fig, use_container_width=True)
```

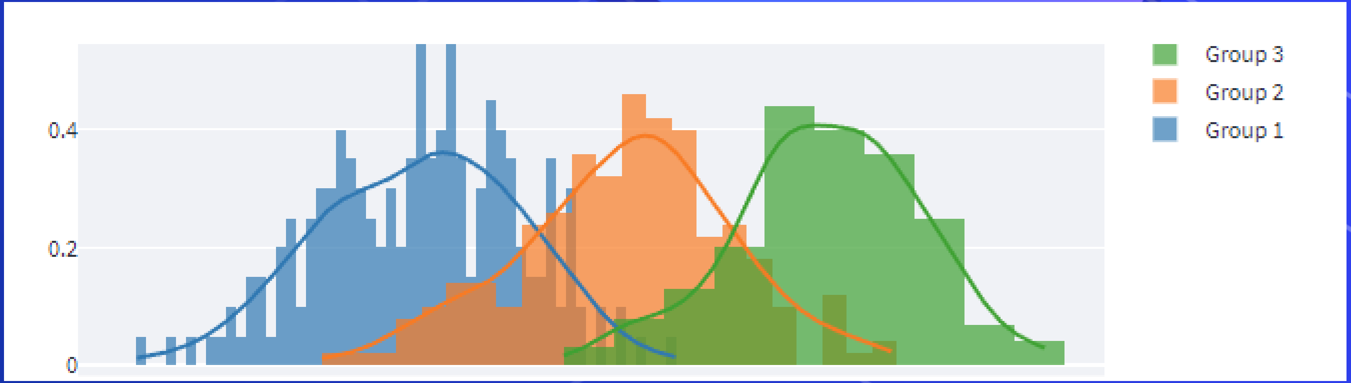


Chart Elements

- *st.line_chart* : Display a line chart
- *st.bar_chart* : Display a bar chart

```
chart_data = pd.DataFrame(
     np.random.randn(20, 3),
     columns=['a', 'b', 'c'])

st.line_chart(chart_data)
```

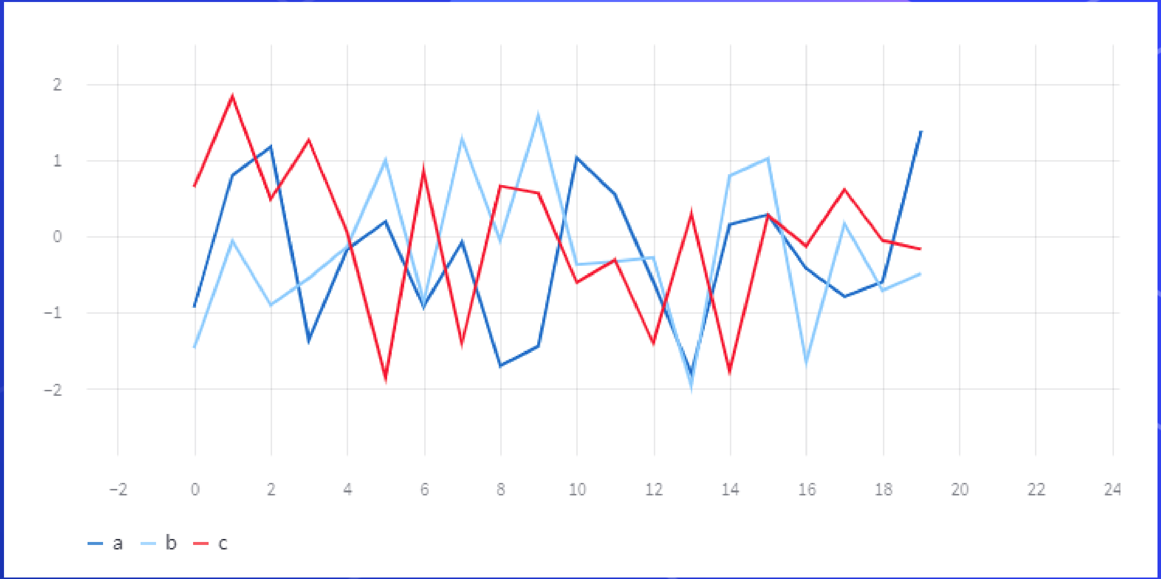
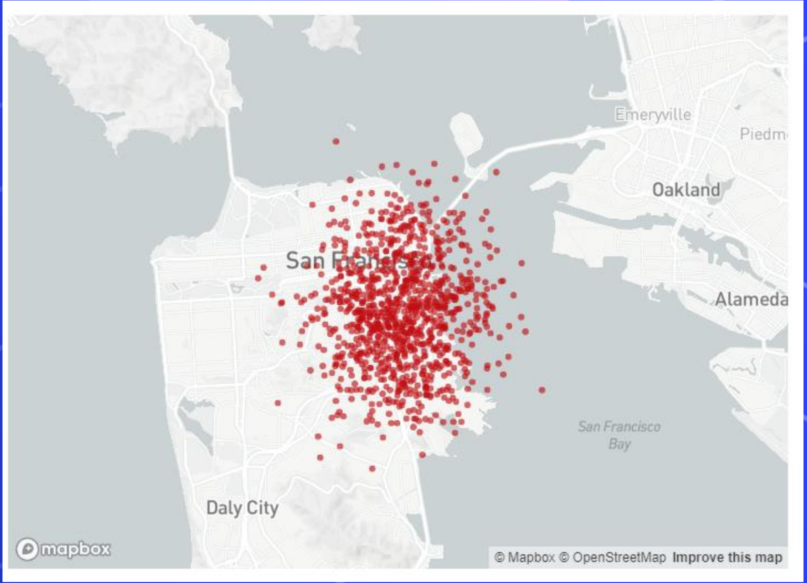


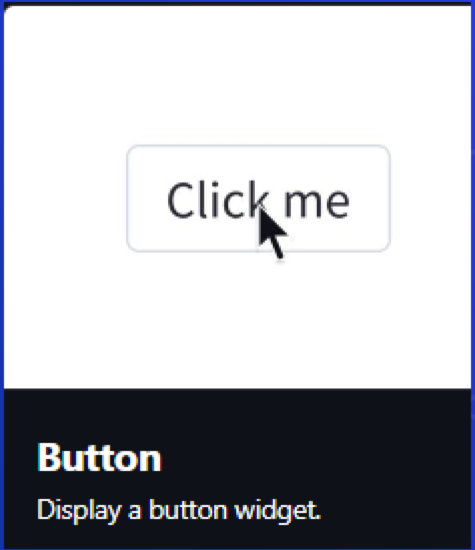
Chart Elements

- *st.map* : Display a map with points on it
- The data must have columns called *'lat', 'lon',* or *'latitude', 'longitude'.*

```
df = pd.DataFrame(
    np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],
    columns=['lat', 'lon'])

st.map(df)
```



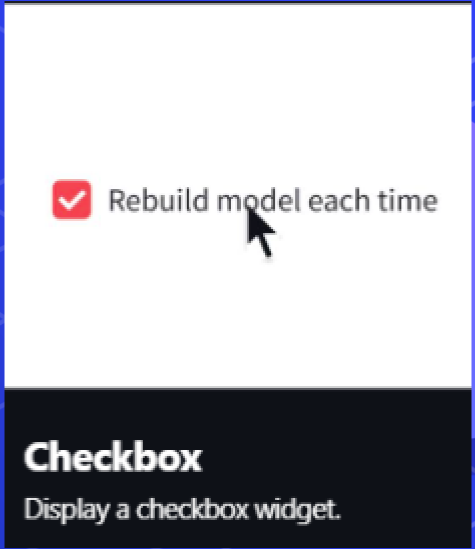
Input Widgets



Button

Display a button widget.

```
if st.button('Say hello'):
     st.write('Why hello there')
```

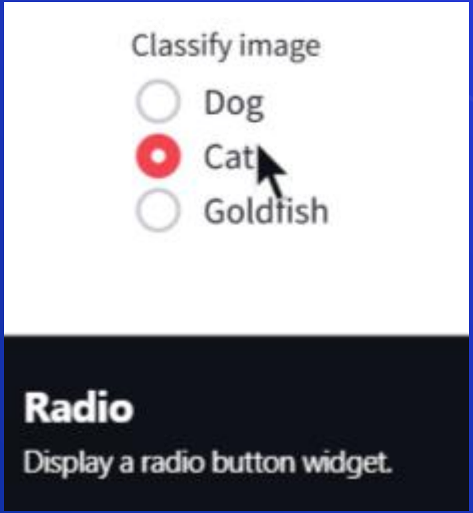


Checkbox

Display a checkbox widget.

```
agree = st.checkbox('I agree')

if agree:
     st.write('Great!')
```

Input Widgets



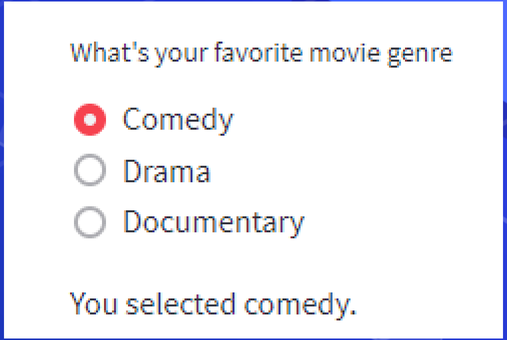
Classify image

Dog
[✓ Cat]
Goldfish

Radio

Display a radio button widget.

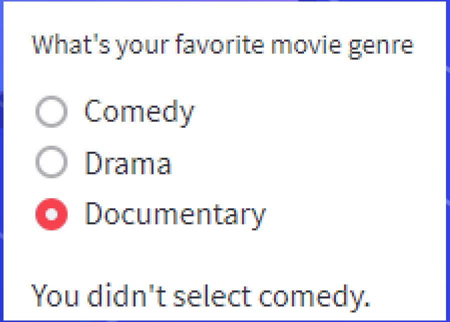
```
genre = st.radio(
      "What's your favorite movie genre",
      ('Comedy', 'Drama', 'Documentary'))

if genre == 'Comedy':
      st.write('You selected comedy.')
 else:
      st.write("You didn't select comedy.")
```



What's your favorite movie genre

[✓ Comedy]
Drama
Documentary

You selected comedy.

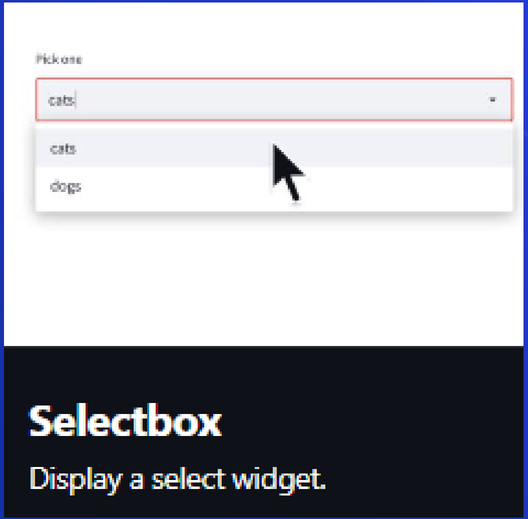


What's your favorite movie genre

Comedy
Drama
[✓ Documentary]

You didn't select comedy.

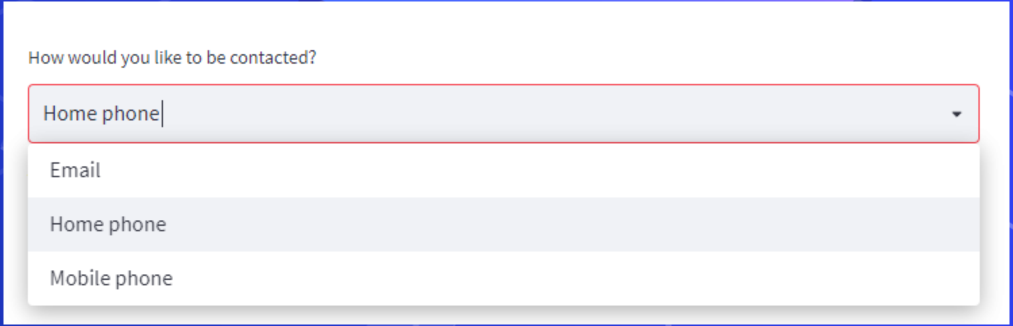
Input Widgets



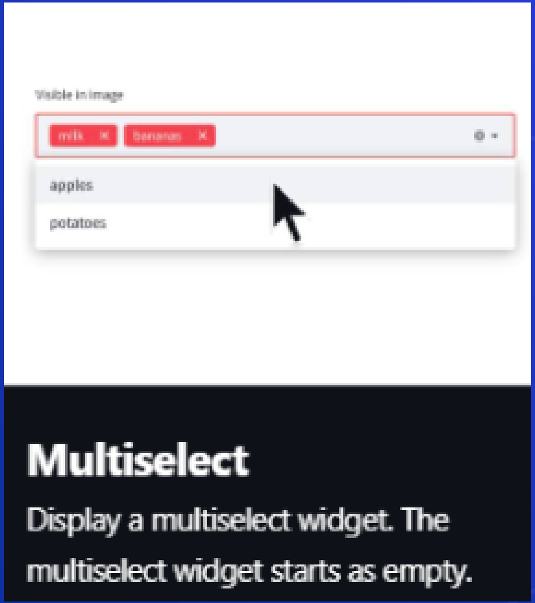
Selectbox

Display a select widget.

```python
option = st.selectbox(
     'How would you like to be contacted?',
     ('Email', 'Home phone', 'Mobile phone'))

st.write('You selected:', option)
```



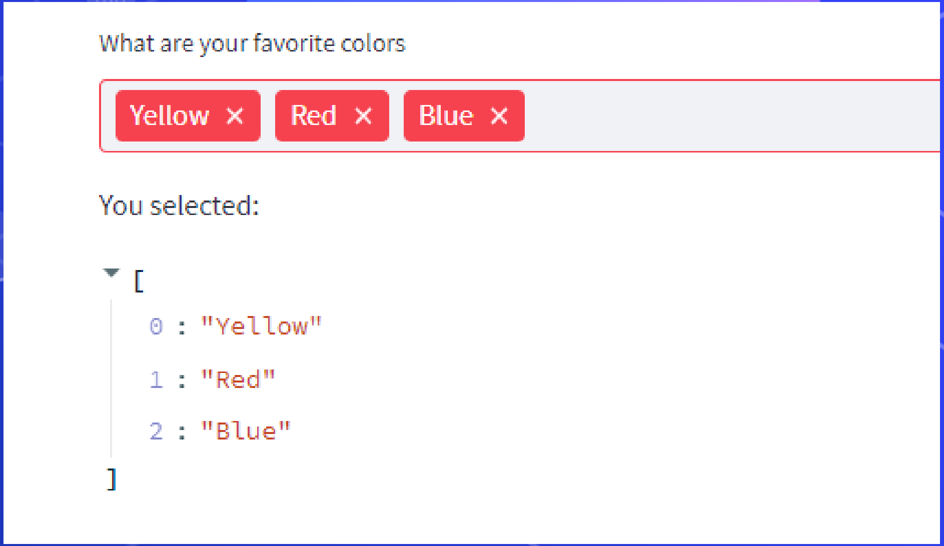
Input Widgets



Multiselect

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

```python
options = st.multiselect(
     'What are your favorite colors',
     ['Green', 'Yellow', 'Red', 'Blue'],
     ['Yellow', 'Red'])

st.write('You selected:', options)
```

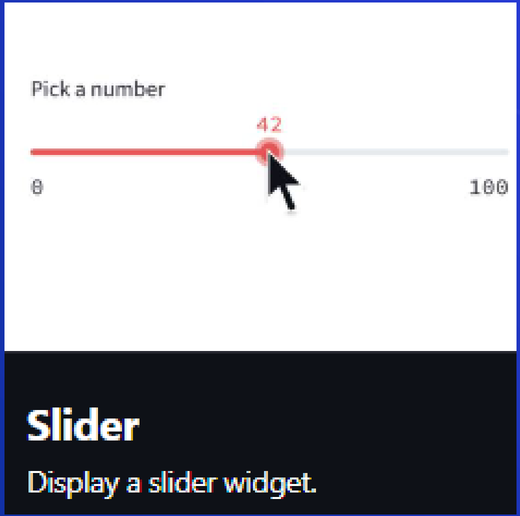


What are your favorite colors

You selected:

```
[
  0 : "Yellow"
  1 : "Red"
  2 : "Blue"
]
```

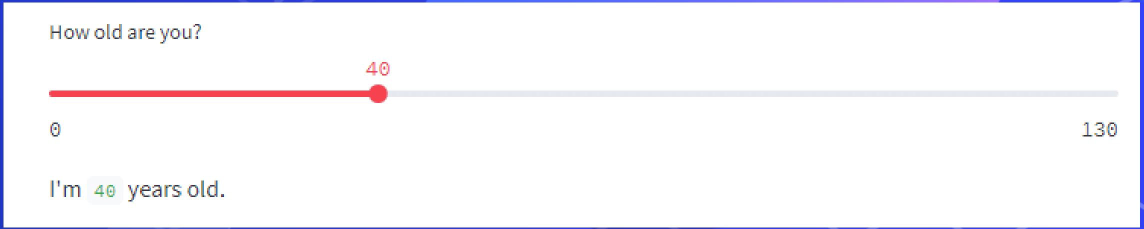
Input Widgets



Slider

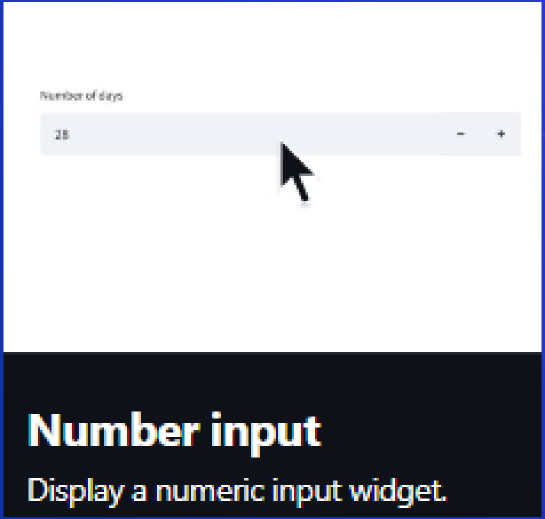
Display a slider widget.

```
age = st.slider('How old are you?', 0, 130, 25)
st.write("I'm ", age, 'years old')
```



I'm `40` years old.

Input Widgets

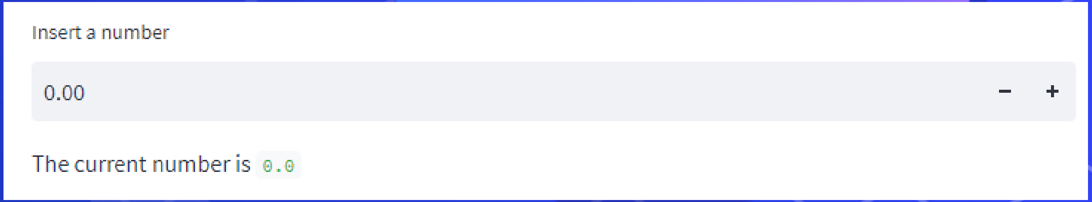

Number of days

28

Number input

Display a numeric input widget.

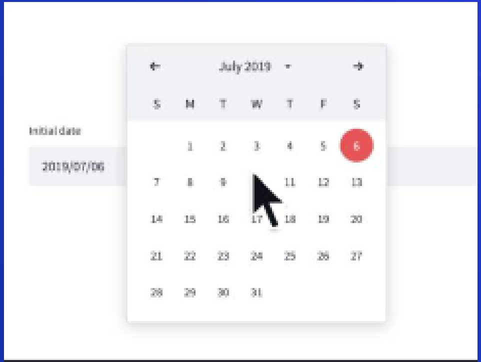
```
number = st.number_input('Insert a number')
st.write('The current number is ', number)
```


Insert a number

0.00

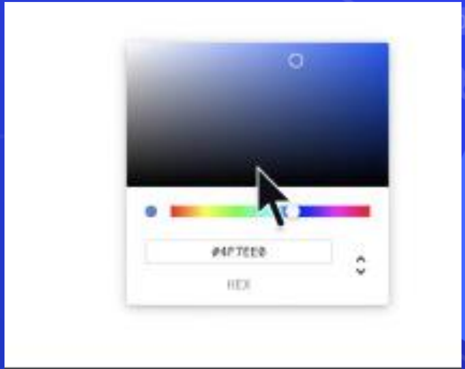
The current number is `0.0`

Input Widgets



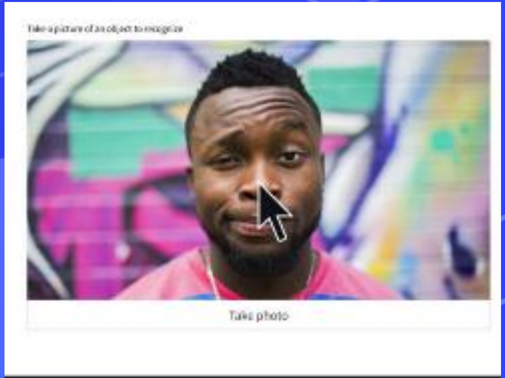
Date input

Display a date input widget.



Color picker

Display a color picker widget.



Camera input

Display a widget that allows users to upload images directly from a camera.

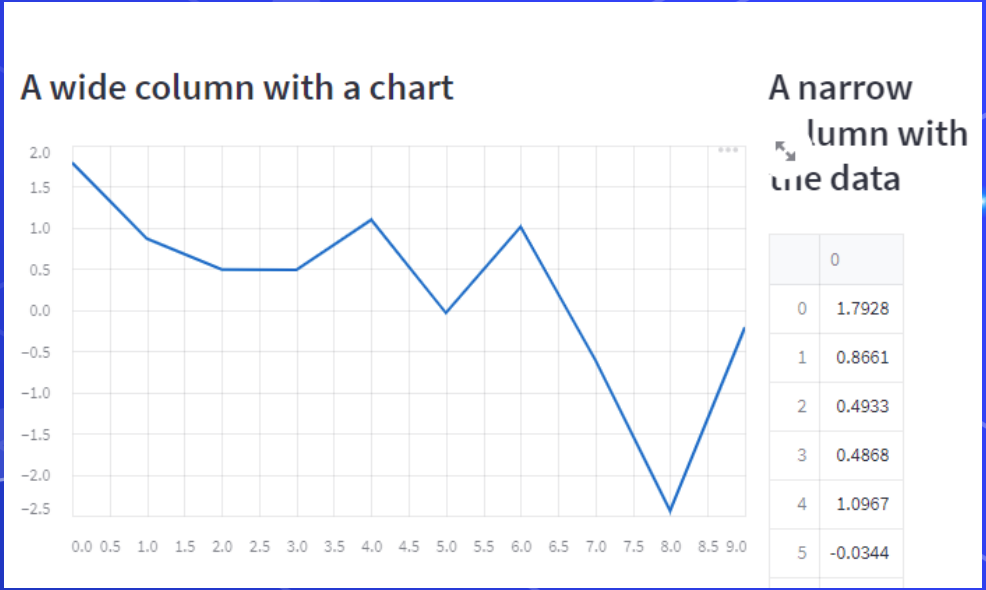
Complex Layouts : Sidebar

Lorem ipsum dolor sit amet
Tincidunt lobortis
Non diam phasellus vestibulum
Sidebar
Display items in a sidebar.

```python
# Using object notation
add_selectbox = st.sidebar.selectbox(
    "How would you like to be contacted?",
    ("Email", "Home phone", "Mobile phone")
)

# Using "with" notation
with st.sidebar:
    add_radio = st.radio(
        "Choose a shipping method",
        ("Standard (5-15 days)", "Express (2-5 days)")
    )
```

Complex Layouts: Columns

```
col1, col2 = st.columns([3, 1])
data = np.random.randn(10, 1)

col1.subheader("A wide column with a chart")
col1.line_chart(data)

col2.subheader("A narrow column with the data")
col2.write(data)
```



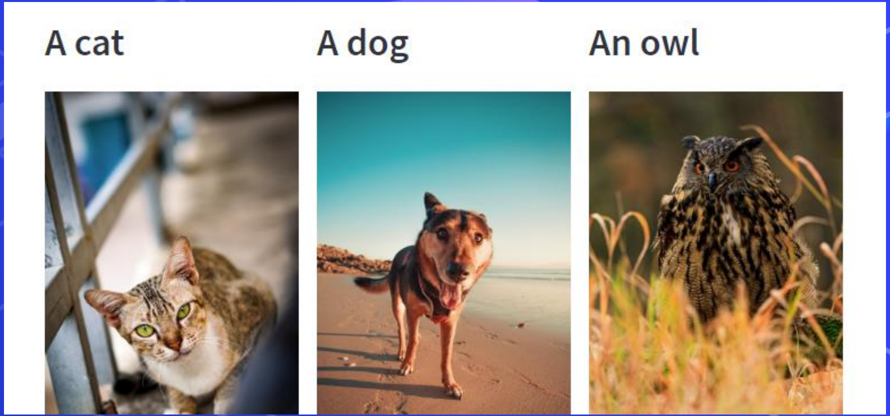
A wide column with a chart

A narrow lumn with ue data

	0
0	1.7928
1	0.8661
2	0.4933
3	0.4868
4	1.0967
5	-0.0344

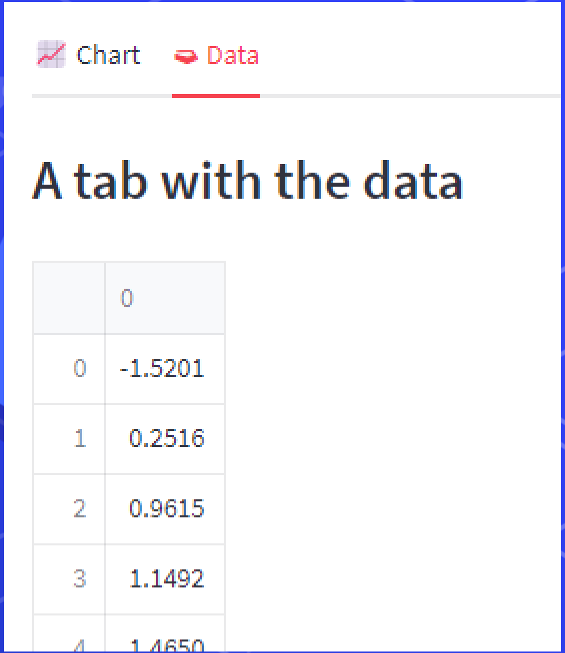
Complex Layouts: Columns

```
col1, col2, col3 = st.columns(3)

with col1:
    st.header("A cat")
    st.image("https://static.streamlit.io/examples/cat.jpg")

with col2:
    st.header("A dog")
    st.image("https://static.streamlit.io/examples/dog.jpg")

with col3:
    st.header("An owl")
    st.image("https://static.streamlit.io/examples/owl.jpg")
```



Complex Layouts: Tabs

```
tab1, tab2 = st.tabs(["📈 Chart", "🗃 Data"])
data = np.random.randn(10, 1)

tab1.subheader("A tab with a chart")
tab1.line_chart(data)

tab2.subheader("A tab with the data")
tab2.write(data)
```



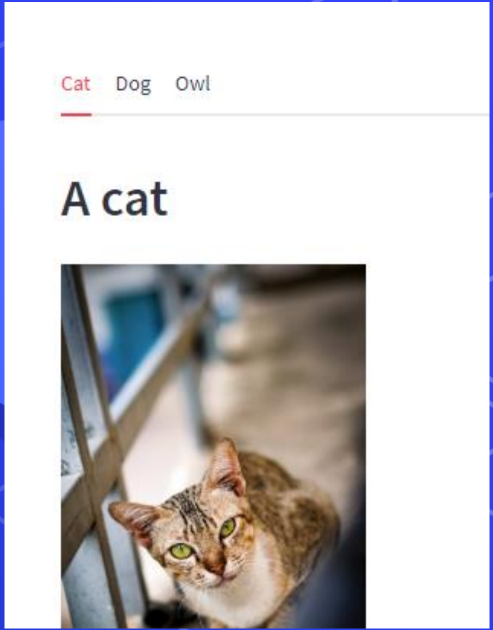
📈 Chart 🗃 Data

A tab with the data

	0
0	-1.5201
1	0.2516
2	0.9615
3	1.1492
4	1.4650

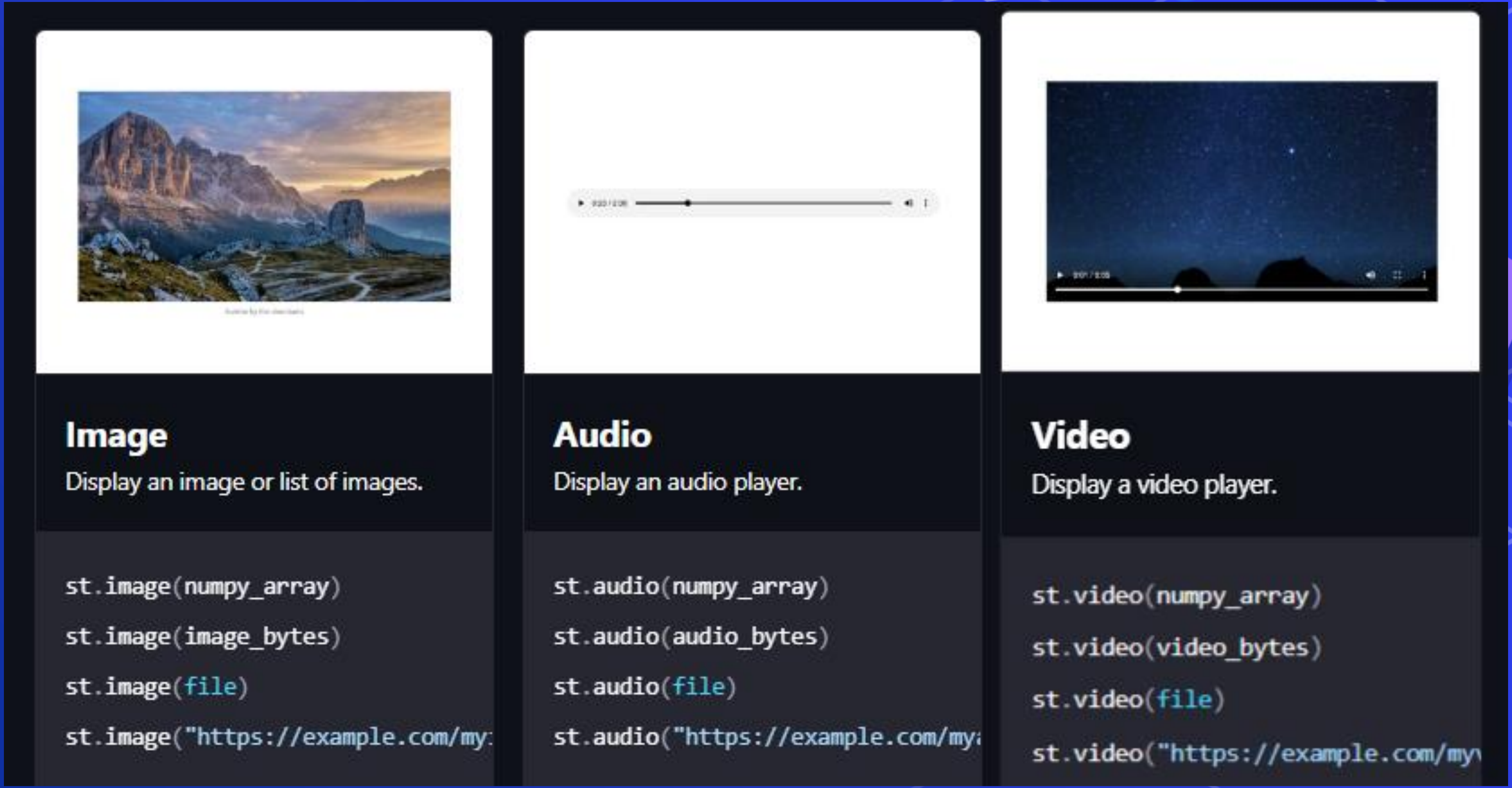
Complex Layouts: Tabs

```
tab1, tab2, tab3 = st.tabs(["Cat", "Dog", "Owl"])

with tab1:
    st.header("A cat")
    st.image("https://static.streamlit.io/examples/cat.jpg", width=200)

with tab2:
    st.header("A dog")
    st.image("https://static.streamlit.io/examples/dog.jpg", width=200)

with tab3:
    st.header("An owl")
    st.image("https://static.streamlit.io/examples/owl.jpg", width=200)
```



A cat

Media Elements



Image

Display an image or list of images.

```
st.image(numpy_array)
st.image(image_bytes)
st.image(file)
st.image("https://example.com/my
```

Audio

Display an audio player.

```
st.audio(numpy_array)
st.audio(audio_bytes)
st.audio(file)
st.audio("https://example.com/my
```

Video

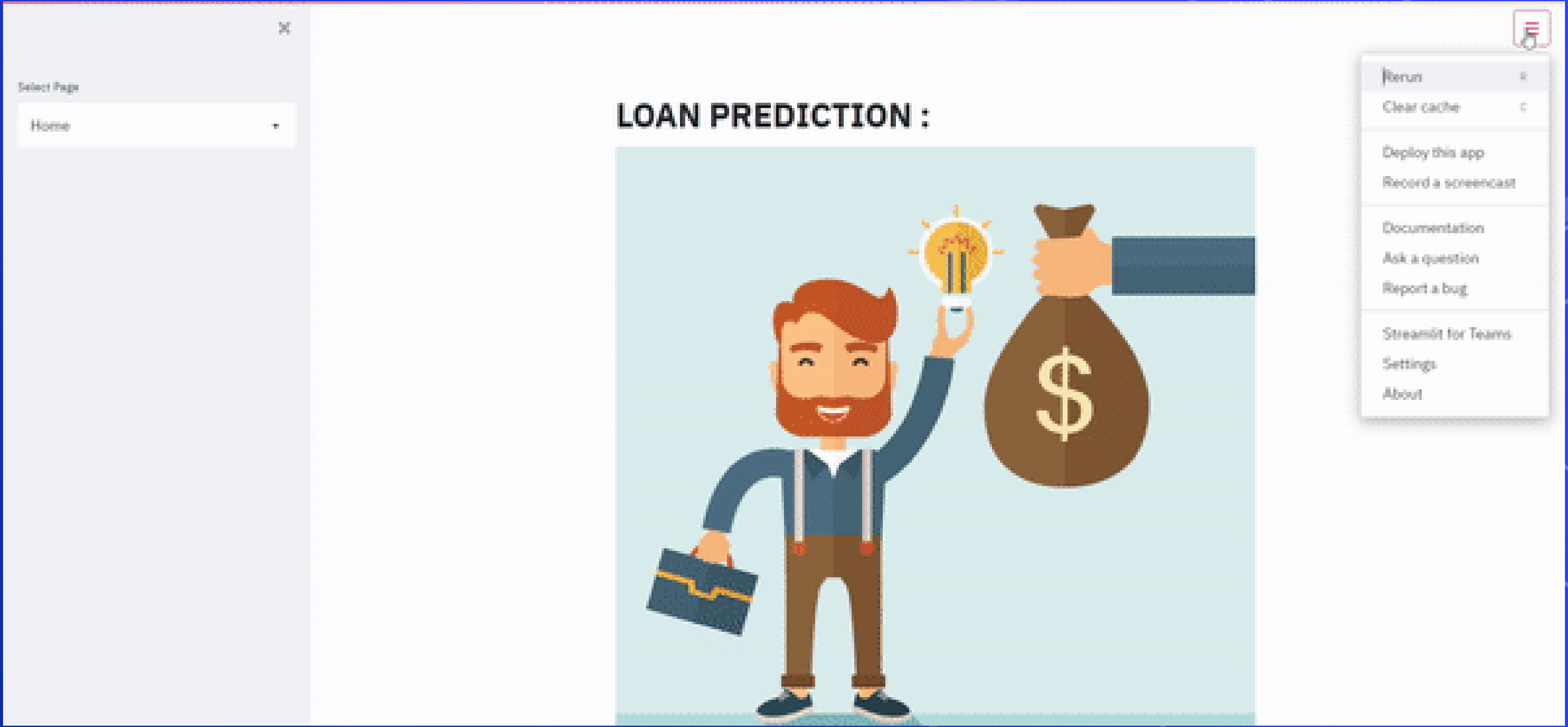
Display a video player.

```
st.video(numpy_array)
st.video(video_bytes)
st.video(file)
st.video("https://example.com/my
```

Display progress and status

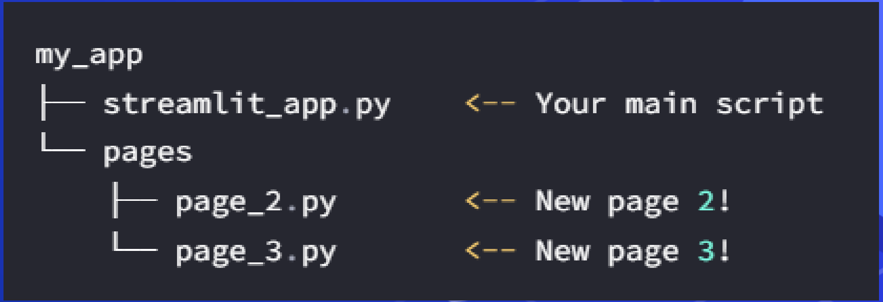
- Streamlit provides a few methods that allow you to add animation to your apps.
- These animations include progress bars, status messages (like warnings), and celebratory balloons.
- You can check here: https://docs.streamlit.io/library/api-reference/status

Themes


Select Page
Home
LOAN PREDICTION :
Rerun
Clear cache
Deploy this app
Record a screencast
Documentation
Ask a question
Report a bug
Streamlit for Teams
Settings
About

Multi-Pages

1. Create a main script named streamlit_app.py.
2. In the same folder, create a new pages folder.
3. Add new .py files in the pages folder. Your filesystem will look like this:



```
my_app
├── streamlit_app.py    <-- Your main script
└── pages
    ├── page_2.py       <-- New page 2!
    └── page_3.py       <-- New page 3!
```

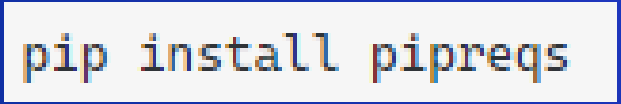
4. Run streamlit run streamlit_app.py as usual.

The streamlit app.py script will now correspond to your app's main page.

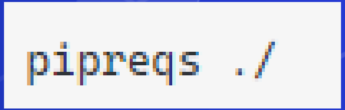
You'll see the other scripts from the pages folder in the sidebar page selector.

Deployment

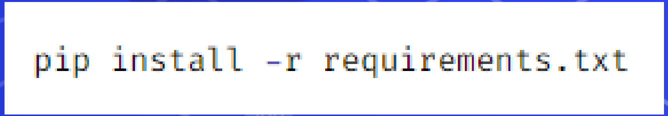
- *You must add all project files (.py) to a github repo including saved models and images.*
- *Add requirements.txt to avoid any dependencies issues*
- *It is also useful for anyone to install required modules to run your project on his machine*



```
pip install pipreqs
```



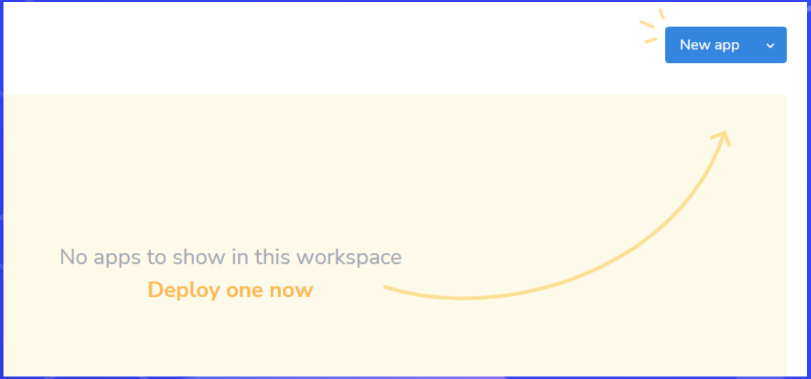
```
pipreqs ./
```

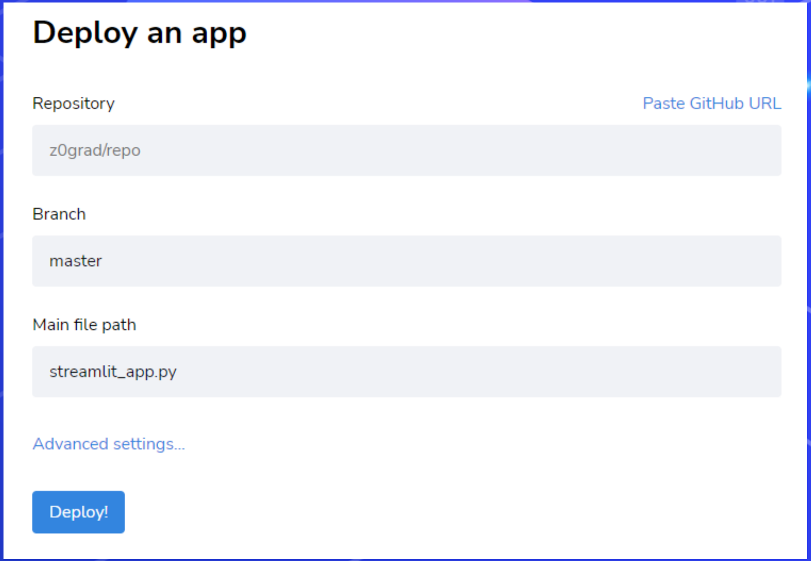


```
pip install -r requirements.txt
```

Deployment

- *Make a streamlit account*
- *Connect it with github*
- *You will have an empty workspace*
- *Click New app*
- *Add your Repo Path and your app.py File, or use **Paste Github URL** to put the github url of app.py*



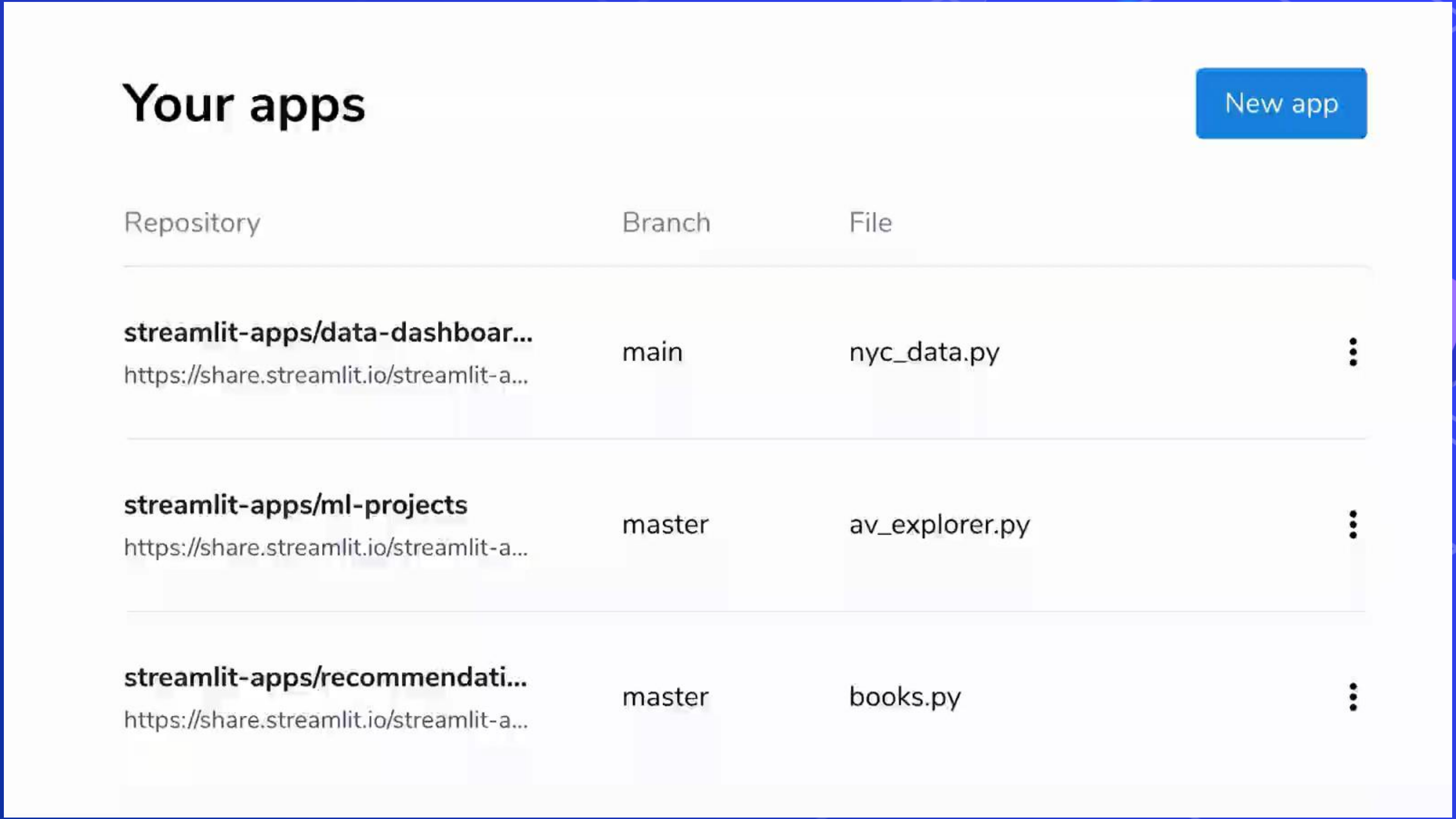

Deploy an app

Repository: z0grad/repo

Branch: master

Main file path: streamlit_app.py

Deployment



Your apps

Repository	Branch	File
streamlit-apps/data-dashboar... https://share.streamlit.io/streamlit-a...	main	nyc_data.py
streamlit-apps/ml-projects https://share.streamlit.io/streamlit-a...	master	av_explorer.py
streamlit-apps/recommendati... https://share.streamlit.io/streamlit-a...	master	books.py