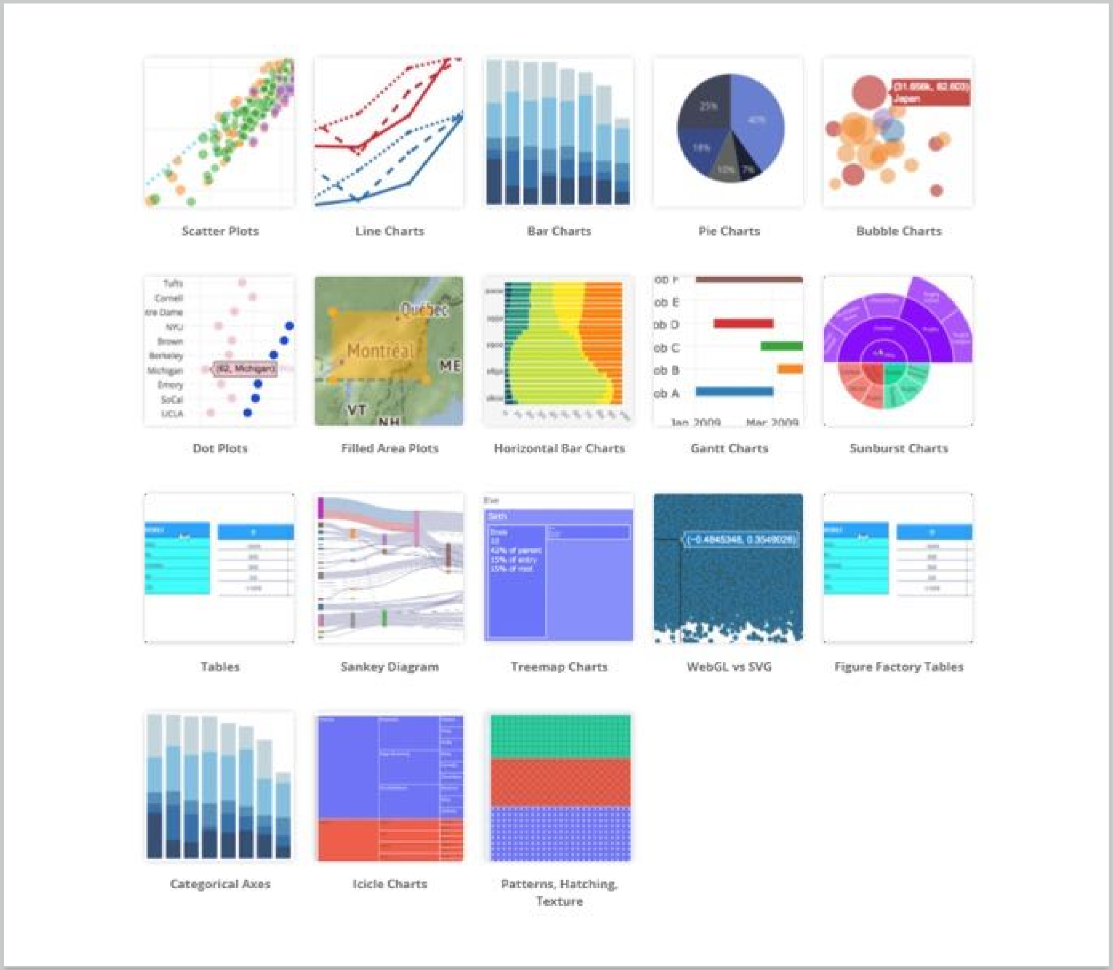
plotly

https://plotly.com

Python Plotly

About Plotly:

- **Plotly** is a Montreal based technical computing company
- Plotly products: tools for data visualization such as **Plotly**, **Dash** and **Chart Studio** (web service).
- Plotly is an open-source Javascript library allowing creation of ineractive graphs in browser windows
- Plotly API exist for different programming languages (Python, R, Julia, MATLAB, .NET, C#, etc.)
- Plotly Javascript receives plot data and configuration in JSON format
- Graphs can also be exported in various raster formats as well as vector image formats
- Most commonly Plotly is used in Jupyter notebooks or web pages.



Installation of Python package

https://plotly.com/python/getting-started/

```
pip install plotly
# or
conda install -c plotly plotly
```

There are three main sub-modules in Plotly:

- **plotly.plotly** – communication with server
- **plotly.graph_objects** – definitions of graph objects that make up the plots (Figure, Data, Layout, Scatter, Box, Histogram etc., 3D plots, etc.).
  All graph objects are dictionary- and list-like objects used to generate and/or modify Plotly plots.
- **plotly.tools**

The **plotly.tools** module:

.. functions for subplot generation

.. embedding plots in IPython notebooks

.. saving and retrieving your credentials

A plot is represented by Figure object

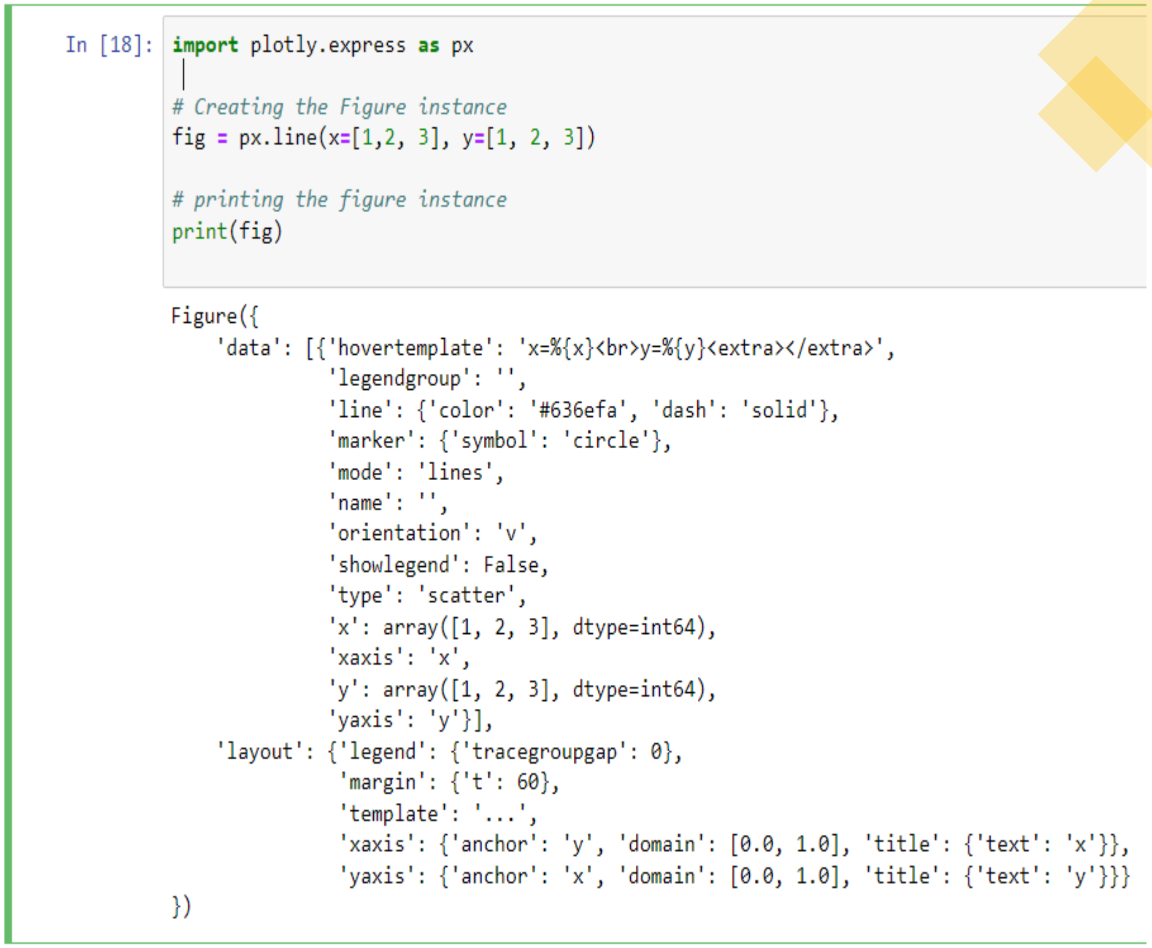
as defined in **plotly.graph_objs** module

Figure is serialized as JSON before it gets passed to plotly.js

Example of JSON representation of a Figure object:

Note:

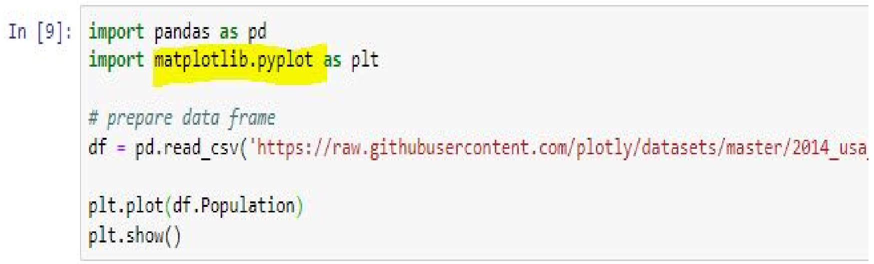
plotly.express sub-module can create the entire Figure in few lines of code. It uses the graph_objects internally.



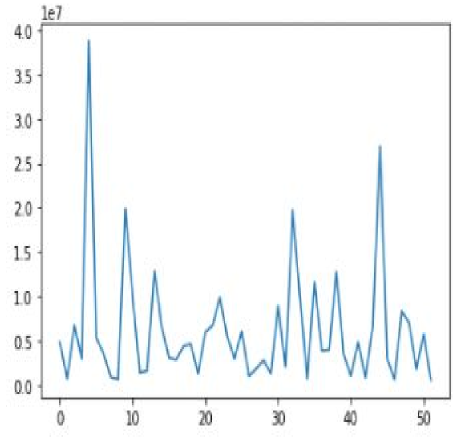
```
In [18]: import plotly.express as px

         # Creating the Figure instance
         fig = px.line(x=[1,2, 3], y=[1, 2, 3])

         # printing the figure instance
         print(fig)
```

```
Figure({
    'data': [{'hovertemplate': 'x=%{x}<br>y=%{y}<extra></extra>',
              'legendgroup': '',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array([1, 2, 3], dtype=int64),
              'xaxis': 'x',
              'y': array([1, 2, 3], dtype=int64),
              'yaxis': 'y'}],
    'layout': {'legend': {'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'x'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'y'}}}
})
```

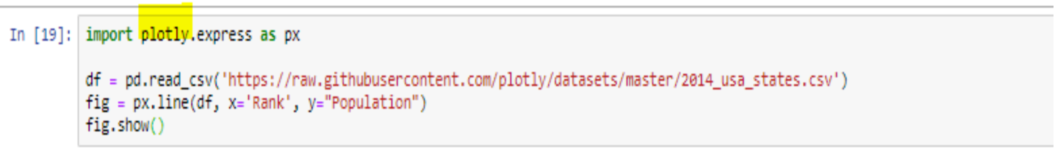
Compare static Matplotlib plot with Interactive Plotly



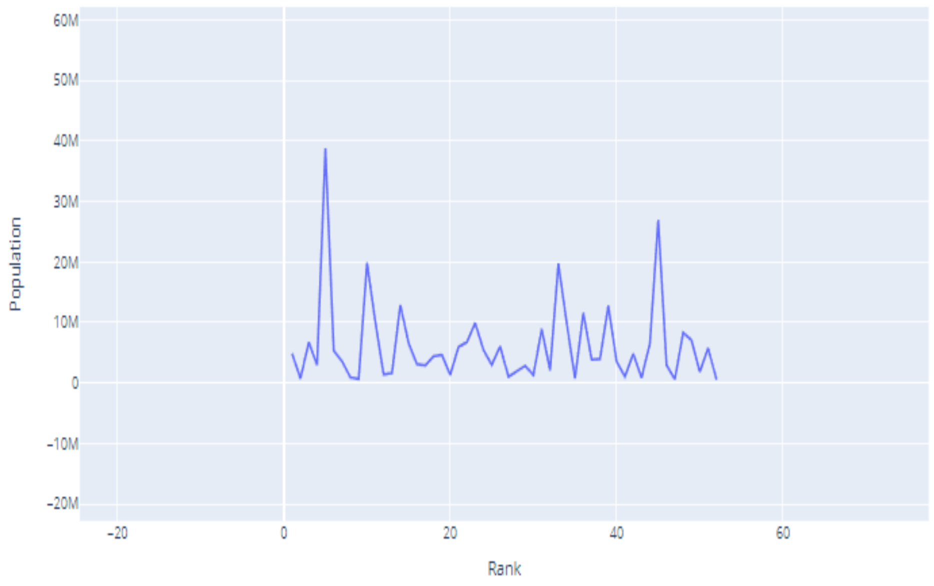
```
In [9]: import pandas as pd
        import matplotlib.pyplot as plt

        # prepare data frame
        df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2014_usa

        plt.plot(df.Population)
        plt.show()
```



Static

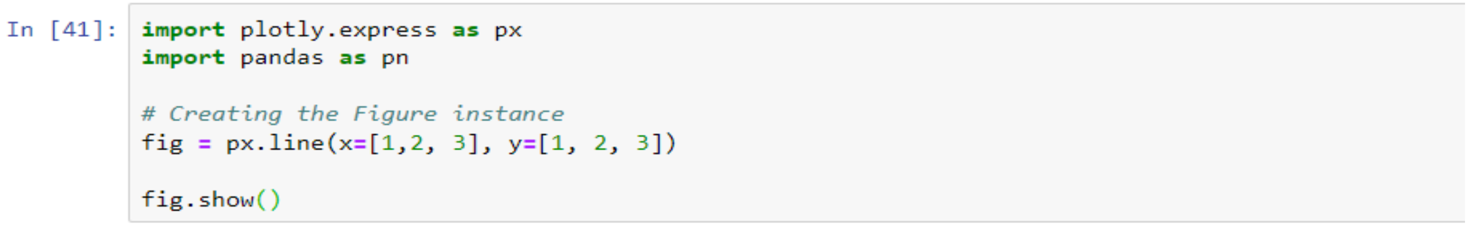


```
In [19]: import plotly.express as px

         df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2014_usa_states.csv')
         fig = px.line(df, x='Rank', y="Population")
         fig.show()
```

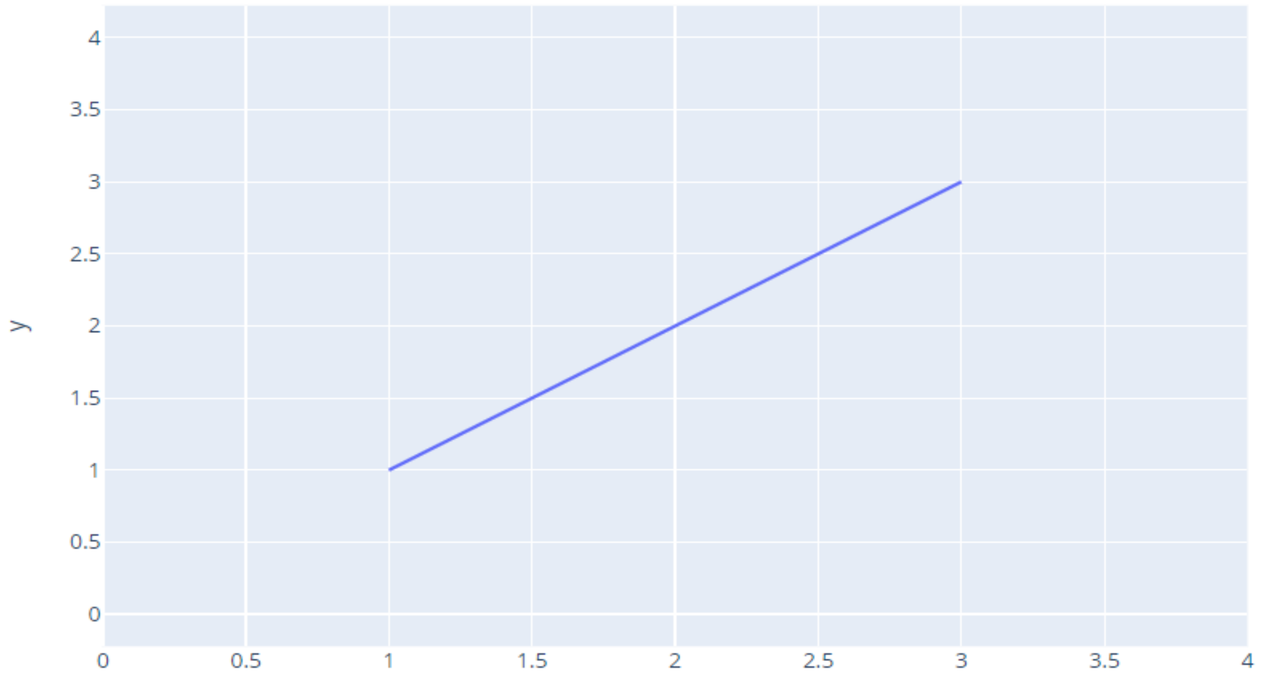


Interactive and modern presentation

Plotly.express examples (quick way to create power graphs with just a few attributes) - start

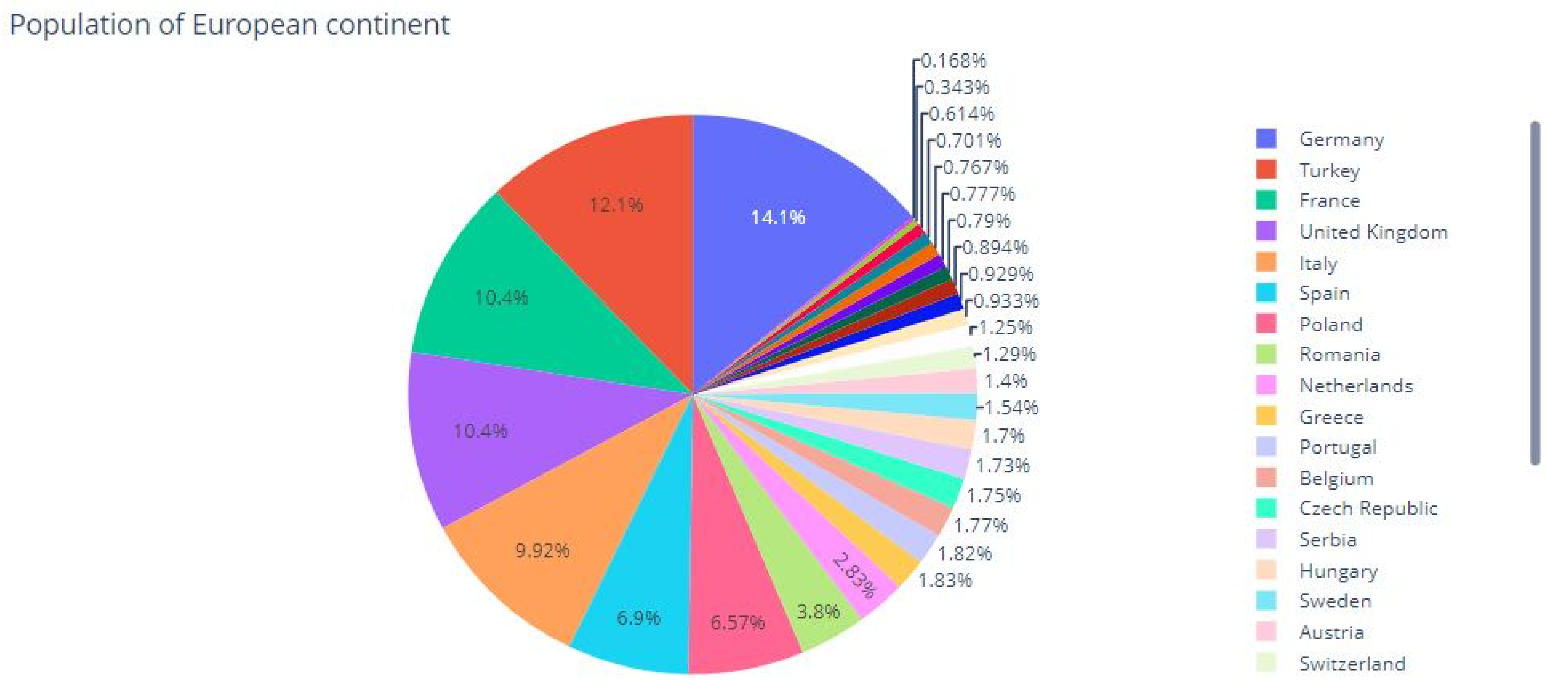


```
In [41]: import plotly.express as px
         import pandas as pn

         # Creating the Figure instance
         fig = px.line(x=[1,2, 3], y=[1, 2, 3])

         fig.show()
```



Plotly Pie

```
In [46]: import plotly.express as px
         import pandas as pd

         df = px.data.gapminder().query("year == 2007").query("continent == 'Europe'")
         df.loc[df['pop'] < 2.e6, 'country'] = 'Other countries' # Represent only large countries
         fig = px.pie(df, values='pop', names='country', title='Population of European continent')
         fig.show()
```

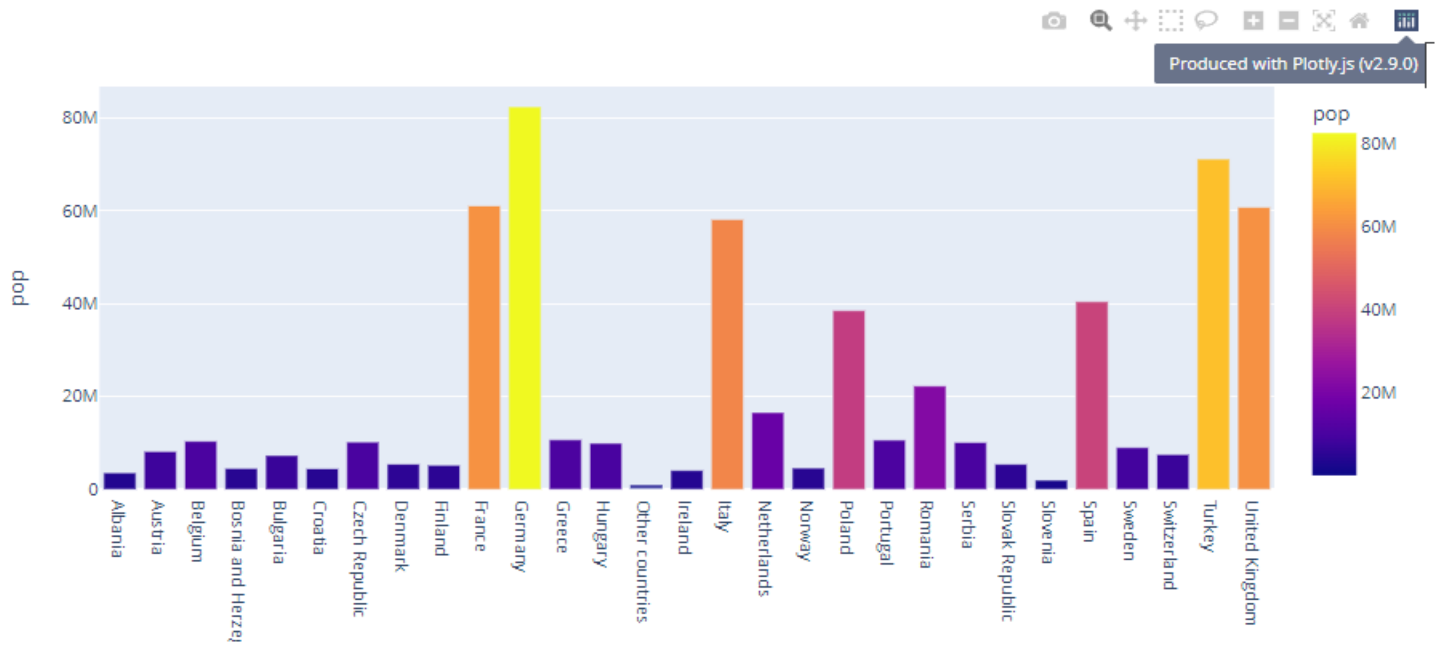


Plotly Bar

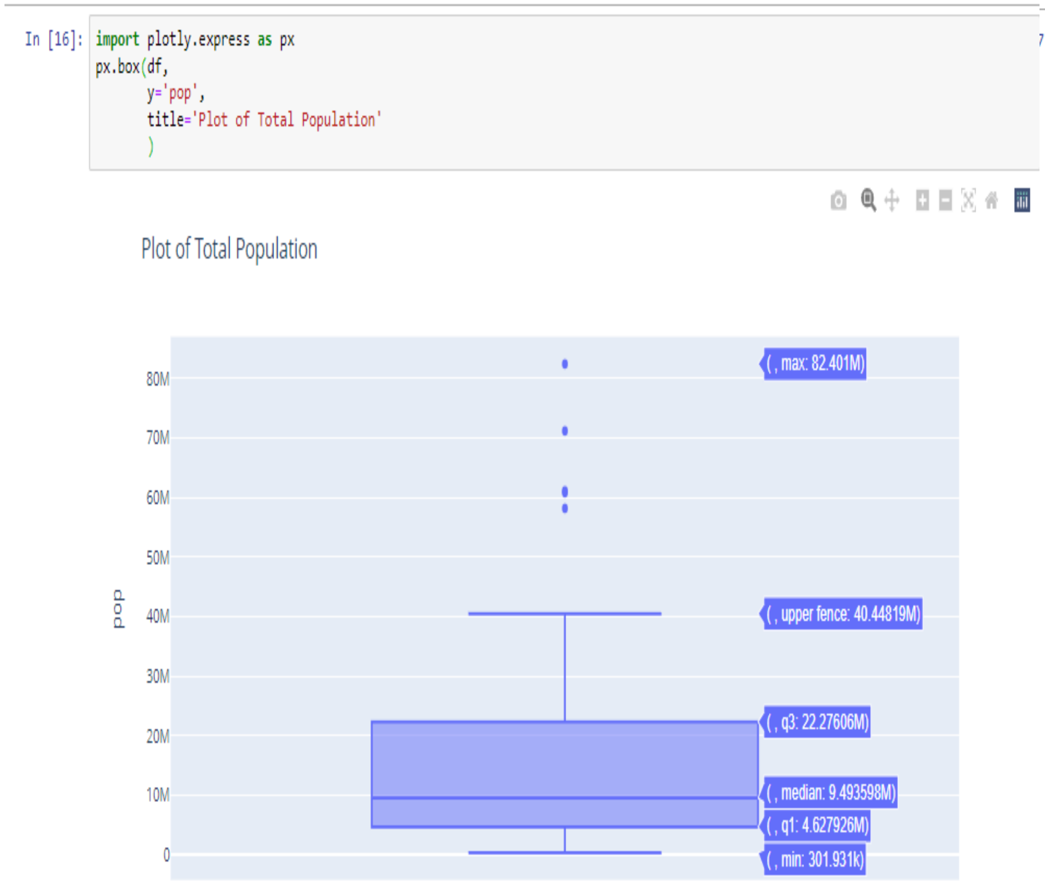
```
Out[60]:
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
23	Albania	Europe	2007	76.423	3600523	5937.029526	ALB	8
83	Austria	Europe	2007	79.829	8199783	36126.492700	AUT	40
119	Belgium	Europe	2007	79.441	10392226	33692.605080	BEL	56
155	Bosnia and Herzegovina	Europe	2007	74.852	4552198	7446.298803	BIH	70
191	Bulgaria	Europe	2007	73.005	7322858	10680.792820	BGR	100

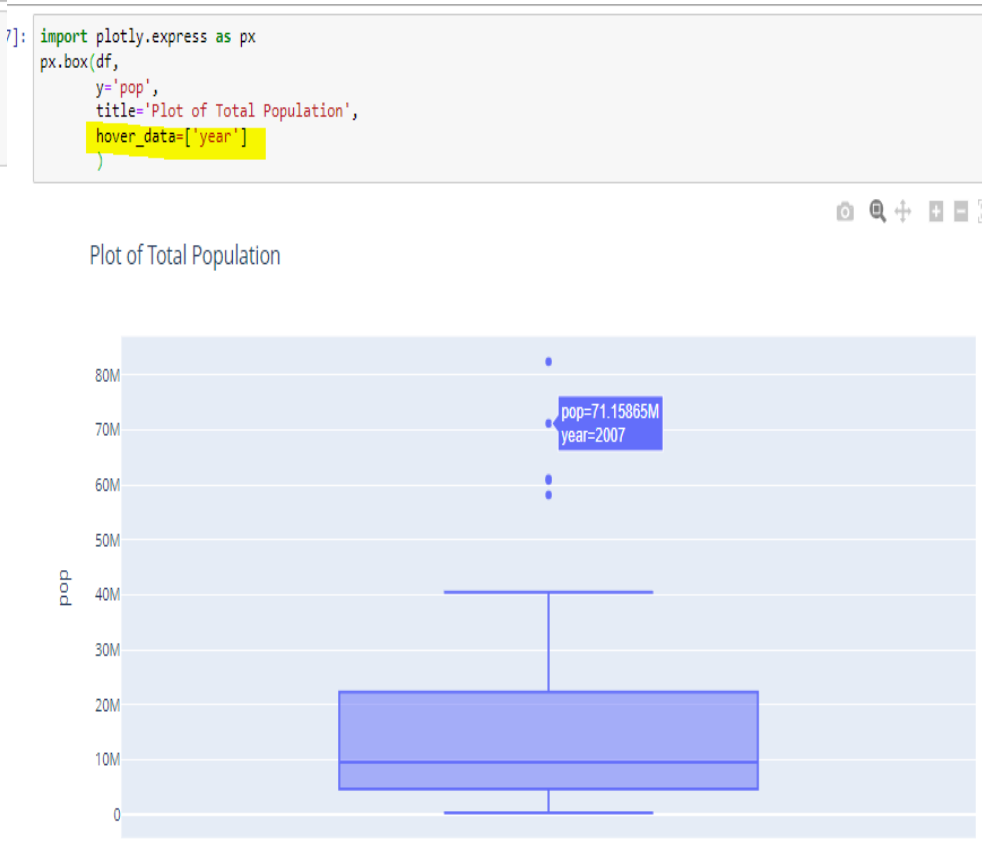
```
In [59]: import plotly.express as px
         import pandas as pd
         px.bar(df, x='country', y='pop', color='pop')
```



Plotly Box with hover



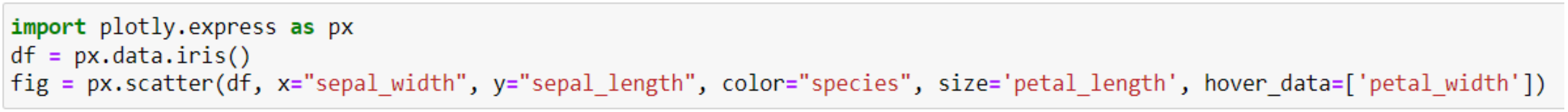
```
In [16]: import plotly.express as px
         px.box(df,
                y='pop',
                title='Plot of Total Population'
               )
```



```
7]: import plotly.express as px
    px.box(df,
           y='pop',
           title='Plot of Total Population',
           hover_data=['year']
          )
```

Plotly Express vs Plotly Graph Objects

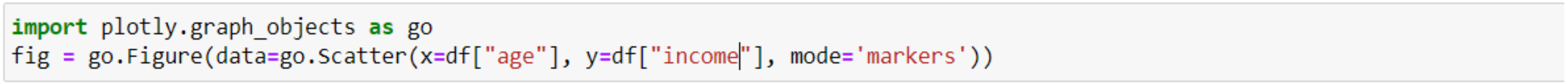
plotly.express it's a quick way to create power graphs with just a few attributes.
For more advanced graphs with Plotly Express High-level Interface see documentation
https://plotly.com/python-api-reference/plotly.express.html)



```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species", size='petal_length', hover_data=['petal_width'])
```

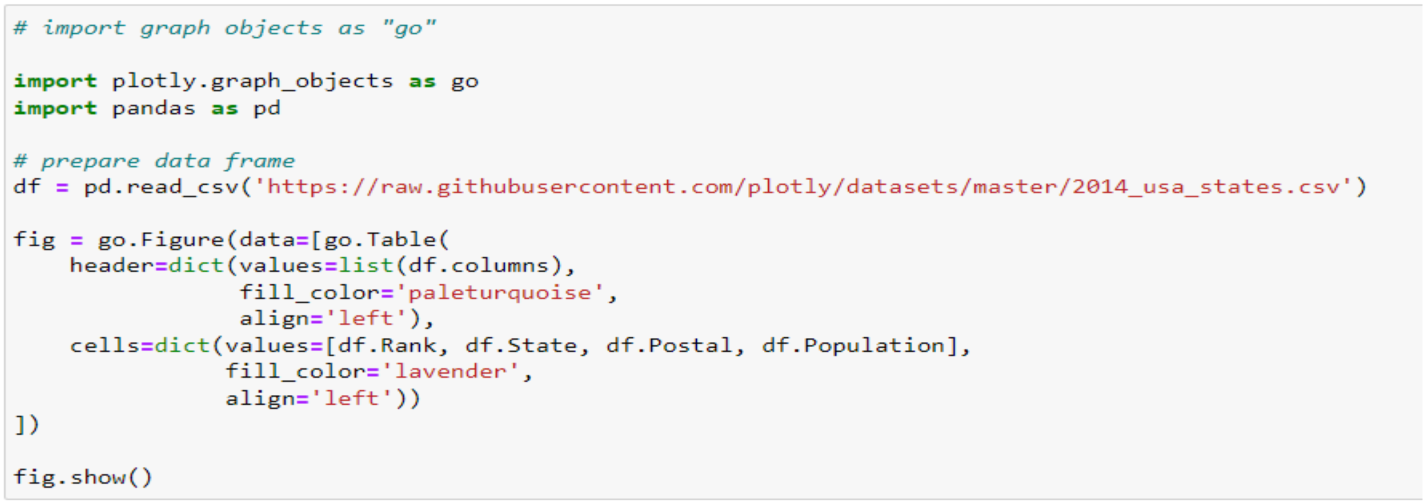
plotly.graph_objects you have to build everything from the bottom up and define attributes: data, layout, frame , etc.
For customization use Plotly Graph_Objects documentation for figure updates:
https://plotly.com/python/reference/index/

- **Layout** - represents the chart (frames, title, color, tick, hover, legend)
- **Traces** - represent the data (inside the layout)

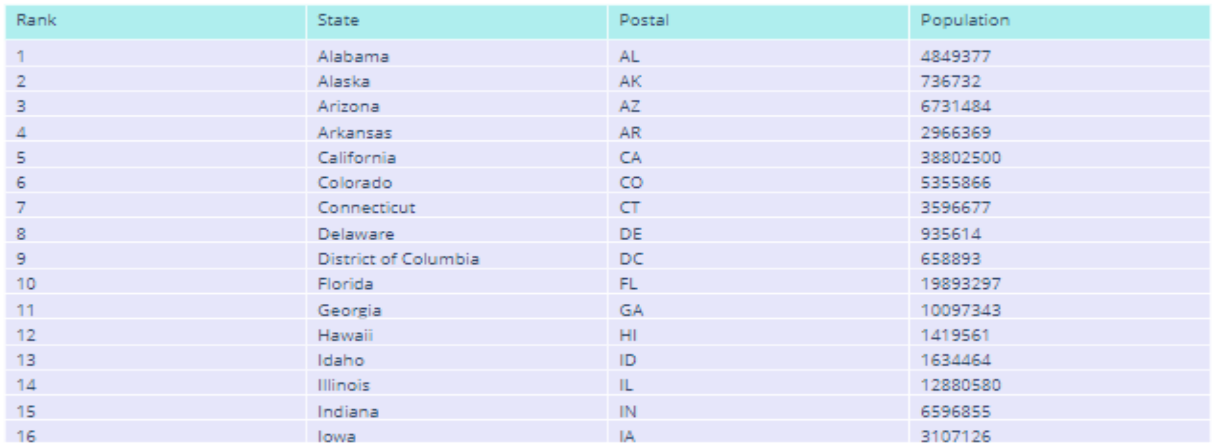


```
import plotly.graph_objects as go
fig = go.Figure(data=go.Scatter(x=df["age"], y=df["income"], mode='markers'))
```

Plotly Table

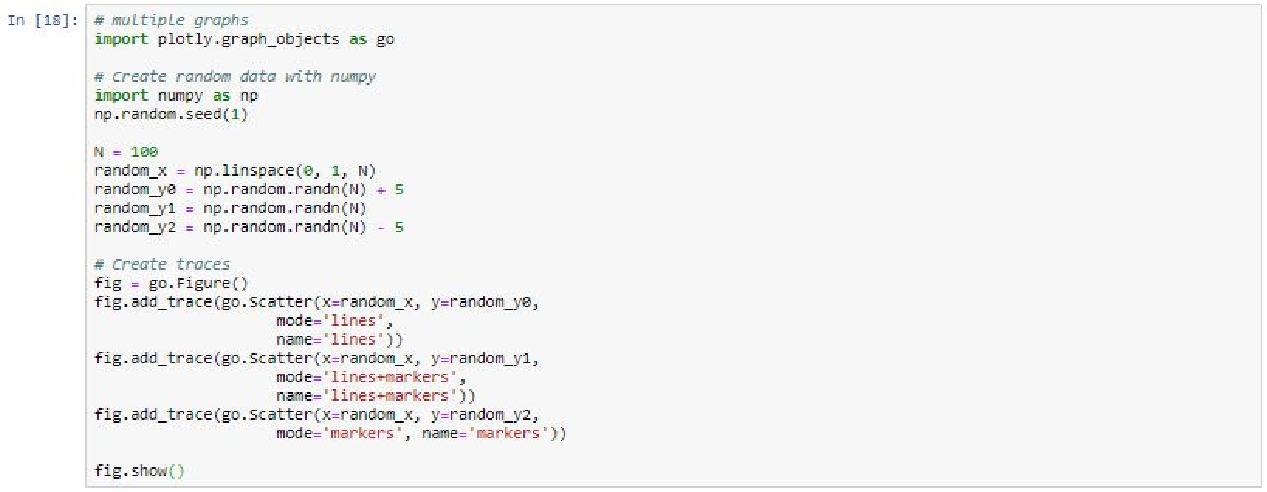


```python
# import graph objects as "go"

import plotly.graph_objects as go
import pandas as pd

# prepare data frame
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2014_usa_states.csv')

fig = go.Figure(data=[go.Table(
    header=dict(values=list(df.columns),
                fill_color='paleturquoise',
                align='left'),
    cells=dict(values=[df.Rank, df.State, df.Postal, df.Population],
               fill_color='lavender',
               align='left'))
])

fig.show()
```

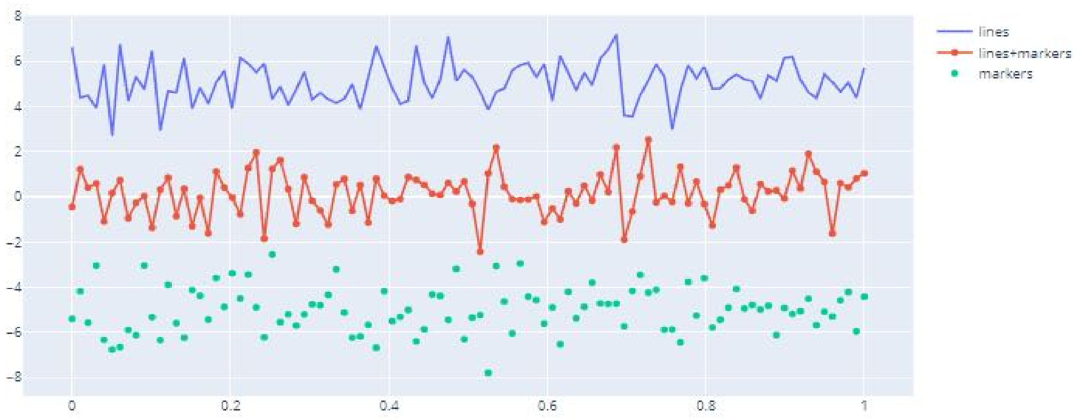


Rank	State	Postal	Population
1	Alabama	AL	4849377
2	Alaska	AK	736732
3	Arizona	AZ	6731484
4	Arkansas	AR	2966369
5	California	CA	38802500
6	Colorado	CO	5355866
7	Connecticut	CT	3596677
8	Delaware	DE	935614
9	District of Columbia	DC	658893
10	Florida	FL	19893297
11	Georgia	GA	10097343
12	Hawaii	HI	1419561
13	Idaho	ID	1634464
14	Illinois	IL	12880580
15	Indiana	IN	6596855
16	Iowa	IA	3107126

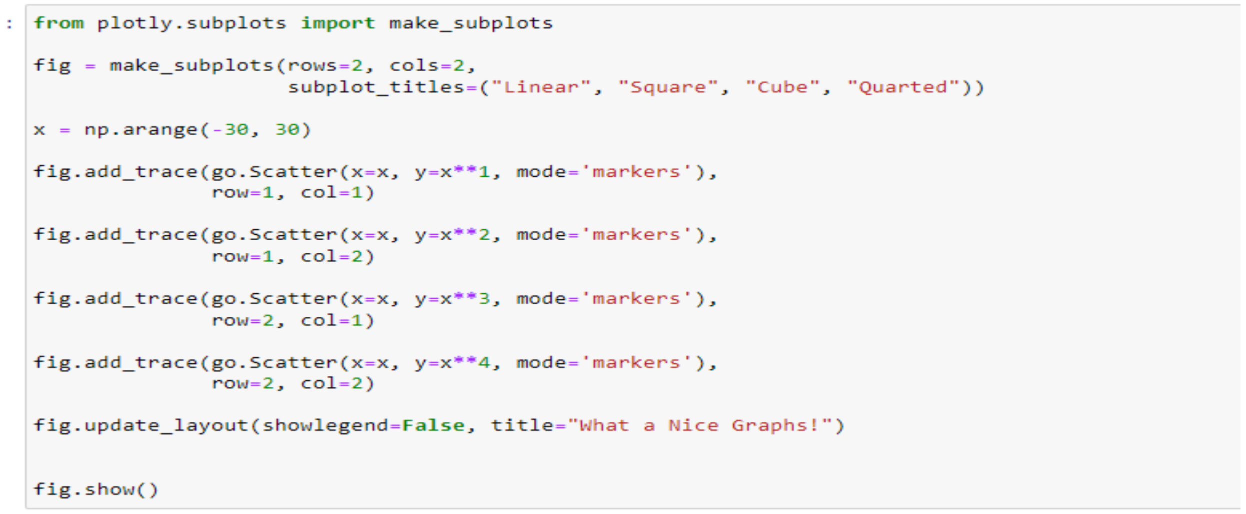
Multiple graphs on the same Chart with **plotly.graph.objects**



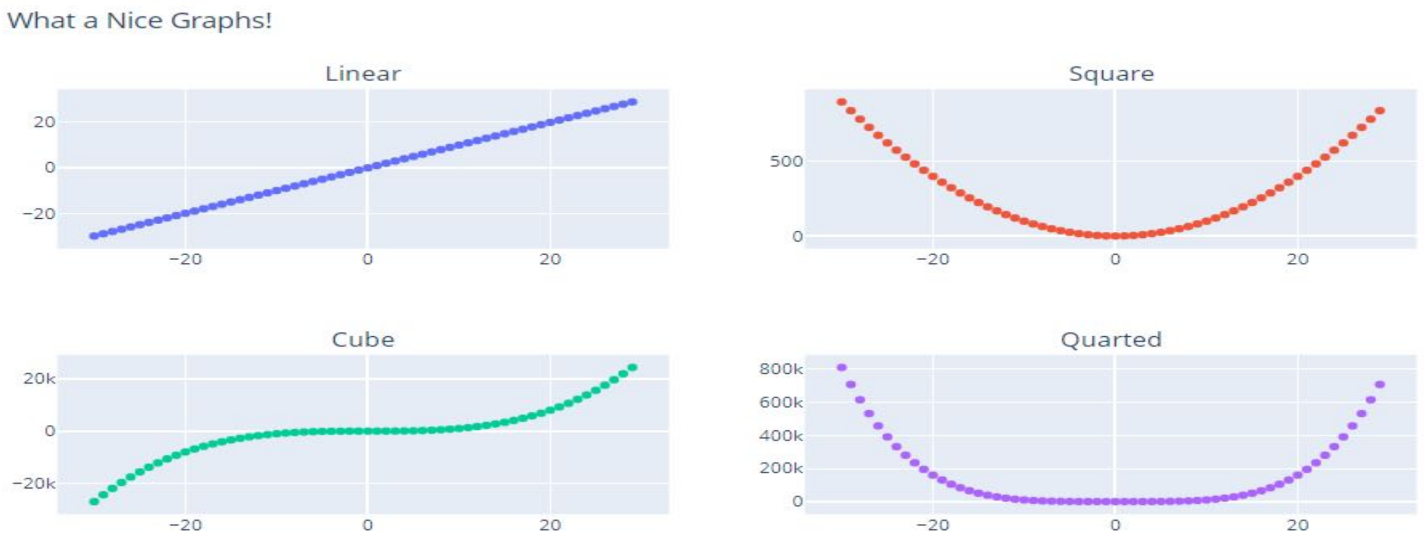
```
In [18]: # multiple graphs
         import plotly.graph_objects as go

         # Create random data with numpy
         import numpy as np
         np.random.seed(1)

         N = 100
         random_x = np.linspace(0, 1, N)
         random_y0 = np.random.randn(N) + 5
         random_y1 = np.random.randn(N)
         random_y2 = np.random.randn(N) - 5

         # Create traces
         fig = go.Figure()
         fig.add_trace(go.Scatter(x=random_x, y=random_y0,
                             mode='lines',
                             name='lines'))
         fig.add_trace(go.Scatter(x=random_x, y=random_y1,
                             mode='lines+markers',
                             name='lines+markers'))
         fig.add_trace(go.Scatter(x=random_x, y=random_y2,
                             mode='markers', name='markers'))

         fig.show()
```



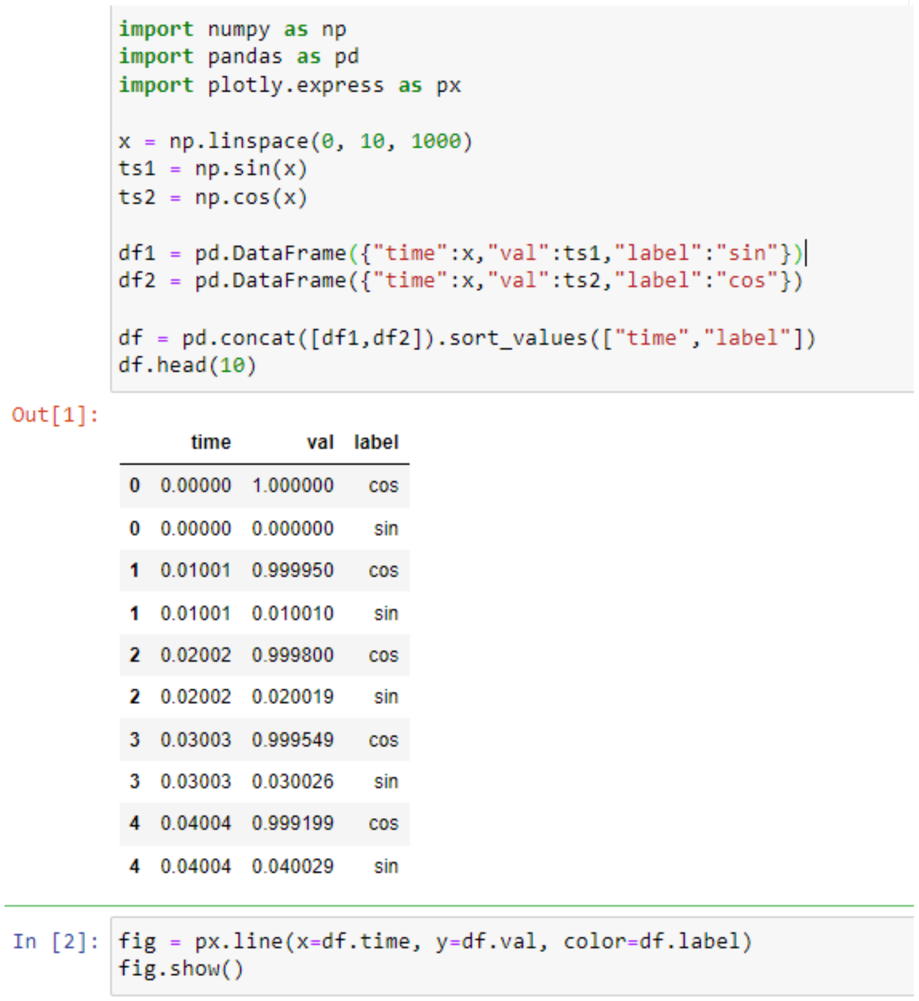
Plotly Subplots



```
from plotly.subplots import make_subplots

fig = make_subplots(rows=2, cols=2,
                    subplot_titles=("Linear", "Square", "Cube", "Quarted"))

x = np.arange(-30, 30)

fig.add_trace(go.Scatter(x=x, y=x**1, mode='markers'),
              row=1, col=1)

fig.add_trace(go.Scatter(x=x, y=x**2, mode='markers'),
              row=1, col=2)

fig.add_trace(go.Scatter(x=x, y=x**3, mode='markers'),
              row=2, col=1)

fig.add_trace(go.Scatter(x=x, y=x**4, mode='markers'),
              row=2, col=2)

fig.update_layout(showlegend=False, title="What a Nice Graphs!")


fig.show()
```



Plotly and Multiple Time Series

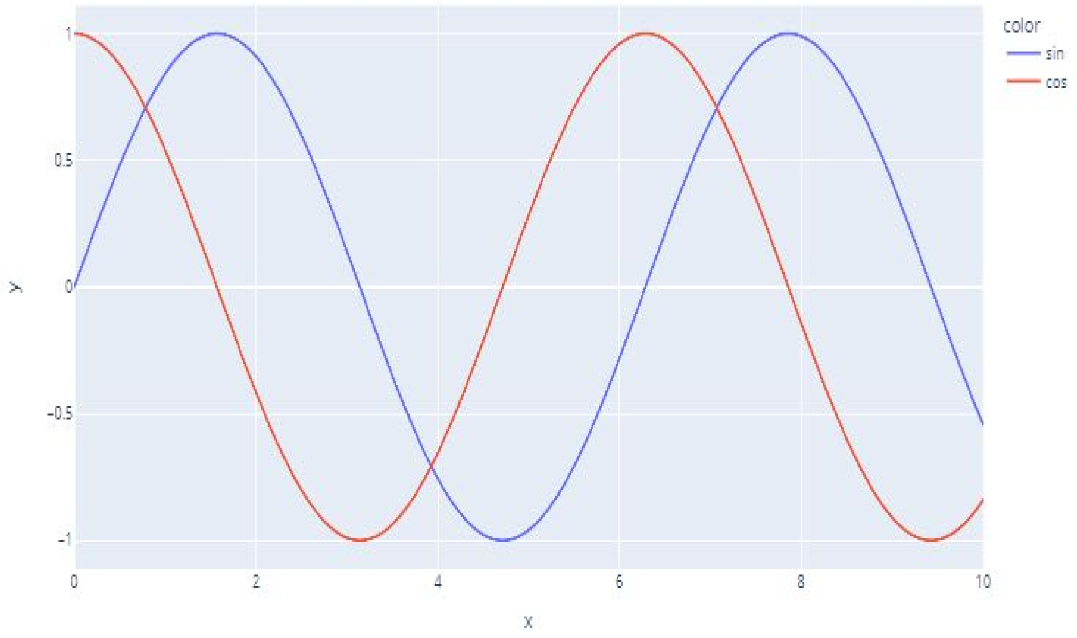


```
import numpy as np
import pandas as pd
import plotly.express as px

x = np.linspace(0, 10, 1000)
ts1 = np.sin(x)
ts2 = np.cos(x)

df1 = pd.DataFrame({"time":x,"val":ts1,"label":"sin"})
df2 = pd.DataFrame({"time":x,"val":ts2,"label":"cos"})

df = pd.concat([df1,df2]).sort_values(["time","label"])
df.head(10)
```

Out[1]:

	time	val	label
0	0.00000	1.000000	cos
0	0.00000	0.000000	sin
1	0.01001	0.999950	cos
1	0.01001	0.010010	sin
2	0.02002	0.999800	cos
2	0.02002	0.020019	sin
3	0.03003	0.999549	cos
3	0.03003	0.030026	sin
4	0.04004	0.999199	cos
4	0.04004	0.040029	sin

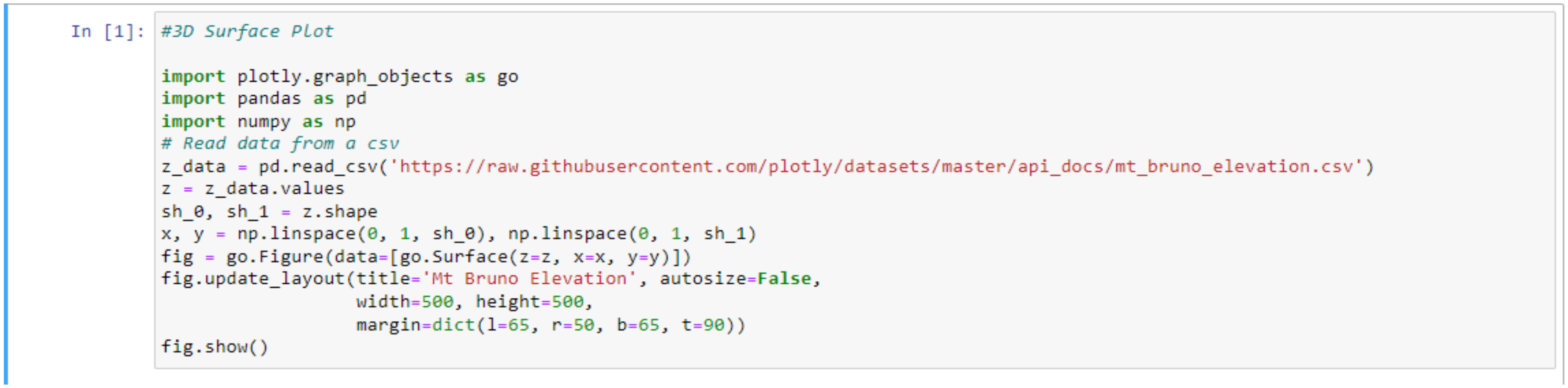
In [2]:
```
fig = px.line(x=df.time, y=df.val, color=df.label)
fig.show()
```



Plotly can split one column into two lines using another column (label) as criteria

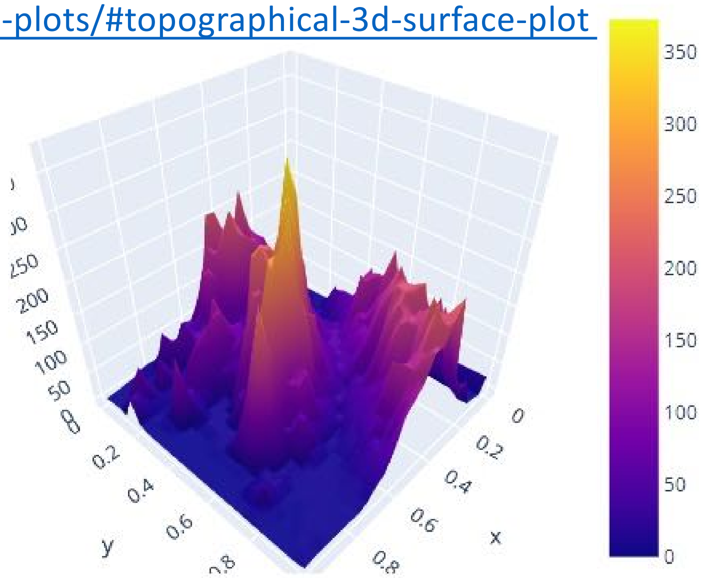
- https://plotly.com/python/time-series/
- https://stackoverflow.com/questions/64158858/plotly-how-to-create-a-line-plot-of-a-time-series-variable-that-has-a-multiple

Plotly 3D Charts



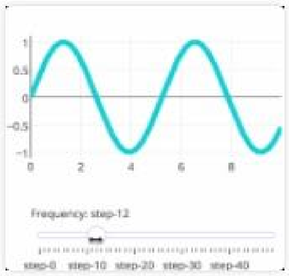
```
In [1]: #3D Surface Plot

        import plotly.graph_objects as go
        import pandas as pd
        import numpy as np
        # Read data from a csv
        z_data = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/api_docs/mt_bruno_elevation.csv')
        z = z_data.values
        sh_0, sh_1 = z.shape
        x, y = np.linspace(0, 1, sh_0), np.linspace(0, 1, sh_1)
        fig = go.Figure(data=[go.Surface(z=z, x=x, y=y)])
        fig.update_layout(title='Mt Bruno Elevation', autosize=False,
                          width=500, height=500,
                          margin=dict(l=65, r=50, b=65, t=90))
        fig.show()
```

https://plotly.com/python/3d-surface-plots/#topographical-3d-surface-plot

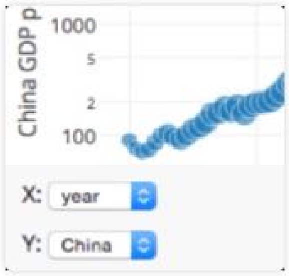


Plotly and Custom Controls

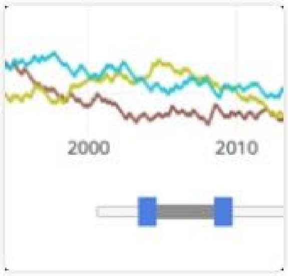
Custom Buttons



Sliders



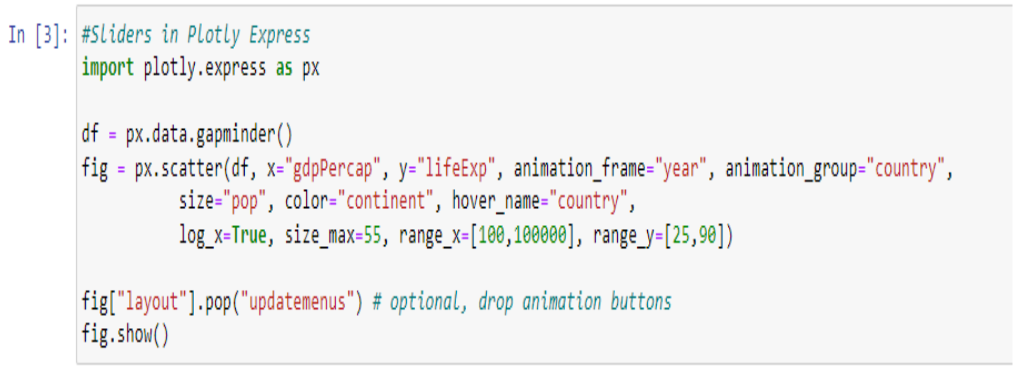
Dropdown Menus



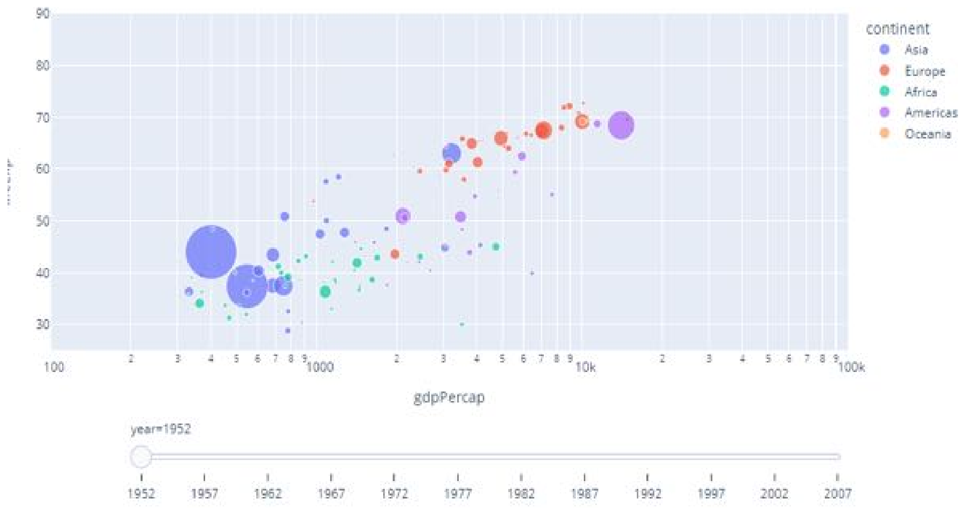
Range Slider and Selector

Reference
https://plotly.com/python/#controls

https://plotly.com/python/reference/layout/updatemenus/



```
In [3]: #Sliders in Plotly Express
        import plotly.express as px

        df = px.data.gapminder()
        fig = px.scatter(df, x="gdpPercap", y="lifeExp", animation_frame="year", animation_group="country",
                   size="pop", color="continent", hover_name="country",
                   log_x=True, size_max=55, range_x=[100,100000], range_y=[25,90])

        fig["layout"].pop("updatemenus") # optional, drop animation buttons
        fig.show()
```



Dash - Plotly Charts in Web Pages

Dash is a way to use Plotly in web pages.
You can create Graphs in python on server side using the same syntax as in Jupyter.

Dash app gallery: https://dash.gallery/Portal/

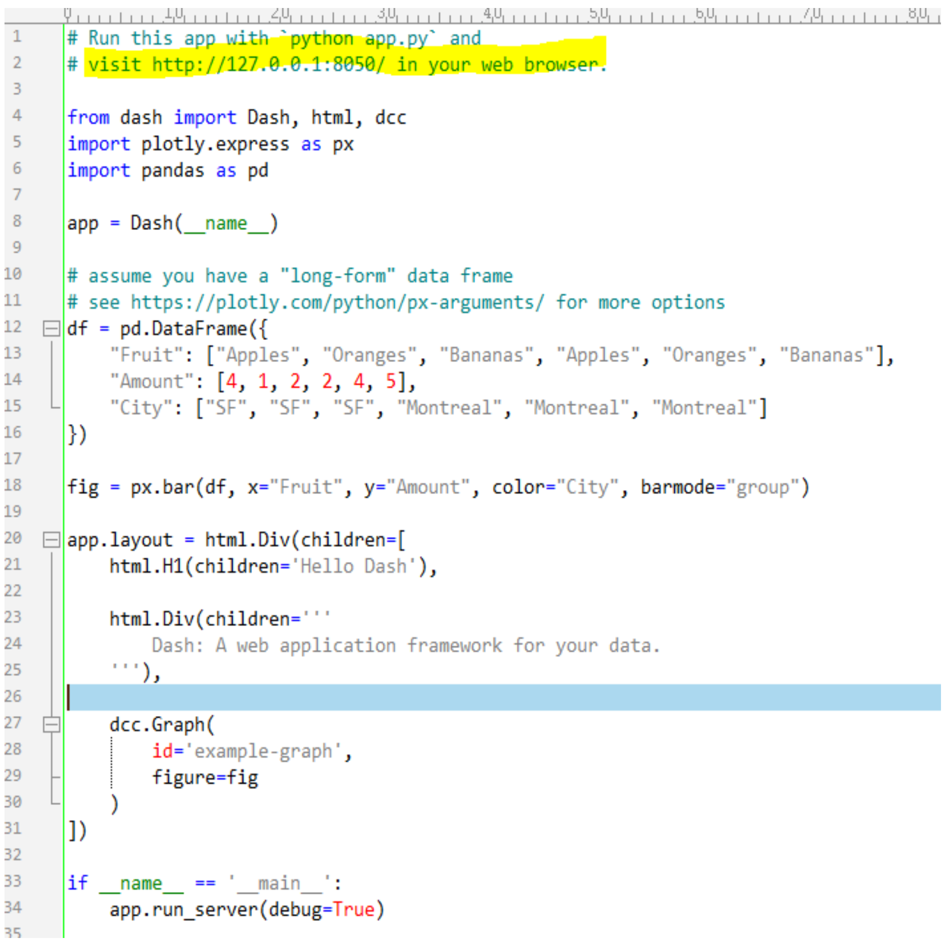
Installation instructions: https://dash.plot.ly/installation.

```
pip install dash
```

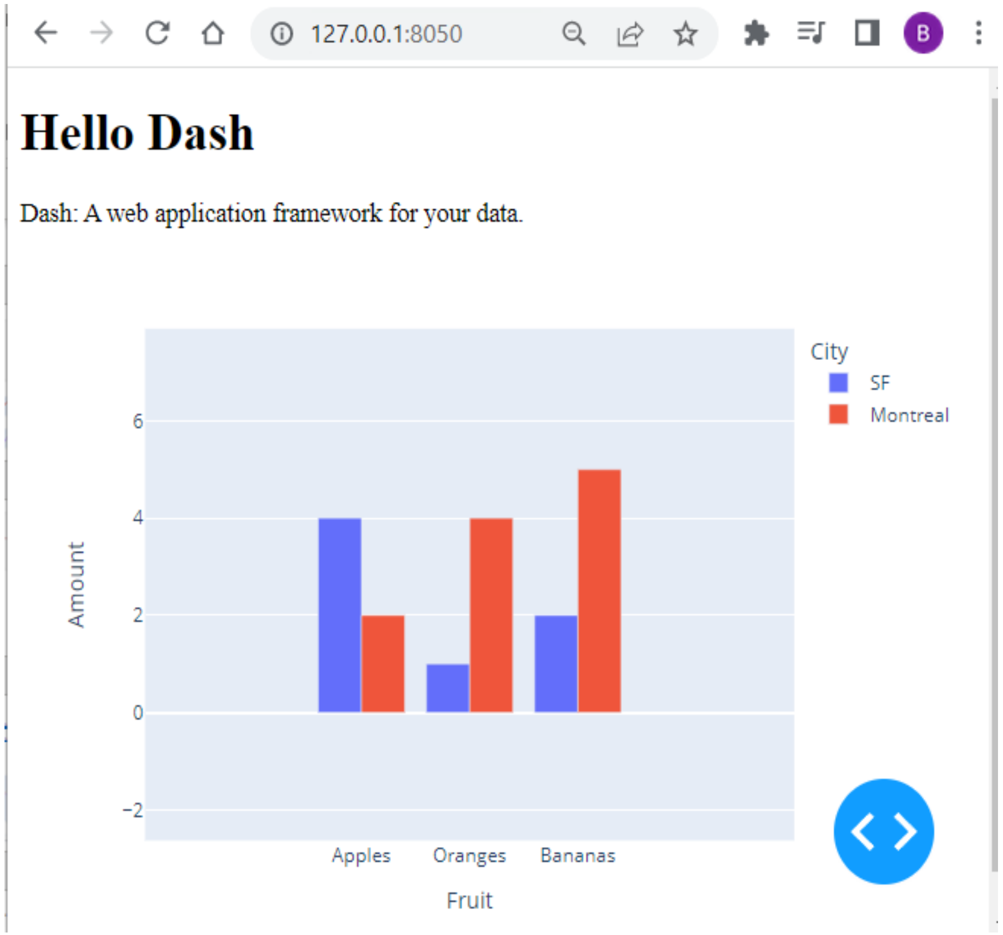
Dash components

- Dash Core Components (buttons, sliders, dropdown, etc.)
  https://dash.plotly.com/dash-core-components
- Plotly Python Open Source Graphing Library (Plotly Graphs)
  https://plotly.com/python/
- Dash Callbacks
  https://dash.plotly.com/basic-callbacks

Dash is running on http://127.0.0.1:8050/ (Press CTRL+C to quit)



```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.

from dash import Dash, html, dcc
import plotly.express as px
import pandas as pd

app = Dash(__name__)

# assume you have a "long-form" data frame
# see https://plotly.com/python/px-arguments/ for more options
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 your data.
    '''),

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

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



127.0.0.1:8050

Hello Dash

Dash: A web application framework for your data.

References:

Plotly library - to be used from Jupyter:

- https://plotly.com
- Gallery and getting started - https://plotly.com/python/
- Plotly Express: https://plotly.com/python-api-reference/plotly.express.html
- Python API reference for plotly: https://plotly.com/python-api-reference/

Dash - same library to be used in a web site:

- https://plotly.com/dash/
- https://dash.gallery/Portal/
- https://dash.plotly.com/
- Dash App Gallery: https://dash-gallery.plotly.host/Portal/
- Dash Components: https://dash.plotly.com/dash-core-components
- The Callback: https://dash.plotly.com/basic-callbacks
- Dash Plotly Community Forum: https://community.plotly.com/c/dash/16

Comparisons of different libraries:

- https://pythonplot.com – very good comparison
- https://ritza.co/articles/matplotlib-vs-seaborn-vs-plotly-vs-MATLAB-vs-ggplot2-vs-pandas/
- https://www.justintodata.com/python-data-visualization-libraries/
- https://www.projectpro.io/article/python-data-visualization-libraries/543
- https://geo-python-site.readthedocs.io/en/stable/lessons/L7/python-plotting.html