

Streamlit



¿Qué es Streamlit?

Streamlit es una herramienta de visualización Open Source orientado a Python con el que crear apps web. Entre sus características más destacables, tenemos:

1. Desarrollo con pocas líneas de código
2. Curva de aprendizaje sencilla
3. Implementación sencilla
4. Compatible con librerías de visualización (Matplotlib)
5. Cacheo de los datos que acelera el cómputo
6. No es necesario conocer lenguajes orientados a aplicaciones (HTML, CSS,...)

Instalación

- Se puede instalar a través de Anaconda : “conda install streamlit” o con pip, “pip install streamlit”.
- Luego, deberemos activar el kernel para que lo vea jupyter, “python -m ipykernel install --user --name=streamlit”.

Componentes de una app

- Las aplicaciones de Streamlit no requieren saber lenguajes de programación Web, como HTML5 o JAVA.
- Cada aplicación se ejecuta en un servidor al cual los usuarios se conectan.
 - Streamlit arranca un **servidor local** (por defecto en <http://localhost:8501>)

Componentes de una app

- Cuando ejecutas :

```
streamlit run nombre_archivo.py
```

Tu navegador se abre automáticamente para mostrar la app.

Puedes seguir programando en **Visual Studio** y ver los cambios actualizados en el navegador (Streamlit recarga automático al guardar).

Componentes de una app

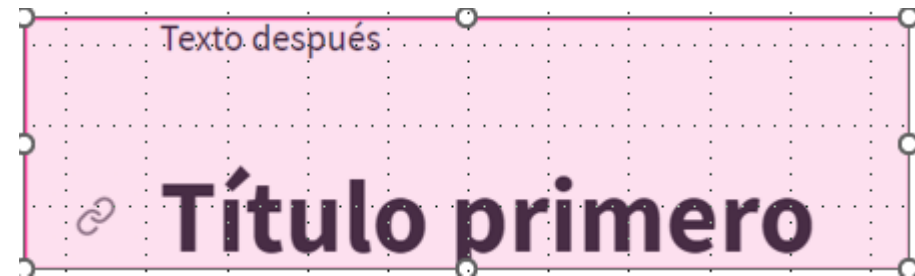
El código en Streamlit, se ejecuta de arriba abajo, por lo que los elementos de la app se irán renderizando en este orden.

Para importarlo, se usa el estándar “st”, “import streamlit as st”.



```
import streamlit as st

# Main
st.title("Título primero")
st.markdown("Texto después")
```



```
import streamlit as st

# Main
st.markdown("Texto después")
st.title("Título primero")
```

Componentes de una app

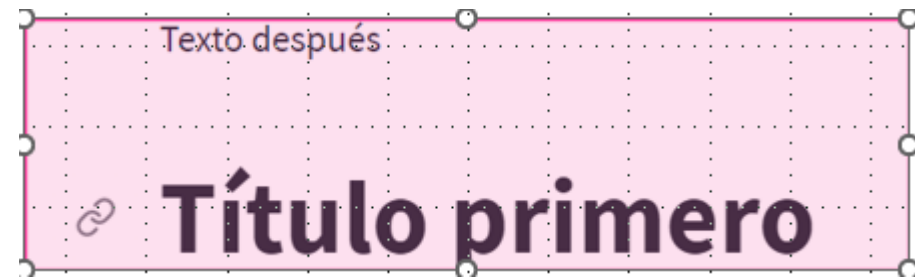
El código en Streamlit, se ejecuta de arriba abajo, por lo que los elementos de la app se irán renderizando en este orden.

Para importarlo, se usa el estándar “st”, “import streamlit as st”.



```
import streamlit as st

# Main
st.title("Título primero")
st.markdown("Texto después")
```



```
import streamlit as st

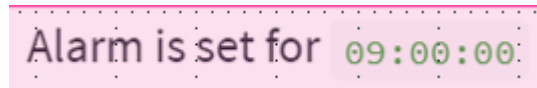
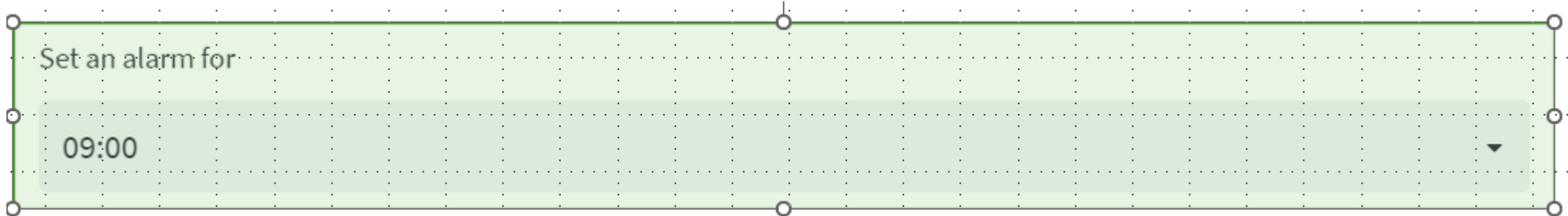
# Main
st.markdown("Texto después")
st.title("Título primero")
```

Componentes de una app

Todas las aplicaciones, presentan **entradas** y **salidas**.

Las **entradas**, permiten a los usuarios elegir las opciones.

Las **salidas**, permiten a los usuarios observar los resultados.



Ejercicio

Ejercicio 1. Crea tu primera aplicación de Streamlit 'Hola mundo'.

Con esto veremos si tu sistema está listo para empezar a trabajar con él.
Lo único que importa ahora es que tengas una app que compile sin errores.

Más adelante, sabréis qué es lo que hace.

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

# Main
st.title("Esta es mi primera
aplicación")
datos = pd.DataFrame(
    np.random.randn(20, 3),
    columns = ['a', 'b', 'c'])

st.line_chart(datos)
st.dataframe(datos)
```

```
(streamlit) D:\TARTANGA\2025-2026\PIA\2. Visualizacion de datos\Streamlit>streamlit run Ejercicio2.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
```

Componentes de una app



- En Streamlit tenemos dos tipos de elementos:
- ENTRADAS (Inputs): Son widgets interactivos donde el usuario introduce información o hace selecciones. Son como los botones y campos de un formulario web.
- SALIDAS (Outputs): Son elementos que muestran información al usuario. Solo sirven para visualizar, no para interactuar.
- La clave es: las ENTRADAS devuelven valores que podemos usar en nuestro código Python, las SALIDAS solo muestran contenido.

Entradas (devuelven valores)

Widget	Uso	Ejemplo
checkbox	Sí/No	<code>st.checkbox("Texto")</code>
radio	Una opción (botones)	<code>st.radio("Texto", ["A", "B"])</code>
selectbox	Una opción (desplegable)	<code>st.selectbox("Texto", ["A", "B"])</code>
multiselect	Múltiples opciones	<code>st.multiselect("Texto", ["A", "B"])</code>
select_slider	Selector con opciones	<code>st.select_slider("Texto", ["A", "B"])</code>
slider	Rango numérico	<code>st.slider("Texto", 0, 100)</code>
number_input	Número específico	<code>st.number_input("Texto", 0, 100)</code>
text_input	Texto libre	<code>st.text_input("Texto")</code>
date_input	Fecha	<code>st.date_input("Texto")</code>
file_uploader	Subir archivos	<code>st.file_uploader("Texto", type=['csv'])</code>


Entradas (devuelven valores)

1. Selección Simple

Widget	Uso	Retorna	Ejemplo Visual
<code>st.checkbox()</code>	Casilla Sí/No	True o False	<input checked="" type="checkbox"/> Mostrar datos
<code>st.radio()</code>	Botones de opción (1 de N)	String o valor	<input checked="" type="radio"/> Opción A <input type="radio"/> B <input type="radio"/> C
<code>st.selectbox()</code>	Lista desplegable (1 de N)	String o valor	 Elegir ciudad ▼
<code>st.toggle()</code>	Interruptor ON/OFF	True o False	 Modo oscuro





Entradas (devuelven valores)

2. Selección múltiple



Widget	Uso	Retorna	Ejemplo
<code>st.multiselect()</code>	Selección múltiple	Lista []	<input checked="" type="checkbox"/> Madrid <input checked="" type="checkbox"/> Barcelona <input type="checkbox"/> Valencia
<code>st.select_slider()</code>	Slider con opciones predefinidas	String o valor	←---  ----→ [S, M, L, XL]

Entradas (devuelven valores)

3. Valores numéricos



Widget	Uso	Retorna	Ejemplo
<code>st.slider()</code>	Deslizador numérico	<code>int</code> , <code>float</code> o <code>tupla</code>	 <code>[0...100]</code>
<code>st.number_input()</code>	Campo numérico con +/-	<code>int</code> o <code>float</code>	 <code>[25]</code>  

4. Texto



widget	Uso	Retorna	Ejemplo
<code>st.text_input()</code>	Entrada de texto corto	<code>String</code>	 Tu nombre: <code>[_____]</code>
<code>st.text_area()</code>	Entrada de texto largo	<code>String</code>	 Comentarios: <code>[_____]</code>

Entradas (devuelven valores)

5. Fechas y Horas




Widget	Uso	Retorna	Ejemplo
<code>st.date_input()</code>	Selector de fecha	<code>datetime.date</code>	 01/01/2024
<code>st.time_input()</code>	Selector de hora	<code>datetime.time</code>	 14:30

6. Archivos

widget	Uso	Retorna	Ejemplo
<code>st.file_uploader()</code>	Subir archivos	Objeto archivo o <code>None</code>	 Arrastra archivo aquí
<code>st.camera_input()</code>	Capturar foto con cámara	Imagen o <code>None</code>	 Tomar foto








Entradas (devuelven valores)

7Otros

Widget	Uso	Retorna	Ejemplo
<code>st.color_picker()</code>	Selector de color	String (hex)	 #FF5733
<code>st.button()</code>	Botón de acción	True (al pulsar)	 Enviar
<code>st.download_button()</code>	Botón de descarga	True (al pulsar)	 Descargar CSV

Salidas (solo muestran el contenido)

1. Textos y Títulos

Widget	Uso	Ejemplo	Tamaño
<code>st.title()</code>	Título principal	<code>st.title("Mi App")</code>	 Grande
<code>st.header()</code>	Encabezado de sección	<code>st.header("Sección 1")</code>	 Medio
<code>st.subheader()</code>	Sub-encabezado	<code>st.subheader("Apartado")</code>	 Normal+
<code>st.text()</code>	Texto plano simple	<code>st.text("Hola")</code>	 Pequeño
<code>st.write()</code>	Texto versátil (detecta formato)	<code>st.write("Texto o datos")</code>	 Normal
<code>st.markdown()</code>	Texto con formato Markdown	<code>st.markdown("**Negrita**")</code>	 Normal
<code>st.caption()</code>	Texto pequeño/aclaración	<code>st.caption("Nota pequeña")</code>	 Muy pequeño









Salidas (solo muestran el contenido)

2. Datos y Tablas

Widget	Uso	Ejemplo	Interactivo
<code>st.dataframe()</code>	Tabla interactiva	<code>st.dataframe(df, height=300)</code> # Puedes ajustar altura	✓ Sí (scroll, ordenar)
<code>st.table()</code>	Tabla estática	<code>st.table(df)</code>	✗ No
<code>st.metric()</code>	Métricas con delta	<code>st.metric("KPI", 100, delta=5)</code>	✗ No
<code>st.json()</code>	Mostrar JSON	<code>st.json({"key": "value"})</code>	✓ Sí (expandible)

Salidas (solo muestran el contenido)

3. Mensajes y Alertas

Widget	Uso	Color	Icono
<code>st.success()</code>	Mensaje de éxito	 Verde	
<code>st.info()</code>	Información	 Azul	
<code>st.warning()</code>	Advertencia	 Amarillo	
<code>st.error()</code>	Error	 Rojo	

Salidas (solo muestran el contenido)

4. Multimedia

Widget	Uso	Ejemplo
<code>st.image()</code>	Mostrar imagen	<code>st.image("foto.jpg", caption="Mi foto")</code>
<code>st.audio()</code>	Reproducir audio	<code>st.audio("audio.mp3")</code>
<code>st.video()</code>	Reproducir video	<code>st.video("video.mp4")</code>

Salidas (solo muestran el contenido)

5. Gráficos

Widget	Librería	Ejemplo
<code>st.plotly_chart()</code>	Plotly	<code>st.plotly_chart(fig)</code>
<code>st.pyplot()</code>	Matplotlib	<code>st.pyplot(fig)</code>
<code>st.altair_chart()</code>	Altair	<code>st.altair_chart(chart)</code>
<code>st.vega_lite_chart()</code>	Vega-Lite	<code>st.vega_lite_chart(spec)</code>
<code>st.line_chart()</code>	Nativo Streamlit	<code>st.line_chart(df)</code>
<code>st.bar_chart()</code>	Nativo Streamlit	<code>st.bar_chart(df)</code>
<code>st.area_chart()</code>	Nativo Streamlit	<code>st.area_chart(df)</code>
<code>st.map()</code>	Mapa (Mapbox)	<code>st.map(df)</code>

Salidas (solo muestran el contenido)

6. Otros

Widget	Uso	Ejemplo
<code>st.progress()</code>	Barra de progreso	<code>st.progress(0.7)</code>
<code>st.spinner()</code>	Animación de carga	<code>with</code> <code>st.spinner("Cargando..."):</code>
<code>st.balloons()</code>	Animación festiva	<code>st.balloons()</code>
<code>st.snow()</code>	Animación de nieve	<code>st.snow()</code>

Salidas

```
import streamlit as st

# Main
st.title("Aplicación")
st.header("Blablabla")
st.markdown("Apartado 1")
st.subheader("Sub-apartado 1")
st.caption("this is the caption")
st.code("x=3")
```

También podemos añadir fotos, vídeos y audios usando una línea

```
st.image("ruta_foto.jpg")
st.audio("ruta_audio.mp3")
st.video("ruta_video.mp4")
```

Aplicación

Blablabla

Apartado 1

Sub-apartado 1

this is the caption

x=3



$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Salidas (solo muestran el contenido)

1. Textos y Títulos

st.write() detecta automáticamente el tipo de dato

st.write("Texto") # Muestra texto

st.write(df) # Muestra dataframe

st.write([1, 2, 3]) # Muestra lista

st.write({"a": 1}) # Muestra dict

Otros son específicos

st.text("Solo texto") # Solo para texto

st.dataframe(df) # Solo para dataframes

SIN HTML (seguro)

st.markdown("Negrita** normal")**

CON HTML (más libertad)

```
st.markdown(  
    "<h1 style='color: red;'>Rojo</h1>",  
    unsafe_allow_html=True  
)
```


Salidas

```
import streamlit as st
from sklearn.datasets import load_iris
```

```
# Carga de datos
iris = load_iris()
```

```
# Main
st.title("Mostramos tabla de iris")
st.table(iris['data'])
```

Mostramos tabla de iris

	0	1	2	3
0	5.1000	3.5000	1.4000	0.2000
1	4.9000	3.0000	1.4000	0.2000
2	4.7000	3.2000	1.3000	0.2000
3	4.6000	3.1000	1.5000	0.2000
4	5.0000	3.6000	1.4000	0.2000
5	5.4000	3.9000	1.7000	0.4000
6	4.6000	3.4000	1.4000	0.3000
7	5.0000	3.4000	1.5000	0.2000
8	4.4000	2.9000	1.4000	0.2000
9	4.9000	3.1000	1.5000	0.1000
10	5.4000	3.7000	1.5000	0.2000
11	4.8000	3.4000	1.6000	0.2000
12	4.8000	3.0000	1.4000	0.1000
13	4.3000	3.0000	1.1000	0.1000
14	5.8000	4.0000	1.2000	0.2000
15	5.7000	4.4000	1.5000	0.4000
16	5.4000	3.9000	1.3000	0.4000
17	5.1000	3.5000	1.4000	0.3000
18	5.7000	3.8000	1.7000	0.3000
19	5.1000	3.8000	1.5000	0.3000

```
import streamlit as st
from sklearn.datasets import load_iris
```

```
# Carga de datos
iris = load_iris()
```

```
# Main
st.title("Mostramos tabla de iris")
st.dataframe(iris['data'])
```

Mostramos tabla de iris

	0	1	2	3
0	5.1000	3.5000	1.4000	0.2000
1	4.9000	3.0000	1.4000	0.2000
2	4.7000	3.2000	1.3000	0.2000
3	4.6000	3.1000	1.5000	0.2000
4	5.0000	3.6000	1.4000	0.2000
5	5.4000	3.9000	1.7000	0.4000
6	4.6000	3.4000	1.4000	0.3000
7	5.0000	3.4000	1.5000	0.2000
8	4.4000	2.9000	1.4000	0.2000
9	4.9000	3.1000	1.5000	0.1000

Made with Streamlit

Salidas

A través de la función “`st.pyplot`” podemos mostrar cualquier plot generado con “`matplotlib`”.

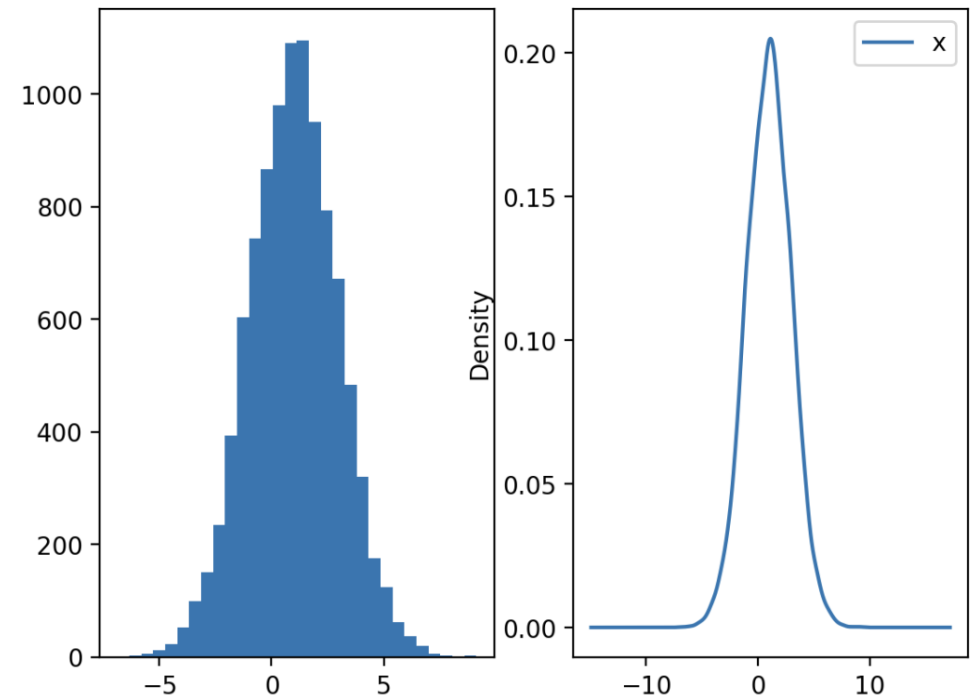
```
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Carga de datos
rand=np.random.normal(1, 2, size=10000)
df = pd.DataFrame({'x' : rand})

# Main
fig, ax = plt.subplots(ncols = 2)
ax[0].hist(rand, bins=30)
df.plot(kind='density', ax = ax[1])

st.header("Distribución normal: histograma y densidad")
st.pyplot(fig)
```

Distribución normal: histograma y densidad



Flujo en Streamlit: **Entradas** → **Procesamiento** → **Salidas**

Un cuadro de mando con Streamlit tiene dos elementos:

1. Interfaz de usuario (UI): Define las **entradas** y **salidas**.

2. Lógica de servidor: Define los cálculos que transforman las **entradas** en **salidas**

🛠️ ENTRADAS (Widgets)

- ✅ `st.checkbox()` → bool
- 🔵 `st.radio()` → opción
- 📁 `st.selectbox()` → opción
- 🔧 `st.slider()` → número
- 📄 `st.text_input()` → texto
- 1234 `st.number_input()` → número
- 🔴 `st.button()` → bool



⚙️ PROCESAMIENTO (Python)

```
if aceptar: ...  
resultado = modelo.predict(...)  
gráfico = sns.scatterplot(...)  
df_filtrado = df.query(...)  
cálculo = sum(lista)  
...
```



📊 SALIDAS (Visualización)

```
st.title("Mi App")  
st.header("Resultados")  
st.write("Texto, tablas...")  
st.pyplot(fig)  
st.dataframe(df)  
st.metric("Accuracy", "95%")
```

Conjunto de datos

Vamos trabajar sobre el conjunto de datos “penguins”. Para ello, activamos el kernel e instalamos la librería con

```
import seaborn as sns
penguins = sns.load_dataset('penguins')
```

	<u>species</u>	island	bill_length_mm	<u>bill_depth_mm</u>	<u>flipper_length_mm</u>	body_mass_g	sex	<u>year</u>
0	<u>Adelie</u>	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	<u>Adelie</u>	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	<u>Adelie</u>	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	<u>Adelie</u>	<u>Torgersen</u>	NaN	NaN	NaN	NaN	NaN	2007
4	<u>Adelie</u>	Torgersen	36.7	19.3	193.0	3450.0	female	2007
...
339	Chinstrap	Dream	55.8	19.8	207.0	4000.0	male	2009
340	Chinstrap	Dream	43.5	18.1	202.0	3400.0	<u>female</u>	2009
341	Chinstrap	Dream	49.6	18.2	193.0	3775.0	male	2009
342	Chinstrap	Dream	50.8	19.0	210.0	4100.0	male	2009
343	<u>Chinstrap</u>	Dream	50.2	18.7	198.0	3775.0	female	2009

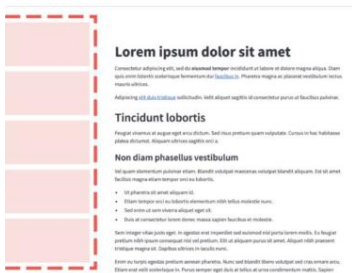
LAYOUTS

Tenemos todos los elementos, pero hasta ahora solo hemos sido capaces de ponerlos uno detrás de otros. Usando los layouts, organizamos el contenido según nos plazca.

En Streamlit, existen 6 objetos que permiten hacer esto: sidebar, columns, tabs, expander, container y empty.



Layouts



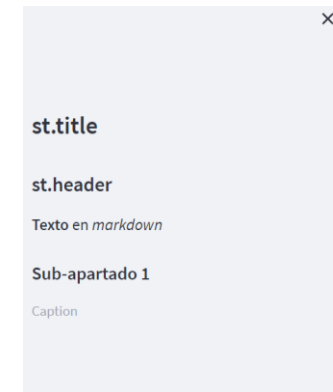
“**sidebar**” es una barra lateral colapsable que se añadirá automáticamente a la aplicación cuando sea usada. Normalmente, se usa para poner todos los **filtros** presentes en ella.

```
import streamlit as st
```

```
st.sidebar.title("st.title")
st.sidebar.header("st.header")
st.sidebar.markdown("**Texto** en _markdown_")
st.sidebar.subheader("Sub-apartado 1")
st.sidebar.caption("Caption")
```

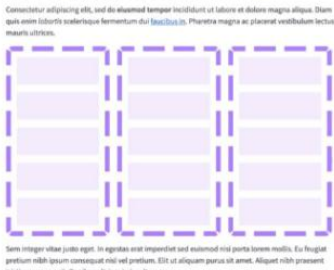
```
import streamlit as st
```

```
with st.sidebar:
    st.title("st.title")
    st.header("st.header")
    st.markdown("**Texto** en _markdown_")
    st.subheader("Sub-apartado 1")
    st.caption("Caption")
```



Layouts

Lorem ipsum dolor sit amet



“**columns**” inserta contenedores uno al lado del otro y devuelve una lista de columnas, que se usarán para ir poniendo los elementos que queramos. Es muy recomendable usar la sintaxis “**with()**”.

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

spec (<i>int or list of numbers</i>)	Si un int: especifica el número de columnas que se van a insertar y todas las columnas tienen el mismo ancho. Si es una lista de números: crea una columna para cada número y el ancho de cada columna es proporcional al número proporcionado. Los números pueden ser ints o floats, pero deben ser positivos. Por ejemplo, <code>st.columns([3, 1, 2])</code> crea 3 columnas donde la primera columna es 3 veces el ancho de la segunda y la última columna es 2 veces ese ancho.
gap (<i>string ("small", "medium", or "large")</i>)	Una cadena opcional, que indica el tamaño del espacio entre cada columna. El valor predeterminado es un pequeño espacio entre columnas. Este argumento sólo puede ser suministrado por palabra clave.

```
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

col1, col2 = st.columns([3, 1])
data = np.random.randn(10, 1)
fig, ax = plt.subplots()
ax.plot(data)

with col1:
    st.subheader("Un gráfico que ocupa mucho")
    st.pyplot(fig)

with col2:
    st.subheader("Tabla pequeña")
    st.write(data)
```

```
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

col1, col2 = st.columns([3, 1])
data = np.random.randn(10, 1)
fig, ax = plt.subplots()
ax.plot(data)

col1.subheader("Un gráfico que ocupa mucho")
col1.pyplot(fig)

col2.subheader("Tabla pequeña")
col2.write(data)
```

Un gráfico que ocupa mucho

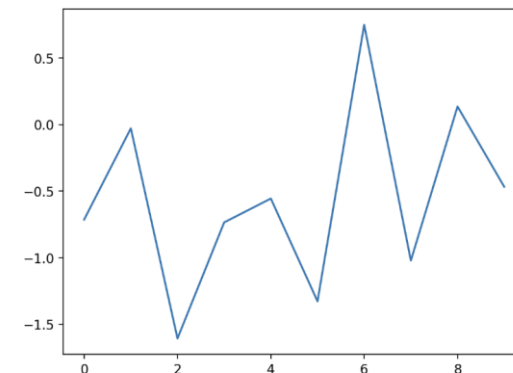
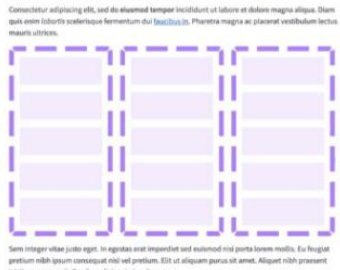


Tabla pequeña

	0
0	-0.7139
1	-0.0289
2	-1.6066
3	-0.7352
4	-0.5562
5	-1.3278
6	0.7488
7	-1.0217
8	0.1353
9	-0.4674

Layouts

Lorem ipsum dolor sit amet



“**columns**” inserta contenedores uno al lado del otro y devuelve una lista de columnas, que se usarán para ir poniendo los elementos que queramos. Es muy recomendable usar la sintaxis “**with()**”.

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

spec (<i>int or list of numbers</i>)	Si un int: especifica el número de columnas que se van a insertar y todas las columnas tienen el mismo ancho. Si es una lista de números: crea una columna para cada número y el ancho de cada columna es proporcional al número proporcionado. Los números pueden ser ints o floats, pero deben ser positivos. Por ejemplo, <code>st.columns([3, 1, 2])</code> crea 3 columnas donde la primera columna es 3 veces el ancho de la segunda y la última columna es 2 veces ese ancho.
gap (<i>string ("small", "medium", or "large")</i>)	Una cadena opcional, que indica el tamaño del espacio entre cada columna. El valor predeterminado es un pequeño espacio entre columnas. Este argumento sólo puede ser suministrado por palabra clave.

Columnas iguales

```
col1, col2 = st.columns(2)
```

#Proporciones específicas

```
col1, col2, col3 = st.columns([3, 1, 1]) # 60% - 20% - 20%
```

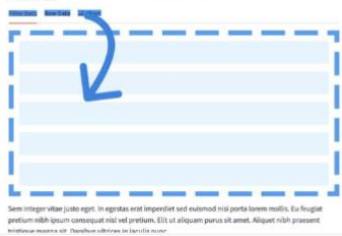
Con espacios entre columnas

```
col1, col2 = st.columns(2, gap="large") # "small", "medium", "large"
```


Layouts

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



“**tab**” inserta contenedores en tabs separadas. En cada uno de ellos podemos poner cuantos elementos queramos. Es muy recomendable usar la sintaxis “**with()**”.

st.tabs(tabs)

tabs (<i>list of strings</i>)	Crea una tab para cada cadena de la lista. La cadena se utiliza como nombre de la pestaña. La primera pestaña está seleccionada de forma predeterminada.
--	--

```
import streamlit as st
```

```
tab1, tab2, tab3 = st.tabs(["Entradas", "Salidas",  
"Reactividad"])
```

```
with tab1:  
    st.header("Tab 1")  
    st.write("Contenido tab 1")
```

```
with tab2:  
    st.header("Tab 2")  
    st.write("Contenido tab 2")
```

```
with tab3:  
    st.header("Tab 3")  
    st.write("Contenido tab 3")
```

Entradas Salidas Reactividad

Tab 1

Contenido tab 1

Entradas Salidas Reactividad

Tab 2

Contenido tab 2

Entradas Salidas Reactividad

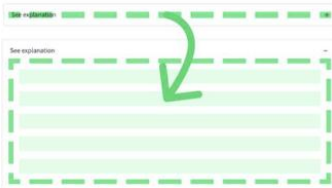
Tab 3

Contenido tab 3

Layouts

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



“**expander**” inserta un contenedor en la aplicación que se puede usar para contener varios elementos, pudiendo ser colapsado o contraído. Cuando se contrae, lo único visible es la etiqueta proporcionada. Es muy recomendable usar la sintaxis “with()”.

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

label (str)	Una cadena para usar como encabezado para el expensor.
expanded (bool)	Si es True, inicializa el expensor en estado "expandido". El valor predeterminado es False (contraído).

```
import streamlit as st
```

```
st.latex(r'''x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}''')
with st.expander("Explicación"):
    st.write("Solución a las ecuaciones de segundo grado")
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Explicación



$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Explicación

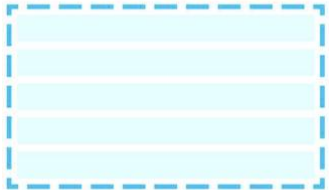


Solución a las ecuaciones de segundo grado

Layouts

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



“**container**” inserta un contenedor invisible en la aplicación que se puede usar para contener varios elementos.

```
import streamlit as st
```

```
container = st.container()  
container.write("Empieza el contenedor")  
st.write("Esto está fuera de él.")
```

```
container.write("Esto está dentro del contenedor.")
```

Empieza el contenedor

Esto está dentro del contenedor.

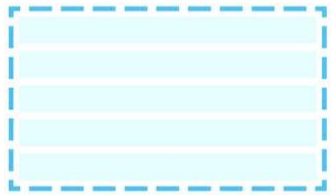
Esto está fuera de él.

Son interesantes a la hora de definir la estructura de la aplicación. Usando la notación “`with()`” nos obligará a mantener un orden en esta y nuestro código será más limpio.

Layouts

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostris exercitationem ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Sem integer vitae justo eget. In egestas erat imperdiet sed euismod nisi porta lorem mollis. Eu feugiat pretium nibh ipsum consequat. Nisi vel pretium. Egestas ut aliquam purus sit amet. Aliquet nibh praesent tristique magna sit. Phasellus convallis elit. Sed ut perspiciatis unde namque posuere congue.

“**empty**” inserta un contenedor en la aplicación que se puede usar para contener un solo elemento. Esto permite, por ejemplo, eliminar elementos en cualquier punto o reemplazar varios elementos a la vez (utilizando un contenedor secundario de varios elementos).

```
import streamlit as st
import time

lugar = st.empty()

lugar.text("¡Hola!")
time.sleep(5)
lugar.text("Procesando los datos...")
time.sleep(5)

with lugar.container():
    st.write("Elemento 1.")
    st.write("Elemento 2.")

time.sleep(5)
lugar.empty()
```

¡Hola!

Procesando los datos...

Elemento 1.

Elemento 2.