Streamlit

May 31, 2023

1 Seminario de Lenguajes - Python

2 Conociendo Streamlit

Material elaborado con colaboración de Ariadna Aspitia

2.1 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.

2.1.1 ¿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

2.1.2 Instalación

- Se recomienda generar un entorno virtual
- Instalar utilizando pip

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

2.2 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.

2.3 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. Esto quiere decir que tiene la capacidad de renderizar diferentes cosas dependiendo lo que reciba como parámetro. Algunas de sus cualidades son:

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. """)

2.3.1 Permite mostrar diferentes tipos de datos

[ ]:
```
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 tiene la capacidad de renderizar diferentes cosas dependiendo lo que␣
 ↪reciba como parámetro.
    """)
st.subheader('Contenido de variables simples')
x = 10
st.write("El valor de x es:", x)

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

2.3.2 Diccionarios

También puede mostrar el contenido de un diccionario, veamos los perfiles de la aplicación y analicemos dos opciones para guardar estos datos:

[ ]:
```
perfiles = [{
       "nickname": "Twist",
```

```
        "name": "Mark Twain",
        "age": 45,
        "gender": "Masculino",
        "avatar": "bear.png"
    },
    {
        "nickname": "Kalpa",
        "name": "Angelica Gorodischer",
        "age": 95,
        "gender": "Femenino",
        "avatar": "giraffe.png"
    }]
```

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

¿Cuál es la diferencia? ¿Cómo es mejor?

2.3.3 ¿Y los dataframes?

[ ]:
```
import pandas as pd
data = {'Nombre': ['Ada', 'María', 'Julieta'],
        'Edad': [25, 30, 35],
        'Ciudad': ['Jujuy', 'Salta', 'Catamarca']}
df = pd.DataFrame(data)
st.write(df)
```

3 Veamos funciones más avanzadas: 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'):
    ...
```

- Organizar con pages, tabs y columns

3.0.1 Pages:

- Estructura dinámica, se debe crear un directorio **pages** y dentro de la misma archivos con numeración:
  - 01_nombre pagina1.py
  - 02_nombre pagina2.py
  - 03_emoji_nombre pagina3.py
- Se genera un sidebar
- Se puede poner emojis

3.1 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:

```
tab1, tab2 = st.tabs(["Básicos", "Input"])

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

3.2 Columnas

- columnas: divide el espacio según la cantidad de columnas, ahora los elementos que quiero en 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")
    ....
```

Creemos por ejemplo de una aplicación con menú sidebar con tres páginas: * inicio.py * pages: * 01_Estadísticas con gráficos.py * 02_Estadísticas con texto.py

En cada página crear tabs con cada item. * ¿En qué parte del código realizamos el análisis con pandas?

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

```
pages-> 01_Estadísticas con gráficos.py

import streamlit as st
import matplotlib.pyplot as plt
from modulo import archivo
st.title("")
```

```
tab1, tab2 = st.tabs(["Uno", "Dos"])
with tab1:
    st.subheader("Gráfico de torta que muestre los porcentajes según el tipo de imagen")
    figura = modulo.funcion()
    st.pyplot(figura)
    st.write("Explicación sobre el análisis")
```

¿Qué dedebe cumplir **modulo** para poder utilizarlo?

4 Visualización de datos:

- Cargar conjuntos de datos utilizando bibliotecas como pandas.
- Utilizar las funciones de visualización de Streamlit, como **st.line_chart()** o **st.bar_chart()**, para mostrar gráficos básicos.
- Personalizar la visualización de datos mediante opciones adicionales.

Las estadísticas se realizan de la misma forma que se realizan con Python, ¿cómo las podemos mostrar utilizando Streamlit? * Generar el análisis y devolver la figura

[ ]:
```
x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)
fig = plt.figure()
plt.plot(x, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Gráfico de ejemplo')
fig.set_size_inches(5.65, 2.5)
st.write(fig)
```

Utilizar widgets o elementos para decidir cuándo mostrar

[ ]:
```
if st.checkbox('Mostrar dataframe'):
    data =analisis_pandas.mostrar_datos_por('alto')

    data


fig = analisis_pandas.evolucion()

st.pyplot(fig)
```

5 Interacción con widgets:

- ¿Qué widgets hay en Streamlit?
- Utilizar widgets de Streamlit, como **st.slider()**, **st.selectbox()**, o **st.checkbox()**, para permitir la interacción del usuario.

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

5.1 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”)

5.2 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”)

5.3 Interactuando con los datos

Ordenar por el criterio seleccionado

```
data = analisis_pandas.mostrar_datos_por()
columnas = data.columns

opcion = st.selectbox("Opción para ordenar", options=columnas)
if opcion:
    data_ord =analisis_pandas.mostrar_datos_por(opcion)
    data_ord
```

- Streamlit te permite realizarlo desde la aplicación

¿Qué widgets nos serviría para las estadísticas solicitadas? * Gráfico de torta que muestre los porcentajes según el tipo de imagen * ¿Cómo elegir el tipo de gráfico: barra- torta?

- Y si queremos sacar tipos de imágenes para redibujar el gráfico cómo haríamos? Probemos con Plotly.

```
import plotly.express as px
fig = px.pie(tipo_counts, values=tipo_counts.values, names=tipo_counts.index)
```

- Gráfico de nube de palabras:
  - de tags:

```
def genero_nube(texto):
    wordcloud = WordCloud(width=800, height=400).generate(texto)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    return plt
```

- de textos que contienen oraciones, ¿qué sucede? ¿cuáles puede ser las palabras que queden como más utilizadas? python tokens = word_tokenize(texto) stopwords_list = stopwords.words('spanish') filtered_tokens = [word for word in tokens if word.lower() not in stopwords_list and len(word) > 2] filtered_text = ' '.join(filtered_tokens) wordcloud = WordCloud(width=800, height=400).generate(filtered_text) fig = plt.figure(figsize=(10, 5)) plt.imshow(wordcloud, interpolation='bilinear') plt.axis('off') NLTK es una librería para el manejo de texto natural, se puede utilizar para analizar gramática, sintáxis, en este caso lo utilizaremos para eliminar palabras establecidas como no significativas y por otro lado eliminamos las que son más cortas que dos letras.

[ ]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

[ ]:
```
texto = '''Yo cuando intento seguir una receta de cocina Cuando alguien dice␣
↪que no le
  gusta el chocolate Yo cuando me dan más tareas justo antes del fin de␣
↪semana Yo cuando
  intento bailar Cuando tienes una idea brillante pero te das cuenta de que␣
↪ya se le
  ocurrió a alguien más Yo cuando intento bailar Yo después de una semana de␣
↪dieta
  Cuando ves el pronóstico del tiempo y llueve todo el fin de semana Yo␣
↪cuando intento
  seguir una receta de cocina Cuando encuentras comida en el refrigerador a␣
↪medianoche
  Cuando intentas mantener la calma pero todo sale mal Yo después de una␣
↪semana de dieta
  Cuando finalmente llega el viernes Yo cuando veo un perro en la calle␣
↪Cuando intentas
  mantener la calma pero todo sale mal Yo cuando veo un perro en la calle Yo␣
↪cuando alguien
    me despierta antes de tiempo Cuando ves a alguien hablando por teléfono␣
↪en el cine
```

```
      Yo cuando alguien me despierta antes de tiempo Cuando intentas programar␣
 ↪sin café Y
      o cuando veo un perro en la calle Yo cuando llega el postre'''
tokens = word_tokenize(texto)
stopwords_list = stopwords.words('spanish')
filtered_tokens = [word for word in tokens if word.lower() not in␣
 ↪stopwords_list and len(word) > 2]
filtered_text = ' '.join(filtered_tokens)
wordcloud = WordCloud(width=800, height=400).generate(filtered_text)
fig = plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

6 Guardar datos en la sesión

- 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")
```

Accedo a las variables de la sesión: * st.session_state[“shared”] * st.session_state.shared

7 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,))
```