Introducción a Dash

Dash

- Dash es un framework de Python para la creación de aplicaciones web analíticas.
- Con las siguientes características:
  - Ligero: Las aplicaciones Dash requieren muy poco tiempo para empezar y son extremadamente ligeras en de Python puro.
  - Control directo: Dash proporciona una interfaz sencilla para vincular controles de interfaz de usuario, como controles deslizantes, desplegables y gráficos, con el código de análisis de datos de Python.
  - Completamente personalizable: Cada elemento estético de una aplicación Dash es personalizable. Las aplicaciones Dash se crean y se publican en la Web, por lo que está disponible toda la potencia de CSS y HTML.

Ejemplos

https://dash-gallery.plotly.host/Portal/

Instalación

- Para instalar dash:

```
pip install dash
```

- Nos instala las siguientes librerías: dash_html_components, dash_core_components, dash_table, plotly
- Ejemplos de gráficos interactivos: https://plotly.com/python/plotly-express/

Dash layout

- Las aplicaciones dash se componen de dos partes:
  - "layout": describe como la aplicación se distribuye y su aspecto estético.
  - "callbacks": da la interactividad a la aplicacción; lo veremos en la sigueinte parte.
- Dash tiene clases de python para todos los componentes visuales, estos se encuentran en las librerias dash_core_components y dash_html_components.

Ejemplo 1

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.express as px
import pandas as pd

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__,
external_stylesheets=external_stylesheets)

df = pd.DataFrame({
    "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
    "Amount": [4, 1, 2, 2, 4, 5],
    "City": ["SF", "SF", "SF", "Montreal", "Montreal", "Montreal"]
})

fig = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

app.layout = html.Div(children=[
    html.H1(children='Hello Dash'),

    html.Div(children='''
        Dash: A web application framework for Python.
    '''),

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

if __name__ == '__main__':
    app.run_server(debug=True)
```

```
app.layout = html.Div(children=[
    html.H1(children='Hello Dash'),

    html.Div(children='''
        Dash: A web application framework for Python.
    '''),

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

- El layout está compuesto de 3 componentes:
- La librería dash_html_components tiene un componente para cada tag HTML.
- `html.H1(children='Hello Dash')` genera un código html: `html <h1>Hello Dash</h1>`

- Los componentes y su comportamiento se describen usando keyarguments.
- El argumento children puede ponerse o no, puede contener una lista, un string, un único componente.
- `html.H1(children='Hello Dash')` es lo mismo que `html.H1('Hello Dash')`.

- Para que las fuentes y el estilo de la aplicación sean más bonitos podemos usar CSS. En este caso tomamos uno de dash:

```
external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
```

to get the same look and feel of these examples.

- Ejecutamos nuestra primera aplicación con:

```
python example_1.py
```

- Podemos ver la app abriendo un navegador en http://127.0.0.1:8050/

Ejercicio

- Ejecuta la aplicacion `example_1.py`.

- Puedes hacer cambios y verlos en directo si se inicializa la app con:

```
app.run_server(debug=True)
```

- Hace que con los cambios se refresque automáticamente el navegador.

Ejercicio

- Prueba a cambiar el título y la x por la y en el gráfico.

- La librería dash_html_components contiene un componente para cada tag HTML.
- El estilo de estos se puede modificar usando el agumento style.
- En `example_2.py` puedes ver una versión modificada de la app anterior.
- Modificamos el estilo en línea:

```
html.H1('Hello Dash', style={'textAlign': 'center', 'color': '#7FDBFF'}) is rendered in the Dash application as <h1 style="text-align: center; color: #7FDBFF">Hello Dash</h1>.
```

- Existen algunas diferencias entre los dash_html_components y la forma en la que los declaramos en html:
  - La propiedad de estilo se definen como un diccionario, no separado por dos puntos.
  - Las claves son camelCased. Por ejemplo: text-align es textAlign
  - El atributo HTML class es className en dash.
  - Los children se especifican con el keyword children.

Componentes resusables.

- Al escribir código en dash generamos muchas veces ficheros muy complejos
- Podemos crear componentes reusables.
- En `example_3.py` tenemos una tabla generada a partir de un dataframe.

Visualizaciones

- La librería dash_core_components incluye el componente Graph.
- Graph renderiza visualizaciones interactivas usando la libreria opensource plotly.js.
- Plotly Tiene 35 tipos distintos de visualizaciones.
- En https://plotly.com/python/ puedes ver ejemplos.

- La forma más simple de generar gráficos con plotly es usando plotly express.
- plotly express es la librería simplificada de plotly.

```
import plotly.express as px
```

- Podemos definir un gráfico de la siguiente manera:

```
fig = px.scatter(df, x="gdp per capita", y="life expectancy",
                 size="population", color="continent", hover_name="country",
                 log_x=True, size_max=60)
```

- Para incluirlo en la aplicación como:

```
app.layout = html.Div([
    dcc.Graph(
        id='life-exp-vs-gdp',
        figure=fig
    )
])
```

- Los gráficos que no son de plotly express se incorporan usando la librería:

```
import plotly.graph_objects as go
```

- Podemos generar gráficos de velas:

```
fig = go.Figure(
        go.Candlestick(
        x=data_to_plot.index,
        open=data_to_plot['open'],
        high=data_to_plot['high'],
        low=data_to_plot['low'],
        close=data_to_plot['close']
    ))
```

Ejercicio:

- Ejecuta el ejemplo 4.

Markdown

- Dash nos da todos los componentes de html en su librería dash_html_components.
- Puede ser tedioso tener que codificar todo esto en python.
- Para escribir texto podemos usar markdown:

```
app.layout = html.Div([
    dcc.Markdown(children=markdown_text)
])
```

- Puedes ver un ejemplo en: `example_5.py`

Core Components

- Los dash_core_components incluyen un conjunto de componentes de alto nivel como son dropdowns, graphs, markdown blocks, sliders, etc.
- Los compoenetes core los podemos encontrar en: https://dash.plotly.com/dash-core-components
- Todos ellos son descritos de forma declarativa.
- En el `example_6.py` podemos ver algunos de ellos y la forma en la que se declaran.

Conclusiones

- El layout describe como se muestra nuestra aplicación.
- El layout en un árbol jerárquico de componentes.
- La librería dash_html_components nos proporciona todas las clases de los tag HTML.
- La librería dash_core_components nos proporciona componentes de control y gráficos.

Referencia:

- dash_core_components gallery: https://dash.plotly.com/dash-core-components
- dash_html_components gallery: https://dash.plotly.com/dash-html-components