Politecnico di Torino

Applicazioni Web

Introduzione a Streamlit

Applicazioni Web

- Streamlit
- Elementi di testo
- Widget di input
- Visualizzazione dati
- Elementi aggiuntivi
- Layout

Streamlit

Applicazione web

Librerie Python

- Python offre diverse librerie per l'analisi, la manipolazione dei dati e lo sviluppo di interfacce per facilitare la creazione di applicazioni di analisi dati

pandas

Libreria utilizzata per lavorare con i dataset. Permette di analizzare, pulire, esplorare e manipolare i dati

NumPy

Libreria (Numeric Python) che permette di lavorare con dati numerici, con strutture dati multidimensionali (i.e., array, matrix)

Streamlit

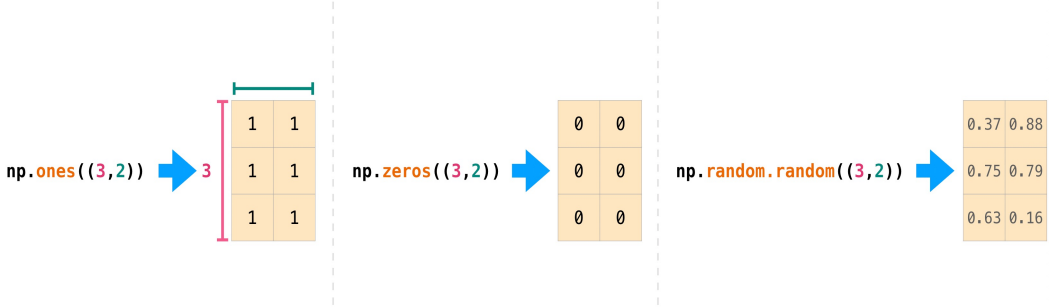
Libreria open-source che facilita la creazione e lo sviluppo di applicazioni web personalizzate

NumPy

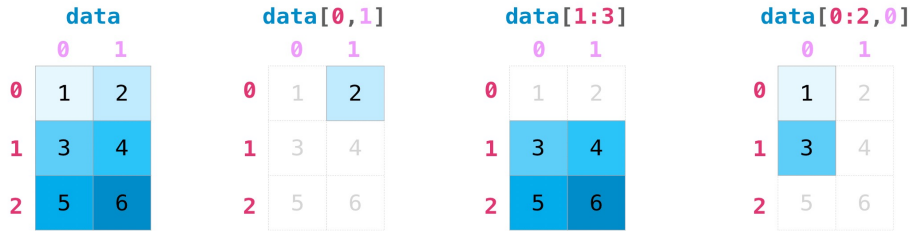
- I principali vantaggi di NumPy sono quelli di aumentare **flessibilità** ed **efficienza** delle operazioni rispetto alle strutture native di Python
  - `import numpy as np`
- La struttura dati ruota attorno al concetto di ***array***, una griglia di valori riferita come ***ndarray*** (N-dimensional array)
- Le dimensioni vengono chiamate ***axes***
- Numpy è la base di altre librerie Python avanzate (e.g., Pandas, Scikit-learn)

https://numpy.org/doc/stable/index.html

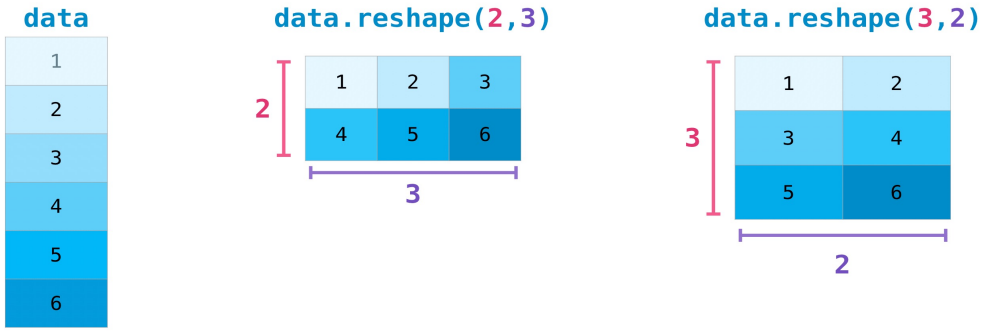
NumPy - Esempi



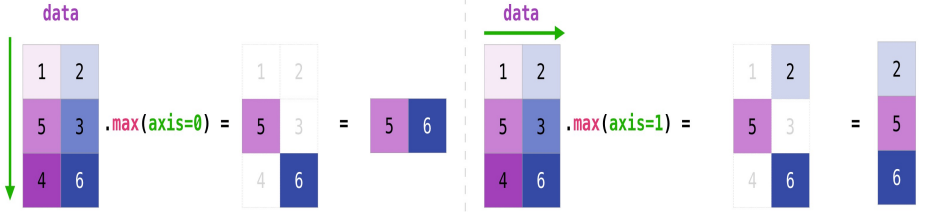
Creazione di una matrice



Indexing e slicing



Reshape di un array

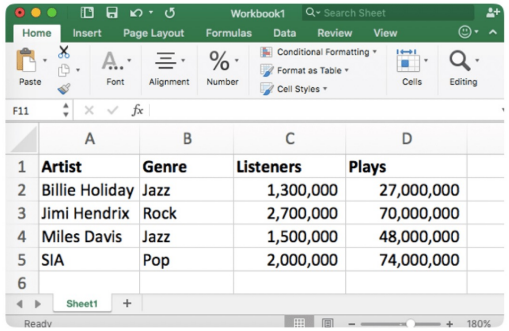


Aggregazioni sui diversi axes

Pandas

- Una libreria fondamentale soprattutto in ambito Data Science insieme a NumPy (su cui è basato)
  - `import numpy as np`
  - `import pandas as pd`
- Ci sono due data structure fondamentali: **Series** (sequenza 1-D di elementi omogenei) e **DataFrame** (array 2-D pensati come tabelle, ogni colonna è una Series e ha un nome)
- Esempio: esplorare, analizzare e visualizzare i dati da un file CSV

music.csv



	A	B	C	D
1	Artist	Genre	Listeners	Plays
2	Billie Holiday	Jazz	1,300,000	27,000,000
3	Jimi Hendrix	Rock	2,700,000	70,000,000
4	Miles Davis	Jazz	1,500,000	48,000,000
5	SIA	Pop	2,000,000	74,000,000
6				

`pandas.read_csv('music.csv')`

	Artist	Genre	Listeners	Plays
0	Billie Holiday	Jazz	1,300,000	27,000,000
1	Jimi Hendrix	Rock	2,700,000	70,000,000
2	Miles Davis	Jazz	1,500,000	48,000,000
3	SIA	Pop	2,000,000	74,000,000

https://pandas.pydata.org/docs/index.html

Pandas

- **Data Loading**: un DataFrame può essere creato partendo da Series, numpy array, dizionari, JSON, CSV…
- **Data Cleaning**: diverse funzioni per la pulizia dei dati, rimozione dei dati duplicati, sostituzione dei valori mancanti con valori di default, conversione dei tipi di dati…
- **Data Manipulation**: filtraggio delle righe, ordinamento dei dati, creazione di nuove colonne, aggregazione dei dati…
- **Data Analysis**: visualizzazione di statistiche descrittive, creazione di tabelle pivot, creazione di grafici e altre analisi avanzate…
- **Data Visualization**: visualizzazione dei dati con diverse tipologie di grafici, utilizzando la libreria Matplotlib

Pandas - Esempi

```python
import pandas as pd

# Creazione di un dataframe da un dizionario
data = {'Nome': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Mauro'],
        'Eta': [25, 33, 47, 19, 28, 17],
        'Citta': ['Roma', 'Milano', 'Napoli', 'Torino', 'Firenze', 'Torino']}
df = pd.DataFrame(data)

# Stampa le prime 5 righe del dataframe
print(df.head())
```

	Nome	Eta	Citta
0	Alice	25	Roma
1	Bob	33	Milano
2	Charlie	47	Napoli
3	David	19	Torino
4	Eva	28	Firenze

```python
#Stampa l'elenco di colonne
print(df.columns)

# Filtra le righe con età maggiore di 30 anni
df_filtrato = df[df['Eta'] > 30]
print(df_filtrato)
```

```
Index(['Nome', 'Eta', 'Citta', 'Maggiorenne'], dtype='object')
```

	Nome	Eta	Citta
1	Bob	33	Milano
2	Charlie	47	Napoli

```python
# Aggrega i dati per città e calcola la media dell'età
df_aggregato = df.groupby('Citta')['Eta'].mean()
print(df_aggregato)
```

```
Citta
Firenze    28.0
Milano     33.0
Napoli     47.0
Roma       25.0
Torino     18.0
Name: Eta, dtype: float64
```

```python
# Crea una nuova colonna che indica se la persona è maggiorenne o minorenne
df['Maggiorenne'] = df['Eta'].apply(lambda x: 'Sì' if x >= 18 else 'No')

# Elimina la colonna Eta
df_output=df.drop(columns=['Eta'])
print(df_output)

# Salva il dataframe su file CSV
df_output.to_csv('persone.csv', index=False)
```

	Nome	Citta	Maggiorenne
0	Alice	Roma	Sì
1	Bob	Milano	Sì
2	Charlie	Napoli	Sì
3	David	Torino	Sì
4	Eva	Firenze	Sì
5	Mauro	Torino	No

persone.csv

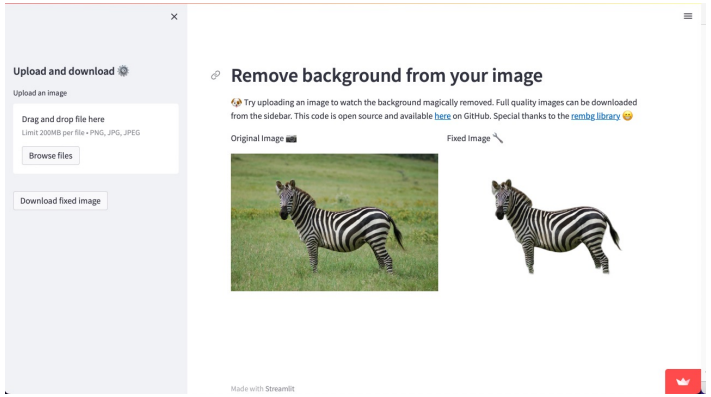
Perché Streamlit?

Streamlit

- Libreria open-source di Python che facilita la creazione e lo sviluppo di applicazioni web personalizzate
- Ideale per supportare progetti di **data science** e machine learning
- Si possono creare interfacce interattive
- Pensato per i neofiti, non sono espressamente richieste competenze di front-end
- Grazie a *widget* ed elementi a disposizione, si possono creare pagine web con poche righe di codice
- Compatibile con la maggior parte di librerie Python

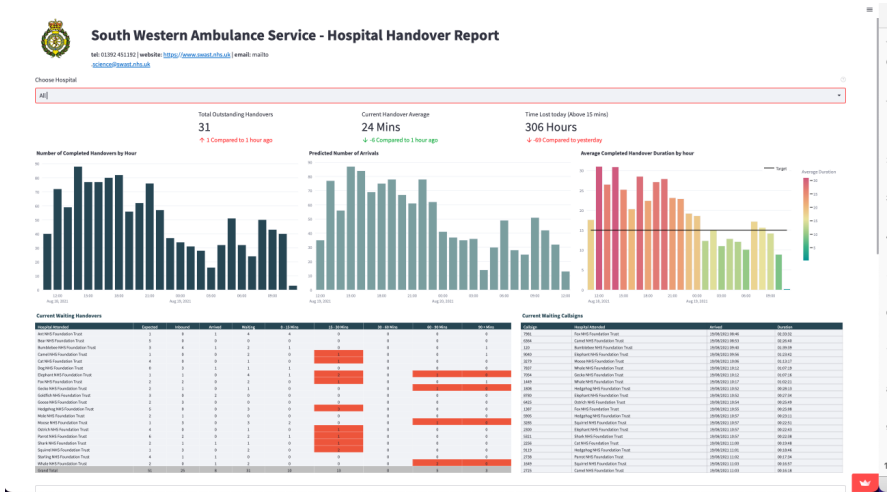
Galleria d'esempi

- Ci sono a disposizione diversi *template* e applicazioni create dalla community (https://streamlit.io/gallery)



Background Removal

Bundesliga analyzer



SWAST - Hospital Handover Report

Installazione

- Python 3.7 – Python 3.11
- Utilizzare un ***virtual environment*** è sempre consigliato (*pipenv, poetry, venv…*)
- Installare streamlit
  - `pip install streamlit`
- Testare l'installazione
  - `streamlit hello`
- Lanciare la propria applicazione
  - `streamlit run your_script.py [-- script args]`

  Oppure

  - `python -m streamlit run your_script.py`

Configurazione

- Diverse possibilità per definire le opzioni di configurazione (e.g. porta server, tema…) mediante:
  1. un ***global config file*** (da creare):
     - *~/.streamlit/config.toml* per macOS/Linux
     - *%userprofile%/.streamlit/config.toml* per Windows
  2. un file di configurazione ***per-project***:
     - *$CWD/.streamlit/config.toml* dove *$CWD* è la cartella da cui Streamlit è stato avviato
  3. ***flag*** da linea di comando:
     - `streamlit run your_script.py --server.port 80`

Telemetria

- Vengono raccolte informazioni statistiche sull'utilizzo da parte degli utenti
- Per disattivare la telemetria, è necessario specificare l'opzione di configurazione

```
[browser]
gatherUsageStats = false
```

Il server deve essere riavviato per aggiornare le opzioni di configurazione

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

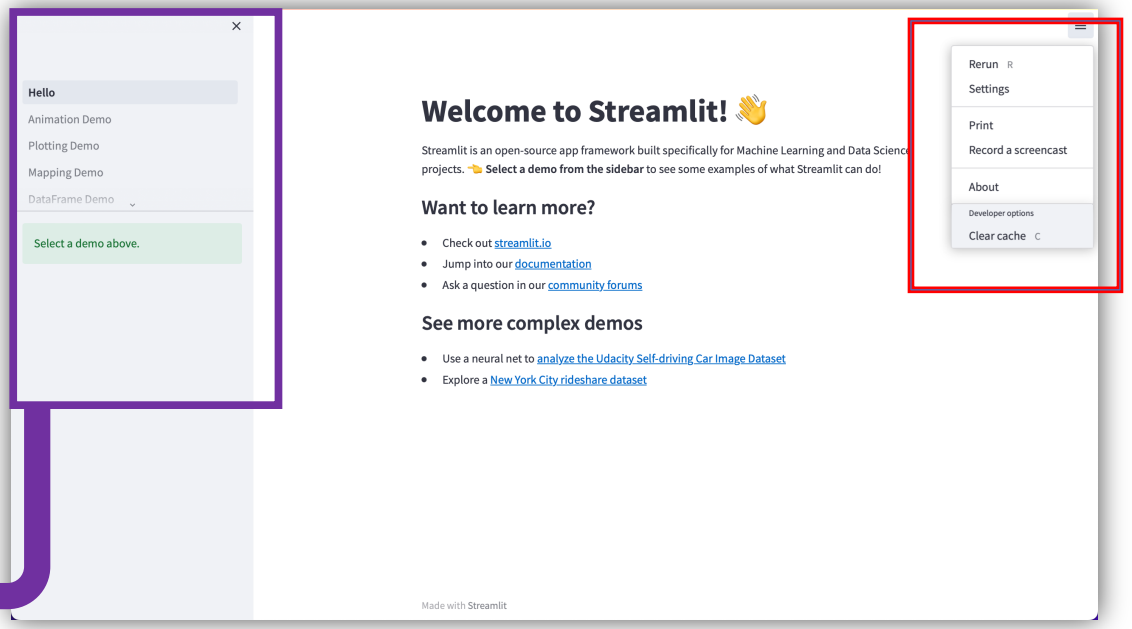
Avvio

```
(streamlitTutorial) (base) ➜  streamlitTutorial streamlit hello

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

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.89:8501

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

  May you create awesome apps!
```

Comando per avviare Streamlit

URL per raggiungere il web server alla porta 8501

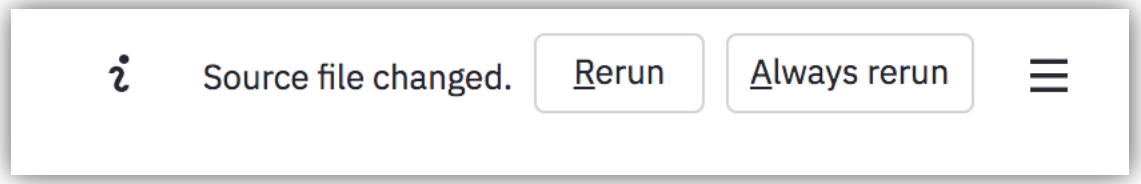
Hamburger menu



Sidebar con accesso a demo di esempio

Sviluppo

- Ogni volta che lo script Python viene salvato, l'applicazione si aggiorna con un click su ***Rerun***, senza bisogno di riavviare il server
- Scegliendo ***Always rerun***, l'applicazione si aggiorna in automatico a ogni salvataggio, consentendo di vedere immediatamente i cambiamenti
- Ogni volta che qualcosa deve essere aggiornato a schermo (incluse le interazioni dell'utente), Streamlit lancia interamente lo script *top-to-bottom*
- Il server si può fermare con
  `Ctrl+C`



Struttura del progetto

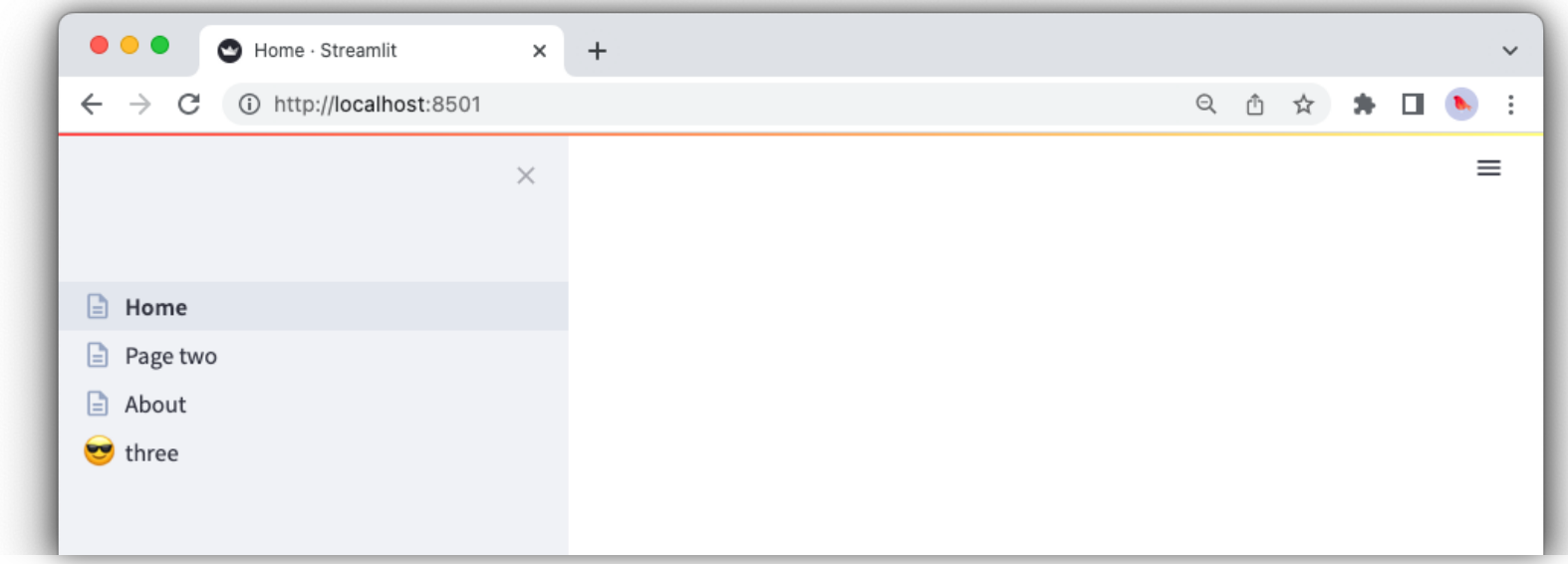
- Prima di sviluppare l'app, è importante definire la struttura della *directory* del progetto
- Bisogna definire un ***entrypoint file*** che rappresenta la pagina principale da mostrare all'utente
- Le altre pagine aggiuntive vanno inserite in una sotto-cartella ***pages***
- Le pagine condividono globalmente gli stessi moduli Python

```
Home.py # This is the file you run with "streamlit run"
└── pages/
    └── About.py # This is a page
    └── 2_Page_two.py # This is another page
    └── 3_😎_three.py # So is this
```

```python
#Home.py
import streamlit as st
```

Pagine dell'applicazione

- Le pagine sono definite dai file *.py* all'interno della cartella *"pages/"*
- I nomi dei file vengono trasformati nei nomi delle pagine
- L'ordine è dato dal numero che precede il titolo e/o dall'ordine alfabetico del titolo stesso
- Il numero usato come prefisso nel nome del file non viene interpretato come parte del titolo



Configurazione della pagina

- Impostare la configurazione di default della pagina
  - `st.set_page_config(page_title=None, page_icon=None, layout="centered", initial_sidebar_state="auto", menu_items=None)`

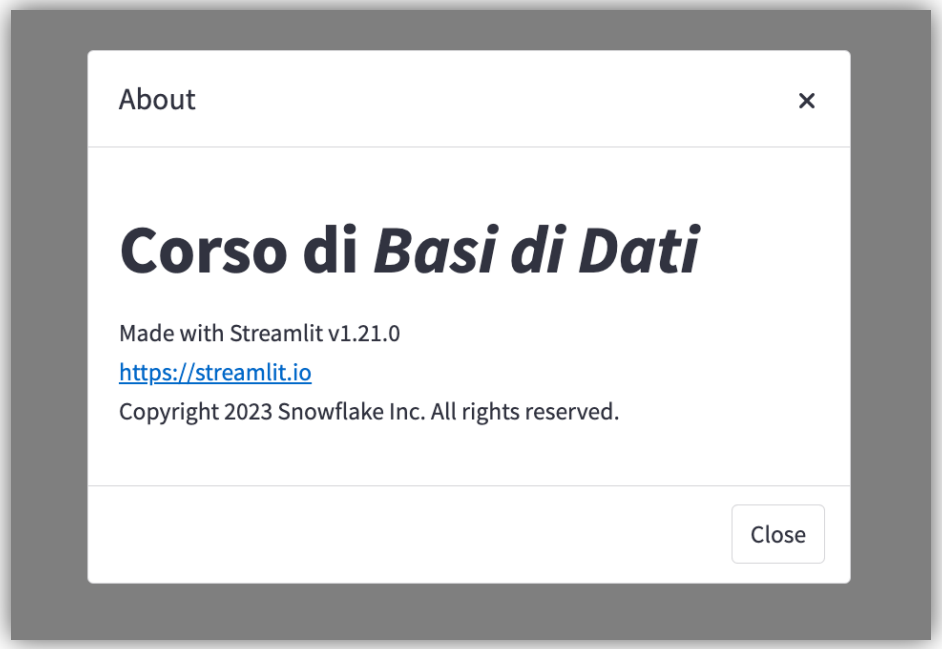
```python
import streamlit as st

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded"
)
```

Deve essere il primo comando Streamlit e impostato una volta soltanto!

Personalizzazione dell'hamburger menu

- Utilizzando il parametro *menu_items* è possibile personalizzare gli elementi da mostrare nell'hamburger menu
- Va formattato secondo un dizionario in cui la chiave è l'elemento che si vuole modificare

```python
import streamlit as st

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded",
    menu_items={
        'Get Help': 'https://dbdmg.polito.it/',
        'Report a bug': "https://dbdmg.polito.it/",
        'About': "# Corso di *Basi di Dati*"
    }
)
```



Elementi di Streamlit

- Widget ed elementi specifici per i diversi tipi di attività e di input
  - integrazione rapida di diverse funzionalità nella propria applicazione
  - consultabili attraverso la documentazione ufficiale: https://docs.streamlit.io/library/api-reference
- Categorie più significative:
  - Elementi di testo
  - Widget di input
  - Layout
  - Visualizzazione di dati e grafici
  - Elementi aggiuntivi

Argomenti degli elementi

- I vari elementi sono integrabili senza particolari configurazioni
  - personalizzazione mediante determinati argomenti
- Alcuni argomenti sono comuni a tutti (o a gran parte) degli elementi:
  - ***label***: descrive all'utente la funzionalità dell'elemento (e.g. il nome di un tasto cliccabile)
  - ***label_visibility*****:** determina la visibilità della label (i.e. "visible", "hidden", "collapsed"); la label dovrebbe essere sempre definita
  - ***disabled***: flag booleano per disattivare un elemento. Utile per rendere disponibile un widget solo se si verifica una determinata condizione
  - ***use_container_width***: flag booleano per adattare le dimensioni del widget a quelle del container di cui fa parte
  - ***key:*** stringa o numero per identificare univocamente il widget. Se omesso, viene generato in base al contenuto

Diversi elementi non possono avere la stessa key!

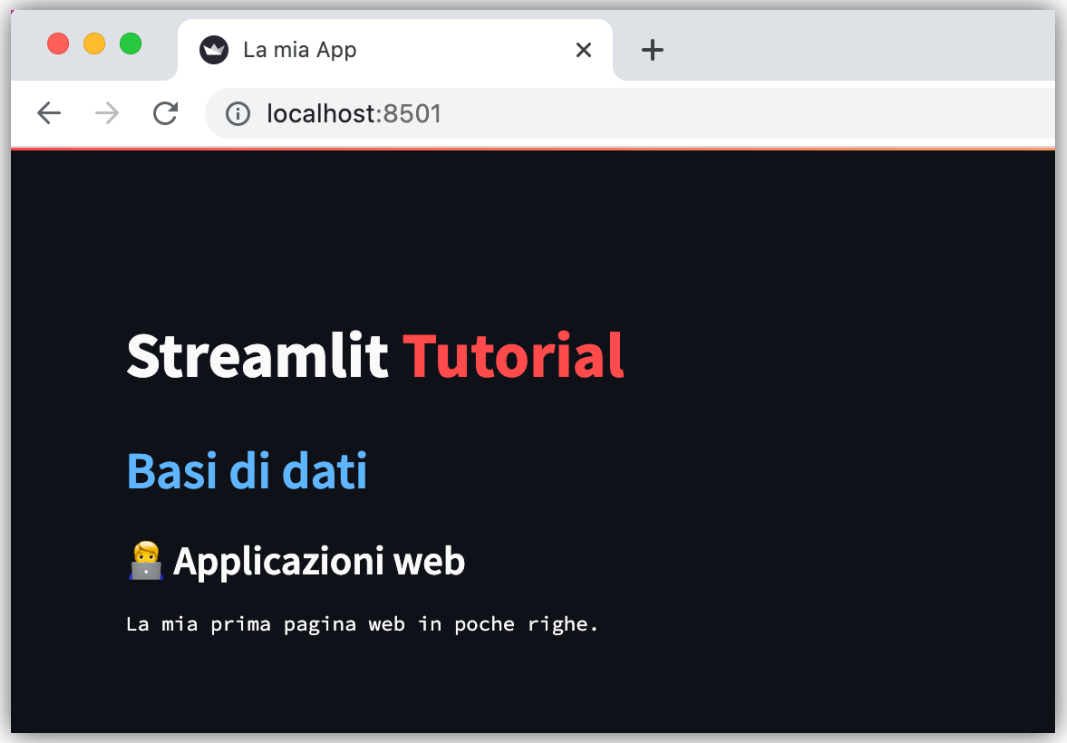
Elementi di testo

Applicazioni Web

Elementi di testo

- Diversi elementi testuali pronti all'uso, con la possibilità di personalizzare il colore e inserire *emoji*:
  - Titolo
  - Intestazione
  - Sotto-intestazione
  - Testo

```python
import streamlit as st

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded",

)
st.title("Streamlit :red[Tutorial]")
st.header(":blue[Basi di dati]")
st.subheader("👨‍💻 Applicazioni web")
st.text("La mia prima pagina web in poche righe.")
```



Markdown

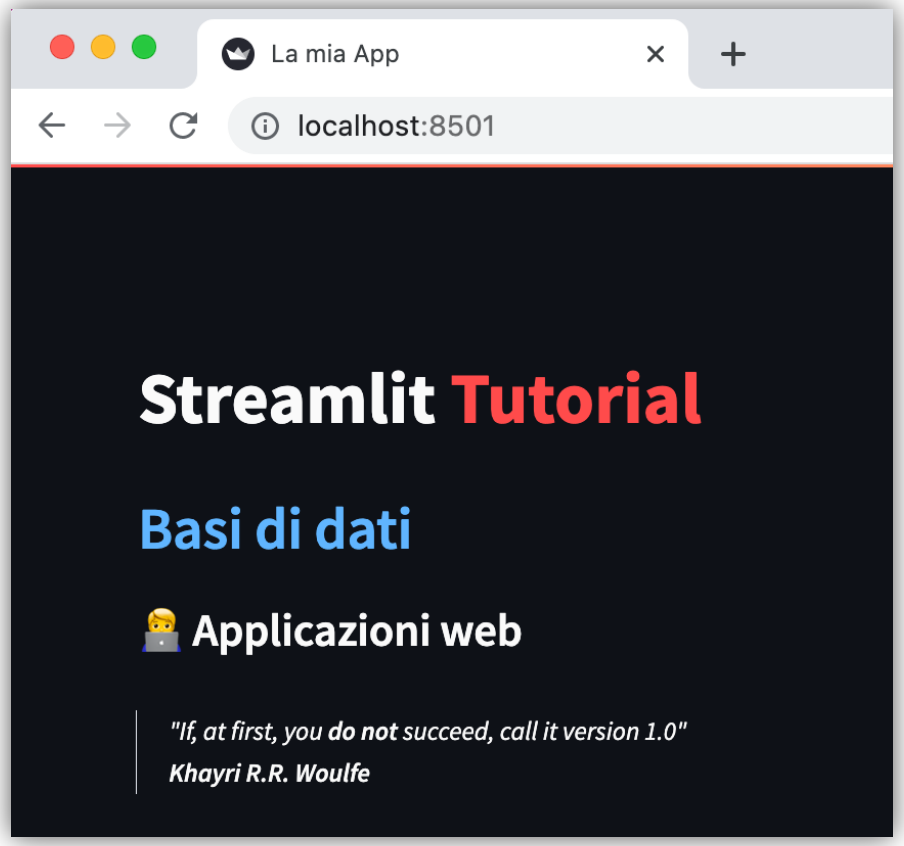
- E' possibile inserire stringhe formattate secondo il linguaggio ***markdown***
- Il markdown viene utilizzato per formattare testi in modo semplice e rapido, essendo più leggibile rispetto ad altri linguaggi markup
- La sintassi più comune *(N.B. gli spazi a volte sono necessari!):*

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

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

Esempio di Markdown

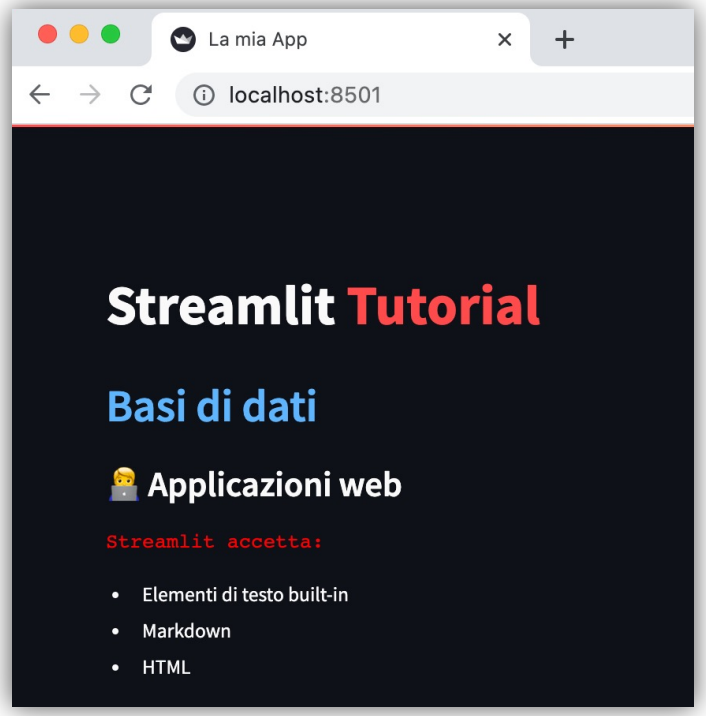
```python
import streamlit as st

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded",

)
st.markdown("# Streamlit :red[Tutorial]")
st.markdown("## :blue[Basi di dati]")
st.markdown("### 👷 Applicazioni web")
st.markdown("""> _"If, at first, you **do not** succeed, call it version 1.0"
        **Khayri R.R. Woulfe**_""")
```



Markdown e HTML

- E' possibile anche utilizzare il markdown per inserire codice HTML
- Utile per particolari personalizzazioni
- E' necessario abilitare l'uso di codice HTML
  - la funzione è disattivata di default per evitare l'inserimento di codice non sicuro da parte dello sviluppatore

```python
import streamlit as st

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded",

)

st.markdown("# Streamlit :red[Tutorial]")
st.markdown("## :blue[Basi di dati]")
st.markdown("### 👷 Applicazioni web")
html_string="""<p style="font-family:courier;color:red">Streamlit accetta:</p>
        <ul>
            <li> Elementi di testo built-in </li>
            <li> Markdown </li>
            <li> HTML </li>
        </ul>"""
st.markdown(html_string,unsafe_allow_html=True)
```



Write

- Permette di scrivere nell'app gli argomenti che gli vengono passati
  - `st.write(*args, unsafe_allow_html=False, **kwargs)`
- Widget universale e flessibile che ha un comportamento diverso in base all'argomento passato
  - accetta diversi tipi di argomenti e li renderizza di conseguenza
  - possono essere passati più argomenti che verranno rappresentati
  - permette di rappresentare diversi oggetti Python (e.g. figure, dataframe, dizionari, errori, funzioni e moduli) anche in modalità interattiva

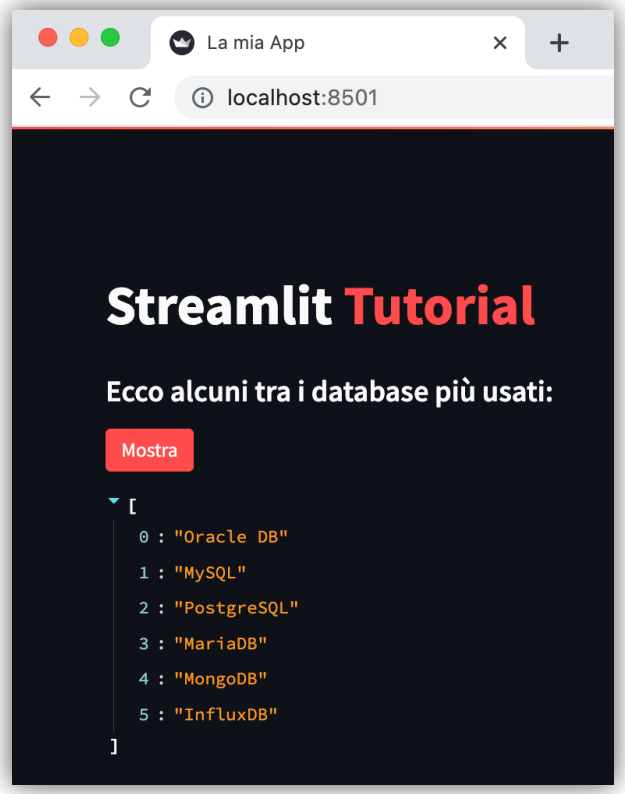
Widget di input

Applicazioni Web

Button

- Permette di mostrare un semplice pulsante che può essere cliccato dall'utente
  - `st.button(label, key=None, help=None, on_click=None, args=None, kwargs=None, type="secondary", disabled=False, use_container_width=False)`

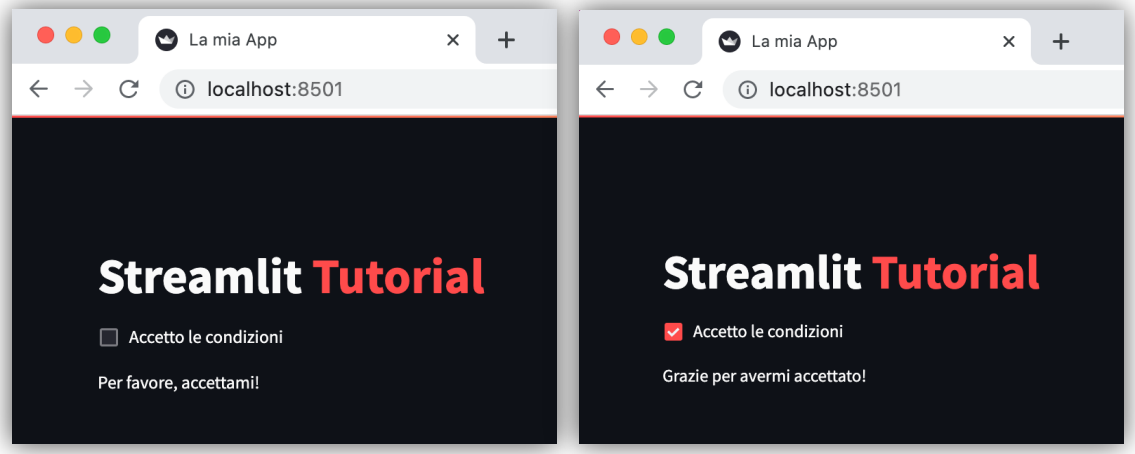
```python
import streamlit as st

st.markdown("# Streamlit :red[Tutorial]")
st.markdown("#### Ecco alcuni tra i database più usati:")

db_list=["Oracle DB","MySQL","PostgreSQL","MariaDB","MongoDB","InfluxDB"]

if st.button("Mostra",type="primary"):
    st.write(db_list)
```



Checkbox

- Permette di mostrare un ***checkbox*** da spuntare ed eseguire un'azione di conseguenza
  - `st.checkbox(label, value=False, key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False, label_visibility="visible")`
- Restituisce True o False in base allo stato del checkbox

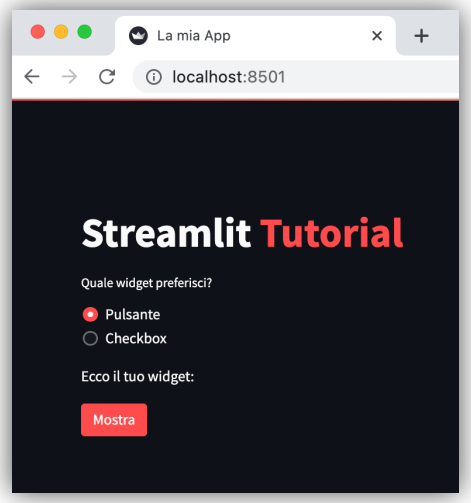
```python
import streamlit as st

st.markdown("# Streamlit :red[Tutorial]")
if st.checkbox("Accetto le condizioni"):
    st.write("Grazie per avermi accettato!")
else:
    st.write("Per favore, accettami!")
```

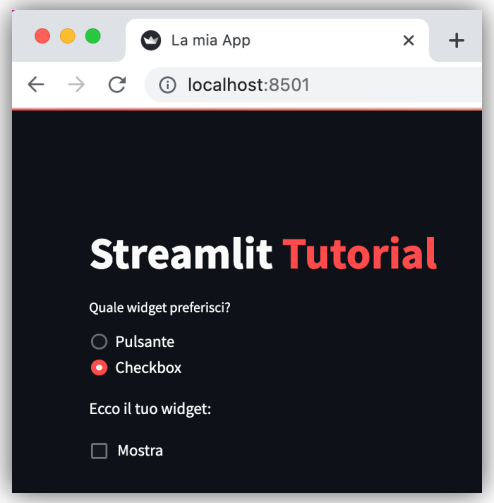


Radio Button

- Permette di inserire un ***radio button*** con cui l'utente può effettuare una scelta esclusiva tra le alternative proposte
  - `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")`
- Restituisce l'opzione scelta

```python
import streamlit as st

st.markdown("# Streamlit :red[Tutorial]")
widget=st.radio("Quale widget preferisci?",["Pulsante","Checkbox"])

st.write("Ecco il tuo widget:")
if widget=="Pulsante":
    if st.button("Mostra",type="primary"):
        #Esegui l'operazione che desideri
        pass
else:
    if st.checkbox("Mostra"):
        #Esegui l'operazione che desideri
        pass
```

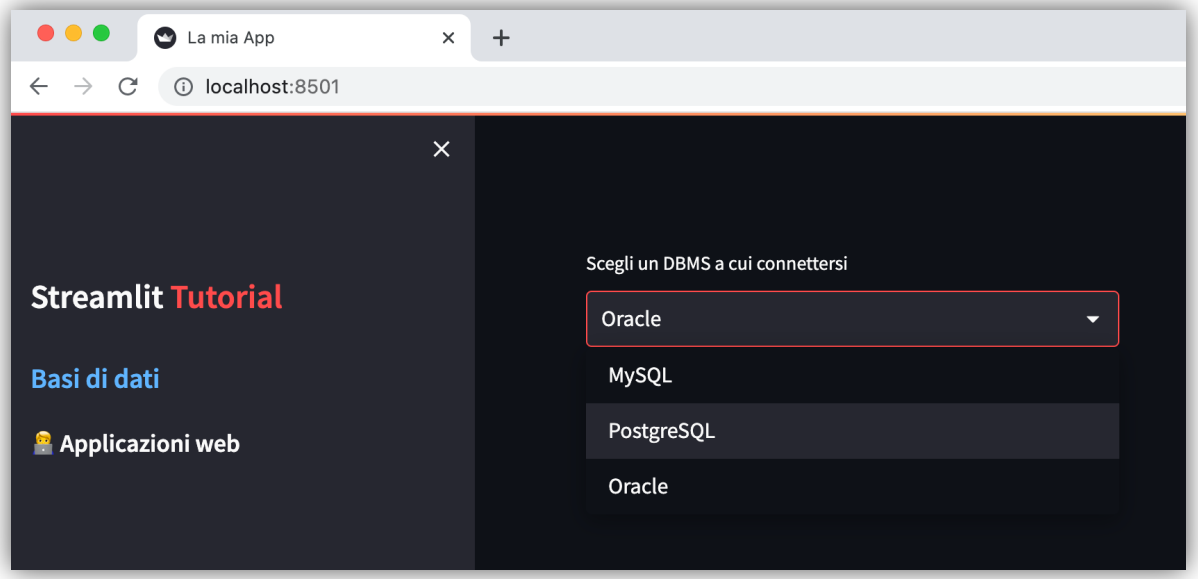




Select Box

- Permette di inserire un ***box di selezione*** a tendina con cui l'utente può scegliere tra le varie alternative
  - `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")`
- Restituisce l'opzione scelta

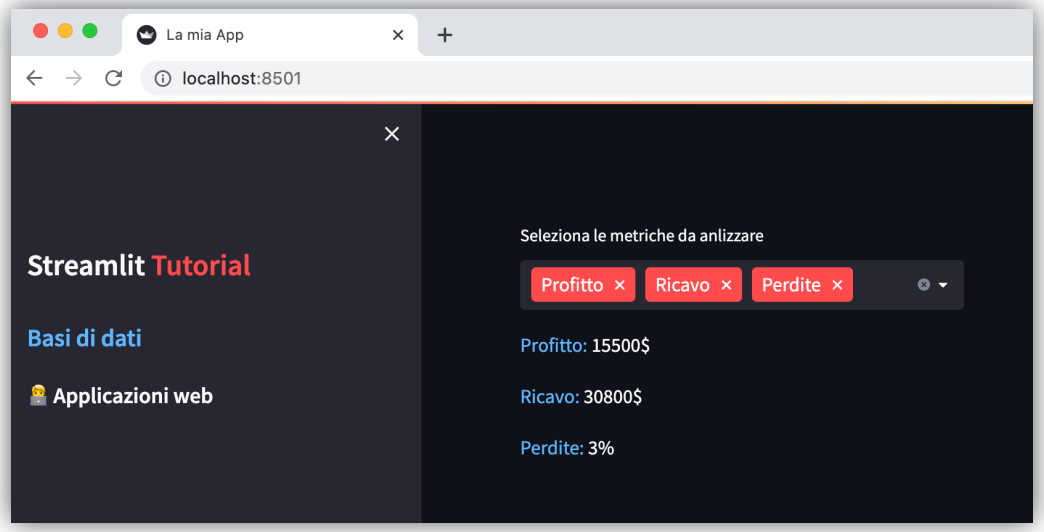
```python
import streamlit as st

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👨‍💻 Applicazioni web")

option = st.selectbox(
    'Scegli un DBMS a cui connetterti',
    ('MySQL', 'PostgreSQL', 'Oracle'))
```



Multiselect

- Permette all'utente di scegliere alternative multiple tra quelle proposte
  - `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)`
- Il parametro *default* specifica la liste di opzioni selezionate all'avvio
- Il parametro *max_selections* definisce il numero massimo di opzioni selezionabili
- Restituisce la lista delle opzioni selezionate

```python
import streamlit as st

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

mydict={"Profitto":"15500$","Ricavo":"30800$","Perdite":"3%"}

options = st.multiselect(
    'Seleziona le metriche da anlizzare',
    ['Profitto', 'Ricavo', 'Perdite'],['Profitto'])

for option in options:
    st.write(f':blue[{option}:] {mydict[option]}')
```



Slider

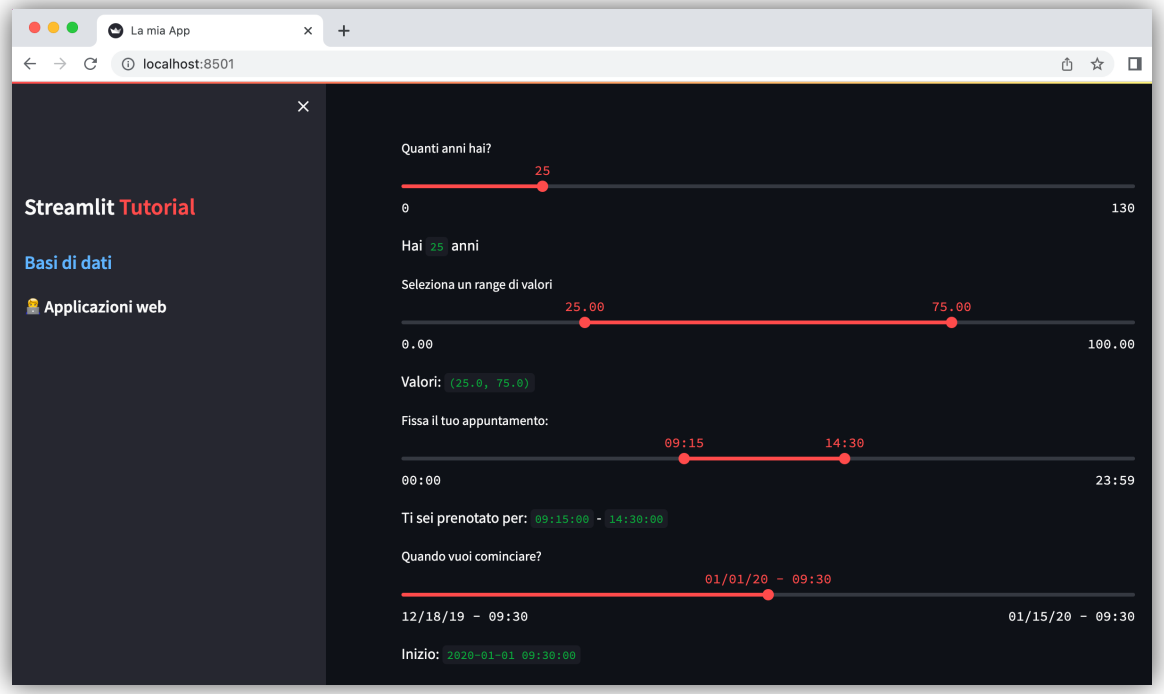
- Offre uno ***slider*** che accetta: int, float, time, date e datetime
  - `st.slider(label, min_value=None, max_value=None, value=None, step=None, format=None, key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False, label_visibility="visible")`
- Permette di selezionare sia un valore singolo che un range di valori
- Restituisce il valore selezionato o la tupla (per i range)
- I parametri *min_value* (*default* 0 se int, 0.0 se float) e *max_value* (*default* 100 se int, 1.0 se float) definiscono rispettivamente il minimo e massimo valore ammesso

Slider

- Offre uno ***slider*** che accetta: int, float, time, date e datetime
  - `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")`
- Il parametro *value* definisce il valore assunto quanto lo slider è caricato per la prima volta
  - se impostato con una tupla crea uno slider con il range selezionabile
  - di default è impostato su *min_value*
- *Il parametro step* definisce l'intervallo tra un valore e l'altro (*default* 1 se int, 0.01 se float)

Esempio Slider

```python
import streamlit as st
from datetime import datetime,time

#slider semplice
age = st.slider('Quanti anni hai?', 0, 130, 25)
st.write("Hai", age, 'anni')

#slider con range
values = st.slider(
    'Seleziona un range di valori',
    0.0, 100.0, (25.0, 75.0))
st.write('Valori:', values)

#range time slider
appointment = st.slider(
    "Fissa il tuo appuntamento:",
    value=(time(11, 30), time(12, 45)))
st.write(f"Ti sei prenotato per:", appointment[0],'-',appointment[1])

#datetime slider
start_time = st.slider(
    "Quando vuoi cominciare?",
    value=datetime(2020, 1, 1, 9, 30),
    format="MM/DD/YY - hh:mm")
st.write("Inizio:", start_time)
```



Text e Number

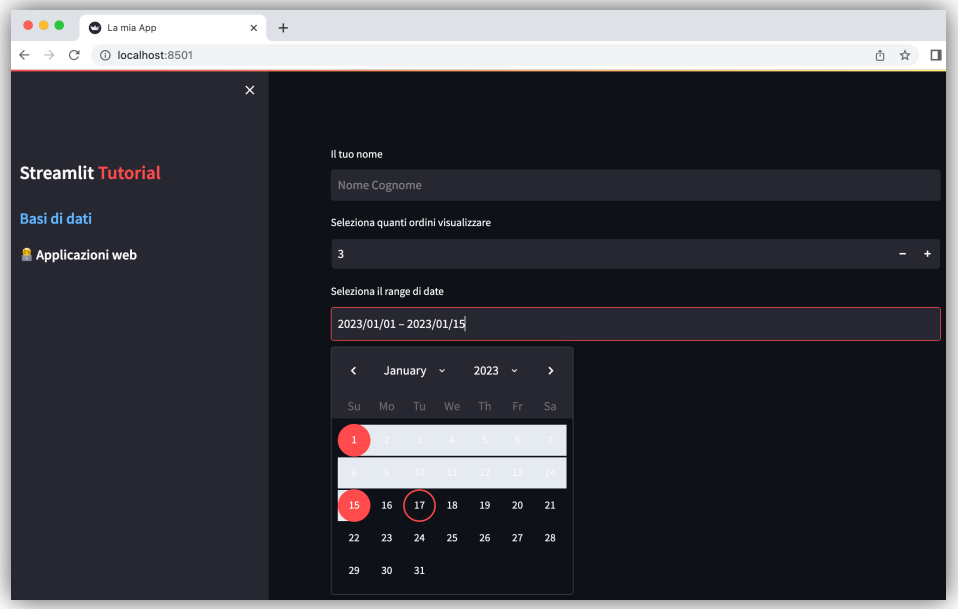
- Il ***text input*** offre la possibilità di un input testuale single-line
  - `st.text_input(label, value="", max_chars=None, key=None, type="default", help=None, autocomplete=None, on_change=None, args=None, kwargs=None, *, placeholder=None, disabled=False, label_visibility="visible")`

- Il ***number input*** permette di passare un numero che può essere scritto da tastiera o usando i tasti '+' e '-'
  - `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

- Offre un widget ideale per la selezione di una ***data*** su calendario
  - `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")`
- Il parametro *value* accetta anche lista/tupla per abilitare un range di date

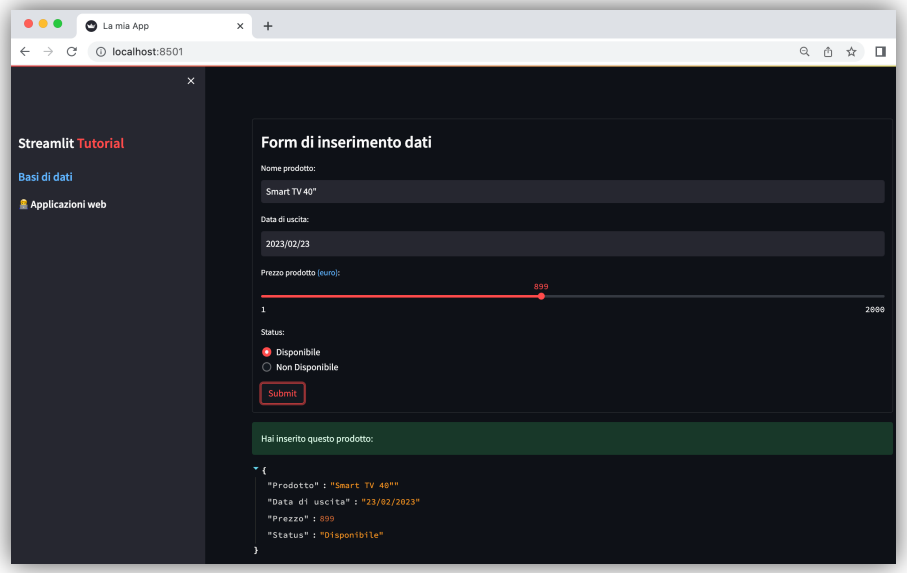
```python
import streamlit as st
import datetime

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👨‍💻 Applicazioni web")

name = st.text_input('Il tuo nome', placeholder="Nome Cognome")
number = st.number_input('Seleziona quanti ordini visualizzare',min_value=1,value=3)
date = st.date_input(
    "Seleziona il range di date",
    value=(datetime.date(2023, 1, 1),datetime.date(2023, 1, 15)))
```



Form

- Permette di raggruppare diversi elementi in un ***form*** (container)
  - `st.form(key, clear_on_submit=False)`
- Ha integrato un pulsante *Submit* che raccoglie tutti i valori acquisiti dai diversi widget
- Il parametro *clear_on_submit* se True resetta i valori dei widget dopo il click dell'utente sul pulsante Submit

```python
import streamlit as st
import datetime

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

with st.form("form"):
   st.subheader("Form di inserimento dati")
   product=st.text_input("Nome prodotto:")
   data=st.date_input("Data di uscita:",value=datetime.datetime(2023,1,1))

   price = st.slider("Prezzo prodotto :blue[(euro)]:",1,2000)
   status = st.radio("Status:",("Disponibile","Non Disponibile"))

   # Every form must have a submit button.
   submitted = st.form_submit_button("Submit")

if submitted:
    st.success("Hai inserito questo prodotto:")
    st.write({"Prodotto": product, "Data di uscita": data.strftime('%d/%m/%Y'), "Prezzo": price, "Status":status})
```



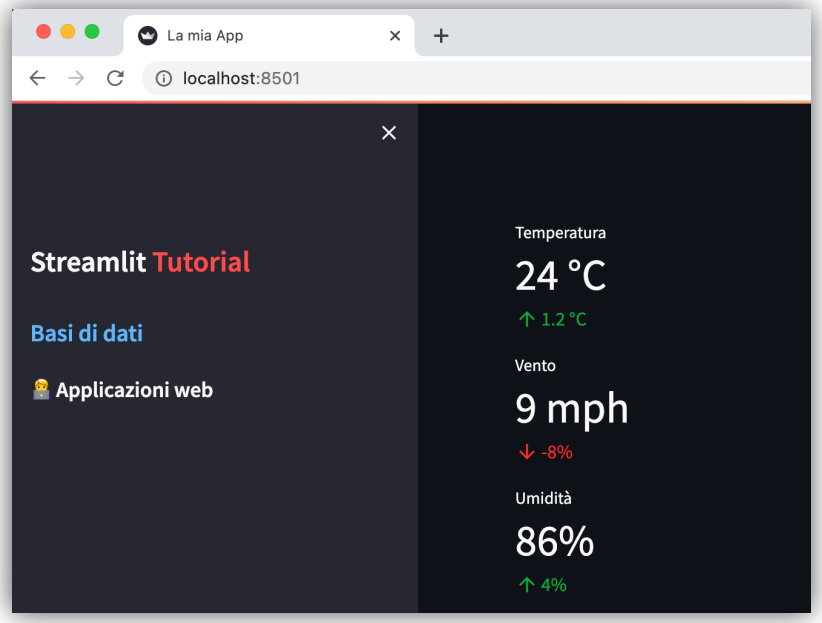
Visualizzazione dati

Applicazioni Web

Metrics

- Visualizza una ***metrica*** con un font specifico, offrendo la possibilità di aggiungere un indicatore relativo alla variazione
  - `st.metric(label, value, delta=None, delta_color="normal", help=None, label_visibility="visible")`
- Il parametro *delta* indica la variazione

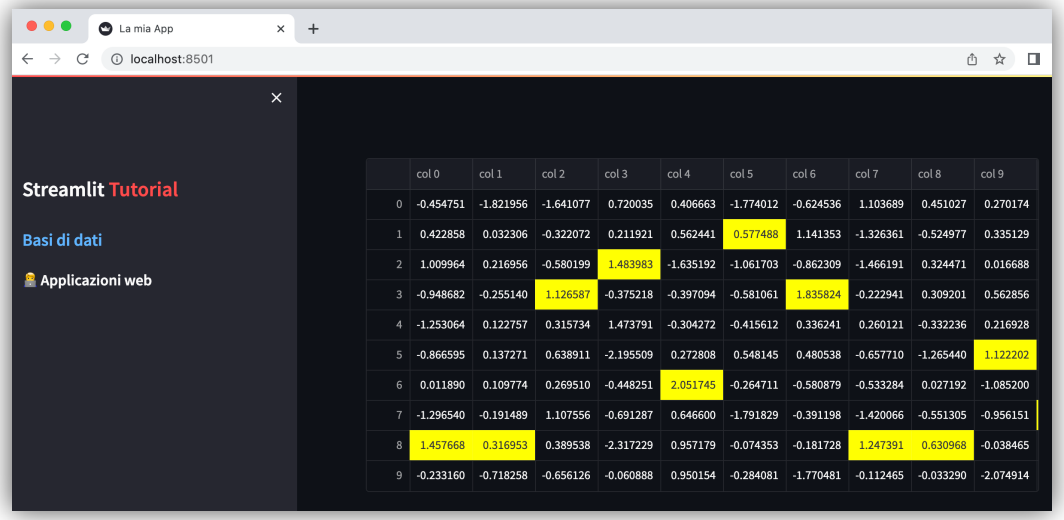
```python
import streamlit as st
import datetime

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👨‍💻 Applicazioni web")

st.metric("Temperatura", "24 °C", "1.2 °C")
st.metric("Vento", "9 mph", "-8%")
st.metric("Umidità", "86%", "4%")
```



Dataframe

- Permette di visualizzare i ***dataframe*** pandas sotto forma di tabelle interattive
  - `st.dataframe(data=None, width=None, height=None, *, use_container_width=False)`

```python
import streamlit as st
import numpy as np
import pandas as pd

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

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))
```



	col 0	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8	col 9
0	-0.454751	-1.821956	-1.641077	0.720035	0.406663	-1.774012	-0.624536	1.103689	0.451027	0.270174
1	0.422858	0.032306	-0.322072	0.211921	0.562441	0.577488	1.141353	-1.326361	-0.524977	0.335129
2	1.009964	0.216956	-0.580199	1.483983	-1.635192	-1.061703	-0.862309	-1.466191	0.324471	0.016688
3	-0.948682	-0.255140	1.126587	-0.375218	-0.397094	-0.581061	1.835824	-0.222941	0.309201	0.562856
4	-1.253064	0.122757	0.315734	1.473791	-0.304272	-0.415612	0.336241	0.260121	-0.332236	0.216928
5	-0.866595	0.137271	0.638911	-2.195509	0.272808	0.548145	0.480538	-0.657710	-1.265440	1.122202
6	0.011890	0.109774	0.269510	-0.448251	2.051745	-0.264711	-0.580879	-0.533284	0.027192	-1.085200
7	-1.296540	-0.191489	1.107556	-0.691287	0.646600	-1.791829	-0.391198	-1.420066	-0.551305	-0.956151
8	1.457668	0.316953	0.389538	-2.317229	0.957179	-0.074353	-0.181728	1.247391	0.630968	-0.038465
9	-0.233160	-0.718258	-0.656126	-0.060888	0.950154	-0.284081	-1.770481	-0.112465	-0.033290	-2.074914

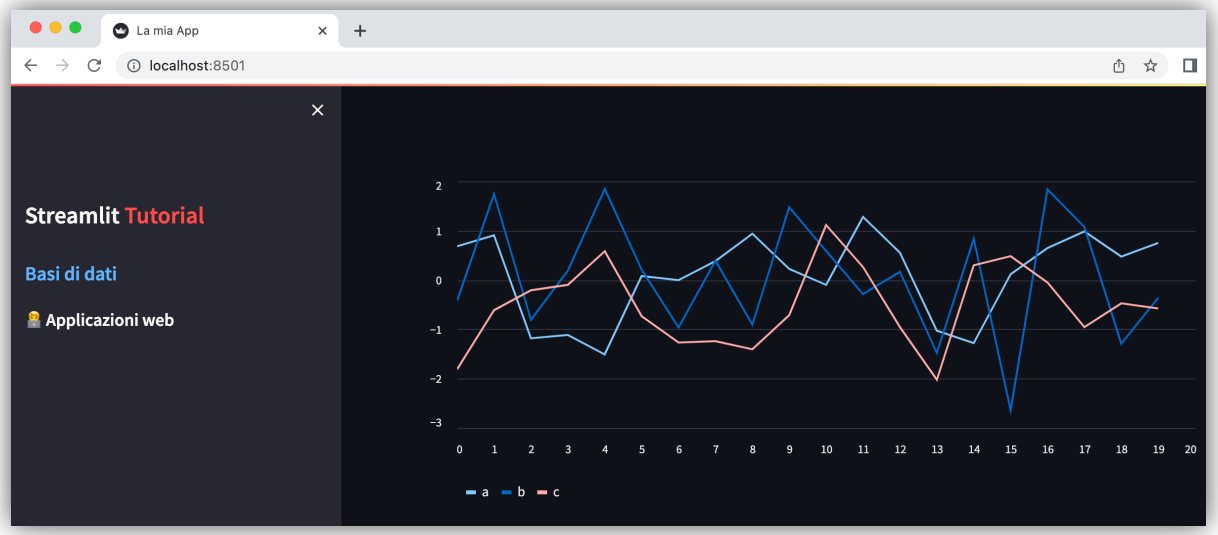
Grafici

- Sono supportate diverse librerie per la rappresentazione grafica dei dati attraverso dei ***chart interattivi***
  - Matplotlib
  - Plotly
  - Altair
  - deck.gl (mappe e grafici 3D)
- Per velocizzare l'integrazione dei grafici più comuni, alcuni sono integrati nativamente in Streamlit (con meno personalizzazioni):
  - Line chart
  - Area Chart
  - Bar Chart
  - Scatterplot on map

Line Chart

- Permette di rappresentare un **line chart** ed è basato su Altair
  - `st.line_chart(data=None, *, x=None, y=None, width=0, height=0, use_container_width=True)`
- Ideale per plot semplici da includere rapidamente e con facilità
- I parametri *x* e *y* specificano il nome delle colonne da usare sugli assi
- I parametri *width* e *height* specificano le dimensioni in pixel

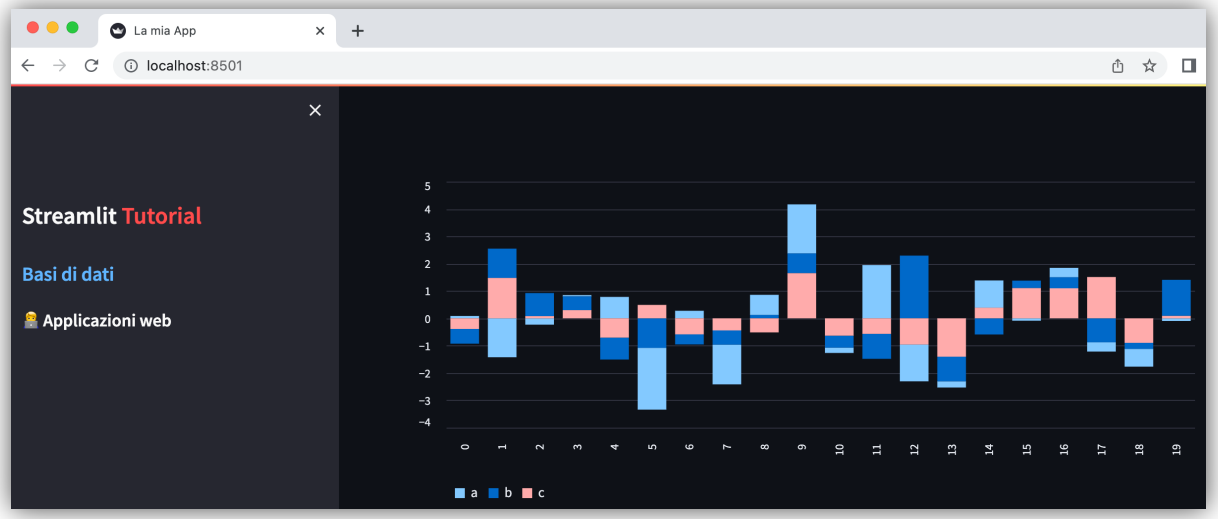
```python
import streamlit as st
import numpy as np
import pandas as pd

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

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

st.line_chart(chart_data)
```



Bar Chart

- Permette di rappresentare un **bar chart** ed è basato su Altair
  - `st.bar_chart(data=None, *, x=None, y=None, width=0, height=0, use_container_width=True)`
- Ideale per plot semplici da includere rapidamente e con facilità
- I parametri *x* e *y* specificano il nome delle colonne da usare sugli assi
- I parametri *width* e *height* specificano le dimensioni in pixel

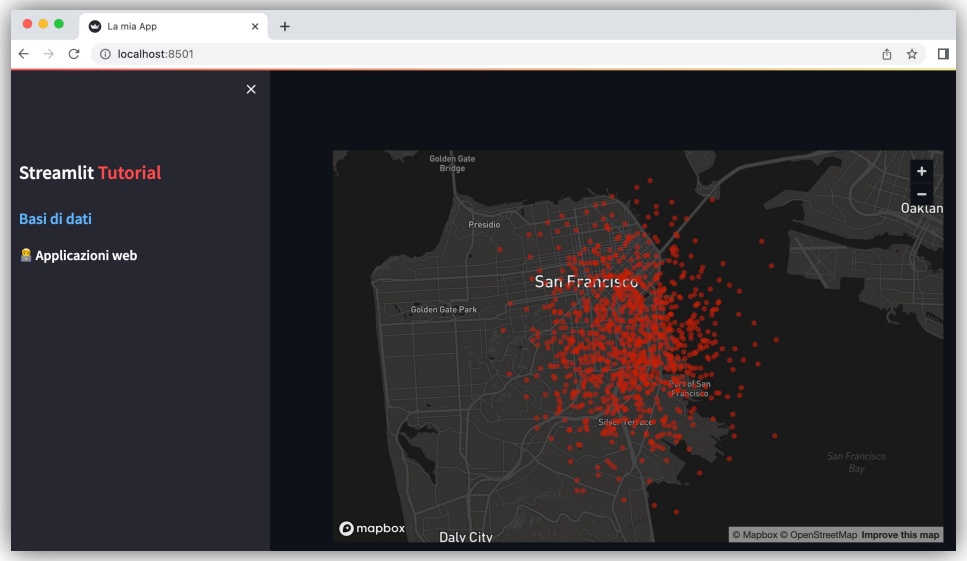
```python
import streamlit as st
import numpy as np
import pandas as pd

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

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

st.bar_chart(chart_data)
```



Map

- Permette di visualizzare ***punti su mappa*** ed è basato su Pydeck
  - `st.map(data=None, zoom=None, use_container_width=True)`
- Il parametro *data* deve avere <u>obbligatoriamente</u> due colonne:
  1. Latitudine chiamata *'lat', 'latitude', 'LAT', 'LATITUDE'*
  2. Longitude chiamata *'lon', 'longitude', 'LON', 'LONGITUDE'*
- La mappa si appoggia sul servizio esterno Mapbox e necessita di un token (al momento è offerto automaticamente da Streamlit)

```python
import streamlit as st
import numpy as np
import pandas as pd

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

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

st.map(df)
```



Elementi aggiuntivi

Applicazioni Web

Elementi aggiuntivi

- Session state
- Elementi per personalizzare l'applicazione
- Ad esempio:
  - messaggi di stato
  - progress bar
  - spinner

Session State

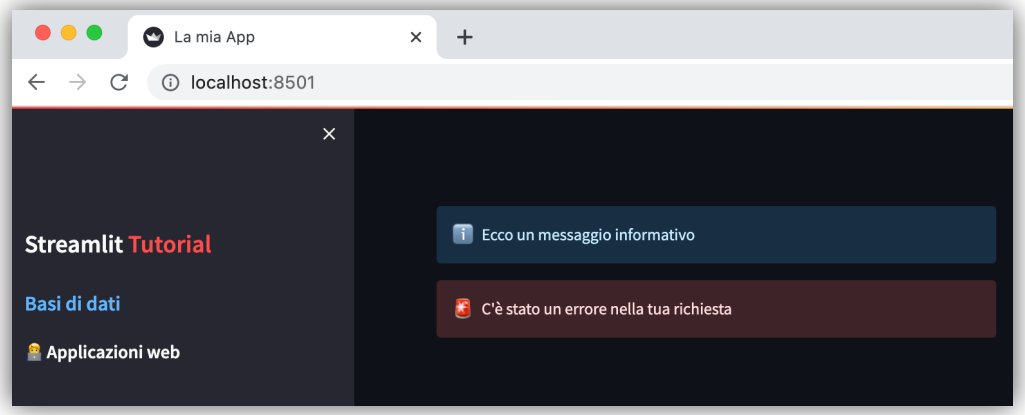
- Modalità con cui condividere variabili tra varie run e pagine, simile a un dizionario Python
  - `st.session_state`
- Bisogna inizializzare la variabile prima di provare ad accedervi o un'eccezione viene sollevata
- Ogni widget con una *key* è automaticamente aggiunto al Session State

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

# Assegnare il valore
if 'key' not in st.session_state:
    st.session_state.key = 'value'
```

Messaggi di stato

- I messaggi di stato sono utili per renderizzare warning, errori o messaggi di successo
  - `st.error(body, *, icon=None)`
  - `st.warning(body, *, icon=None)`
  - `st.info(body, *, icon=None)`
  - `st.success(body, *, icon=None)`

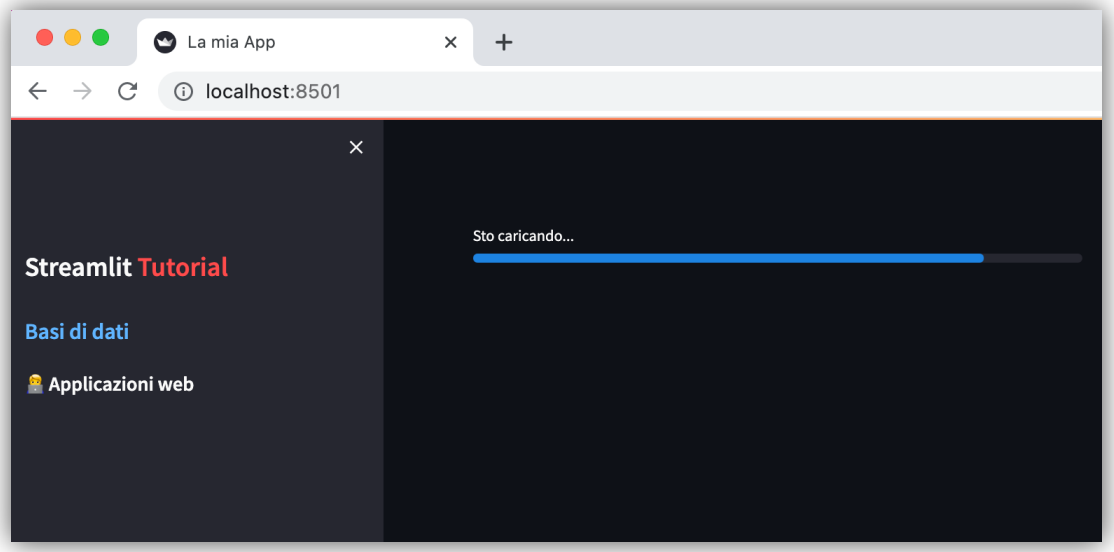
```python
import streamlit as st

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

st.info('Ecco un messaggio informativo', icon="ℹ️")
st.error("C'è stato un errore nella tua richiesta", icon="🚨")
```



Progress bar

- Per visualizzare lo stato di avanzamento si può usare una ***progress bar***
  - `st.progress(value, text=None)`
- Il parametro *valore* è compreso tra 0 e 100 per interi, tra 0.0 e 1.0 per float
- Il parametro *text* è il testo da mostrare sopra la progress bar

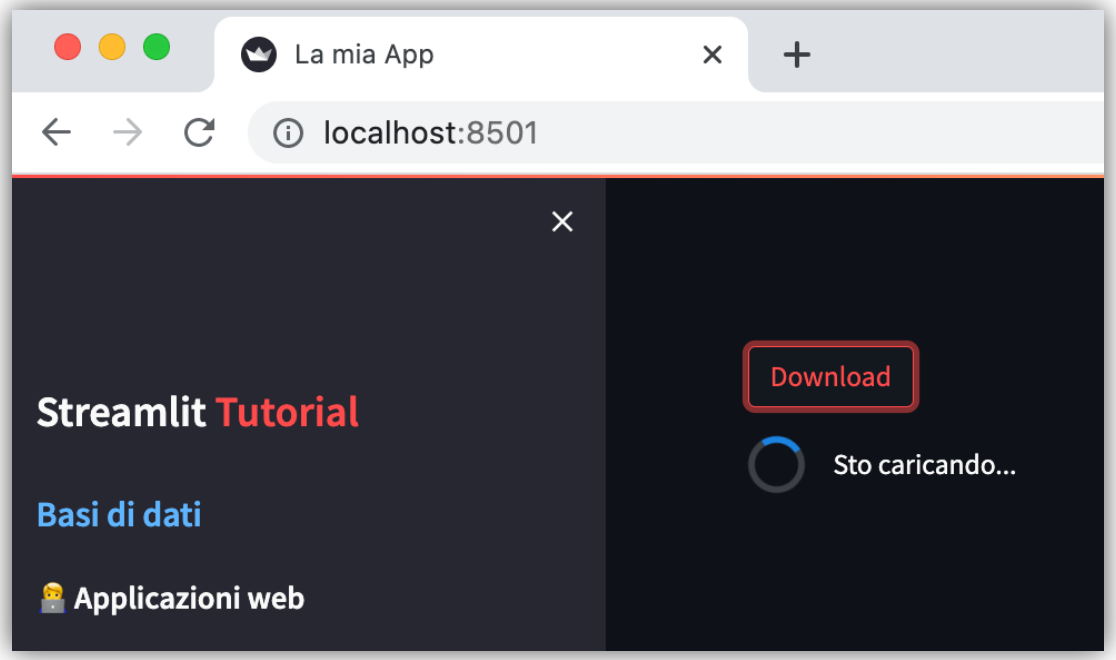
```python
import streamlit as st
import time

col1,col2=st.columns(2)
st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

progress_text = "Sto caricando..."
my_bar = col1.progress(0, text=progress_text)

for percent_complete in range(100):
    time.sleep(0.1)
    my_bar.progress(percent_complete + 1, text=progress_text)
```



Spinner

- Per mostrare un testo temporaneo mentre un blocco di codice viene eseguito, si può usare uno ***spinner*** da mostrare all'utente
  - `st.spinner(text="In progress...")`
- Il parametro *text* è il messaggio da mostrare insieme allo spinner finché l'esecuzione non termina

```python
import streamlit as st
import time

col1,col2=st.columns(2)

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

if col1.button("Download"):
    with st.spinner('Sto caricando...'):
        time.sleep(5)
    col2.success('Done!')
```



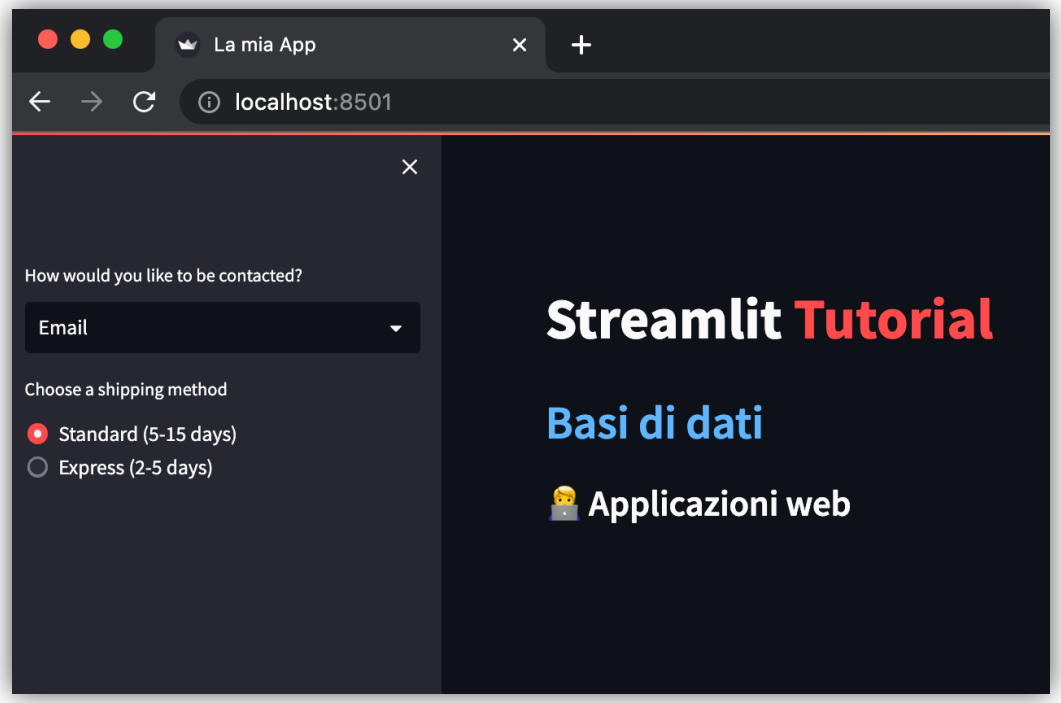
Layout

Applicazioni Web

Sidebar

- La ***sidebar*** è molto utile per aggiungere elementi sulla sinistra, lasciando all'utente piena concentrazione sull'applicativo principale
- Accetta sia la ***object notation*** che la ***with notation***
- Gli elementi di layout possono in genere essere usati come oggetti Streamlit e quindi contenere diversi elementi

```python
import streamlit as st

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded",

)

st.title("Streamlit :red[Tutorial]")
st.header(":blue[Basi di dati]")
st.subheader("👨‍💻 Applicazioni web")

# 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)")
    )
```

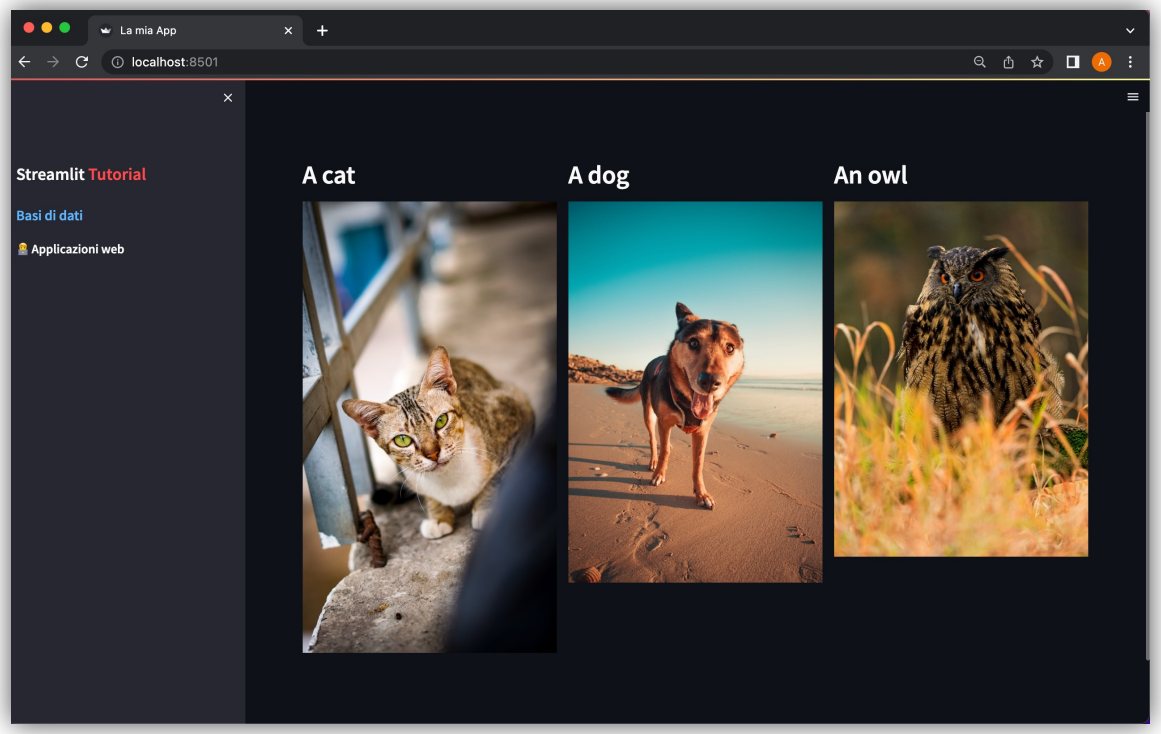


Colonne

- Per suddividere lo spazio in contenitori affiancati, è possibile dividere la pagina in **colonne**
  - `st.columns(spec, *, gap="small")`
- Il parametro *spec* può essere un intero o una lista di numeri
  - se un intero, indica il numero di colonne da inserire, tutte con la stessa larghezza
  - se una lista, viene creata una colonna per ciascun numero con larghezza proporzionale al numero specificato
- Il parametro *gap* indica la distanza tra le colonne
- Viene restituita la lista di container (i.e. le colonne)
- Accetta sia la ***object notation*** che la ***with notation***

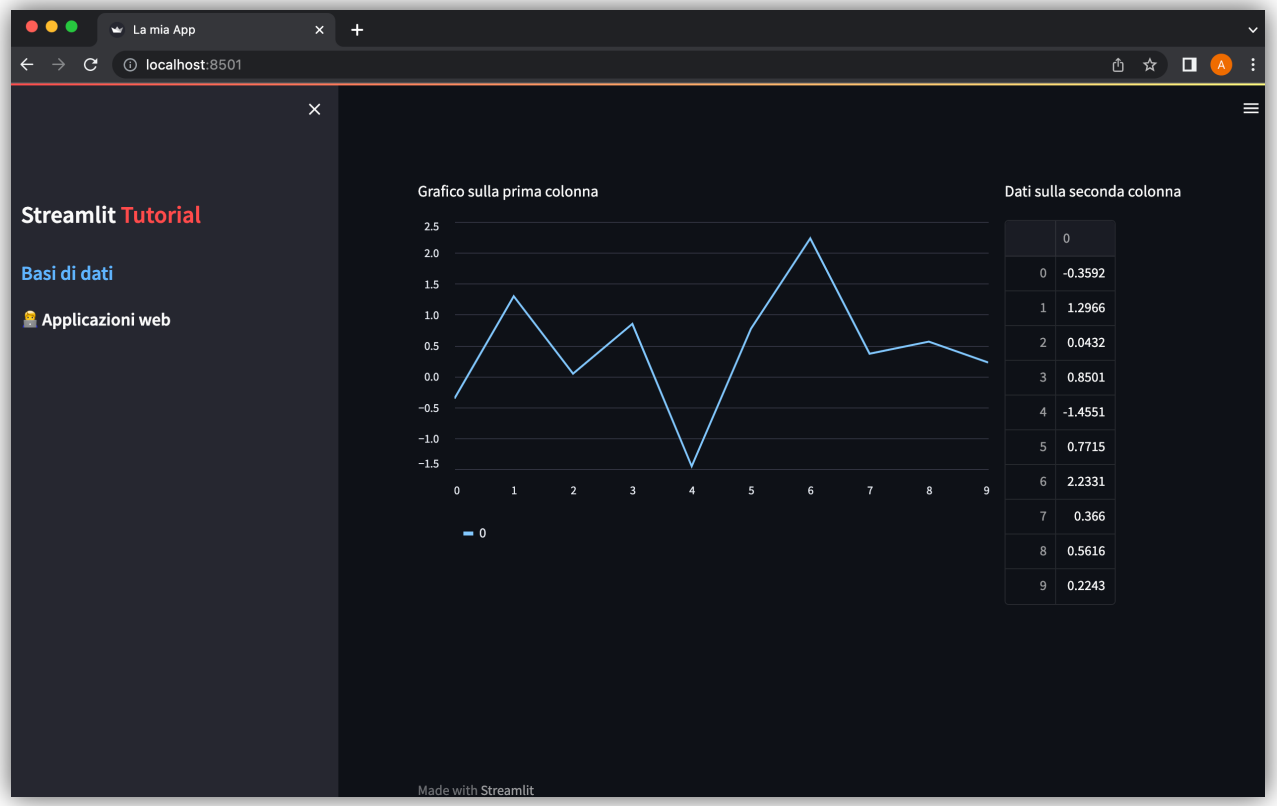
Colonne: esempio with notation

```python
import streamlit as st

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded",

)

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👨‍💼 Applicazioni web")


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")
```



Colonne: esempio oggetto

```python
import streamlit as st
import numpy as np

st.set_page_config(
    page_title="La mia App",
    layout="wide",
    initial_sidebar_state="expanded",

)

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

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

col1.write("Grafico sulla prima colonna")
col1.line_chart(data)

col2.write("Dati sulla seconda colonna")
col2.write(data)
```



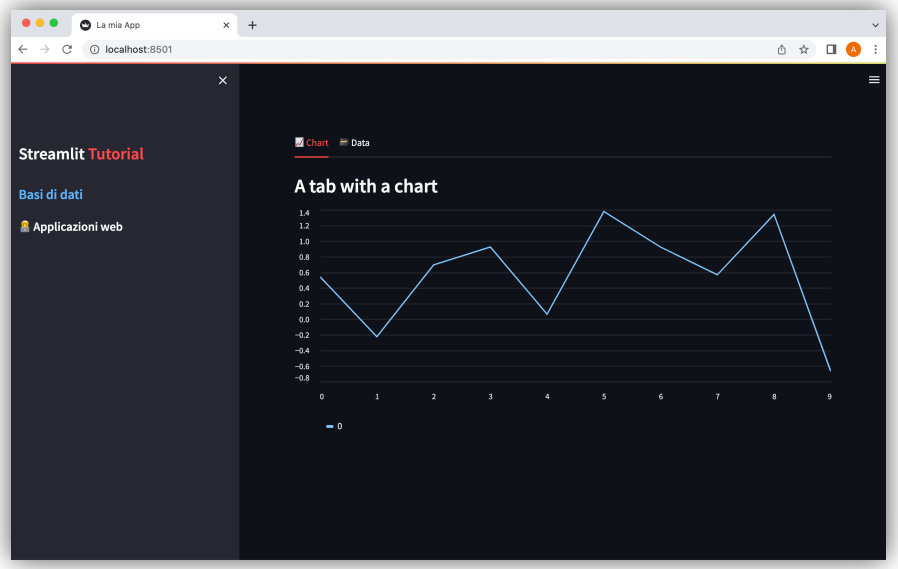
Tabs

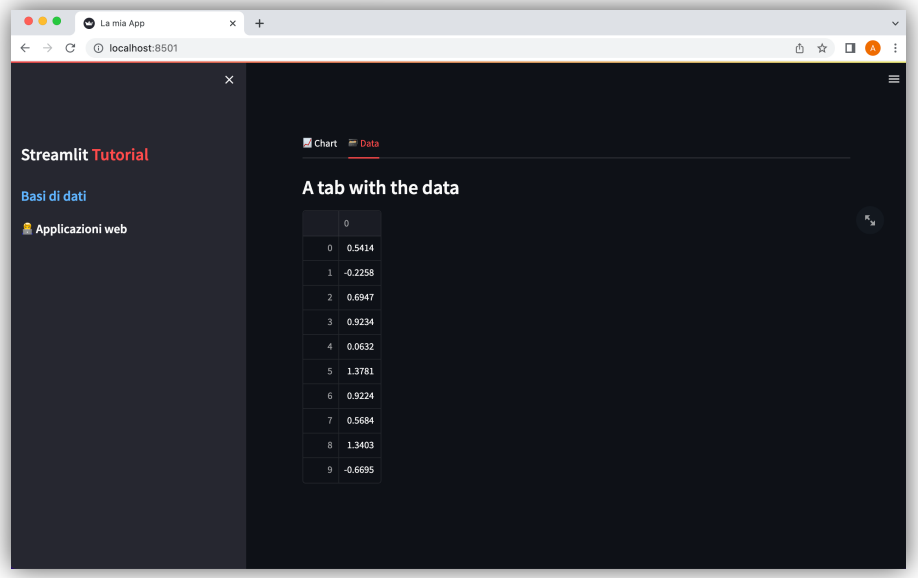
- I tab permettono un'organizzazione più strutturata del contenuto
- L'utente può agilmente navigare tra un tab e l'altro
- Per avere dei container separati, si possono usare i ***tab***
  - `st.tabs(tabs)`

- Il parametro *tabs* è una lista di stringhe, in cui ciascuna rappresenta il nome di un tab
- Il primo tab è quello selezionato di default
- Come per le colonne, restituisce una lista di container
- Accetta la ***with notation***

Esempio Tabs

```python
import streamlit as st
import numpy as np

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)econda colonna")
col2.write(data)
```



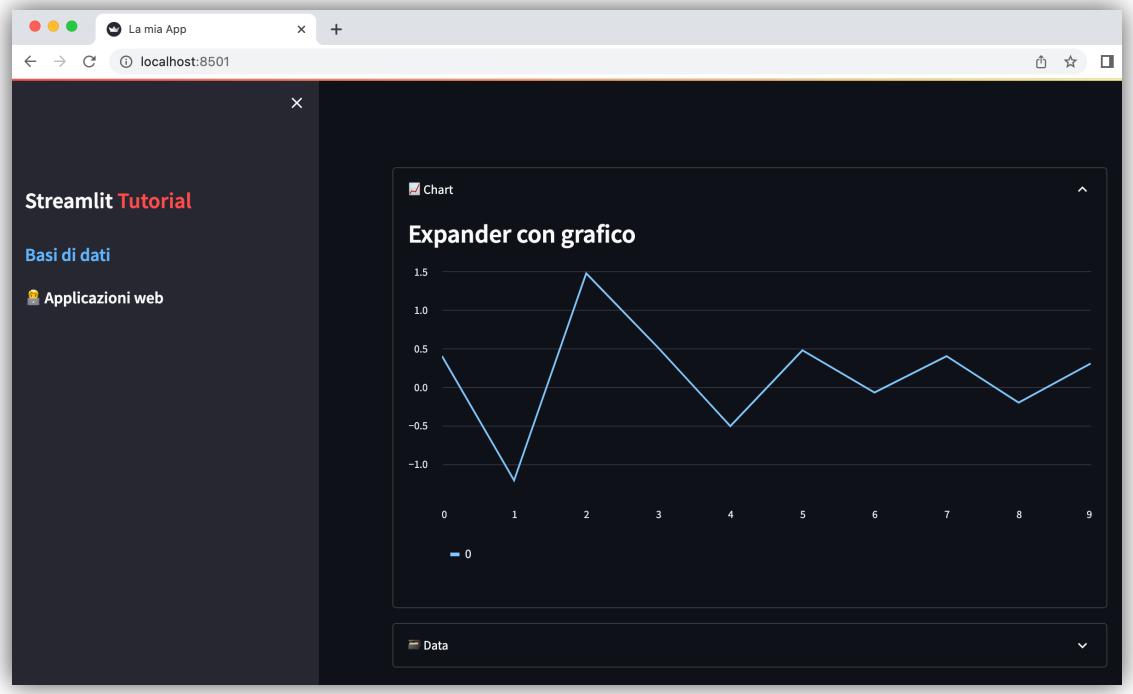


Expander

- Gli ***expander*** permettono di definire dei container che l'utente può scegliere se aprire o chiudere
  - `st.expander(label, expanded=False)`

- Il parametro *label* rappresenta il nome dell'expander
- Il parametro *expanded* rappresenta lo stato di default dell'expander, se aperto o chiuso

```python
import streamlit as st
import datetime

st.sidebar.title("Streamlit :red[Tutorial]")
st.sidebar.header(":blue[Basi di dati]")
st.sidebar.subheader("👷 Applicazioni web")

data = np.random.randn(10, 1)

with st.expander("📈 Chart", expanded=True):
    st.subheader("Expander con grafico")
    st.line_chart(data)

with st.expander("🗃️ Data", expanded=False):
    st.subheader("Expander con i dati")
    st.write(data)
```



Editare i Temi

- E' possibile modificare lo stile e i colori dell'interfaccia per creare un'applicazione personalizzata
- Dal menu andare su *Settings* e poi su *Theme*
- Si può scegliere tra la modalità ***Light*** e quella ***Dark*** oppure modificarli creando il proprio tema (modificando i colori e il font)
- In questo modo è possibile sperimentare live le proprie personalizzazioni prima di copiarle nel file di configurazione all'interno della sezione `[theme]`