Seminario de Lenguajes - Python

Cursada 2024

🫀 Conociendo Streamlit

Instalación de Streamlit:

- ⚡ Introducción a Streamlit y su instalación en el entorno de desarrollo.
- ⚡ Creación de un entorno virtual y activación del mismo.
- ⚡ Instalación de Streamlit a través de pip.

¿Qué es?

Streamit es una herramienta que permite generar web apps interactivas sencillas

Características:

- ✅ Es sencillo para comenzar a desarrollar web apps
- ✅ Orientado a la ciencia de datos
- ✅ Permite la integración con las librerías más usadas de Python directamente
- ✅ Actualización en tiempo real, las modificaciones se visualizan sin volver a ejecutar

Instalación

- Se recomienda generar un entorno virtual
- Instalar utilizando pip

```
$ python3 -m venv env
$ source env/bin/activate
$ (env) $ pip install streamlit
```

Creación de una aplicación básica:

- Importar el módulo Streamlit en un script de Python.
- Utilizar la función **st.write()** para mostrar texto en la aplicación.
- Ejecutar la aplicación utilizando el comando **streamlit run archivo.py**.
- Organizar la información.

Veamos un ejemplo

st.write() es una función para mostrar texto que permite múltiples opciones y puede mostrar varios tipos de datos.

- Se considera el Swiss Knife de Streamlit. Acepta casi cualquier tipo de dato

1. Se le pueden pasar cualquier cantidad de parámetros, y **TODOS** serán escritos.

2. Su comportamiento depende del tipo de parámetros que le lleguen.

3. Devuelve *None*, por ende, no es un elemento que pueda ser reusado en la ejecución de la aplicación.

st.write(*args, unsafe_allow_html=False, **kwargs)

Algunos de los tipos que soporta son:

- write(string): Imprime el string en formato Markdown, con soporte de expresiones LaTeX, emojis, y texto en color.
- write(data_frame): Muestra el DF en formato de tabla.
- write(error): Imprime la excepción.
- write(mpl_figure): Muestra una figura de Matplotlib.
- write(altair): Muestra un gráfico de Altair.
- write(plotly_fig): Muestra una figura de Plotly.
- write(dict): Muestra un diccionario en un widget interactivo.

st.write en el código

In [2]:
```
import streamlit as st
st.header('Mi primera app con Stremalit')
st.subheader('''Características de st.write
             Muestra texto plano o con formato Markdown''')
st.write('Hola **mundo**')
if st.button('Mostrar opciones'):
    st.write('# Soy un título')
    st.write(""" * se considera el Swiss Knife de Streamlit. Esto quiere decir que tie
    """)
st.subheader('Contenido de variables simples')
x = 10
st.write("El valor de x es:", x)

st.subheader('Contenido de estructuras')
my_list = [1, 2, 3, 4, 5]
st.write("La lista es:", my_list)
```

Diccionarios

También puede mostrar el contenido de un diccionario:

In [3]:
```
profiles = { "Twist":{
        "name": "Mark Twain",
        "age": 45,
        "gender": "Masculino",
        "avatar": "bear.png"
    },"Kalpa":{
        "name": "Angelica Gorodischer",
        "age": 95,
        "gender": "Femenino",
        "avatar": "giraffe.png"
    }}
```

💫 Personalización

- Definir el estilo de la página:
  - **st.set_page_config(layout="wide")**
- Cambiar el título de la aplicación con **st.title()**, **st.header()** y **st.subheader()**.
- Generar divisiones en la pantalla con **st.divide()**
- Expander elementos:

```
with st.expander('Mostrar contenido variables'):
    ...
```

📂Estructuración

Podemos estructura nuestra página usando diferentes recursos:

- Pages
- Containers
  - Tabs
  - Columnas
  - Formularios

📌 Pages

- Estructura dinámica, se debe crear un directorio **pages** y dentro crear archivos con numeración:
  - 01_nombre pagina1.py
  - 02_nombre pagina2.py
  - 03_emoji_nombre pagina3.py
- Se genera un sidebar.
- El orden en que se muestra está dado por la numeración.
- Se pueden poner emojis.

📌Tabs

- Nos permiten organizar nuestros datos
- Se deben definir con una lista los nombres de los tabs y luego incluir el código dentro de cada una

Usamos la sentencia **with**, recuerdan qué hace?

```
tab1, tab2 = st.tabs(["Uno", "Dos"])

with tab1:
    ....
with tab2:
    .....
```

📌 Columnas

- Divide el espacio según la cantidad de columnas **st.columns(number)**.
- Cada columna deben indicarse con el número correspondiente, en lugar de **st**, debe ir **c1** o la columna correspondiente

```
with tab1:
```

```
    c1, c2 = st.columns(2)
    c1.write("Columna 1")
    c2.write("Columna 2")
    ....
```

📌 Formularios

- Debe tener un botón "sumbit" para permite el envío.
- Los datos no se refrescan hasta que se hace clic sobre el botón

```
with my_form:
        questions_data = [1,2,3]

        my_form.write("Dentro del form")
        my_form.write(f"## Información del usuario")
        usuario = my_form.text_input('Usuario', 'completar')
        answer = my_form.radio('Cuánto te gusta Streamlit:', questions_data,
index=None)
        submitted = st.form_submit_button("Responder")
```

Creemos por ejemplo de una aplicación con menú sidebar con tres páginas:

- inicio.py
- pages:
  - 01_Tabs y Form.py
  - 02_🌊_Columnas y Callbacks.py

En cada página crear tabs con cada item.

```
inicio.py
import streamlit as st
st.title("Estadísticas realizadas por el grupo xx")
st.write("Se utilizaron las librerías")
```

📈 Widgets:

- Documentación
- Componentes que nos permiten interactuar
- El valor guardado se renueva cuando se refresca, guardan valor de la sesión

- ¿Qué widgets ya vimos?
  - input
  - radio
  - button
  - number_input

- **st.slider()**
- **st.selectbox()**
- **st.checkbox()**

- Implementar lógica condicional en función de los valores de los widgets.

Input

- button("st.button")
- checkbox("st.checkbox")
- number_input("st.number_input")
  - definir rango de valores posibles: min_value = 0, max_value = 120, value = 20, format = "%i"
- slider("st.slider", value=50, min_value=0, max_value=100, step=1)
- select_slider("st.select_slider", options=["uno", "dos", "tres", "cuatro"])
- date_input(("Inserte una fecha:", datetime.date(2000, 7, 6))
- time_input("st.time_input", datetime.time(8,45))
- color_picker("st.color_picker")

- text_input("st.text_input"):
  - podemos dar formato de color a la etiqueta ":blue[Usuario:]"
  - poner un texto ejemplo: placeholder="Nombre de usuario"
- text_area("st.text_area")
- radio("st.radio", options=["AM", "FM", "Online"])
- selectbox("st.selectbox", options=["Lunes", "Martes", "Miércoles", "Jueves", "Viernes"])
- multiselect("st.multiselect", options=["Tomate", "Palta", "Mayo", "Mostaza", "Ketchup"])
- camera_input("st.camera_input")
- file_uploader("st.file_uploader")

Media

- image("https://images.unsplash.com/photo-1548407260-da850faa41e3")
- audio("https://upload.wikimedia.org/wikipedia/commons/b/bb/Test_ogg_mp3_48kbps.wav")
- video("https://www.youtube.com/watch?v=YaEG2aWJnZ8")

Interactuando con los datos

- El widget guarda un valor al interactuar que podemos consultar

```
answer = my_form.radio('Cuánto te gusta Streamlit:', questions_data,
index=None)
submitted = st.form_submit_button("Responder")
if answer:
   st.write(f'Controlo para mostrar. La respuesta de {usuario} es {answer}')
st.write(f'La respuesta de {usuario} es {answer}')
```

⏳ Guardar datos en la sesión

La **sesión** permite almacenar datos hasta que se refreque la página

- genera un diccionario con las variables que queremos guardar
- se reinicia al recargar la página

```
if "shared" not in st.session_state:
    st.session_state["shared"] = True

cambio = st.button('Cambiar valor booleano')
```

```
if cambio:
    st.session_state["shared"] = not st.session_state["shared"]
    st.write(f"Valor {st.session_state.shared} despues de presionar el
boton")
```

Funcionamiento de refresco

- lee nuevamente en orden las variables de forma lineal

🚨 Callbacks

- **widgets** usando on_change:

```
celsius = st.number_input("Celsius: ", key="cel", on_change =
convertir_cel_fah)
```

- **convertir_cel_fah** es una función declarada que modifica el input en *fahrenheit* el valor correspondiente, la opción **key** permite acceder al input para leer el valor ingresado y mostrar el valor calculado.

```
def convertir_cel_fah():
        st.session_state.fah = st.session_state.cel *1.8 +32
```

- **button** usando on_click:

```
button('Verificar tamaño', key="btn_tam", on_click = mostrar,
args=(archivos_list,dirname,))
```

🚨 Desafío

Convertir en pages el archivo start.py en una app multipage

- Generar una page con cada opción de los radio buttons.

In [ ]: