# Streamlit

Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. All in pure Python. No front-end experience required. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.
developing a dashboard for your machine learning solution has been made incredibly easy

**Pros:**

- Streamlit is really easy to start prototyping in, everything is done in python
- Streamlit is integrated with Python. The usual visualization libraries you may use like matplotlib and plotly can easily be used with Streamlit. it is compatible with major Python libraries used in Data Science such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.
- Less code is needed to create a beautiful application
- No callbacks are needed since widgets are treated as variables

**Cons:**

- Streamlit is constrained in adding options like session management and secure authentication.
- Every time you perform an action in Streamlit, such as clicking a button, all Python code is re-executed
- Cannot create apis, the only way to use it is with its own frontend

### Understanding the Streamlit basic functions

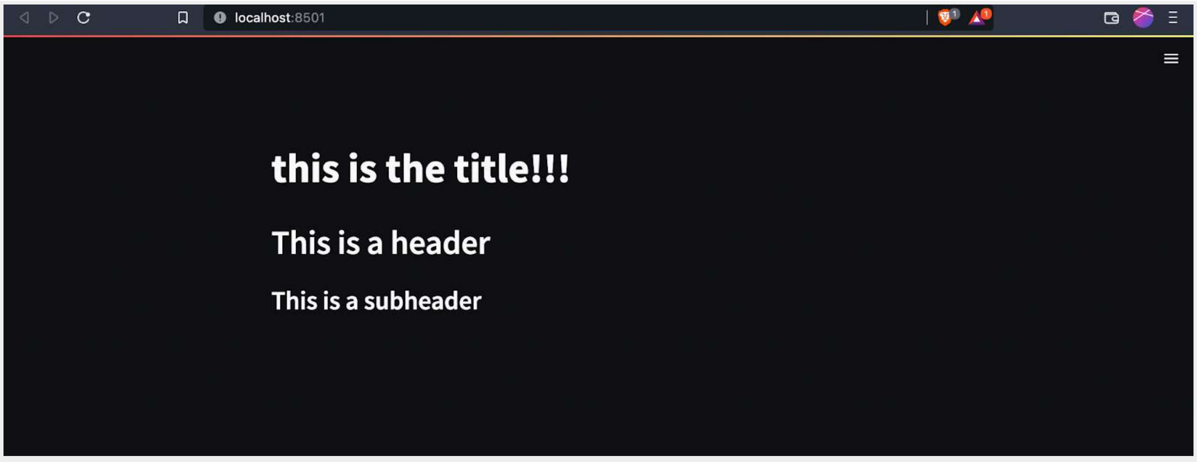
Streamlit lays out everything in the form of elements and widgets. The elements that are interactable are widgets.

```python
# import module
import streamlit as st

# Title
st.title("this is the title!!!")
```

```python
# Header
st.header("This is a header")

# Subheader
st.subheader("This is a subheader")
```



this is the title!!!

This is a header

This is a subheader

Now to run a streamlit app run

```
streamlit run your_script.py
```

As soon as you run the script as shown above, a local Streamlit server will spin up and your app will open in a new tab in your default web browser. The app is your canvas, where you'll draw charts, text, widgets, tables, and more.

What gets drawn in the app is up to you. For example st.text writes raw text to your app, and st.line_chart draws — you guessed it — a line chart. Refer to API documentation to see all commands that are available to you.

**Data flow**

Streamlit's architecture allows you to write apps the same way you write plain Python scripts. To unlock this, Streamlit apps have a unique data flow: any time something must be updated on the screen, Streamlit reruns your entire Python script from top to bottom.

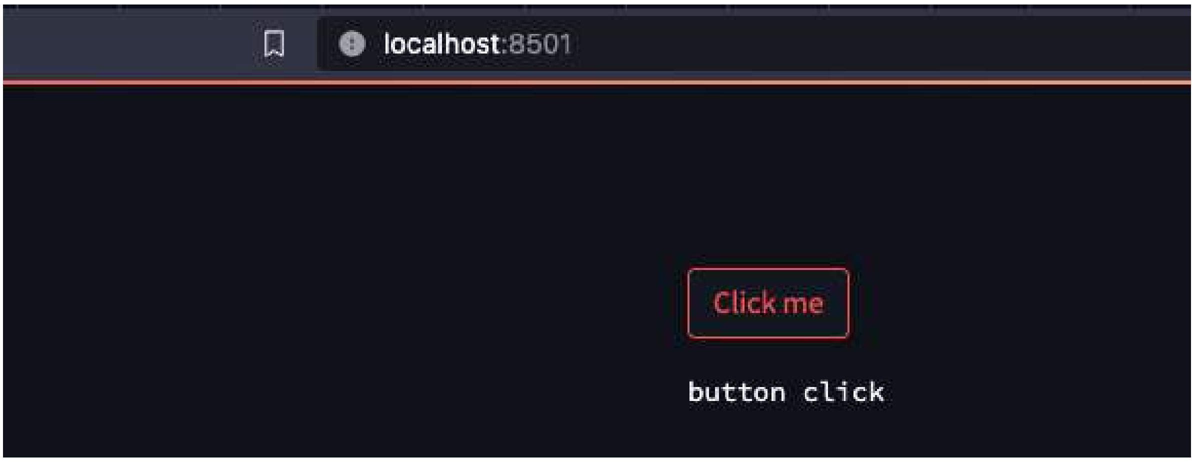
This can happen in two situations:

- Whenever you modify your app's source code.
- Whenever a user interacts with widgets in the app. For example, when dragging a slider, entering text in an input box, or clicking a button.

**Button**

Here we create a button which shows a text after it is clicked

```
# Create a button, that when clicked, shows a text
if(st.button("Click me")):
    st.text("button click")
```



```
button click
```

Now we’ll see how to interact with elements, make function calls and make elements work together.

- A very simple function that returns the square of the number entered.

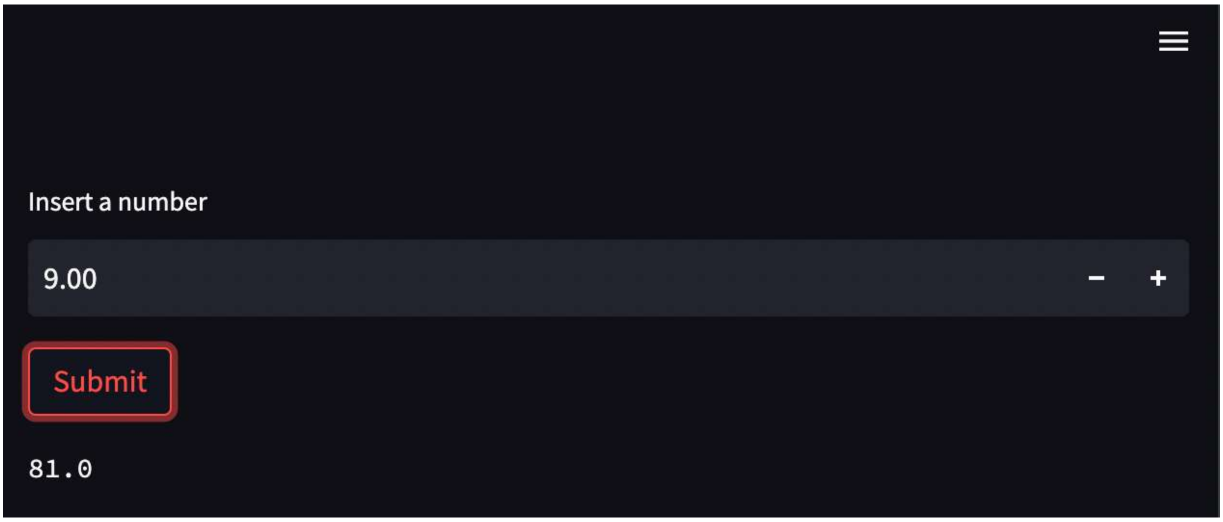
Create a number input widget and store the value in num.
Create a button ‘calculate square’. Which when pressed will pass the num to sqr function and create a text to show that result

```
def sqr(num):

     return num*num


num = st.number_input('Insert a number')

# display the name when the submit button is clicked
# .title() is used to get the input text string
if(st.button(Calculate Square)):
    result = sqr(num)
    st.text(result)
```



```
81.0
```

Streamlit cheat sheet:
https://dev.to/ramanbansal/the-ultimate-streamlit-cheatsheet-for-2023-1pln
Streamlit docs: https://docs.streamlit.io/library/cheatsheet