Cheat Sheet

This is a summary of the docs, as of Streamlit v1.30.0.

Install & Import

```
streamlit run first_app.py

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

Command line

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

Pre-release features

```
pip uninstall streamlit
pip install streamlit-nightly --upgrade
```

Learn more about experimental features

Magic commands

```
# Magic commands implicitly
# call st.write().
"_This_ is some **Markdown***"
```

```
my_variable
"dataframe:", my_data_frame
```

Display text

```
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
```

Display data

```
st.dataframe(my_dataframe)
st.table(data.iloc[0:10])
st.json({"foo":"bar","fu":"ba"})
st.metric("My metric", 42, 2)
```

Display media

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

Display charts

```
st.area_chart(df)
```

```
st.bar_chart(df)
st.line_chart(df)
st.map(df)
st.scatter_chart(df)

st.altair_chart(chart)
st.bokeh_chart(fig)
st.graphviz_chart(fig)
st.plotly_chart(fig)
st.pydeck_chart(chart)
st.pyplot(fig)
st.vega_lite_chart(df)
```

Add widgets to sidebar

```
# Just add it after st.sidebar:
>>> a = st.sidebar.radio("Select one:", [1, 2])

# Or use "with" notation:
>>> with st.sidebar:
>>>   st.radio("Select one:", [1, 2])
```

Columns

```
# Two equal columns:
>>> col1, col2 = st.columns(2)
>>> col1.write("This is column 1")
>>> col2.write("This is column 2")

# Three different columns:
>>> col1, col2, col3 = st.columns([3, 1, 1])
# col1 is larger.

# You can also use "with" notation:
>>> with col1:
```

```
>>>    st.radio("Select one:", [1, 2])
```

Tabs

```
# Insert containers separated into tabs:
>>> tab1, tab2 = st.tabs(["Tab 1", "Tab2"])
>>> tab1.write("this is tab 1")
>>> tab2.write("this is tab 2")

# You can also use "with" notation:
>>> with tab1:
>>>    st.radio("Select one:", [1, 2])
```

Control flow

```
# Stop execution immediately:
st.stop()
# Rerun script immediately:
st.rerun()
# Navigate to another page:
st.switch_page("pages/my_page.py")

# Group multiple widgets:
>>> with st.form(key="my_form"):
>>>    username = st.text_input("Username")
>>>    password = st.text_input("Password")
>>>    st.form_submit_button("Login")
```

Display interactive widgets

```
st.button("Click me")
st.download_button("Download file", data)
st.link_button("Go to gallery", url)
st.data_editor("Edit data", data)
st.checkbox("I agree")
```

```
st.toggle("Enable")
st.radio("Pick one", ["cats", "dogs"])
st.selectbox("Pick one", ["cats", "dogs"])
st.multiselect("Buy", ["milk", "apples", "potato
st.slider("Pick a number", 0, 100)
st.select_slider("Pick a size", ["S", "M", "L"])
st.text_input("First name")
st.number_input("Pick a number", 0, 10)
st.text_area("Text to translate")
st.date_input("Your birthday")
st.time_input("Meeting time")
st.file_uploader("Upload a CSV")
st.camera_input("Take a picture")
st.color_picker("Pick a color")

# Use widgets' returned values in variables:
>>> for i in range(int(st.number_input("Num:"))):
>>>   foo()
>>> if st.sidebar.selectbox("I:",["f"]) == "f":
>>>   b()
>>> my_slider_val = st.slider("Quinn Mallory", 1
>>> st.write(slider_val)

# Disable widgets to remove interactivity:
>>> st.slider("Pick a number", 0, 100, disabled=
```

Build chat-based apps

```
# Insert a chat message container.
>>> with st.chat_message("user"):
>>>    st.write("Hello 👋")
>>>    st.line_chart(np.random.randn(30, 3))

# Display a chat input widget.
>>> st.chat_input("Say something")
```

Learn how to Build a basic LLM chat app

Mutate data

```
# Add rows to a dataframe after
# showing it.
>>> element = st.dataframe(df1)
>>> element.add_rows(df2)

# Add rows to a chart after
# showing it.
>>> element = st.line_chart(df1)
>>> element.add_rows(df2)
```

Display code

```
>>> with st.echo():
>>>     st.write("Code will be executed and printe
```

Placeholders, help, and options

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

# Insert out of order.
>>> elements = st.container()
>>> elements.line_chart(...)
>>> st.write("Hello")
>>> elements.text_input(...)  # Appears above "He

st.help(pandas.DataFrame)
st.get_option(key)
st.set_option(key, value)
st.set_page_config(layout="wide")
```

```
st.query_params[key]
st.query_params.get_all(key)
st.query_params.clear()
```

Connect to data sources

```
st.connection("pets_db", type="sql")
conn = st.connection("sql")
conn = st.connection("snowflake")

>>> class MyConnection(BaseConnection[myconn.MyC
>>>     def _connect(self, **kwargs) -> MyConnect
>>>         return myconn.connect(**self._secrets
>>>     def query(self, query):
>>>         return self._instance.query(query)
```

Optimize performance

Cache data objects

```
# E.g. Dataframe computation, storing downloaded
>>> @st.cache_data
... def foo(bar):
...     # Do something expensive and return data
...     return data
# Executes foo
>>> d1 = foo(ref1)
# Does not execute foo
# Returns cached item by value, d1 == d2
>>> d2 = foo(ref1)
# Different arg, so function foo executes
>>> d3 = foo(ref2)
# Clear all cached entries for this function
>>> foo.clear()
# Clear values from *all* in-memory or on-disk c
>>> st.cache_data.clear()
```

Cache global resources

```
# E.g. TensorFlow session, database connection, e
>>> @st.cache_resource
... def foo(bar):
...   # Create and return a non-data object
...   return session
# Executes foo
>>> s1 = foo(ref1)
# Does not execute foo
# Returns cached item by reference, s1 == s2
>>> s2 = foo(ref1)
# Different arg, so function foo executes
>>> s3 = foo(ref2)
# Clear all cached entries for this function
>>> foo.clear()
# Clear all global resources from cache
>>> st.cache_resource.clear()
```

Deprecated caching

```
>>> @st.cache
... def foo(bar):
...   # Do something expensive in here...
...   return data
>>> # Executes foo
>>> d1 = foo(ref1)
>>> # Does not execute foo
>>> # Returns cached item by reference, d1 == d2
>>> d2 = foo(ref1)
>>> # Different arg, so function foo executes
>>> d3 = foo(ref2)
```

Display progress and status

```
# Show a spinner during a process
>>> with st.spinner(text="In progress"):
>>>     time.sleep(3)
>>>     st.success("Done")

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

>>> with st.status("Authenticating...") as s:
>>>     time.sleep(2)
>>>     st.write("Some long response.")
>>>     s.update(label="Response")

st.balloons()
st.snow()
st.toast("Warming up...")
st.error("Error message")
st.warning("Warning message")
st.info("Info message")
st.success("Success message")
st.exception(e)
```

Personalize apps for users

```
# Show different content based on the user's ema
>>> if st.user.email == "jane@email.com":
>>>     display_jane_content()
>>> elif st.user.email == "adam@foocorp.io":
>>>     display_adam_content()
>>> else:
>>>     st.write("Please contact us to get access
```

Still have questions?

Our forums are full of helpful information and Streamlit experts.

Was this page helpful? Yes No Edit this page on GitHub