1. Pre-requisitos

1. Asegúrese de tener instalado Python en versión 3.7 o superior.
2. Para este taller usaremos Python y en particular las librerías *dash*, *plotly* y *pandas*.
3. Puede instalar las librerías con el comando

   ```
   pip install dash plotly pandas
   ```

   o si prefiere repositorios de anaconda use

   ```
   conda install dash plotly pandas
   ```

   También puede usar el administrador de paquetes de Anaconda.
4. La entrega de este taller consiste en un **reporte** y unos **archivos de soporte**. Cree el archivo de su **reporte** como un documento de texto en el que pueda fácilmente incorporar capturas de pantalla, textos y similares. Puede ser un archivo de word, libre office, markdown, entre otros.
5. En su **reporte** incluya las respuestas solicitadas en el taller, marcando la sección y el numeral al que se refieren.

2. Un primer tablero en Dash

Dash es un framework para el desarrollo de aplicaciones web en python, desarrollado por Plotly y construido sobre Flask. En particular, Dash ofrece facilidades para crear tableros para la visualización de datos interactivos y dinámicos que proveen una buena interfaz al usuario del tablero.

Dash interactúa además con Pandas, la librería estándar para la manipulación de datos en python. Esto permite fácilmente cargar datos desde una fuente externa y generar visualizaciones de gran calidad.

Al ser un framework web, el resultado es una página web que puede desplegarse usando navegadores estándar, ya sea localmente o en un servidor.

En este taller buscamos conocer las facilidades de Dash a través de algunos ejemplos, empezando por el más sencillo y aumentando paulatinamente su nivel de complejidad.

1. Abra el archivo `app1.py` en un editor de python (spyder, visual studio code, etc). Para abrir los archivos de base de este taller evite usar cuadernos de jupyter.
2. Note que en la parte final del archivo se define qué hacer cuando se ejecuta el archivo. En este caso se llama al objeto `app` y se ejecuta su método `run_server`. El objeto `app` se crea en las primeras líneas del script. En su **reporte** incluya el código con el que se crea el objeto `app`.

3. Después de crear el objeto `app` (y extraer el atributo `server`) el programa define un dataframe de pandas en el que se almacena un conjunto de datos. En su **reporte** describa brevemente el conjunto de datos. Use un atributo o método del dataframe para imprimir en consola los nombres de las columnas que componen el conjunto de datos.

4. Ejecute la aplicación usando su editor o ejecutando en consola

   ```
   python app1.py
   ```

5. En su navegador (Chrome, Firefox, Safari, Edge) visite el servidor local usando el puerto 8050, es decir, vaya a la página

   ```
   http://127.0.0.1:8050/
   ```

   En su **reporte** incluya un pantallazo donde aparezca el código en ejecución y la página cargada, lado a lado.

6. Considerando la aplicación resultante, describa en su **reporte** qué hace el comando

   ```
   fig = px.bar(df, x="Fiebre", y="Casos", color="Diagnostico",
       barmode="group")
   ```

   También puede apoyarse en la documentación de plotly:

   - `https://plotly.com/python-api-reference/generated/plotly.express.bar.html`
   - `https://plotly.com/python/bar-charts/`

7. La siguiente (y última sección de código) crea el documento html que es renderizado por el navegador. Note que se crea definiendo secciones (Div) y título (H1), entre otros elementos html. En su reporte describa brevemente qué hacen las líneas

   ```
   dcc.Graph(
     id='example-graph',
     figure=fig
   ),
   ```

   y

   ```
   html.Div(
     className="Columnas",
     children=[
       html.Ul(id='my-list', children=[html.Li(i) for i in df.columns
           ])
     ],
   )
   ```

8. Realice cambios en los datos: modifique los nombres de las columnas y los valores que toman. Realice todos los cambios necesarios hasta obtener una nueva versión del tablero con datos de su elección. Incluya su archivo modificado como parte de la entrega. Incluya un pantallazo de la aplicación en ejecución en su **reporte**.

3. Callbacks

Un mecanismo muy útil de Dash es el uso de callbacks para modificar dinámicamente elementos del tablero. Veamos un ejemplo sencillo de callbacks.

1. Abra el archivo `app2.py` en un editor de python.
2. Note que el código tiene muchas similitudes con el anterior, incluyendo el main y la definición de los objetos `app` y `server`.
3. Describa en su **reporte** el layout de esta aplicación, con sus elementos y el ID de cada elemento.
4. La gran diferencia de esta nueva aplicación radica en la función

   ```
   def update_output_div(input_value):
     return 'Output: {}'.format(input_value)
   ```

   Note que a esta función se le agrega un *decorador*, el cual extiende la función asociando sus entradas y salidas con elementos del documento HTML

   ```
   @app.callback(
     Output(component_id='my-output', component_property='children'),
     [Input(component_id='my-input', component_property='value')]
   )
   ```

5. En su **reporte** describa qué hace la función `update_output_div`.
6. Estudie el decorador y describa en su **reporte** describa qué hace, especialmente qué elementos del documento HTML asocia con la función `update_output_div`.
7. Modifique la frase que retorna la función, por ejemplo incluyendo un texto personalizado, además del texto que ingresa el usuario. Incluya el código modificado como parte de su entrega. Incluya un pantallazo de la aplicación modificada en ejecución en su **reporte**.

4. Más visualizaciones e interacciones

Hasta el momento hemos visto un tipo de visualización muy sencilla. Ahora consideramos una visualización más elaborada.

1. Abra el archivo `app3.py` en un editor de python.
2. Note que el código tiene muchas similitudes con el anterior, no solo en el main, app y server, sino también en la definición de layout y una función con callback.
3. Ejecute la aplicación.
4. Describa en su **reporte** el layout de esta aplicación. ¿Qué elementos tiene?

5. En su **reporte** describa qué hace la función `update_figure`.

6. Describa en su **reporte** qué hace el decorador, especialmente qué elementos del documento HTML asocia con la función `update_figure`.

7. Note que en este caso los datos están en formato CSV, provienen de una URL y se cargan usando el método read_csv de pandas.

8. Cree otro archivo, ya sea de python (.py) o un cuaderno de jupyter (.ipynb) y úselo para explorar los datos cargados. Emplee métodos y atributos de los dataframes en pandas, como

   head() (recuerde que el objeto en este caso se llama `df`, luego el método se aplica sobre el objeto como `df.head()`)

   size

   ndim

   columns

   tail()

   info()

   shape (en este caso nos referimos a un atributo del objeto `df`, luego el llamado es `df.shape`)

   describe()

   sample()

   isnull().sum() (determina cuáles entradas son nulos y suma por columnas)

   nunique() (número de valores únicos)

   value_counts()

   corr()

   En este momento obtener una hoja de resumen de funciones en python para ciencia de datos puede ser útil. Considere por ejemplo ésta: `https://www.utc.fr/~jlaforet/Suppl/python-cheatsheets.pdf`.

9. Para terminar, y aprovechando su conocimiento de los datos, realice *una nueva visualización* de interés. Considere las opciones de gráficas que ofrece plotly.express `https://plotly.com/python/plotly-express/`. También puede ser de interés la Coda a esta sección. Incluya títulos y textos que describan las visualizaciones incluidas.

10. Incluya el código modificado como parte de su entrega. Incluya un pantallazo de la aplicación modificada en ejecución en su **reporte**.

11. **Coda**: a manera de ejemplo adicional incluimos el archivo `app4.py` donde podrá ver elementos adicionales y una actualización un poco más elaborada de una gráfica y sus elementos usando callbacks.