Streamlit-LEARNING

Summary of the docs, as of Streamlit v1.25.0.

Install and import

```
$ pip install streamlit
```

```
# Import convention
>>> import streamlit as st
```

Add widgets to sidebar

```
# Just add it after st.sidebar:
>>> a = st.sidebar.radio('Choose
```

Magic commands

```
'_This_ is some __Markdown__'
a=3
'dataframe:', data
```

Command line

```
$ streamlit --help
$ streamlit run your_script.py
$ streamlit hello
$ streamlit config show
$ streamlit cache clear
$ streamlit docs
```

Streamlit-LEARNING

Display text

```
st.text('Fixed width text
st.markdown('_Markdown_')
st.caption('Balloons. Hunc
st.latex(r''' e^{i\pi} + 1
st.write('Most objects') #
st.write(['st', 'is <', 3]
st.title('My title')
st.header('My header')
st.subheader('My sub')
st.code('for i in range(8)

# * optional kwarg unsafe_
```

Display data

```
st.dataframe(my_dataframe)
st.table(data.iloc[0:10])
st.json({'foo':'bar','fu'
st.metric(label="Temp", va
```

Display media

```
st.image('./header.png')
st.audio(data)
st.video(data)
```

Display interactive widgets

```
st.button('Hit me')
st.data_editor('Edit data
st.checkbox('Check me out
st.radio('Pick one:', ['nc
st.selectbox('Select', [1,
st.multiselect('Multiselec
st.slider('Slide me', min_
st.select_slider('Slide tc
st.text_input('Enter some
st.number_input('Enter a n
st.text_area('Area for tex
st.date_input('Date input
st.time_input('Time entry
st.file_uploader('File up
st.download_button('On the
st.camera_input("一二三,茄子
st.color_picker('Pick a co
```

```
# Use widgets' returned va
>>> for i in range(int(st
>>> if st.sidebar.selectbo
>>> my_slider_val = st.sli
>>> st.write(slider_val)
```

```
# Disable widgets to remov
>>> st.slider('Pick a numb
```

Connect to data sources

```
st.experimental_connectior
conn = st.experimental_cor
conn = st.experimental_cor

>>> class MyConnection(Exp
>>>     def _connect(self,
>>>         return myconn.c
>>>     def query(self, que
>>>         return self._ins
```

Optimize performance

Cache data objects

```
# E.g. Dataframe computati
>>> @st.cache_data
... def foo(bar):
...     # Do something exper
...     return data
# Executes foo
>>> d1 = foo(ref1)
# Does not execute foo
# Returns cached item by v
>>> d2 = foo(ref1)
# Different arg, so functi
>>> d3 = foo(ref2)
# Clear all cached entries
```

Columns

```
col1, col2 = st.columns(2)
col1.write('Column 1')
col2.write('Column 2')

# Three columns with diffe
col1, col2, col3 = st.colu
# col1 is wider

# Using 'with' notation:
>>> with col1:
>>>     st.write('This is
```

Tabs

```
# Insert containers separ
>>> tab1, tab2 = st.tabs(
>>> tab1.write("this is ta
>>> tab2.write("this is ta

# You can also use "with"
>>> with tab1:
>>>   st.radio('Select one
```

Control flow

```
# Stop execution immediate
st.stop()
# Rerun script immediately
st.experimental_rerun()

# Group multiple widgets:
>>> with st.form(key='my_
>>>   username = st.text_
```

Build chat-based apps

```
# Insert a chat message co
>>> with st.chat_message('
>>>     st.write("Hello 👋
>>>     st.line_chart(np.ra

# Display a chat input wid
>>> st.chat_input("Say som
```

Learn how to build chat-based apps

Mutate data

```
# Add rows to a dataframe
# showing it.
>>> element = st.dataframe
>>> element.add_rows(df2)

# Add rows to a chart afte
# showing it.
>>> element = st.line_char
>>> element.add_rows(df2)
```

Display code

```
st.echo()
>>> with st.echo():
>>>     st.write('Code wi
```

Placeholders, help,

```
>>> foo.clear()
# Clear values from *all*
>>> st.cache_data.clear()
```

Cache global resources

```
# E.g. TensorFlow session,
>>> @st.cache_resource
... def foo(bar):
...   # Create and return
...   return session
# Executes foo
>>> s1 = foo(ref1)
# Does not execute foo
# Returns cached item by r
>>> s2 = foo(ref1)
# Different arg, so functi
>>> s3 = foo(ref2)
# Clear all cached entries
>>> foo.clear()
# Clear all global resourc
>>> st.cache_resource.clea
```

Deprecated caching

```
>>> @st.cache
... def foo(bar):
...   # Do something exper
...   return data
>>> # Executes foo
>>> d1 = foo(ref1)
>>> # Does not execute foo
>>> # Returns cached item
>>> d2 = foo(ref1)
>>> # Different arg, so fu
>>> d3 = foo(ref2)
```

```
>>>   password = st.text_
>>>   st.form_submit_butt
```

Personalize apps for users

```
# Show different content
>>> if st.user.email == '
>>>    display_sahil_conte
>>> elif st.user.email ==
>>>    display_mrSpy_conte
>>> else:
>>>    st.write("Please c
```

and options

```
# Replace any single eleme
>>> element = st.empty()
>>> element.line_chart(..
>>> element.text_input(..

# Insert out of order.
>>> elements = st.containe
>>> elements.line_chart(.
>>> st.write("Hello")
>>> elements.text_input(.

st.help(pandas.DataFrame)
st.get_option(key)
st.set_option(key, value)
st.set_page_config(layout=
st.experimental_show(objec
st.experimental_get_query_
st.experimental_set_query_
```

Display progress and status

```
# Show a spinner during a
>>> with st.spinner(text=
>>>    time.sleep(3)
>>>    st.success('Done')

# Show and update progress
>>> bar = st.progress(50)
>>> time.sleep(3)
>>> bar.progress(100)

st.balloons()
st.snow()
st.toast('Mr Stay-Puft')
st.error('Error message')
st.warning('Warning messag
st.info('Info message')
st.success('Success messag
st.exception(e)
```

Display interactive widgets

```
st.subheader("This is My subheader")
    st.header("This is My header")
    st.text("Hi, i am text function and it work like paragraph tag.")
    st.markdown("Markdown")
    st.markdown("---") # Stright line
    st.markdown("Bold text:- **Hello!** ")
    st.markdown("Italic text:- *Hello!* ")
    st.markdown("---") # Stright line
    st.markdown("# h1 tag")
    st.markdown("## h2 tag")
    st.markdown("### h3 tag")
```

```
st.markdown("#### h4 tag")
st.markdown("##### h5 tag")
st.markdown("###### h6 tag")
st.markdown(">hello")
```

This is My subheader

This is My header

Hi, i am text function and it work like paragraph tag.

Markdown

Bold text:- **Hello!**

Italic text:- *Hello!*

---

h1 tag

h2 tag

h3 tag

h4 tag

h5 tag

h6 tag

hello