Software Engineering For Data Science (SEDS)

Class: 2nd Year 2nd Cycle
Branch: AIDS

Dr. Belkacem KHALDI| ESI-SBA

Lecture 11:

Web Development for Data Science: Part III: Building DS App with Streamlit

Web Development for Data Science:

Part III: Building DS App with Streamlit

Building DS App with Streamlit

Why Streamlit?

Challenges in other Approaches

Data Scientists' Significance:	Vital for data-driven decisions, efficiency improvement, and scaling machine learning models.
Challenges in Showcasing Insights:	Data scientists struggle to effectively showcase **dynamic results**, especially for complex analyses and user input scenarios.
Current Approaches' Limitations:	Sending **static visualizations**, creating **Word documents**, or building **web applications** using frameworks like **Flask** or **Django** from scratch are often slow, lack interactivity, and hinder reproducibility.

What Streamlit brings?

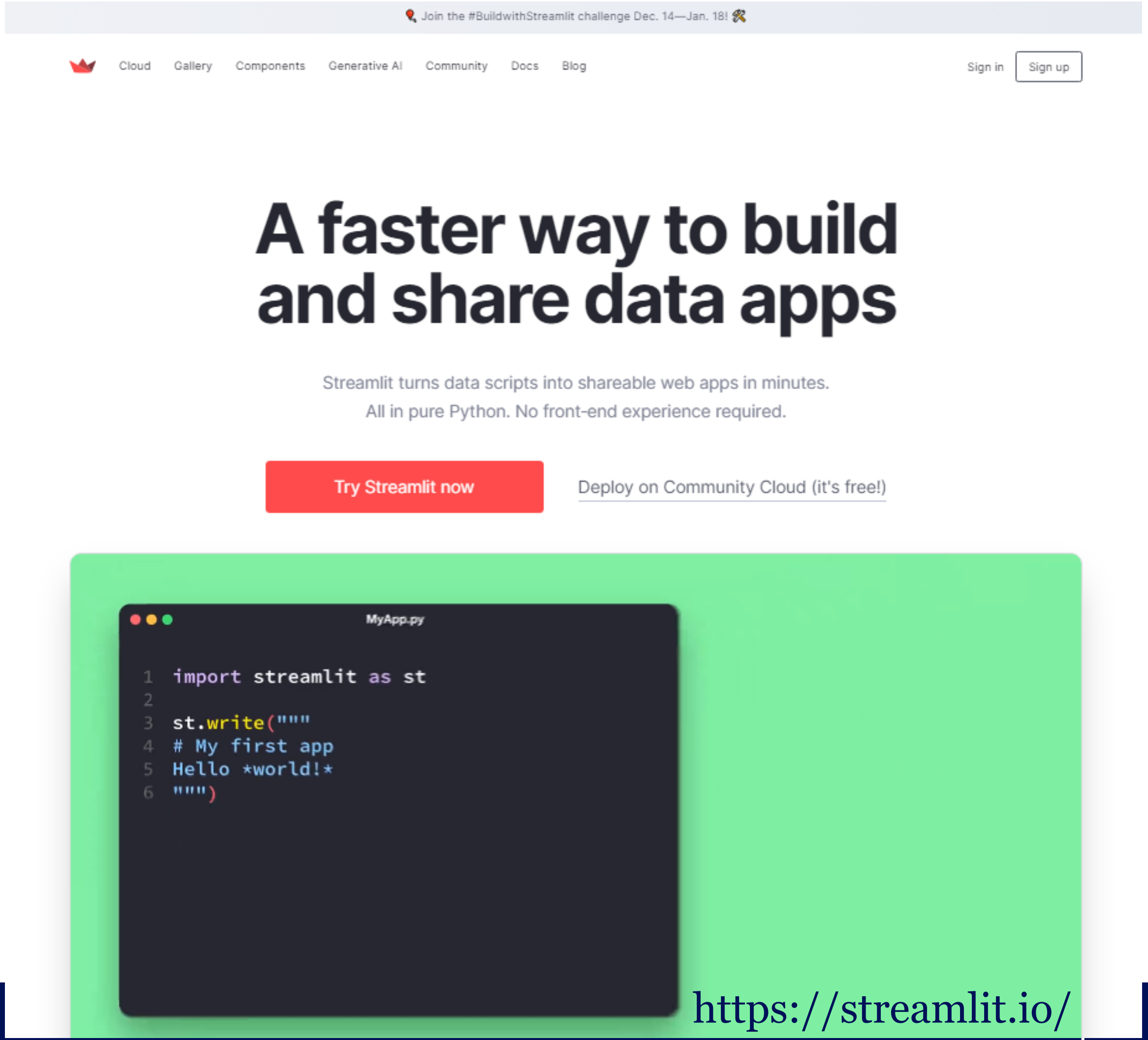
Introduction to Streamlit:	**Streamlit** addresses these challenges by offering a web application framework focused on **speed** and **interaction**, allowing Python developers to **quickly build and deploy applications**.
Streamlit's Features:	**Speed and Interaction:** Streamlit prioritizes fast development and user interactivity.
	User Input Handling: Offers built-in methods for handling user inputs like text and dates.
	Graphing Capabilities: Enables the creation of interactive graphs using popular Python graphing libraries.

Building DS App with Streamlit

What is Streamlit?

Open-source Python library that facilitates the creation and development of custom web applications.

- ❑ Ideal for supporting **machine learning** and **data science** projects.
- ❑ Fast prototyping of **machine learning** and **data science** projects.
- ❑ Support **interactive interfaces.**
- ❑ **Front-end skills** are not expressly required.
- ❑ Thanks to **widgets** and **elements** available, you can create web pages with a few lines of code.
- ❑ Compatible with most **Python libraries**.


Join the #BuildwithStreamlit challenge Dec. 14—Jan. 18!
Cloud
Gallery
Components
Generative AI
Community
Docs
Blog
Sign in
Sign up
Try Streamlit now
Deploy on Community Cloud (it's free!)
MyApp.py

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.

```
import streamlit as st

st.write("""
# My first app
Hello *world!*
""")
```

https://streamlit.io/

Building DS App with Streamlit

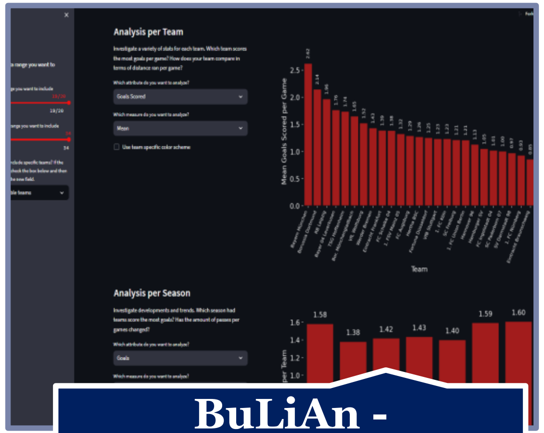
Streamlit

Examples Gallery

There are several templates and applications created by the community

Grouped into several applications categories: LLMs, Data Visulaization, Geography and Society, NLP & Language, ...

https://streamlit.io/gallery



Analysis per Team

Investigate a variety of stats for each team. Which team scores the most goals per game? How does your team compare in terms of distance ran per game?

Which attribute do you want to analyze?

Goals Scored

Which measure do you want to analyze?

Mean

□ Use team specific color scheme

Analysis per Season

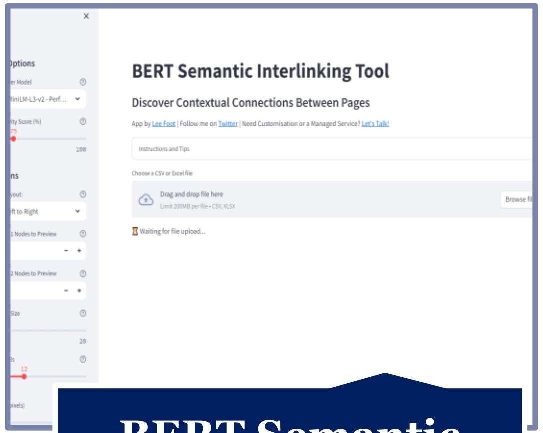
Investigate developments and trends. Which season had teams score the most goals? Has the amount of passes per games changed?

Which attribute do you want to analyze?

Goals

Which measure do you want to analyze?

BuLiAn - Bundesliga Analyzer



BERT Semantic Interlinking Tool

Discover Contextual Connections Between Pages

App by Lee Foot | Follow me on Twitter | Need Customisation or a Managed Service? Let's Talk!

Instructions and Tips

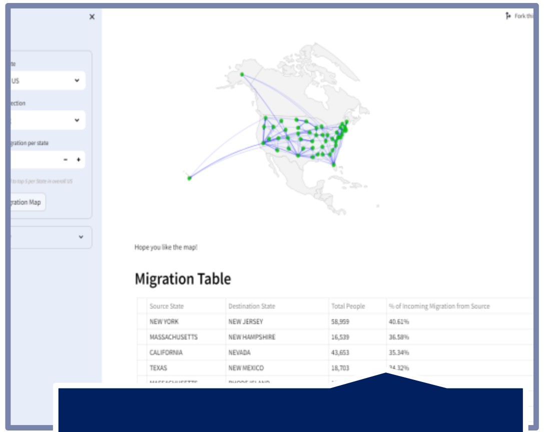
Choose a CSV or Excel file

Drag and drop file here

Limit 200MB per file • CSV, XLSX

⏳ Waiting for file upload...

BERT Semantic Interlinking Tool

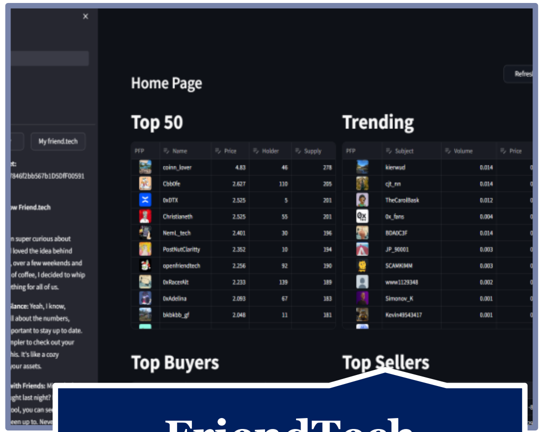


Hope you like the map!

Migration Table

Source State	Destination State	Total People	% of Incoming Migration from Source
NEW YORK	NEW JERSEY	58,959	40.61%
MASSACHUSETTS	NEW HAMPSHIRE	16,539	36.58%
CALIFORNIA	NEVADA	43,653	35.34%
TEXAS	NEW MEXICO	18,703	34.32%

Migration Network



Home Page

Top 50

PFP	Name	Price	Holder	Supply
	coinn_lover	4.83	46	278
	Cbb0fe	2.627	110	205
	0xDTX	2.525	5	201
	Christianeth	2.525	55	201
	NemL_tech	2.401	30	196
	PostNutClaritty	2.352	10	194
	openfriendtech	2.256	92	190
	0xRacerAlt	2.233	139	189
	0xAdelina	2.093	67	183
	bkbkbb_gf	2.048	11	181

Trending

PFP	Subject	Volume	Price
	kierwud	0.014	
	cjt_nn	0.014	
	TheCarolBask	0.012	
	0x_fans	0.004	
	BDA0C3F	0.014	
	JP_90001	0.003	
	SCAMKIMM	0.003	
	www1129348	0.002	
	Simonov_K	0.001	
	Kevin49543417	0.001	

Top Buyers

Top Sellers

FriendTech Dashboard

Building DS App with Streamlit

Streamlit

Installation & Configuration

Python 3.7 – Python 3.11

Using a **virtual environment** is always recommended (**pipenv**, **conda**, **venv**…)

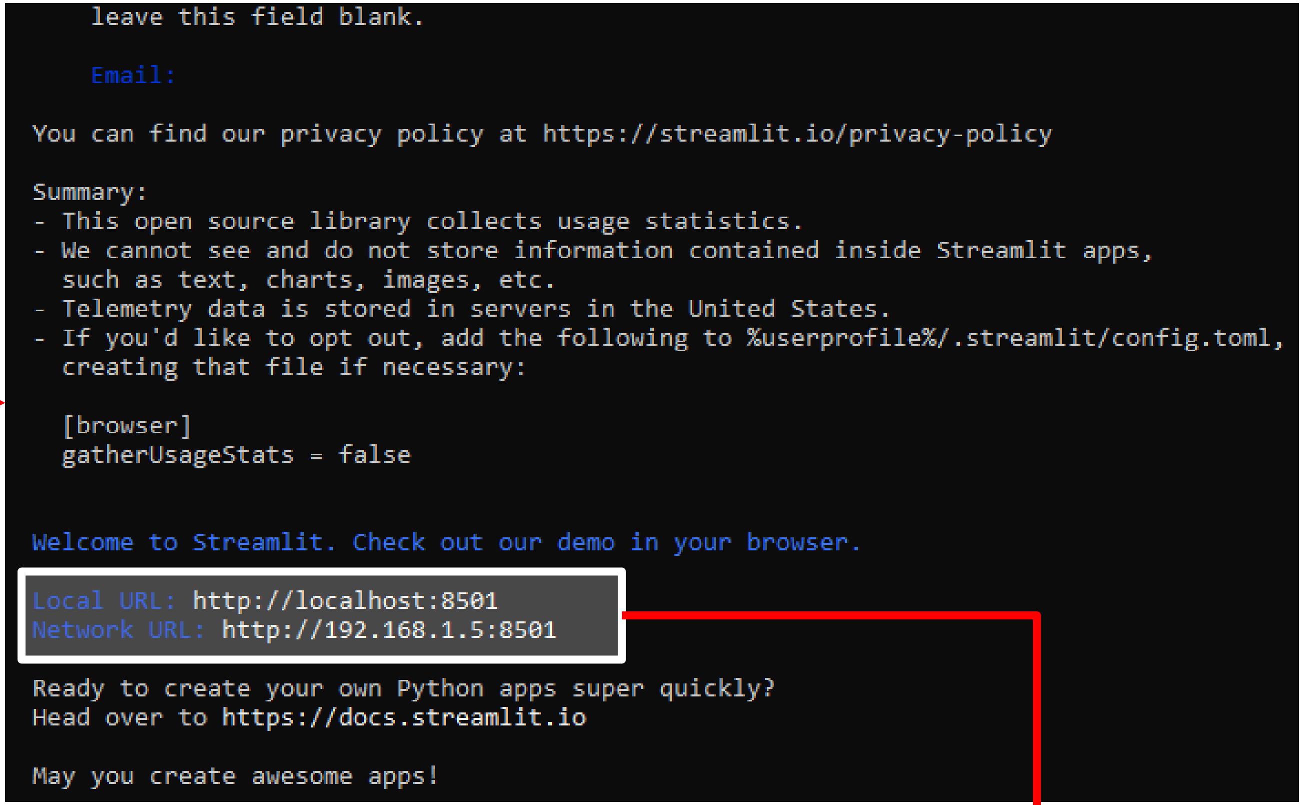
Install Streamlit

```
pip install streamlit
```

```
conda install streamlit
```

Test the installation

```
streamlit hello
```



```
    leave this field blank.

    Email:

You can find our privacy policy at https://streamlit.io/privacy-policy

Summary:
- This open source library collects usage statistics.
- We cannot see and do not store information contained inside Streamlit apps,
  such as text, charts, images, etc.
- Telemetry data is stored in servers in the United States.
- If you'd like to opt out, add the following to %userprofile%/.streamlit/config.toml,
  creating that file if necessary:

  [browser]
  gatherUsageStats = false


Welcome to Streamlit. Check out our demo in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.5:8501

Ready to create your own Python apps super quickly?
Head over to https://docs.streamlit.io

May you create awesome apps!
```

URL to reach the web **server** at **port 8501**

Launch your own application

```
streamlit run your_script.py [-- script args]
```

Or

```
python -m streamlit run your_script.py
```

https://docs.streamlit.io/get-started/installation

Building DS App with Streamlit

Installation & Configuration

Streamlit

Python 3.7 – Python 3.11

Using a **virtual environment** is always recommended (**pipenv**, **conda**, **venv**…)

Install Streamlit

```
pip install streamlit
```

```
conda install streamlit
```

Test the installation

```
streamlit hello
```

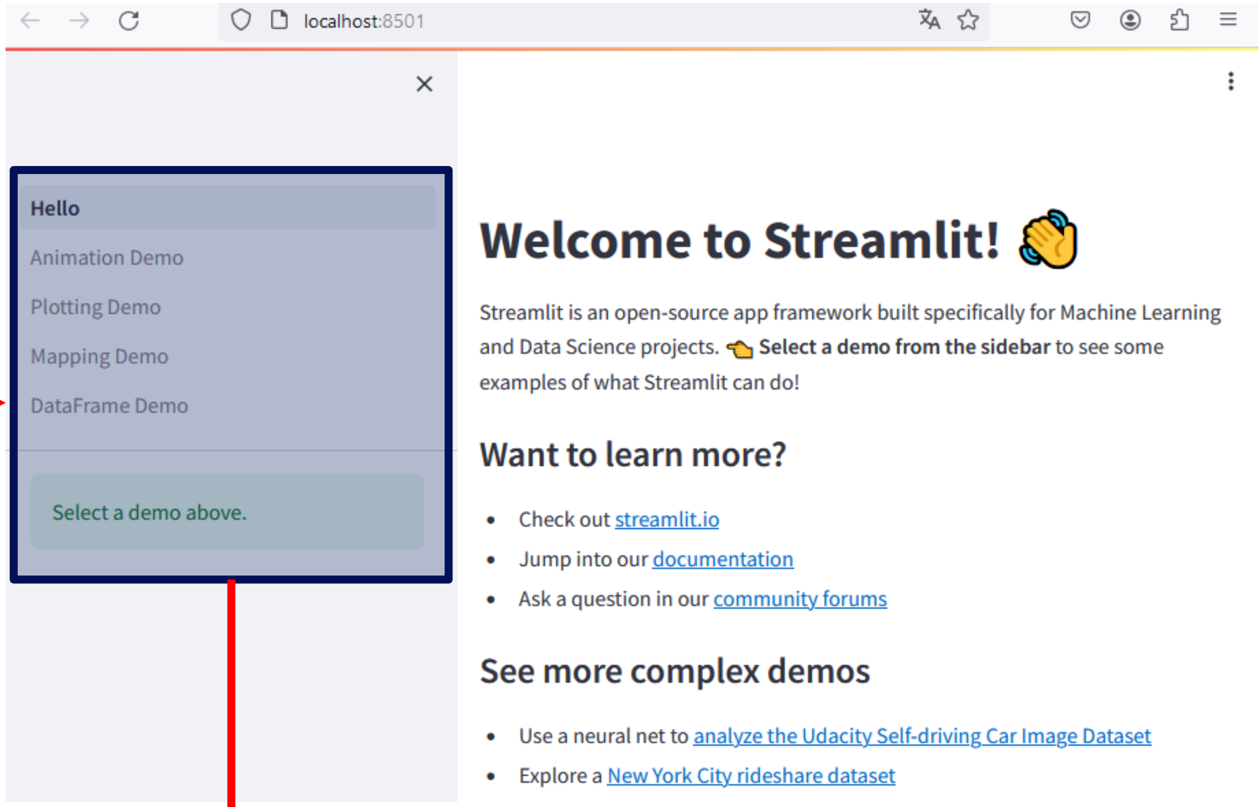
Launch your own application

```
streamlit run your_script.py [-- script args]
```

Or

```
python -m streamlit run your_script.py
```

https://docs.streamlit.io/get-started/installation



Select a demo above.

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

Sidebar with access to **sample demos**

Building DS App with Streamlit

Streamlit

Installation & Configuration

Various possibilities to define configuration options (e.g., **server port**, **theme**…) via:

A global config file (to be created):

for macOS/Linux

```
~/.streamlit/config.toml
```

for Windows

```
%userprofile%/.streamlit/config.toml
```

A per-project configurat-ion file:

```
$CWD/.streamlit/config.toml
```

where **$CWD** is the folder from which **Streamlit** was launched

A command line flag:

```
streamlit run your_script.py --server.port 80
```

config.toml

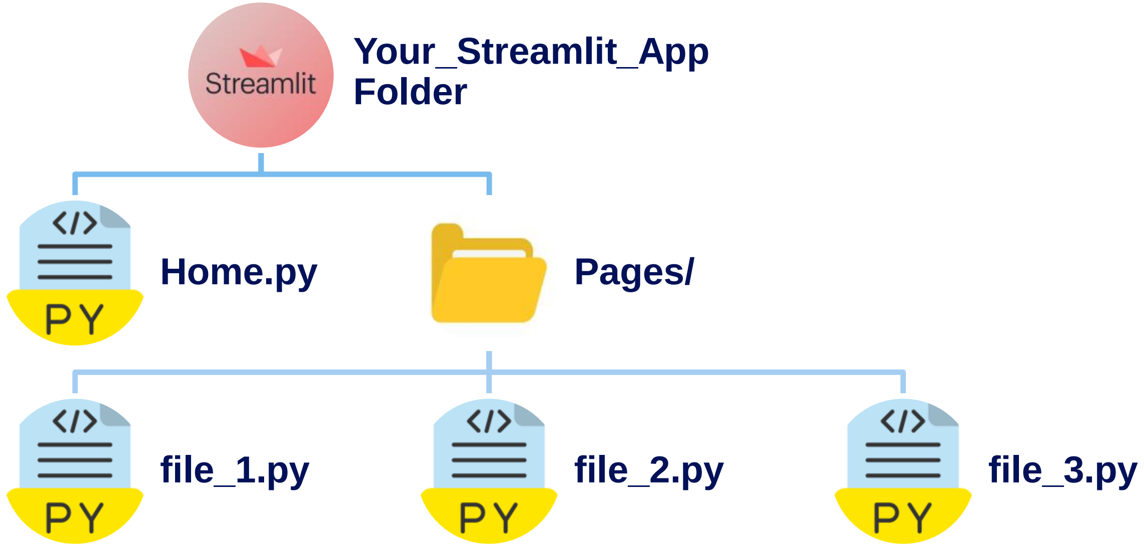
```
[server]
…
# The port where the server will listen for browser connections.
# Default: 8501
port = 8501
…
```

https://docs.streamlit.io/library/advanced-features/configuration

Building DS App with Streamlit

Recommended Project Structure

- ❑ Before you develop your app, it's important to define the project directory structure
- ❑ You need to define an **entrypoint** (e.g.: **Home.py**) file that represents the main page to show to the user
- ❑ Other additional **pages** should be placed in **a sub-folder** Pages
- ❑ **Pages** globally share the same Python modules

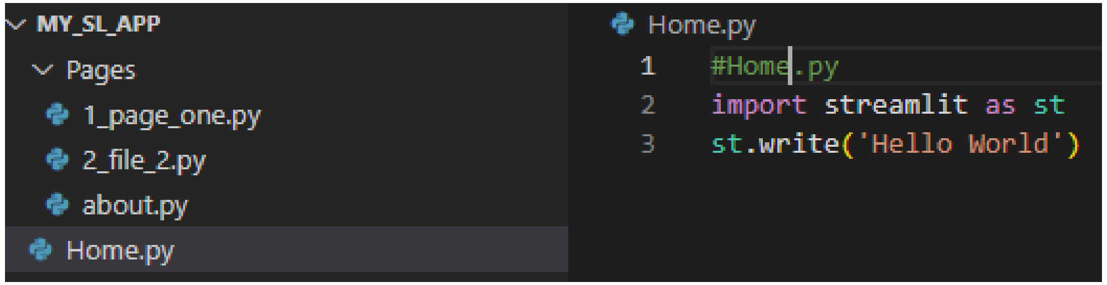


Building DS App with Streamlit

Application Pages

Streamlit

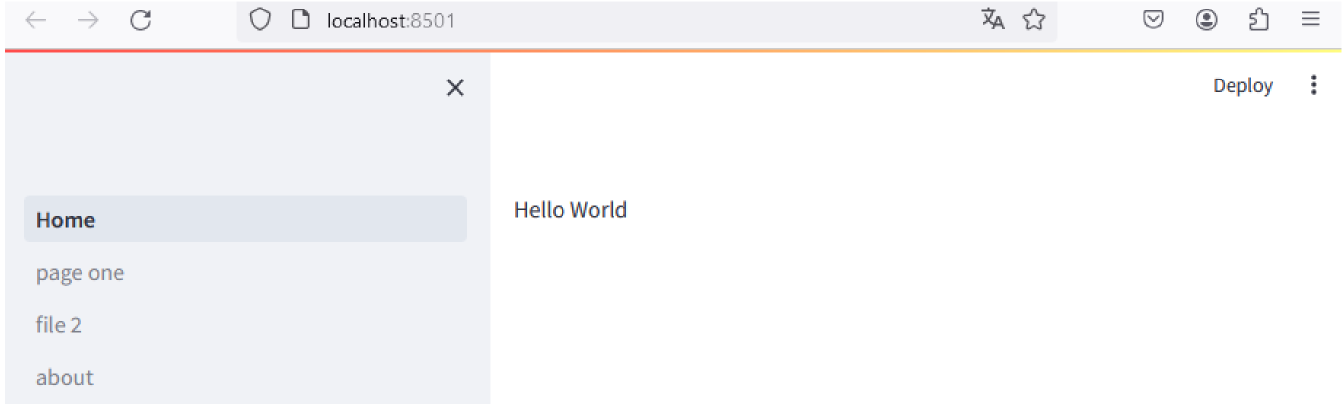
```
/my_sl_app> streamlit run Home.py
```



```
MY_SL_APP
  Pages
    1_page_one.py
    2_file_2.py
    about.py
  Home.py
```

Home.py

```python
#Home.py
import streamlit as st
st.write('Hello World')
```



localhost:8501

Hello World

- ❑ **Pages** are defined by files .py within the "pages/" folder
- ❑ File names are transformed into **page names**
- ❑ The order is given by the number preceding the title and/or by the alphabetical order of the title itself.
- ❑ The number used as a **prefix** in the **file name** is not interpreted as part of the **title**

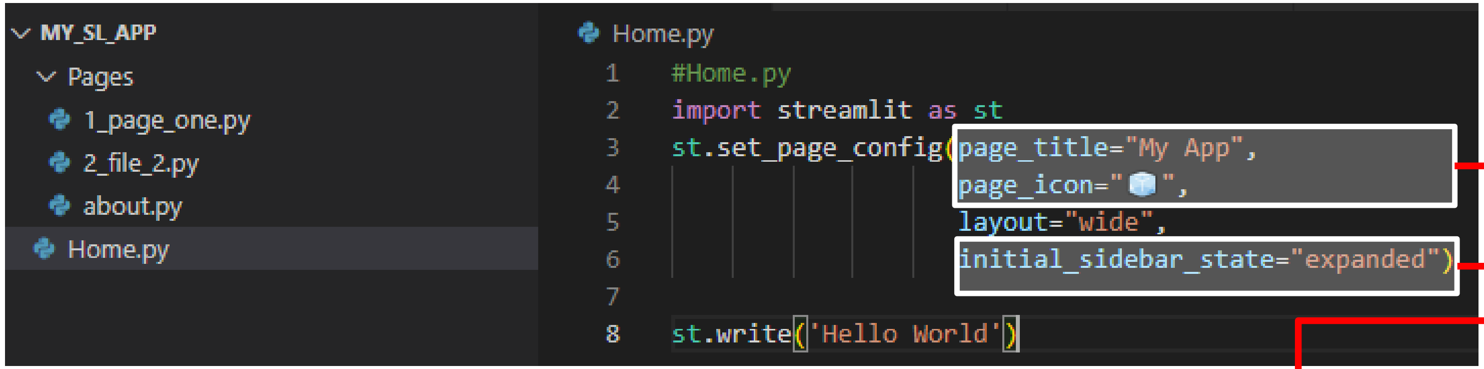
Building DS App with Streamlit

Streamlit

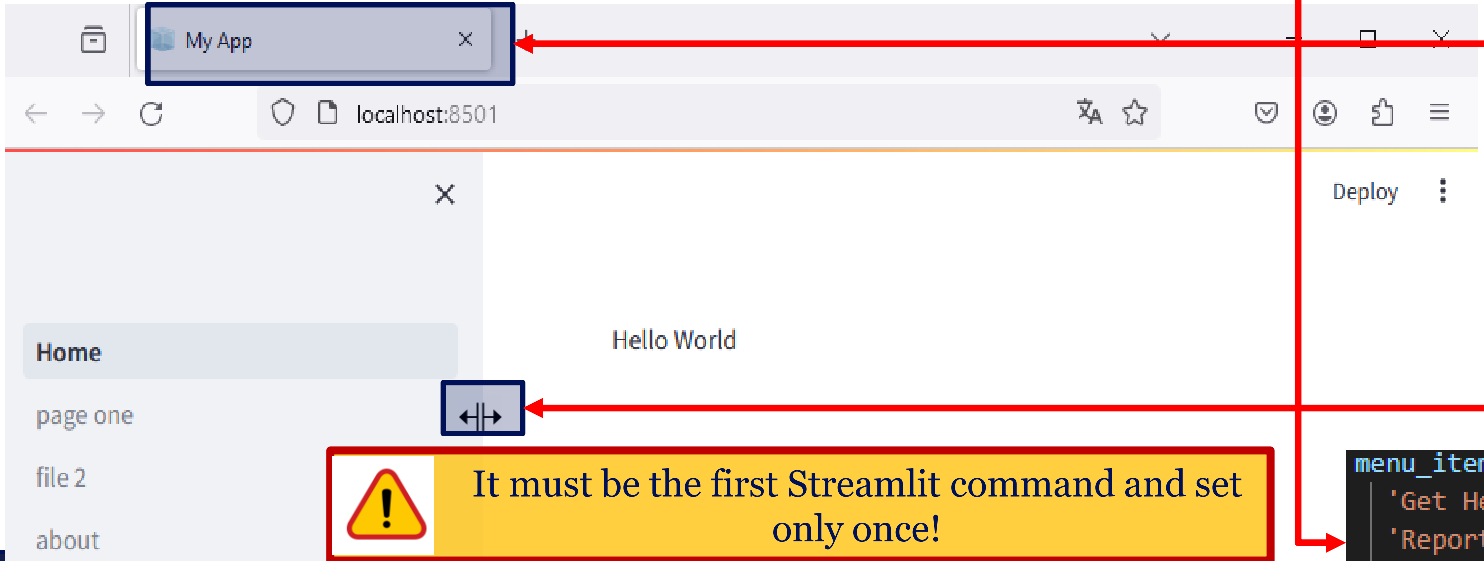
Page Configuration

```
/my_sl_app> streamlit run Home.py
```

https://docs.streamlit.io/1.18.0/library/api-reference/utilities/st.set_page_config



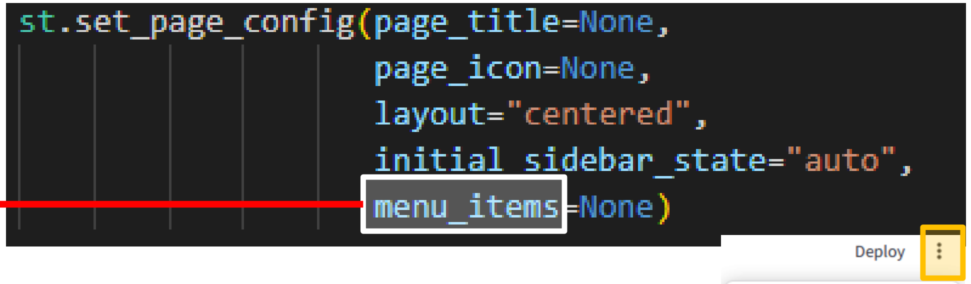
```python
#Home.py
import streamlit as st
st.set_page_config(page_title="My App",
                   page_icon="🧊",
                   layout="wide",
                   initial_sidebar_state="expanded")

st.write('Hello World')
```



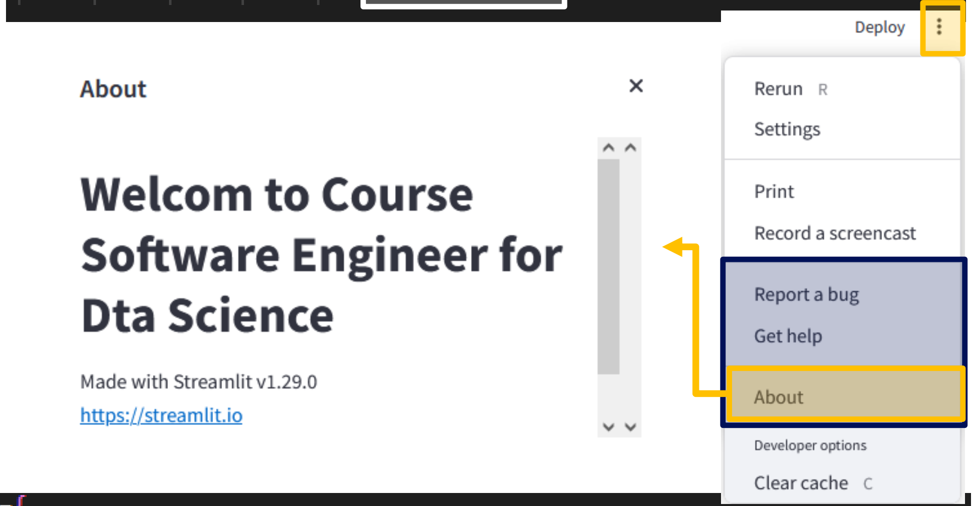
Hello World

⚠ It must be the first Streamlit command and set only once!

Default Page Configuration

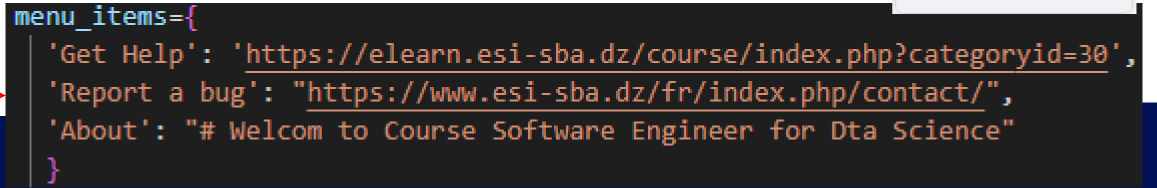


```python
st.set_page_config(page_title=None,
                   page_icon=None,
                   layout="centered",
                   initial_sidebar_state="auto",
                   menu_items=None)
```



Welcom to Course Software Engineer for Dta Science

Made with Streamlit v1.29.0
https://streamlit.io



```python
menu_items={
  'Get Help': 'https://elearn.esi-sba.dz/course/index.php?categoryid=30',
  'Report a bug': "https://www.esi-sba.dz/fr/index.php/contact/",
  'About': "# Welcom to Course Software Engineer for Dta Science"
  }
```

Building DS App with Streamlit

Elements of Streamlit & their Arguments

Streamlit

Widgets & elements specific to different types of activities and inputs

Quickly integrate different features into your application

Available through official documentation: https://docs.streamlit.io/library/api-reference

Most significant categories:

- Text elements
- Input widgets
- Layout
- Visualization of data and graphs
- Additional element

The various **elements** can be integrated without special configurations

□ Personalization via certain arguments.

⚠ Different items cannot have the same key!

Some **arguments** are common to all (or most) of the **elements**:

□ **label**: describes the functionality of the element (e.g. the name of a clickable button).

□ **label_visibility**: determine label visibility (i.e., ”visible”, ”hidden”, ”collapsed”); the label should always be defined.

□ **disabled**: boolean flag to disable an element. Useful for making a widget available only if a certain condition occurs.

□ **use_container_width**: boolean flag to fit the size of the widget to that of the container it is part of.

□ **key**: string or number to uniquely identify the widget. If omitted, it is generated based on content.

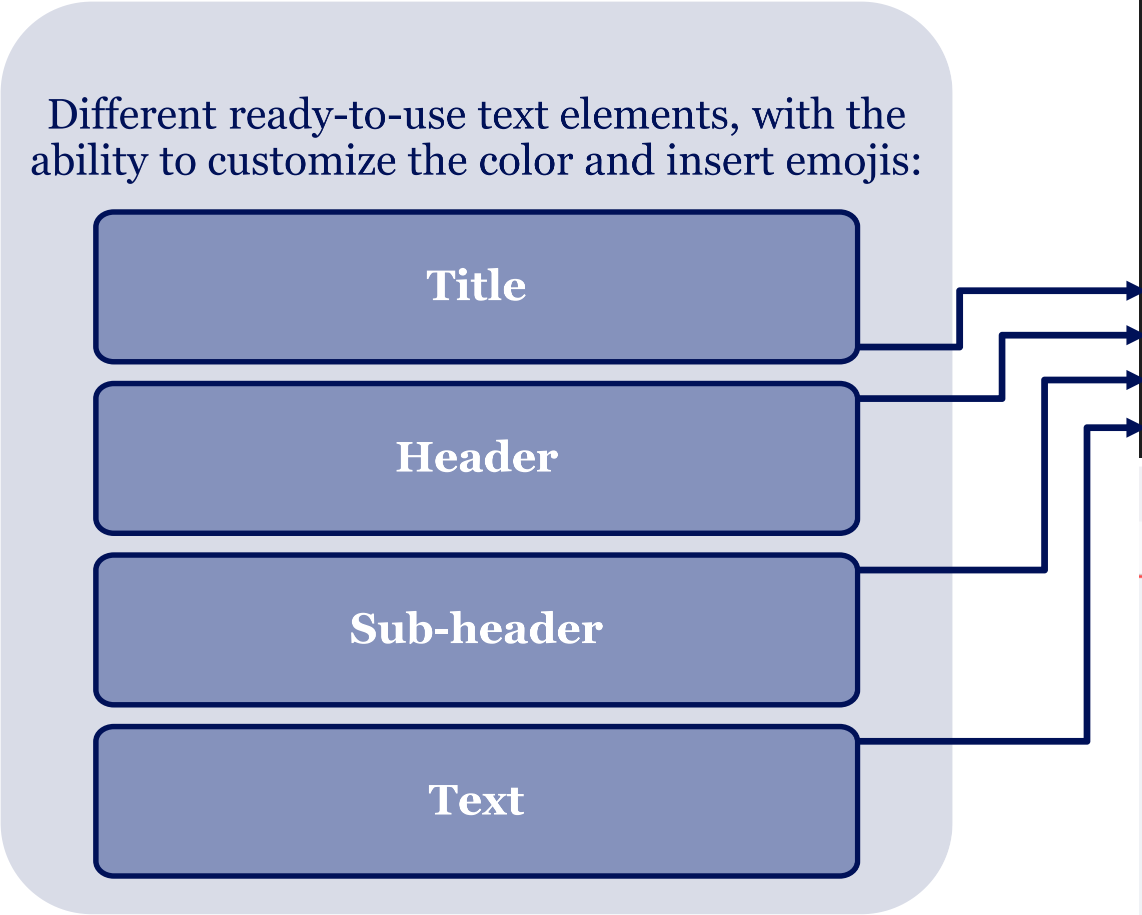
Text Elements Streamlit

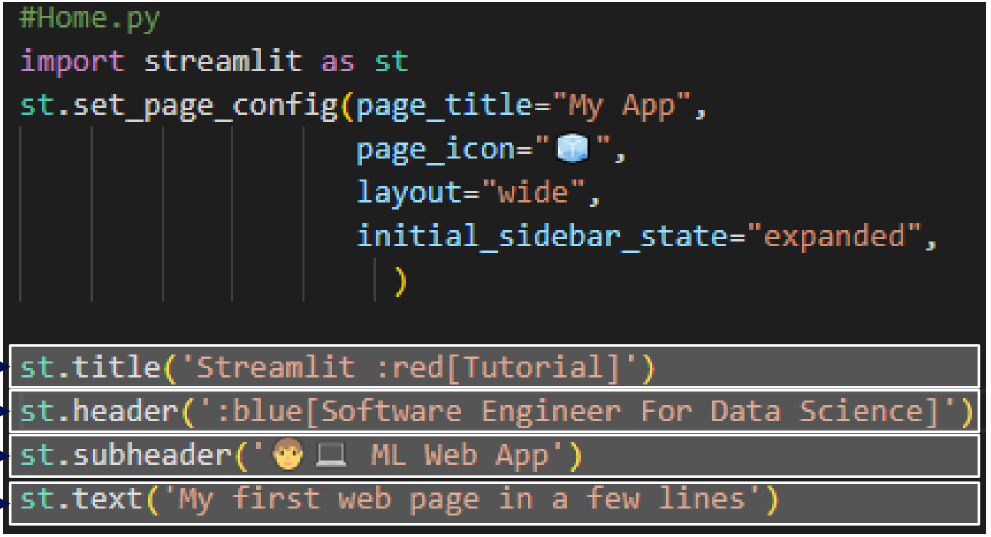
Building DS App with Streamlit

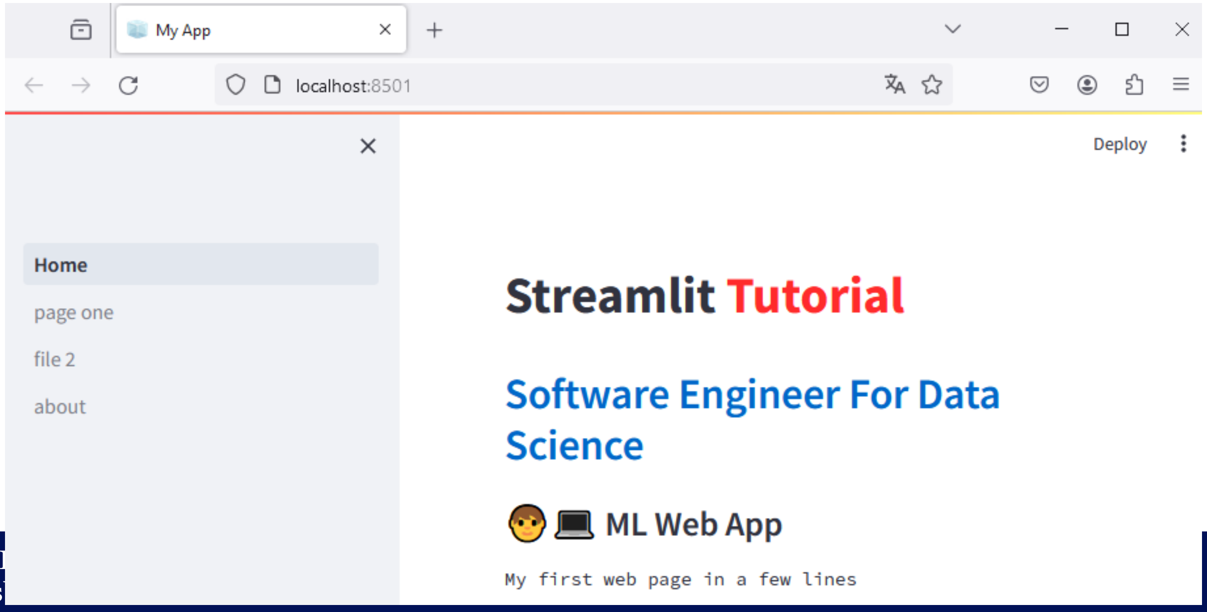
Building DS App with Streamlit

Streamlit

Text elements


Different ready-to-use text elements, with the ability to customize the color and insert emojis:


```python
#Home.py
import streamlit as st
st.set_page_config(page_title="My App",
                   page_icon="🧊",
                   layout="wide",
                   initial_sidebar_state="expanded",
                   )

st.title('Streamlit :red[Tutorial]')
st.header(':blue[Software Engineer For Data Science]')
st.subheader('👦💻 ML Web App')
st.text('My first web page in a few lines')
```



Streamlit Tutorial

Software Engineer For Data Science

👦💻 ML Web App

```
My first web page in a few lines
```

Building DS App with Streamlit

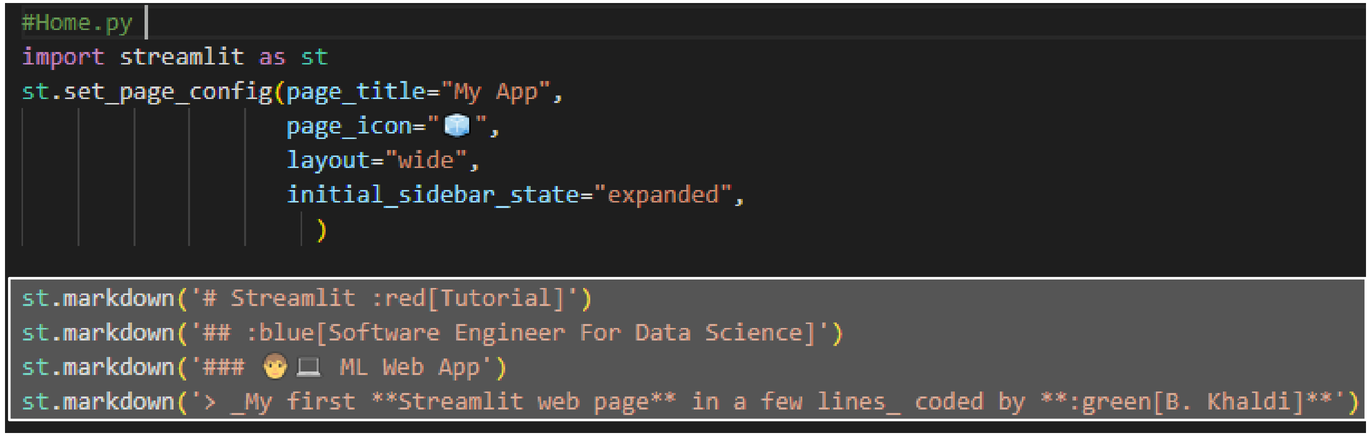
Streamlit

Markdown

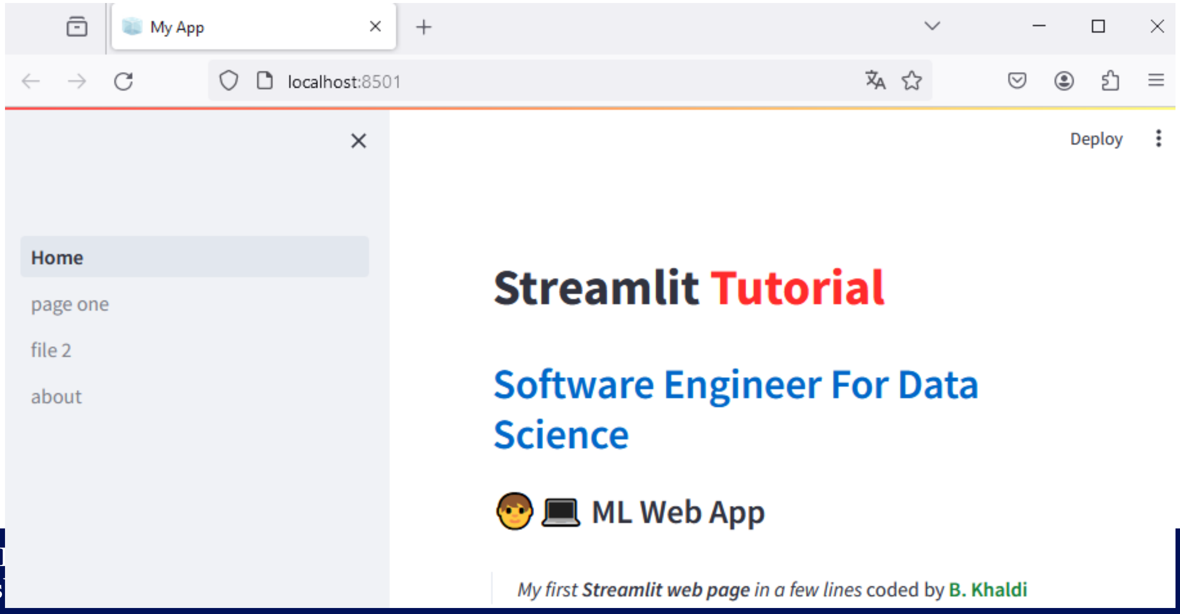
- ❑ It is possible to insert strings formatted according to the markdown language
- ❑ Markdown is used to format text quickly and easily, being more readable than other markup languages
- ❑ The most common syntax (N.B. spaces are sometimes necessary!):

# Header 1	**bold**
## Header 2	> blockquote
### Header 3	* Item 1 * Item 2
italics	Line␣␣ Break

https://www.markdownguide.org/basic-syntax/



```
#Home.py
import streamlit as st
st.set_page_config(page_title="My App",
                   page_icon="🧊",
                   layout="wide",
                   initial_sidebar_state="expanded",
                   )

st.markdown('# Streamlit :red[Tutorial]')
st.markdown('## :blue[Software Engineer For Data Science]')
st.markdown('### 🧑‍💻 ML Web App')
st.markdown('> _My first **Streamlit web page** in a few lines_ coded by **:green[B. Khaldi]**')
```



Streamlit Tutorial

Software Engineer For Data Science

🧑‍💻 ML Web App

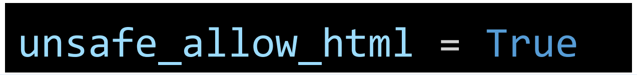
> *My first **Streamlit web page** in a few lines coded by **B. Khaldi***

Building DS App with Streamlit

Streamlit

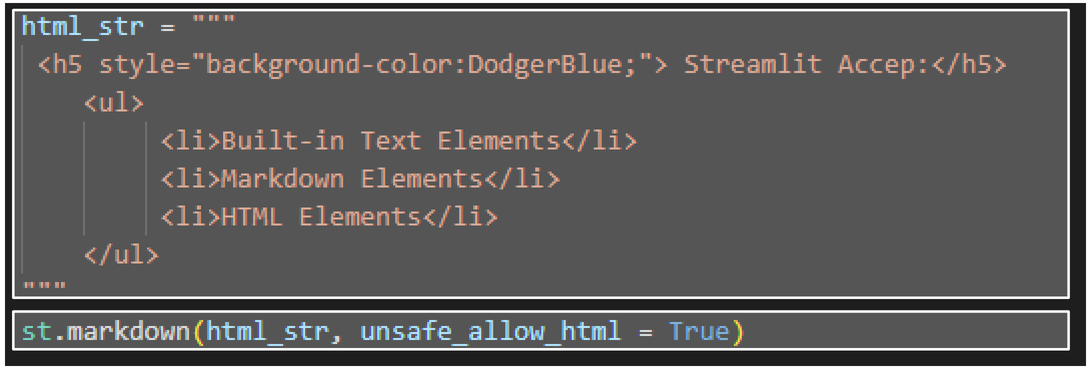
Markdown and HTML

- ❑ You can also use **markdown** to insert **HTML** code
- ❑ Useful for special customizations
- ❑ It is necessary to enable the use of **HTML** code via:



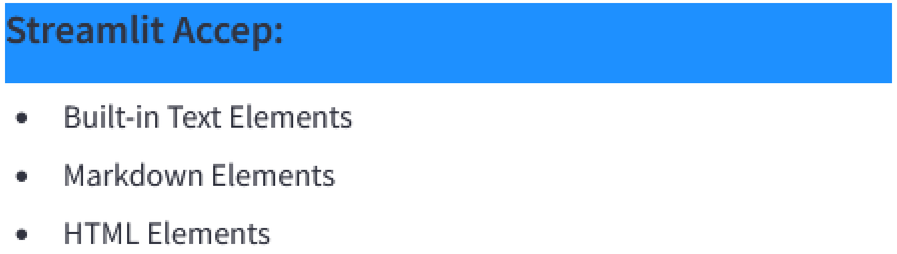
```
unsafe_allow_html = True
```

  - ○ The feature is disabled by default to prevent the developer from inserting **unsafe code**



```
html_str = """
 <h5 style="background-color:DodgerBlue;"> Streamlit Accep:</h5>
    <ul>
         <li>Built-in Text Elements</li>
         <li>Markdown Elements</li>
         <li>HTML Elements</li>
    </ul>
"""
```

```
st.markdown(html_str, unsafe_allow_html = True)
```



Streamlit Accep:

- Built-in Text Elements
- Markdown Elements
- HTML Elements

Input Widgets Streamlit

Building DS App with Streamlit

Building DS App with Streamlit

Streamlit

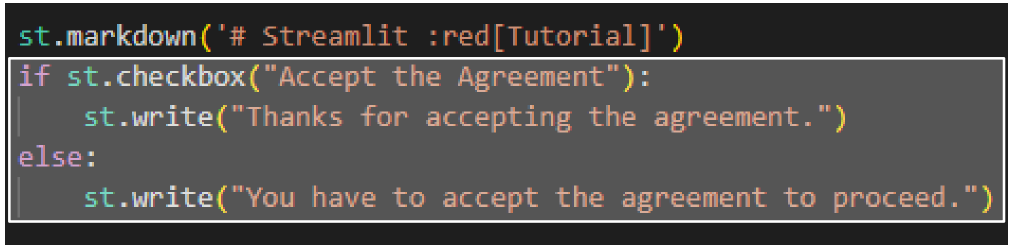
Buttons & Checkboxes

Button

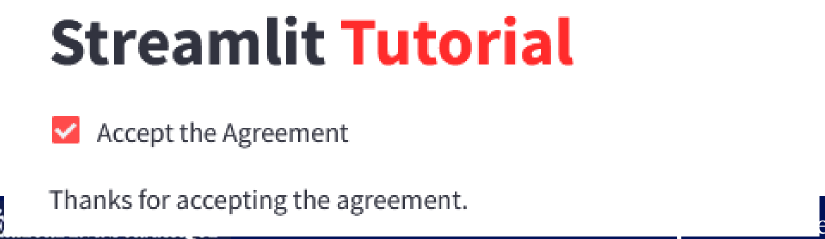
```
st.button(label, key=None, help=None,
on_click=None, args=None, kwargs=None,
type="secondary", disabled=False,
use_container_width=False)
```

Checkbox

```
st.checkbox(label, value=False, key=None,
help=None, on_change=None, args=None,
kwargs=None, *, disabled=False,
label_visibility="visible")
```



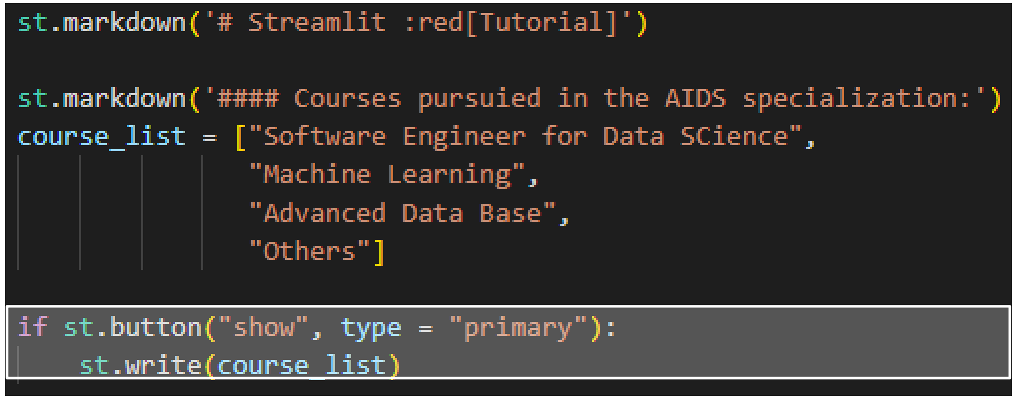
```
st.markdown('# Streamlit :red[Tutorial]')
if st.checkbox("Accept the Agreement"):
    st.write("Thanks for accepting the agreement.")
else:
    st.write("You have to accept the agreement to proceed.")
```



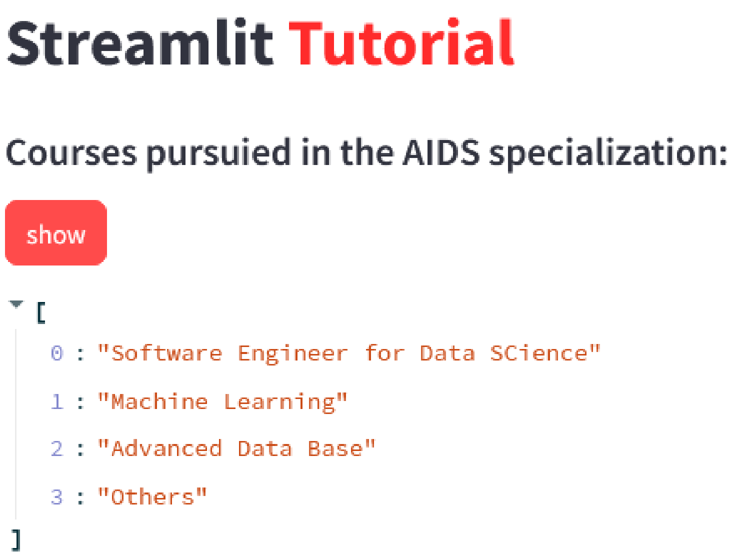
Streamlit Tutorial

☑ Accept the Agreement

Thanks for accepting the agreement.



```
st.markdown('# Streamlit :red[Tutorial]')

st.markdown('#### Courses pursuied in the AIDS specialization:')
course_list = ["Software Engineer for Data SCience",
               "Machine Learning",
               "Advanced Data Base",
               "Others"]

if st.button("show", type = "primary"):
    st.write(course_list)
```



Streamlit Tutorial

Courses pursuied in the AIDS specialization:

```
[
  0 : "Software Engineer for Data SCience"
  1 : "Machine Learning"
  2 : "Advanced Data Base"
  3 : "Others"
]
```

Building DS App with Streamlit

Streamlit

Input Texts

Text

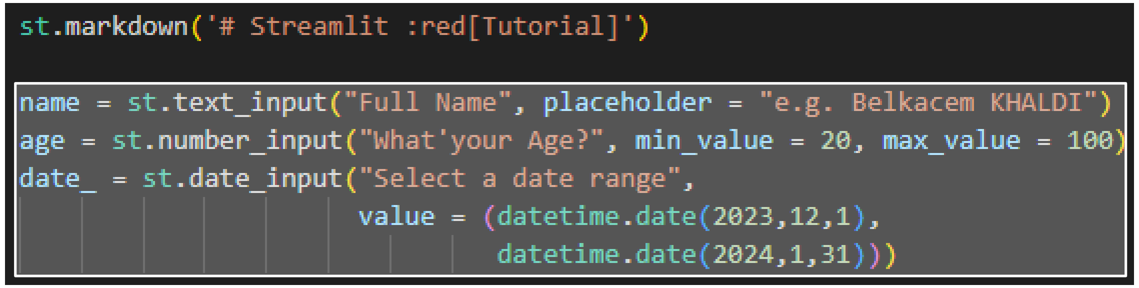
```
st.text_input(label, value="", max_chars=None,
key=None, type="default", help=None,
autocomplete=None, on_change=None, args=None,
kwargs=None, *, placeholder=None,
disabled=False, label_visibility="visible")
```

Number

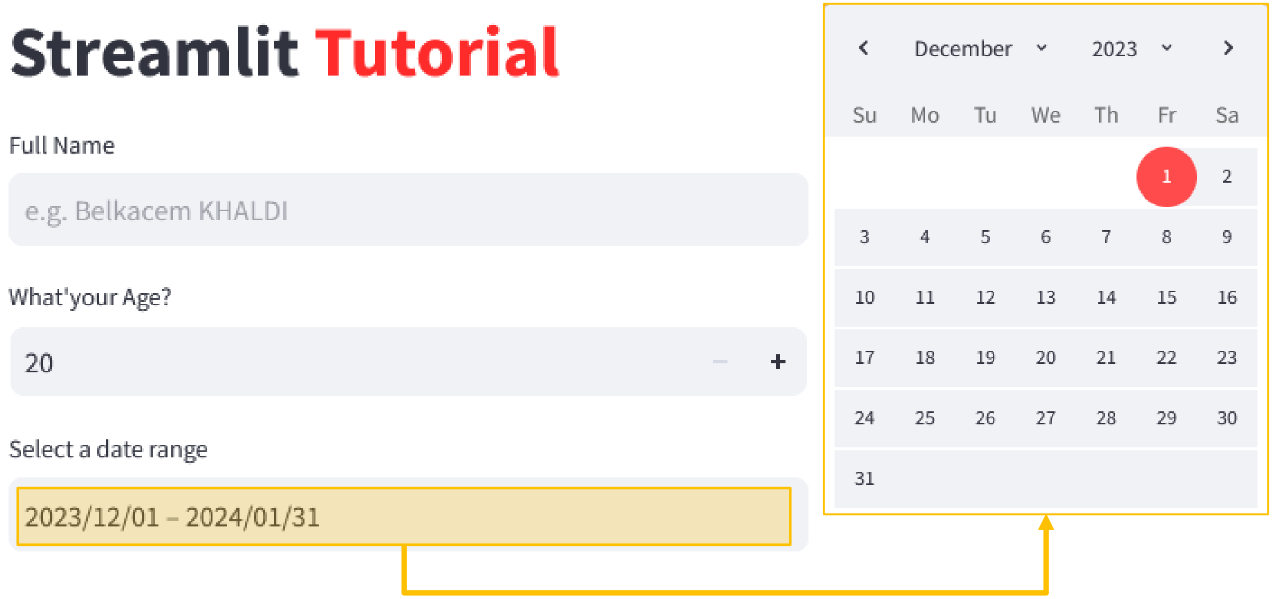
```
st.number_input(label, min_value=None,
max_value=None, value=, step=None, format=None,
key=None, help=None, on_change=None, args=None,
kwargs=None, *, disabled=False,
label_visibility="visible")
```

Date Input

```
st.date_input(label, value=None, min_value=None,
max_value=None, key=None, help=None,
on_change=None, args=None, kwargs=None, *,
disabled=False, label_visibility="visible")
```



```
st.markdown('# Streamlit :red[Tutorial]')

name = st.text_input("Full Name", placeholder = "e.g. Belkacem KHALDI")
age = st.number_input("What'your Age?", min_value = 20, max_value = 100)
date_ = st.date_input("Select a date range",
                        value = (datetime.date(2023,12,1),
                                 datetime.date(2024,1,31)))
```



Streamlit Tutorial

Full Name

e.g. Belkacem KHALDI

What'your Age?

20 − +

Select a date range

2023/12/01 – 2024/01/31

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Building DS App with Streamlit

Select Box & Multiselect

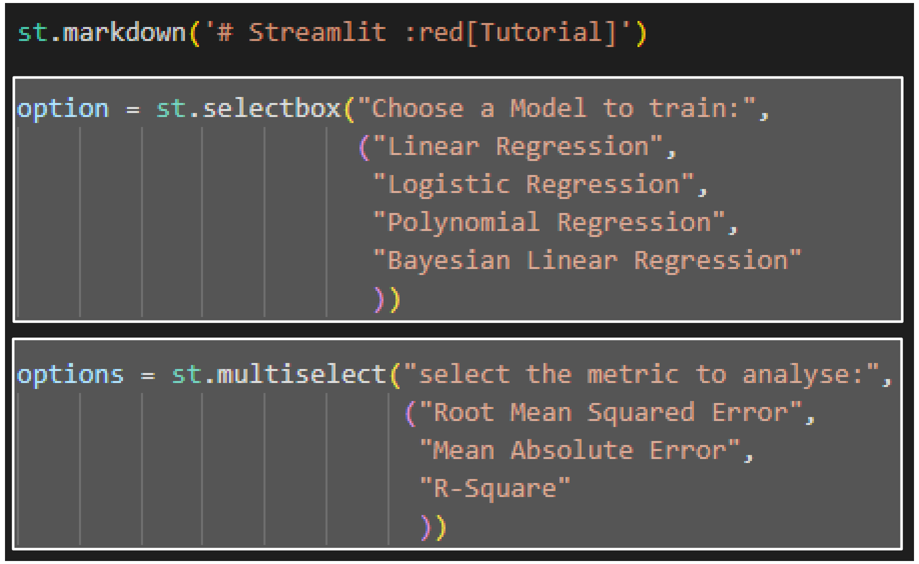
Streamlit

Select Box

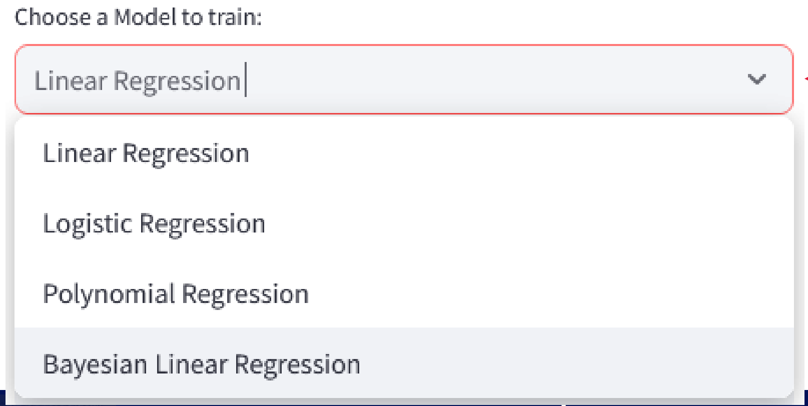
```
st.selectbox(label, options, index=0,
format_func=special_internal_function, key=None,
help=None, on_change=None, args=None, kwargs=None, *,
disabled=False, label_visibility="visible")
```

Multiselect

```
st.multiselect(label, options, default=None,
format_func=special_internal_function, key=None,
help=None, on_change=None, args=None, kwargs=None, *,
disabled=False, label_visibility="visible",
max_selections=None)
```



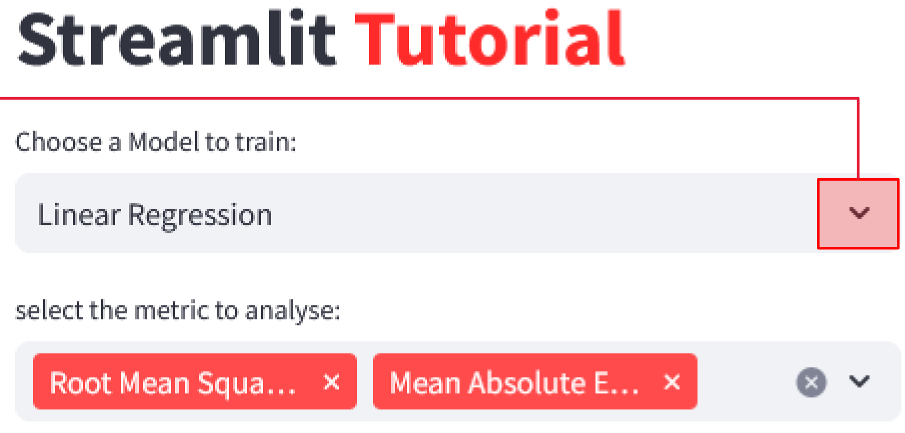
```
st.markdown('# Streamlit :red[Tutorial]')

option = st.selectbox("Choose a Model to train:",
                      ("Linear Regression",
                       "Logistic Regression",
                       "Polynomial Regression",
                       "Bayesian Linear Regression"
                       ))

options = st.multiselect("select the metric to analyse:",
                         ("Root Mean Squared Error",
                          "Mean Absolute Error",
                          "R-Square"
                          ))
```



Choose a Model to train:

Linear Regression

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Bayesian Linear Regression



Streamlit Tutorial

Choose a Model to train:

Linear Regression

select the metric to analyse:

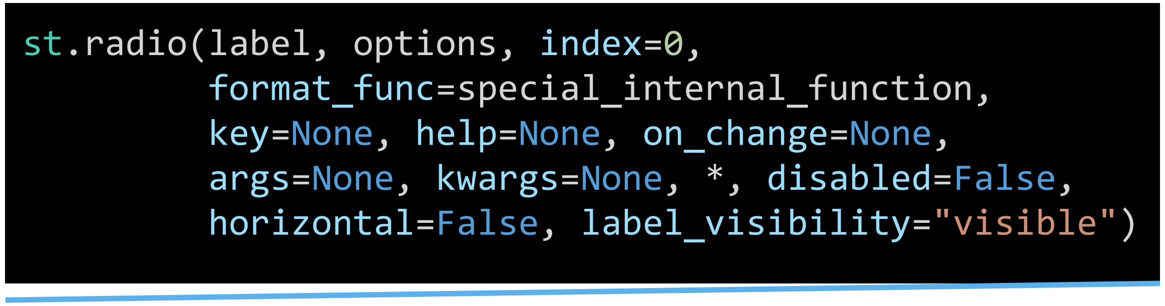
Root Mean Squa... × Mean Absolute E... ×

Building DS App with Streamlit

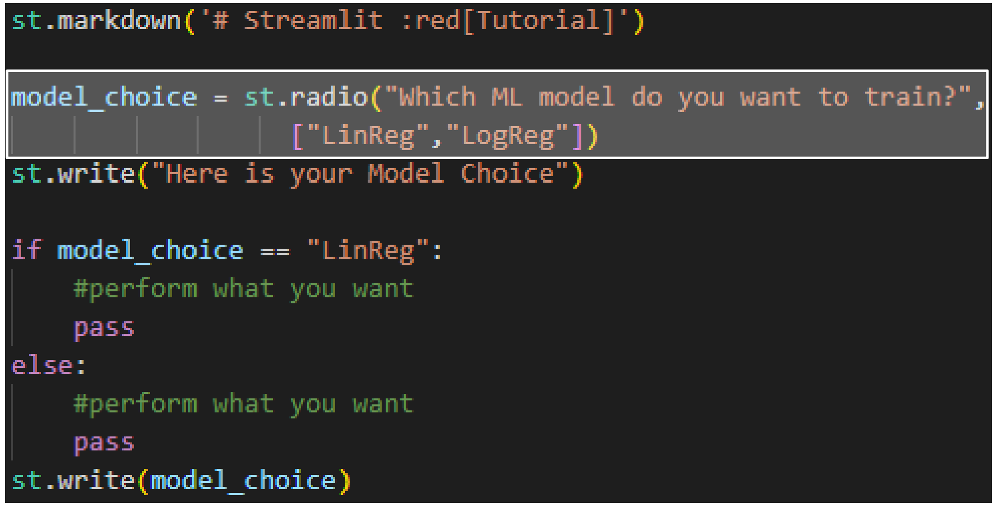
Streamlit

Radio Buttons & Sliders

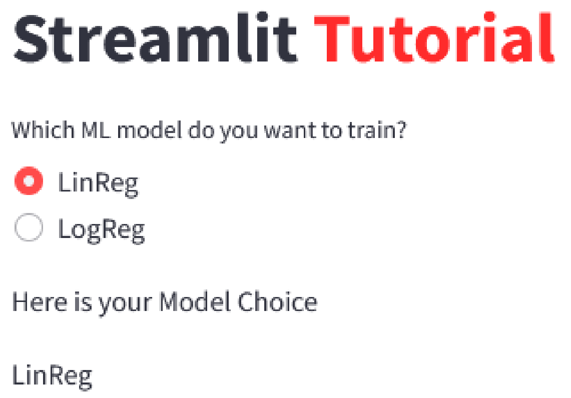
Radio Button



```
st.radio(label, options, index=0,
         format_func=special_internal_function,
         key=None, help=None, on_change=None,
         args=None, kwargs=None, *, disabled=False,
         horizontal=False, label_visibility="visible")
```



```
st.markdown('# Streamlit :red[Tutorial]')

model_choice = st.radio("Which ML model do you want to train?",
                         ["LinReg","LogReg"])
st.write("Here is your Model Choice")

if model_choice == "LinReg":
    #perform what you want
    pass
else:
    #perform what you want
    pass
st.write(model_choice)
```



Streamlit Tutorial

Which ML model do you want to train?

● LinReg

○ LogReg

Here is your Model Choice

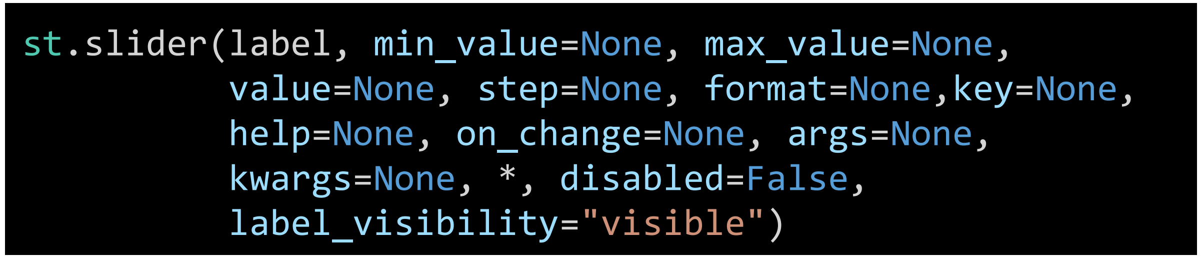
LinReg

Building DS App with Streamlit

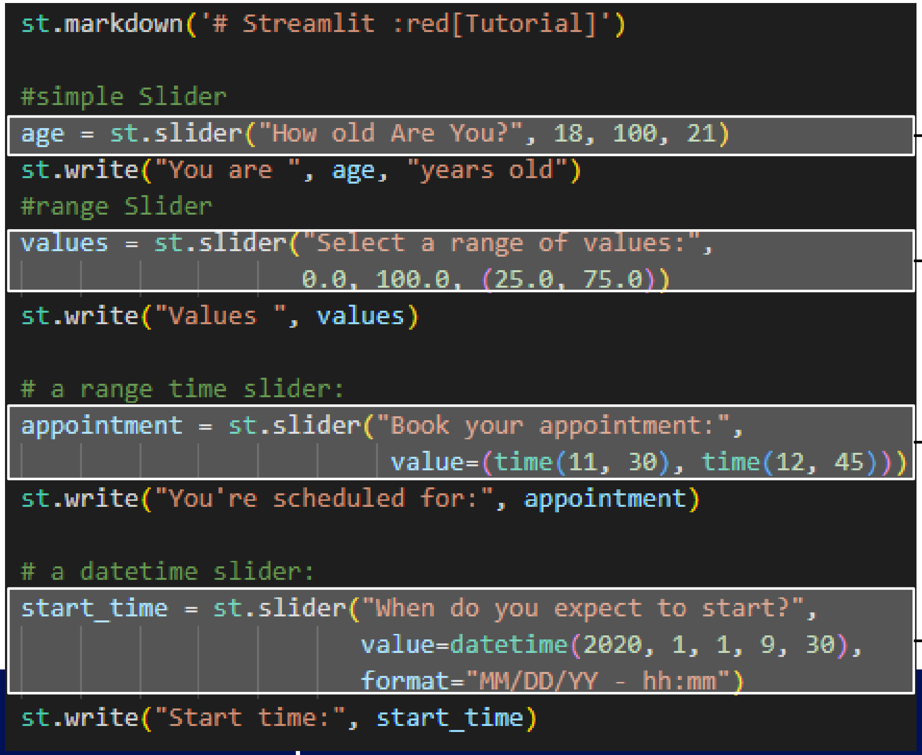
Streamlit

Radio Buttons & Sliders

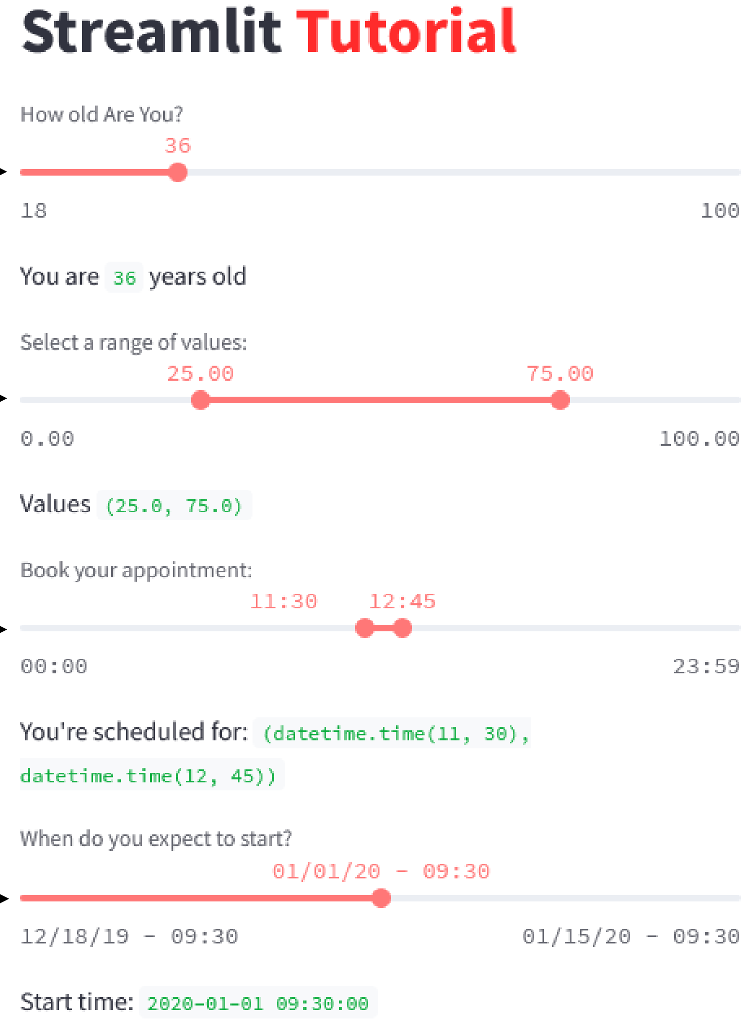
Slider



```
st.slider(label, min_value=None, max_value=None,
          value=None, step=None, format=None,key=None,
          help=None, on_change=None, args=None,
          kwargs=None, *, disabled=False,
          label_visibility="visible")
```



```
st.markdown('# Streamlit :red[Tutorial]')

#simple Slider
age = st.slider("How old Are You?", 18, 100, 21)
st.write("You are ", age, "years old")
#range Slider
values = st.slider("Select a range of values:",
                   0.0, 100.0, (25.0, 75.0))
st.write("Values ", values)

# a range time slider:
appointment = st.slider("Book your appointment:",
                        value=(time(11, 30), time(12, 45)))
st.write("You're scheduled for:", appointment)

# a datetime slider:
start_time = st.slider("When do you expect to start?",
                       value=datetime(2020, 1, 1, 9, 30),
                       format="MM/DD/YY - hh:mm")
st.write("Start time:", start_time)
```



Streamlit Tutorial

How old Are You?

You are 36 years old

Select a range of values:

Values (25.0, 75.0)

Book your appointment:

You're scheduled for: (datetime.time(11, 30), datetime.time(12, 45))

When do you expect to start?

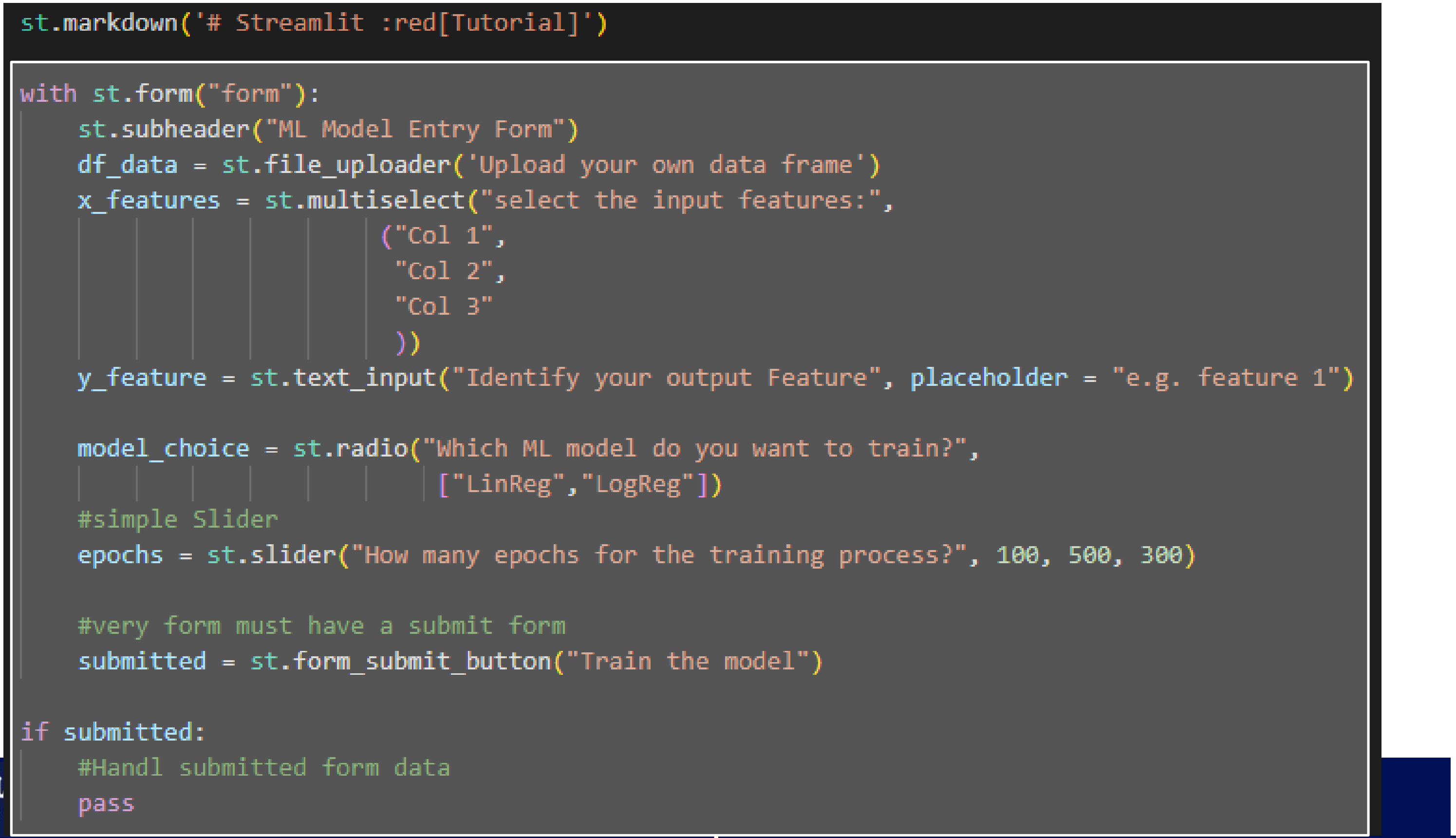
Start time: 2020-01-01 09:30:00

Building DS App with Streamlit

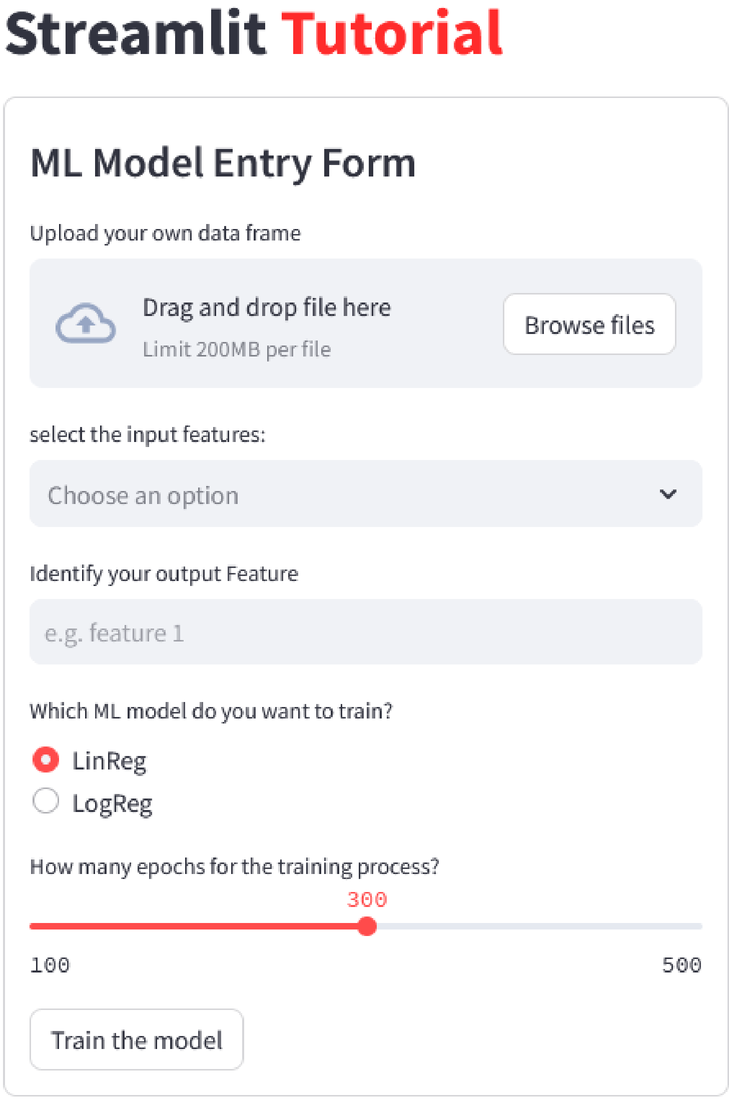
Forms

Form

```
st.form(key,
        clear_on_submit=False
```



```
st.markdown('# Streamlit :red[Tutorial]')

with st.form("form"):
    st.subheader("ML Model Entry Form")
    df_data = st.file_uploader('Upload your own data frame')
    x_features = st.multiselect("select the input features:",
                                ("Col 1",
                                 "Col 2",
                                 "Col 3"
                                 ))
    y_feature = st.text_input("Identify your output Feature", placeholder = "e.g. feature 1")

    model_choice = st.radio("Which ML model do you want to train?",
                                    ["LinReg","LogReg"])
    #simple Slider
    epochs = st.slider("How many epochs for the training process?", 100, 500, 300)

    #very form must have a submit form
    submitted = st.form_submit_button("Train the model")

if submitted:
    #Handl submitted form data
    pass
```



Streamlit Tutorial

ML Model Entry Form

Upload your own data frame

Drag and drop file here

Limit 200MB per file

select the input features:

Choose an option

Identify your output Feature

e.g. feature 1

Which ML model do you want to train?

● LinReg

○ LogReg

How many epochs for the training process?

300

100 500

Data Visualization Streamlit

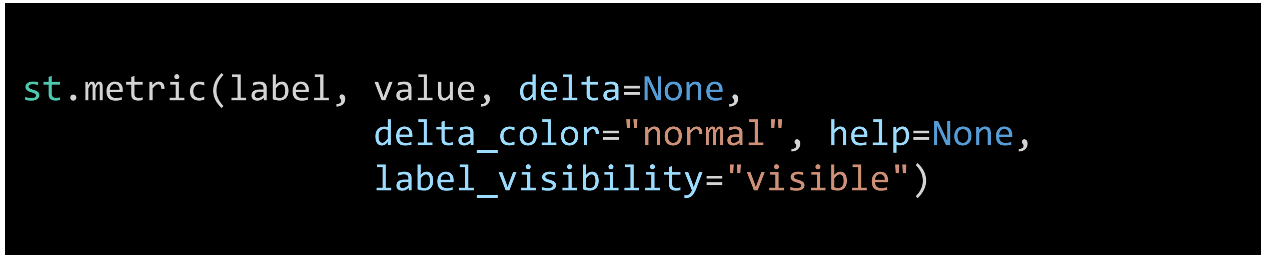
Building DS App with Streamlit

Building DS App with Streamlit

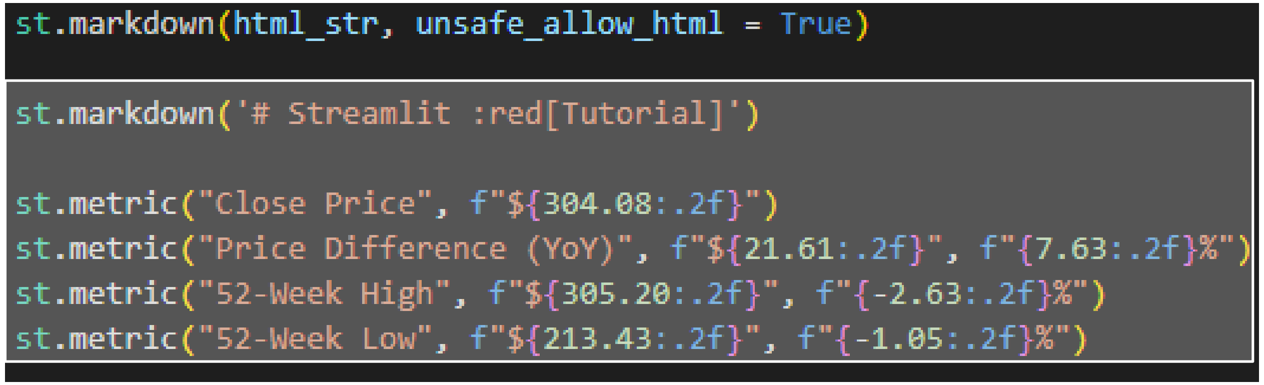
Streamlit

Metrics

Metric

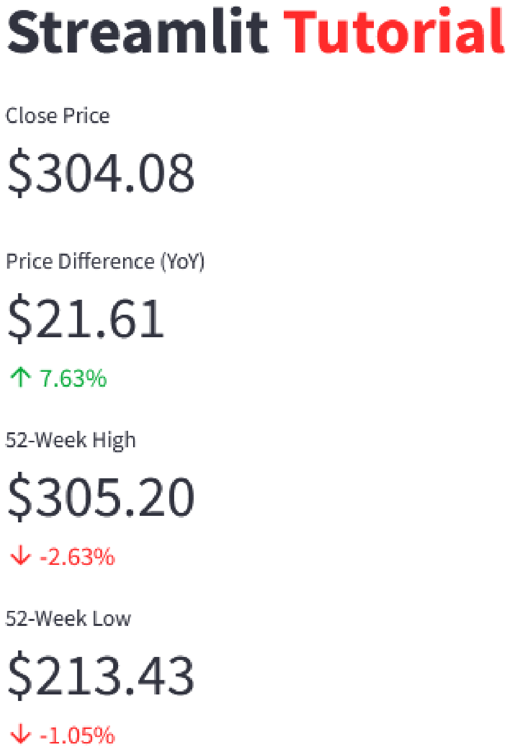


```
st.metric(label, value, delta=None,
                delta_color="normal", help=None,
                label_visibility="visible")
```



```
st.markdown(html_str, unsafe_allow_html = True)
```

```
st.markdown('# Streamlit :red[Tutorial]')

st.metric("Close Price", f"${304.08:.2f}")
st.metric("Price Difference (YoY)", f"${21.61:.2f}", f"{7.63:.2f}%")
st.metric("52-Week High", f"${305.20:.2f}", f"{-2.63:.2f}%")
st.metric("52-Week Low", f"${213.43:.2f}", f"{-1.05:.2f}%")
```



Streamlit Tutorial

Close Price

$304.08

Price Difference (YoY)

$21.61

↑ 7.63%

52-Week High

$305.20

↓ -2.63%

52-Week Low

$213.43

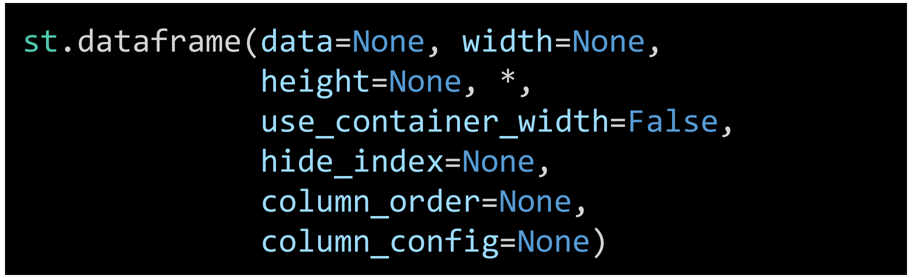
↓ -1.05%

Building DS App with Streamlit

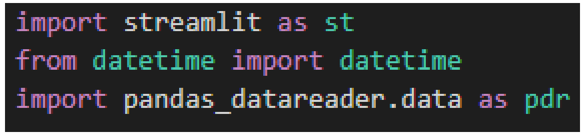
Streamlit

DataFrames

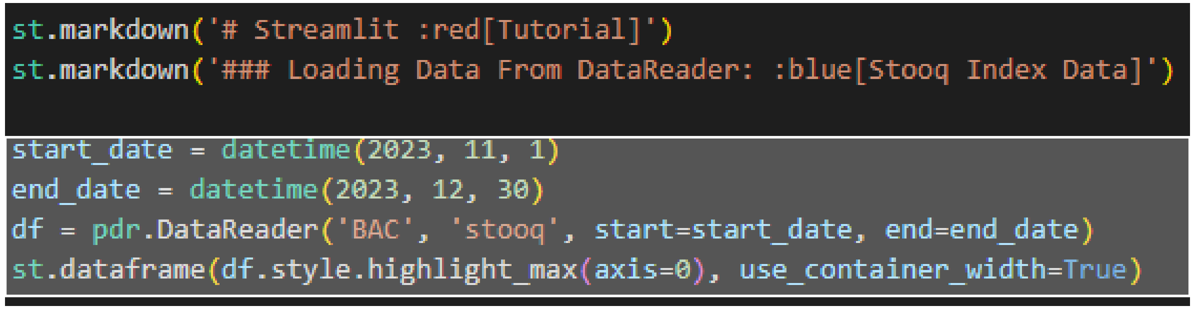
DataFrame



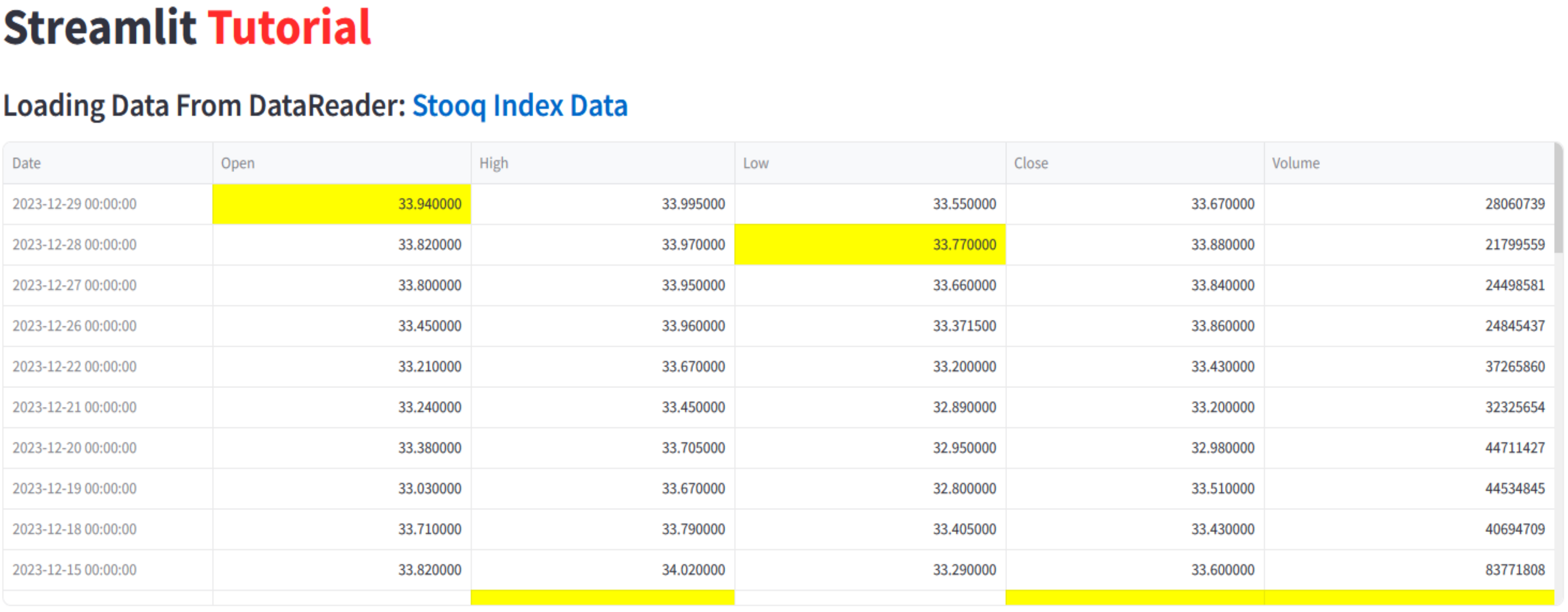
```
st.dataframe(data=None, width=None,
             height=None, *,
             use_container_width=False,
             hide_index=None,
             column_order=None,
             column_config=None)
```



```
import streamlit as st
from datetime import datetime
import pandas_datareader.data as pdr
```



```
st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Loading Data From DataReader: :blue[Stooq Index Data]')

start_date = datetime(2023, 11, 1)
end_date = datetime(2023, 12, 30)
df = pdr.DataReader('BAC', 'stooq', start=start_date, end=end_date)
st.dataframe(df.style.highlight_max(axis=0), use_container_width=True)
```



Streamlit Tutorial

Loading Data From DataReader: Stooq Index Data

Date	Open	High	Low	Close	Volume
2023-12-29 00:00:00	33.940000	33.995000	33.550000	33.670000	28060739
2023-12-28 00:00:00	33.820000	33.970000	33.770000	33.880000	21799559
2023-12-27 00:00:00	33.800000	33.950000	33.660000	33.840000	24498581
2023-12-26 00:00:00	33.450000	33.960000	33.371500	33.860000	24845437
2023-12-22 00:00:00	33.210000	33.670000	33.200000	33.430000	37265860
2023-12-21 00:00:00	33.240000	33.450000	32.890000	33.200000	32325654
2023-12-20 00:00:00	33.380000	33.705000	32.950000	32.980000	44711427
2023-12-19 00:00:00	33.030000	33.670000	32.800000	33.510000	44534845
2023-12-18 00:00:00	33.710000	33.790000	33.405000	33.430000	40694709
2023-12-15 00:00:00	33.820000	34.020000	33.290000	33.600000	83771808

Building DS App with Streamlit

Streamlit

Charts

Several libraries are supported for the graphical representation of data through interactive charts

- □ Matplotlib
- □ Plotly
- □ Altair
- □ deck.gl (maps and 3D graphs)

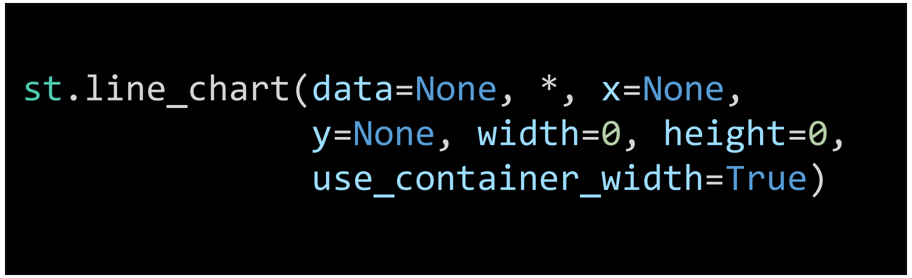
To speed up the integration of the most common charts, some are natively integrated into Streamlit (with less customization):

- □ Line chart
- □ Area chart
- □ Bar chart
- □ Scatterplot on map

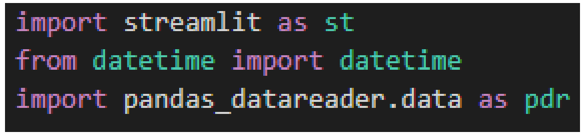
Building DS App with Streamlit

Charts

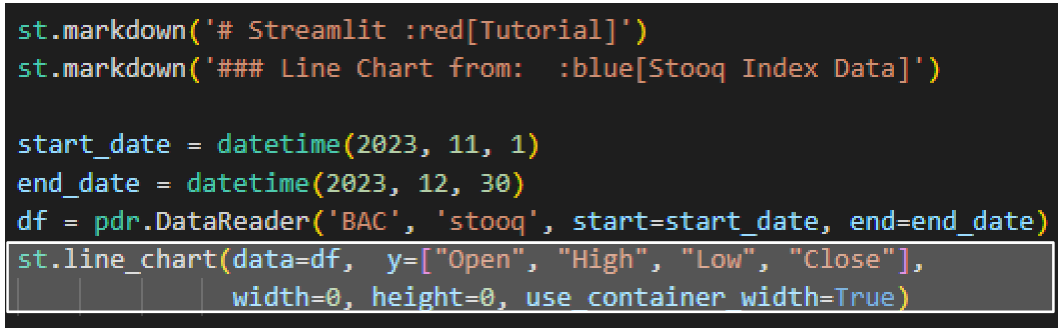
Line Chart



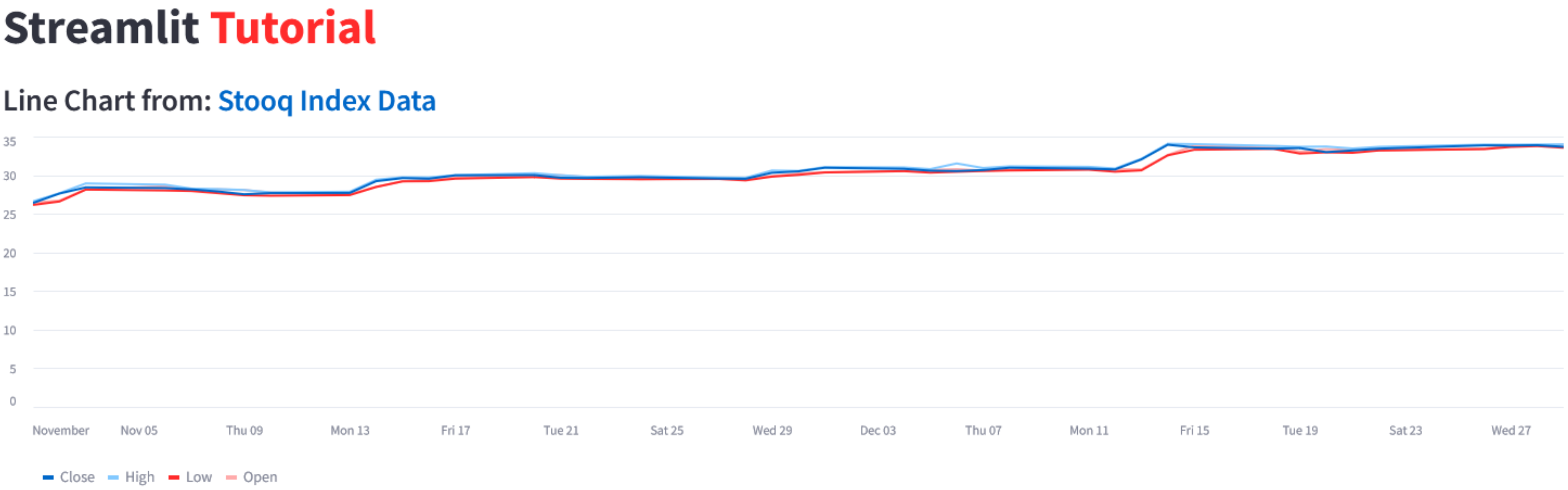
```
st.line_chart(data=None, *, x=None,
              y=None, width=0, height=0,
              use_container_width=True)
```



```
import streamlit as st
from datetime import datetime
import pandas_datareader.data as pdr
```



```
st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Line Chart from:  :blue[Stooq Index Data]')

start_date = datetime(2023, 11, 1)
end_date = datetime(2023, 12, 30)
df = pdr.DataReader('BAC', 'stooq', start=start_date, end=end_date)
st.line_chart(data=df,  y=["Open", "High", "Low", "Close"],
              width=0, height=0, use_container_width=True)
```



Streamlit Tutorial

Line Chart from: Stooq Index Data

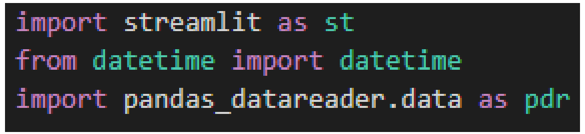
Building DS App with Streamlit

Streamlit

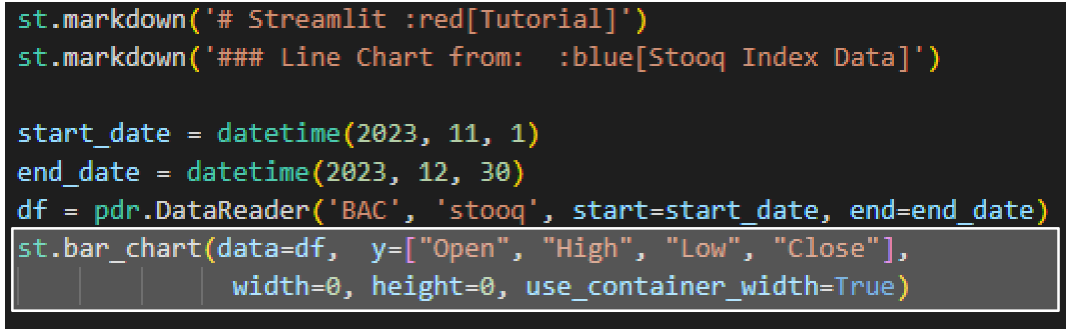
Charts

Bar Chart

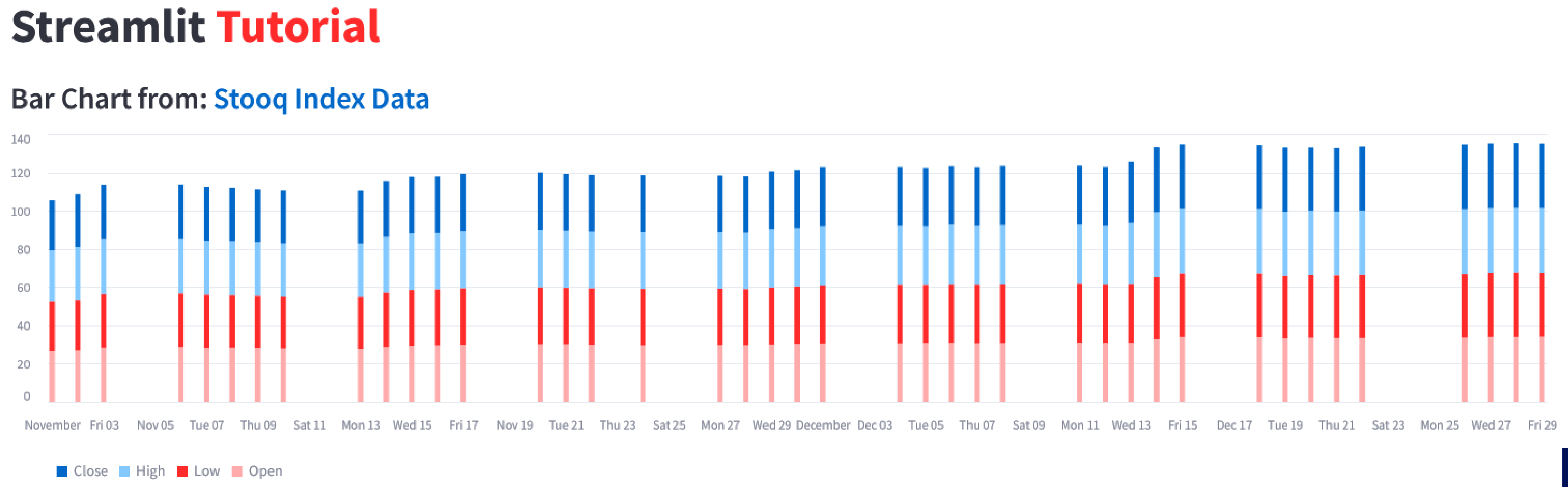
```
st.bar_chart(data=None, *, x=None,
             y=None, width=0, height=0,
             use_container_width=True)
```



```
import streamlit as st
from datetime import datetime
import pandas_datareader.data as pdr
```



```
st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Line Chart from:  :blue[Stooq Index Data]')

start_date = datetime(2023, 11, 1)
end_date = datetime(2023, 12, 30)
df = pdr.DataReader('BAC', 'stooq', start=start_date, end=end_date)
st.bar_chart(data=df,  y=["Open", "High", "Low", "Close"],
                 width=0, height=0, use_container_width=True)
```



Streamlit Tutorial

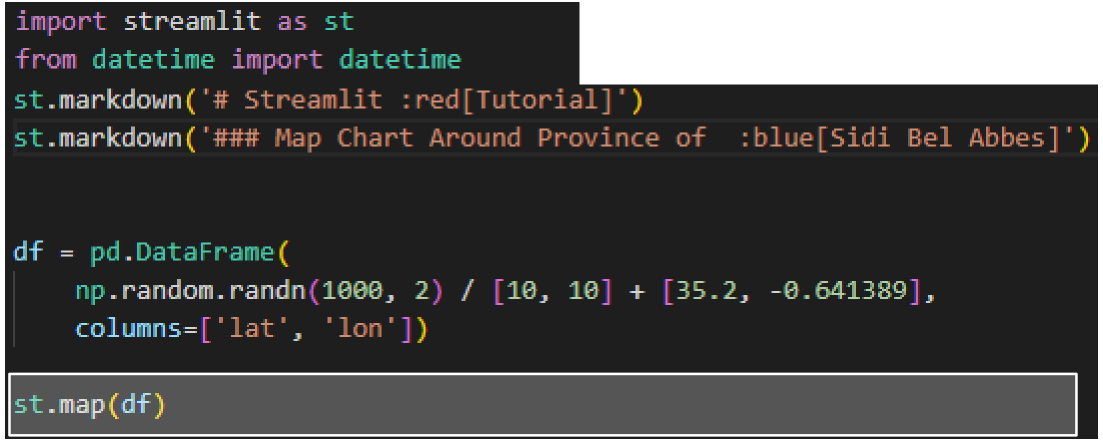
Bar Chart from: Stooq Index Data

Building DS App with Streamlit

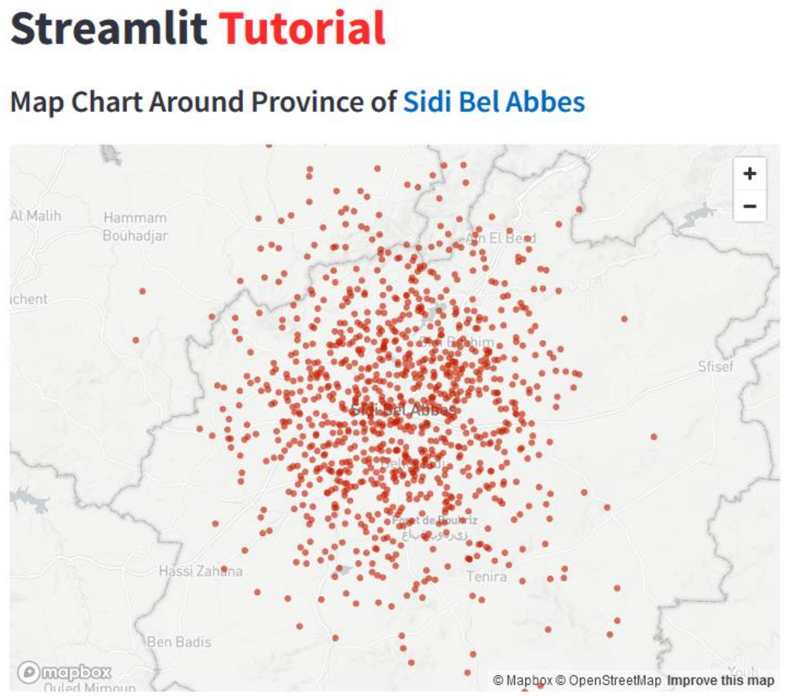
Maps

Map Chart

```
st.map(data=None, *, latitude=None,
       longitude=None, color=None,
       size=None, zoom=None,
       use_container_width=True)
```



```
import streamlit as st
from datetime import datetime
st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Map Chart Around Province of  :blue[Sidi Bel Abbes]')


df = pd.DataFrame(
    np.random.randn(1000, 2) / [10, 10] + [35.2, -0.641389],
    columns=['lat', 'lon'])

st.map(df)
```



Streamlit Tutorial

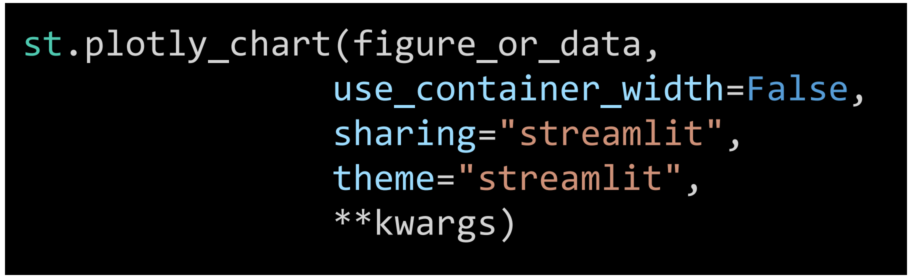
Map Chart Around Province of Sidi Bel Abbes

- ❑ The **data** parameter must have two columns: **'lat'** or **'latitude'**, and **'lon'** or **'longitude'**.
- ❑ The map relies on the external service **Mapbox** and requires a **token** (currently offered automatically by **Streamlit**)

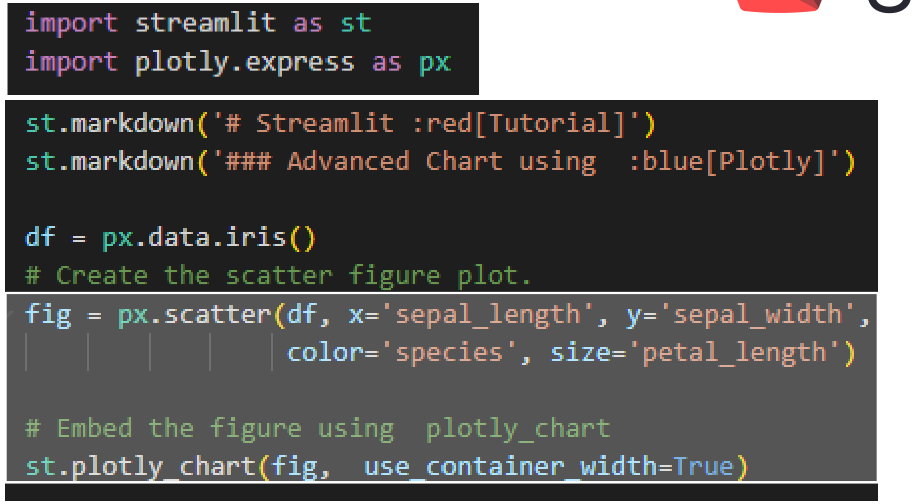
Building DS App with Streamlit

Advanced Charts using Plotly

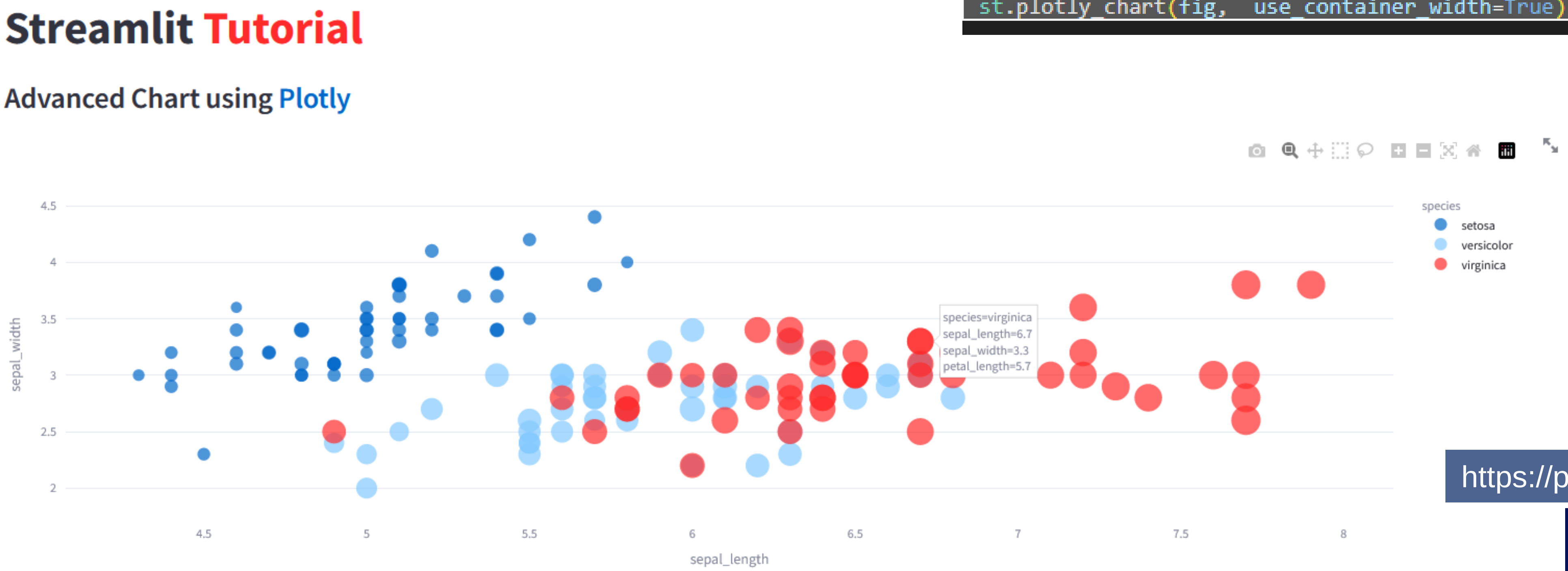
Map Chart



```
st.plotly_chart(figure_or_data,
                use_container_width=False,
                sharing="streamlit",
                theme="streamlit",
                **kwargs)
```



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

st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Advanced Chart using  :blue[Plotly]')

df = px.data.iris()
# Create the scatter figure plot.
fig = px.scatter(df, x='sepal_length', y='sepal_width',
                 color='species', size='petal_length')

# Embed the figure using  plotly_chart
st.plotly_chart(fig,  use_container_width=True)
```



Streamlit Tutorial

Advanced Chart using Plotly

More details about advanced charts using Plotly

https://plotly.com/python/getting-started/

Additional Elements Streamlit

Building DS App with Streamlit

Building DS App with Streamlit

Streamlit

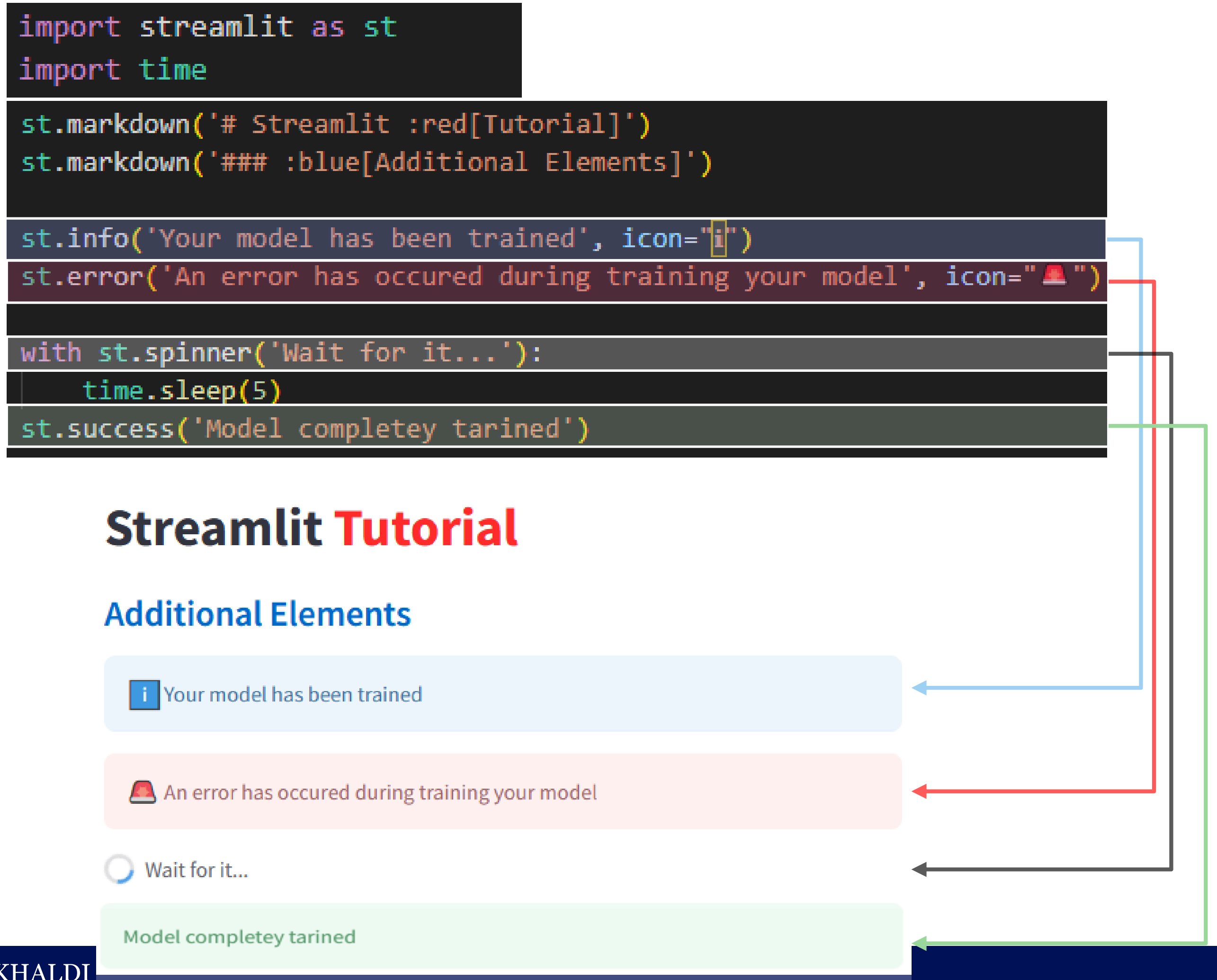
Status Messages & Spinners

Status Message

```
st.error(body, *, icon=None)
st.warning(body, *, icon=None)
st.info(body, *, icon=None)
st.success(body, *, icon=None)
```

Spinner

```
st.spinner(text="In progress...",
           *, cache=False)
```



```
import streamlit as st
import time

st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### :blue[Additional Elements]')

st.info('Your model has been trained', icon="ℹ")
st.error('An error has occured during training your model', icon="🚨")

with st.spinner('Wait for it...'):
    time.sleep(5)
st.success('Model completey tarined')
```

Streamlit Tutorial

Additional Elements

ℹ Your model has been trained

🚨 An error has occured during training your model

Wait for it...

Model completey tarined

Building DS App with Streamlit

Streamlit

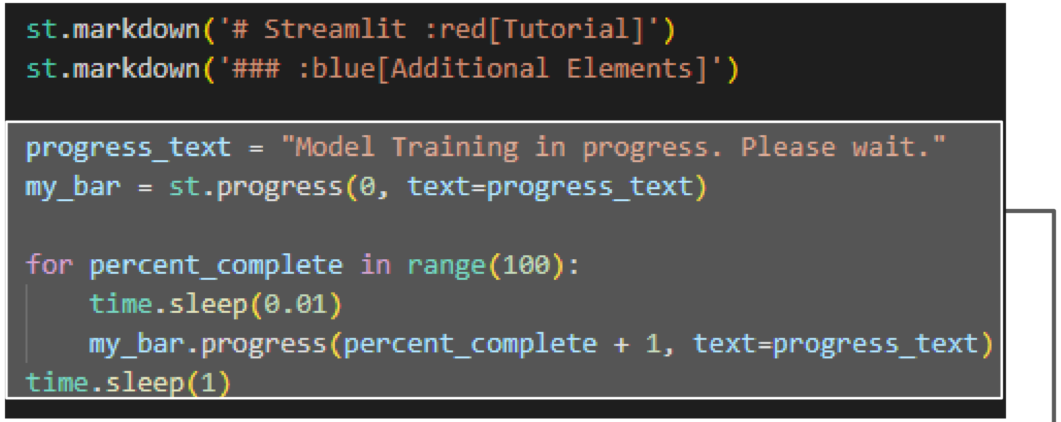
Progress Bars & Session state

Progress Bar

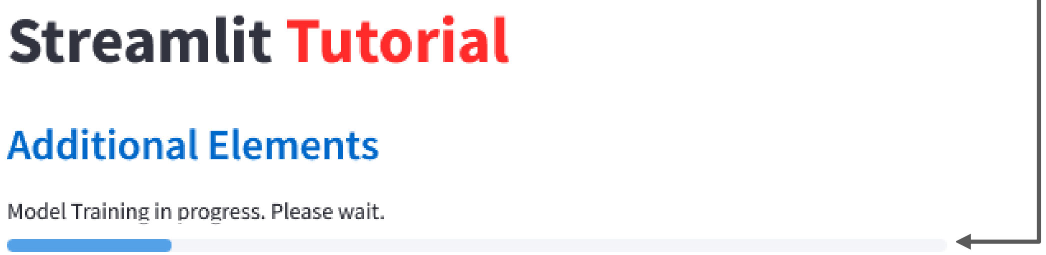
```
st.progress(value, text=None)
```

Session State

```
# Initialization
if 'key' not in st.session_state:
    st.session_state['key'] = 'value'

# or
if 'key' not in st.session_state:
    st.session_state.key = 'value'
#Updating
st.session_state.key = 'value2'
st.session_state['key'] = 'value2'
# Deleting
del st.session_state[key]
```



```
st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### :blue[Additional Elements]')

progress_text = "Model Training in progress. Please wait."
my_bar = st.progress(0, text=progress_text)

for percent_complete in range(100):
    time.sleep(0.01)
    my_bar.progress(percent_complete + 1, text=progress_text)
time.sleep(1)
```

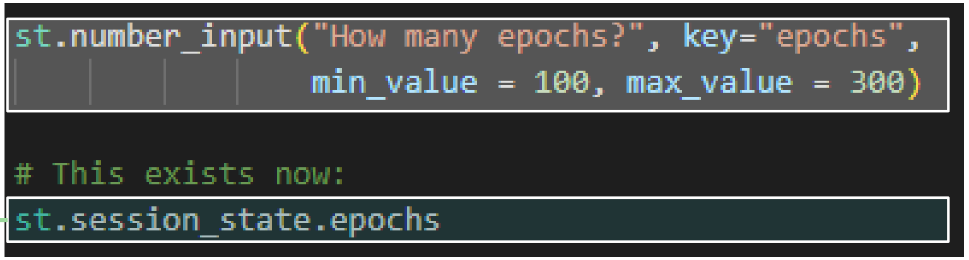


Streamlit Tutorial

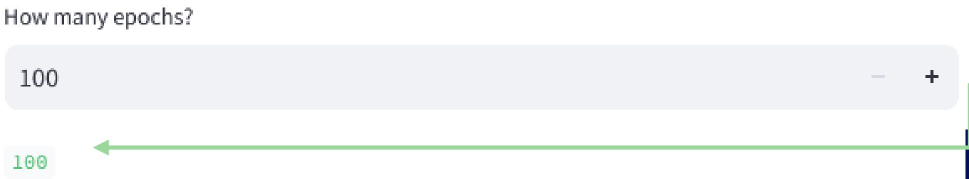
Additional Elements

Model Training in progress. Please wait.

Every **widget** with a **key** is automatically added to **Session State**:



```
st.number_input("How many epochs?", key="epochs",
                min_value = 100, max_value = 300)

# This exists now:
st.session_state.epochs
```



How many epochs?

100

```
100
```

Layout Streamlit

Building DS App with Streamlit

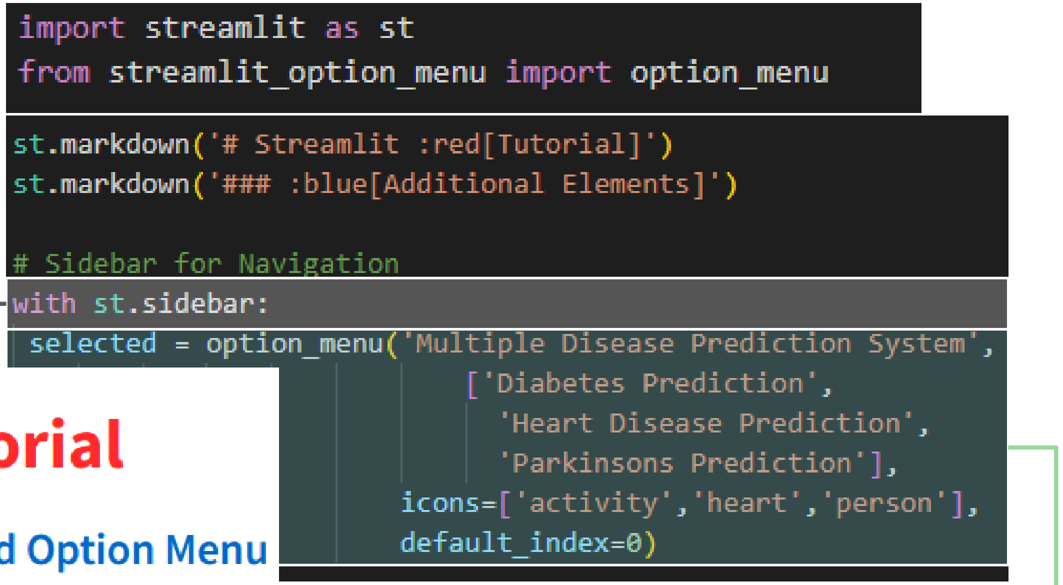
Building DS App with Streamlit

Streamlit

Sidebars, Tabs, Columns, & Expanders

Sidebar

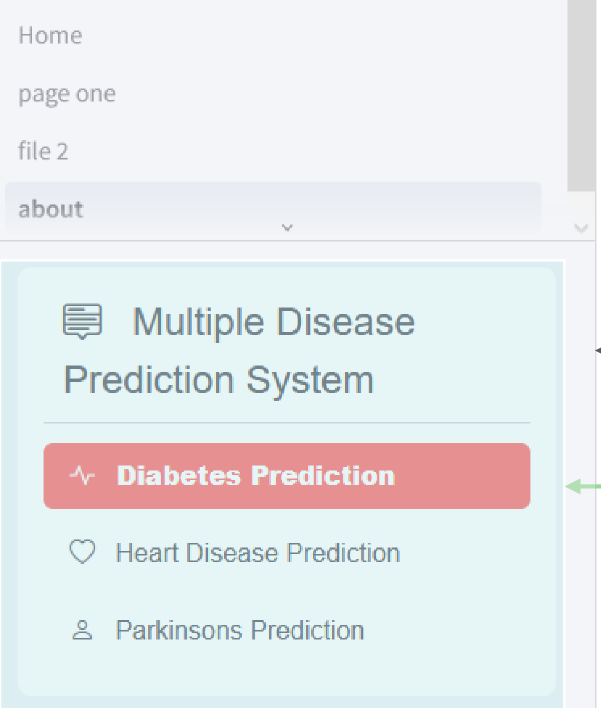
```
with st.sidebar:
     # Add UI elements
```



```
import streamlit as st
from streamlit_option_menu import option_menu

st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### :blue[Additional Elements]')

# Sidebar for Navigation
with st.sidebar:
    selected = option_menu('Multiple Disease Prediction System',
                           ['Diabetes Prediction',
                            'Heart Disease Prediction',
                            'Parkinsons Prediction'],
                           icons=['activity','heart','person'],
                           default_index=0)
```

Streamlit Tutorial

Sidebar with Advanced Option Menu



```
pip install streamlit-option-menu
```

More details about advanced **Streamlit Advanced Components** from the **community**.

https://streamlit.io/components

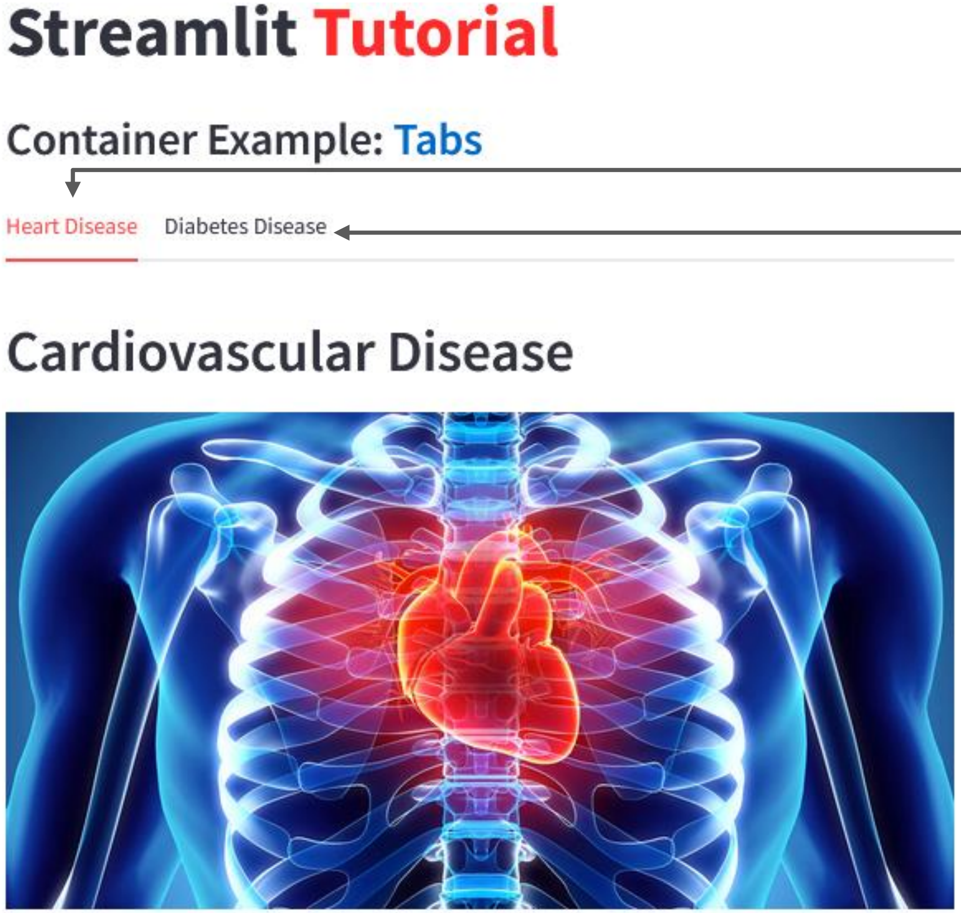
https://pypi.org/search/?q=streamlit+components&o=

Building DS App with Streamlit

Sidebars, Tabs, Columns, & Expanders

Tab

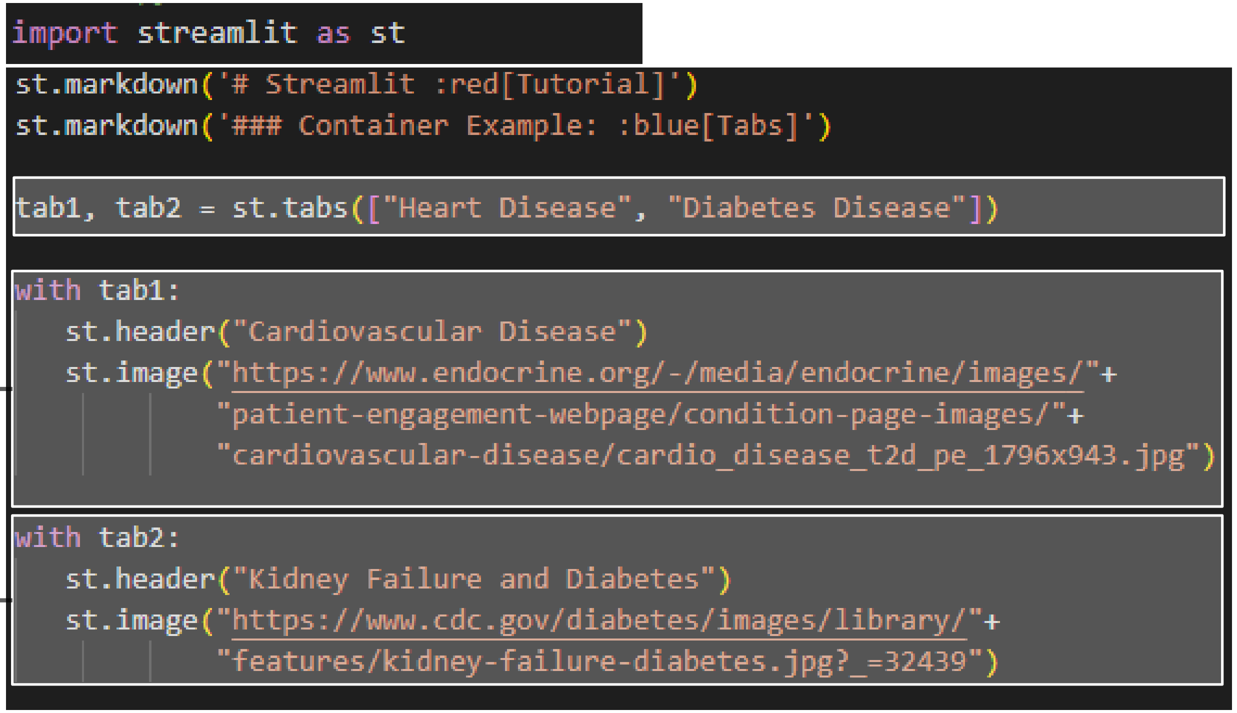
```
st.tabs(tabs)
```



Streamlit Tutorial

Container Example: Tabs

Cardiovascular Disease



```
import streamlit as st
st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Container Example: :blue[Tabs]')

tab1, tab2 = st.tabs(["Heart Disease", "Diabetes Disease"])

with tab1:
    st.header("Cardiovascular Disease")
    st.image("https://www.endocrine.org/-/media/endocrine/images/"+
             "patient-engagement-webpage/condition-page-images/"+
             "cardiovascular-disease/cardio_disease_t2d_pe_1796x943.jpg")

with tab2:
    st.header("Kidney Failure and Diabetes")
    st.image("https://www.cdc.gov/diabetes/images/library/"+
             "features/kidney-failure-diabetes.jpg?_=32439")
```

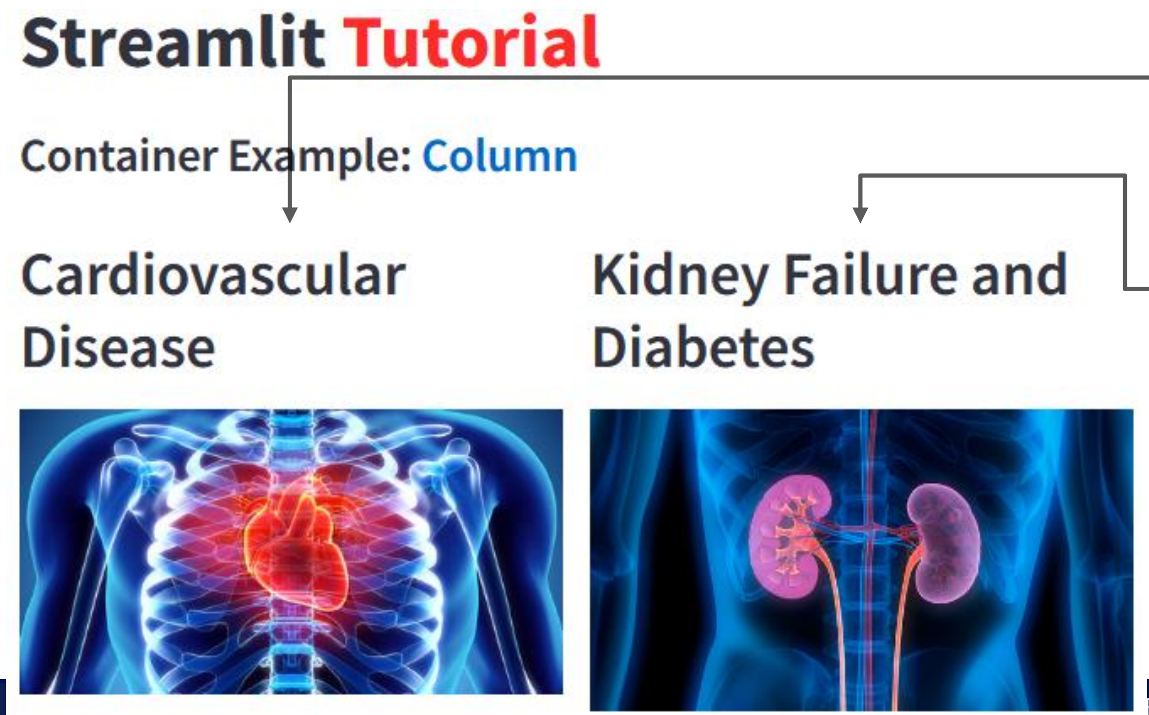
Building DS App with Streamlit

Sidebars, Tabs, Columns, & Expanders

Streamlit

Column

```
st.columns(spec, *, gap="small")
```

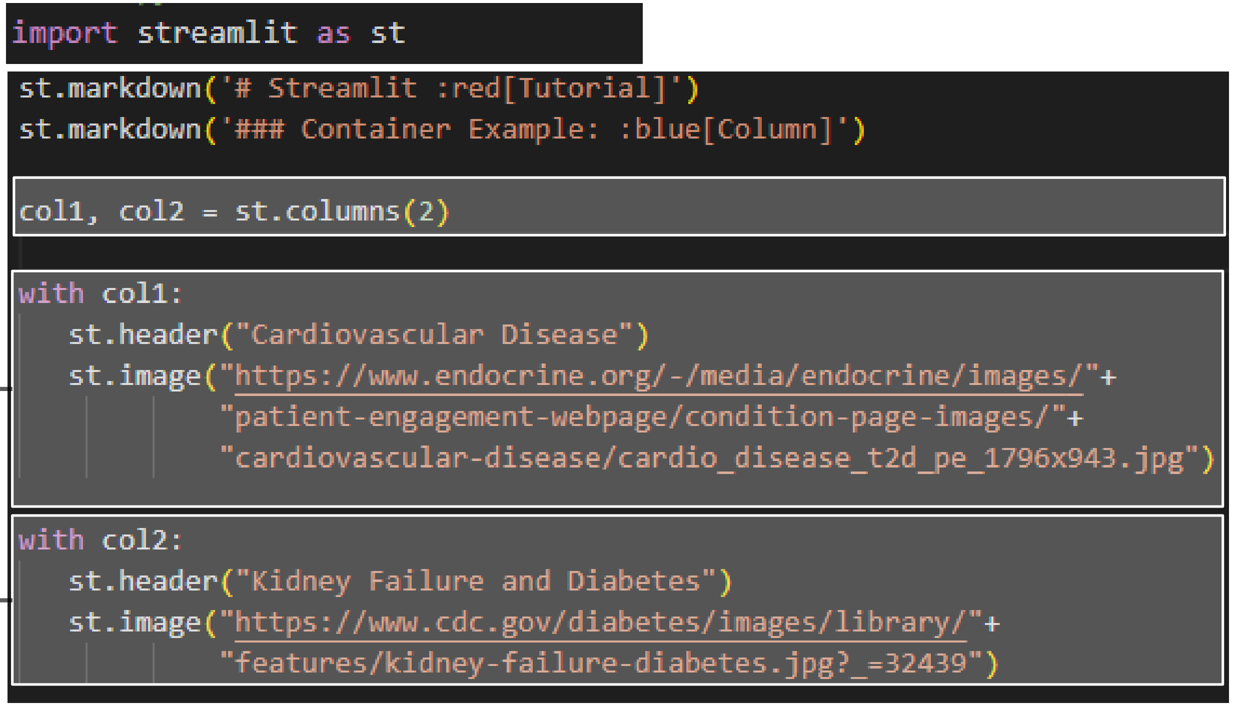


Streamlit Tutorial

Container Example: Column

Cardiovascular Disease

Kidney Failure and Diabetes



```
import streamlit as st
st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Container Example: :blue[Column]')

col1, col2 = st.columns(2)

with col1:
    st.header("Cardiovascular Disease")
    st.image("https://www.endocrine.org/-/media/endocrine/images/"+
             "patient-engagement-webpage/condition-page-images/"+
             "cardiovascular-disease/cardio_disease_t2d_pe_1796x943.jpg")

with col2:
    st.header("Kidney Failure and Diabetes")
    st.image("https://www.cdc.gov/diabetes/images/library/"+
             "features/kidney-failure-diabetes.jpg?_=32439")
```

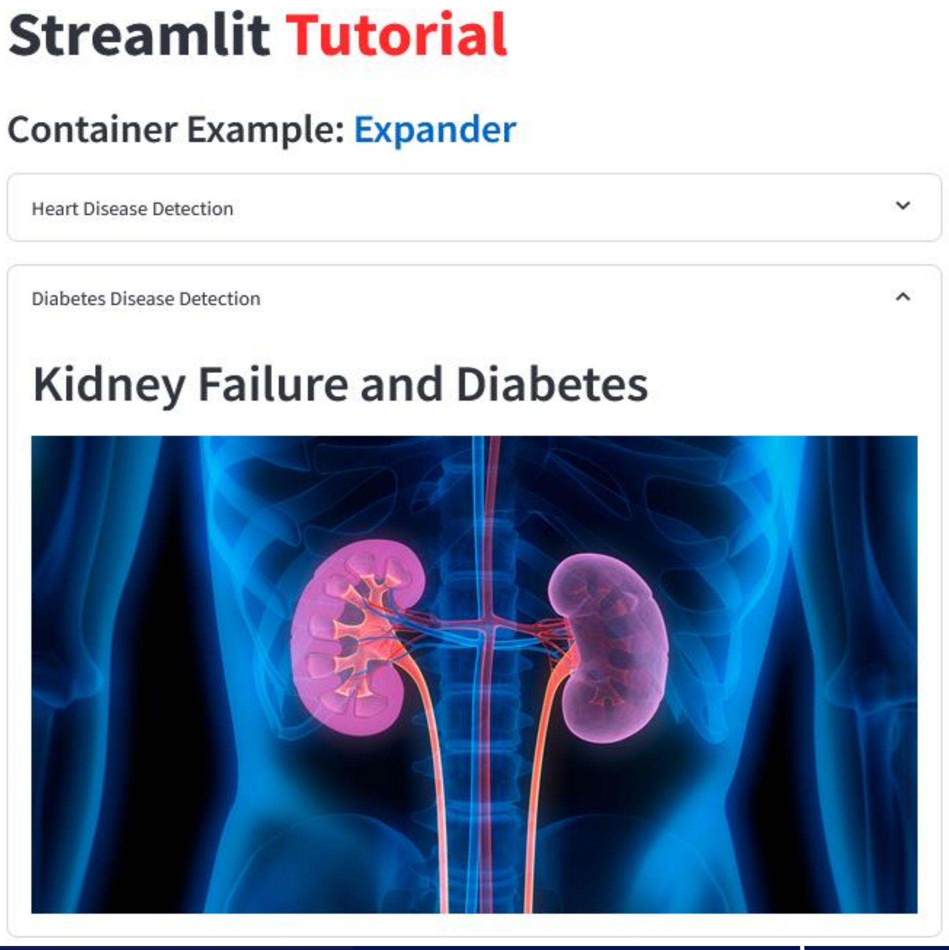
Building DS App with Streamlit

Streamlit

Sidebars, Tabs, Columns, & Expanders

Expander

```
st.expander(label, expanded=False)
```



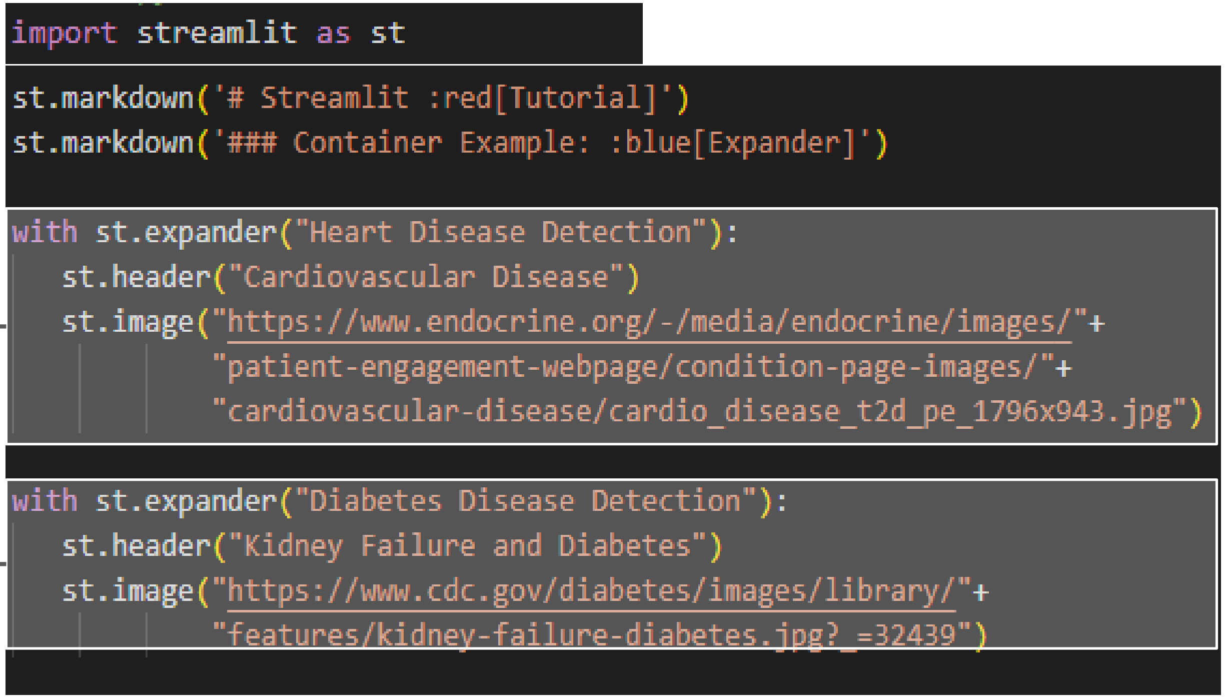
Streamlit Tutorial

Container Example: Expander

Heart Disease Detection

Diabetes Disease Detection

Kidney Failure and Diabetes



```
import streamlit as st

st.markdown('# Streamlit :red[Tutorial]')
st.markdown('### Container Example: :blue[Expander]')

with st.expander("Heart Disease Detection"):
    st.header("Cardiovascular Disease")
    st.image("https://www.endocrine.org/-/media/endocrine/images/"+
             "patient-engagement-webpage/condition-page-images/"+
             "cardiovascular-disease/cardio_disease_t2d_pe_1796x943.jpg")

with st.expander("Diabetes Disease Detection"):
    st.header("Kidney Failure and Diabetes")
    st.image("https://www.cdc.gov/diabetes/images/library/"+
             "features/kidney-failure-diabetes.jpg?_=32439")
```

Caching Mechanisms Streamlit

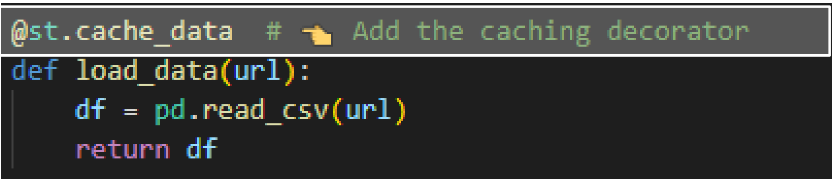
Building DS App with Streamlit

Building DS App with Streamlit

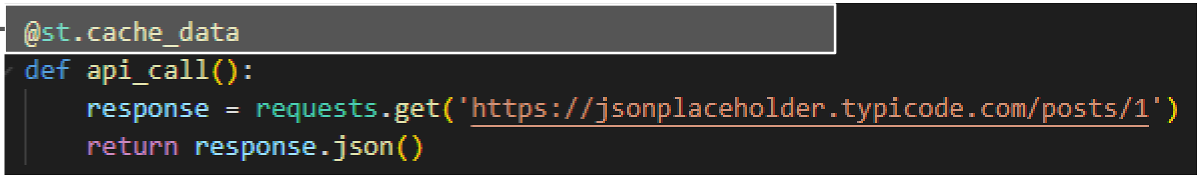
Streamlit

Caching Mechanisms

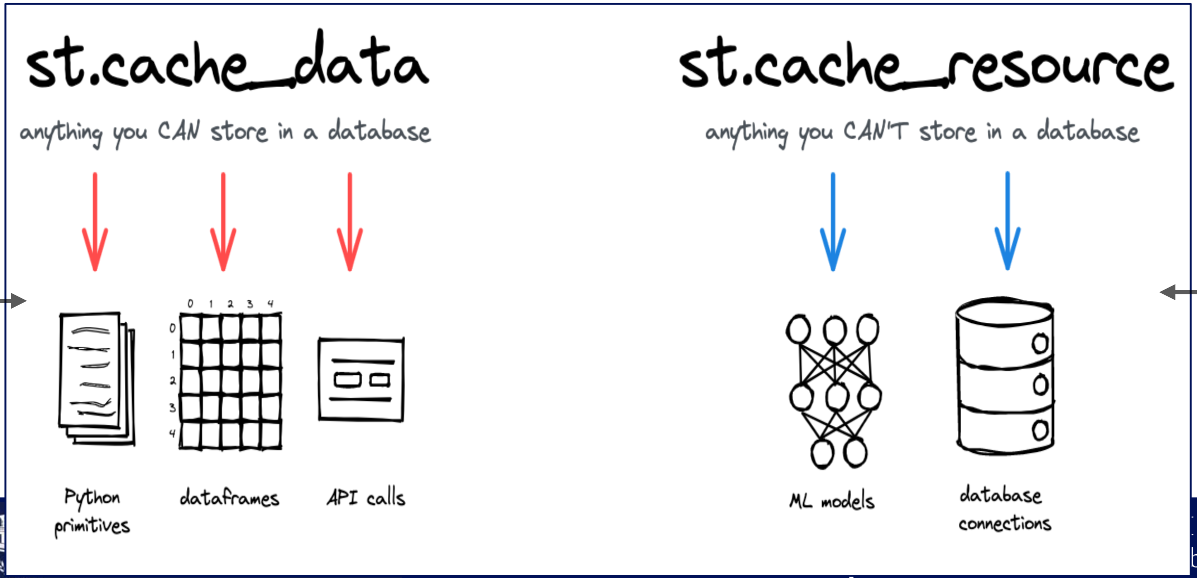
Challenges in Streamlit Execution Model	Rerunning long functions can slow down the app.
	Objects are recreated, posing challenges in persistence
Streamlit's Caching Solution	Addresses challenges with built-in caching.
	Ensures efficiency and persistence across reruns.



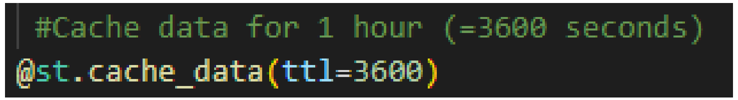
```
@st.cache_data  # 👈 Add the caching decorator
def load_data(url):
    df = pd.read_csv(url)
    return df
```



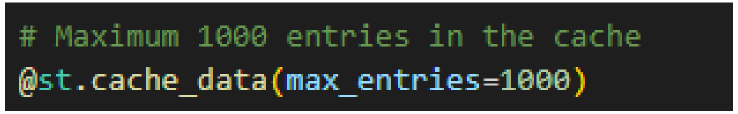
```
@st.cache_data
def api_call():
    response = requests.get('https://jsonplaceholder.typicode.com/posts/1')
    return response.json()
```



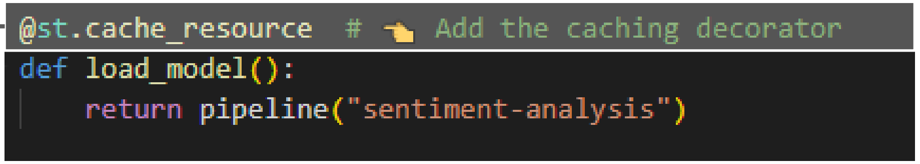
Controlling cache size and duration



```
#Cache data for 1 hour (=3600 seconds)
@st.cache_data(ttl=3600)
```



```
# Maximum 1000 entries in the cache
@st.cache_data(max_entries=1000)
```



```
@st.cache_resource  # 👈 Add the caching decorator
def load_model():
    return pipeline("sentiment-analysis")
```

st.cache_data implicitly uses the **pickle** module. Anything your cached function returns is **pickled** and **stored**, then **unpickled** on retrieval.

st.cache_resource is similar to **st.cache_data**. It does not create a copy of the cached return value but instead stores the object itself in the cache.

Building DS App with Streamlit

Streamlit

Caching Mechanisms – A Pre-Trained Example Model from Hugging Face

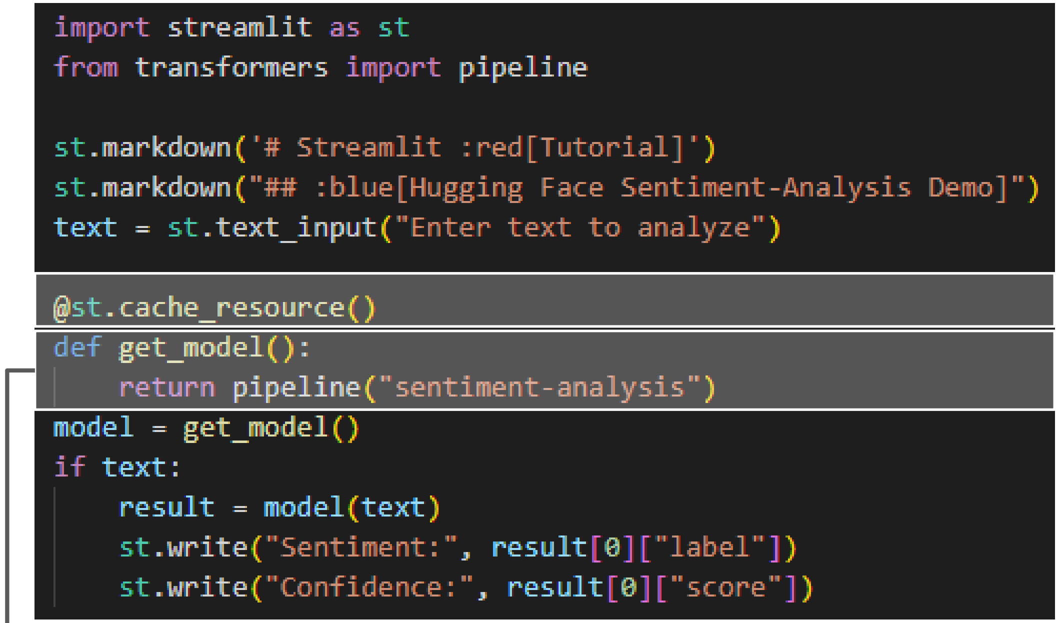
Hugging Face:

Hugging Face

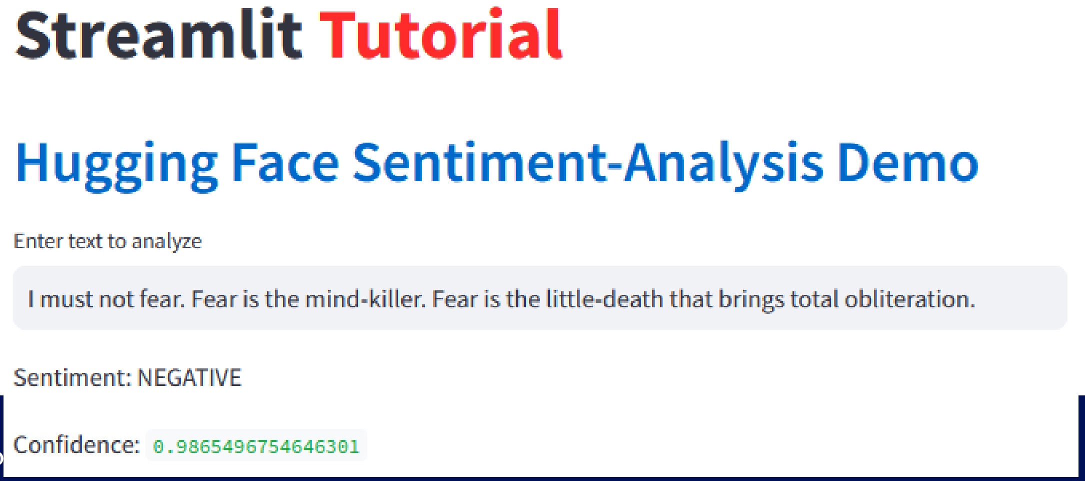
- ❑ **Prioritizing Open Source and Sharing**
  - ❑ Stands out for open-sourcing models and methods.
  - ❑ Facilitates easy access to models from top researchers.
- ❑ **Simplifying ML Model Usage**
  - ❑ Easy integration of models into custom use cases.
  - ❑ Seamless integration with Streamlit.

```
pip install transformers[torch]
```

`Pipeline:` A high-level API that allows you to use a pre-trained model for specific NLP tasks (e.g., “**`sentiement-analysis`**” model).



```python
import streamlit as st
from transformers import pipeline

st.markdown('# Streamlit :red[Tutorial]')
st.markdown("## :blue[Hugging Face Sentiment-Analysis Demo]")
text = st.text_input("Enter text to analyze")

@st.cache_resource()
def get_model():
    return pipeline("sentiment-analysis")
model = get_model()
if text:
    result = model(text)
    st.write("Sentiment:", result[0]["label"])
    st.write("Confidence:", result[0]["score"])
```



Streamlit Tutorial

Hugging Face Sentiment-Analysis Demo

Enter text to analyze

I must not fear. Fear is the mind-killer. Fear is the little-death that brings total obliteration.

Sentiment: NEGATIVE

Confidence: `0.9865496754646301`

Thanks for your Listening