plotly

[Community Forum]

[Dash Docs]

Dash Club Cheat Sheets

1. Getting Started

Installation

```
pip install dash pandas
```

Hello World

```
from dash import Dash, dcc, html, Output,
Input
import plotly.express as px

df = px.data.tips()
app = Dash(__name__)

radio_b = dcc.RadioItems(df.columns, 'time')
my_graph = dcc.Graph()
app.layout = html.Div([radio_b, my_graph])

@app.callback(Output(my_graph, 'figure'),
              Input(radio_b, 'value'))
def update_graph(value):
   fig = px.bar(df, x=value, y="tip")
   return fig
if __name__ == '__main__':
   app.run()
```

2. Common Components

```
dcc.Dropdown(['kiwi', 'banana', 'apple'],
value='orange')

dcc.Slider(0,20,5, value=10)

dcc.RadioItems(['Purple', 'Red','Pink'],
value='Red')

df = px.data.tips()
fig = px.bar(df, x='time',y='tip')
dcc.Graph(figure=fig)

html.Button('Submit', n_clicks=0)

dcc.RangeSlider(0,20,1, value=[5, 15])

dcc.Textarea(
value='Textarea content',
style={'width': '100%',  'height': 300})
```

3. Side-by-Side Layouts

HTML and CSS

```
sheet=['https://codepen.io/chriddyp/pen/bWLwg
P.css']
app = Dash(__name__,
external_stylesheets=sheet)
```

```
pdn = dcc.Dropdown(['App', 'Base', 'Cat'],
'App')

app.layout = html.Div([
   html.Div([
       html.Div([dpdn], className="three
columns"),
       html.Div([dcc.Graph()],
className="nine columns")
   ], className="row")
])
```

Dash Bootstrap

```
app = Dash(__name__,
external_stylesheets=[dbc.themes.BOOTSTRAP])
dpdn = dcc.Dropdown(['App', 'Base', 'Cat'],
'App')
graph = dcc.Graph()
app.layout = dbc.Container([
   dbc.Row([
       dbc.Col([dpdn], width=3),
       dbc.Col([graph], width=9)
   ])
])
```

Dash Enterprise Design Kit

```
controls = [
   ddk.ControlItem(),
   ddk.ControlItem()
]
app.layout = ddk.App([
   ddk.ControlCard(controls, width=30),
   ddk.Card(width=70, children=ddk.Graph())
])
```

4. Callbacks

Declared IDs

```
app.layout = html.Div([
   dcc.Input(id='my-input',
             value='initial value'),
   html.Div(id='my-output')
])

@app.callback(
   Output('my-output','children'),
   Input('my-input', 'value')
)
def update_output(input_value):
   return input_value
```

Randomly Generated IDs

```
text = dcc.Input(value='hello')
user_input= html.Div()
app.layout = html.Div([
   text, user_input
])

@app.callback(
   Output(user_input,'children'),
   Input(text, 'value')
)
def update_output(input_value):
   return input_value
```

Multiple Outputs / Inputs

```
app.layout = html.Div([
   dcc.Input(id='my-input',
             value='hello'),
   dcc.Slider(0,20,5, value=10,
              id='slider'),
   html.Div(id='my-output1'),
   html.Div(id='my-output2')
])
@app.callback(
   Output('my-output1','children'),
   Output('my-output2','children'),
   Input('slider','value'),
   Input('my-input','value')
)
def update_out(slider,input_v):
   return input_v, str(slider)
```

5. Share the App

Dash Enterprise Deploy

Requirements.txt:

```
Dash
gunicorn
```

Procfile:

```
web: gunicorn app:server
```

App.py:

```
from dash import Dash, html
app = Dash(__name__)
server = app.server
app.layout = html.Div('hi')
```

Then, push to deploy with:

```
git push plotly main
```

plotly

Community Forum

Dash Docs

Basic Charts

```
df = px.data.iris()
fig = px.scatter(df, x='sepal_width',
y='sepal_length')

df = px.data.stocks()
fig = px.line(df, x='date', y='GOOG')

df = px.data.medals_long()
fig = px.bar(df, x='nation', y='count',
color='medal')

df = px.data.medals_long()
fig = px.area(df, x='medal', y='count',
color='nation')

df = px.data.tips()
fig = px.pie(df, values='tip', names='day')

df = px.data.tips()
fig = px.treemap(df,
path=[px.Constant("all"), 'sex', 'day',
'time'], values='total_bill')

df = px.data.tips()
fig = px.sunburst(df, path=['day', 'time',
'sex'],values='total_bill')

df = px.data.tips()
fig = px.icicle(df, path=[px.Constant("all"),
'day', 'time', 'sex'], values='total_bill')
fig.update_traces(root_color='lightgrey')
```

Statistical Charts

```
df = px.data.tips()
fig = px.density_heatmap(df, x='total_bill',
y='tip')

df = px.data.tips()
fig = px.histogram(df, x='total_bill')

df = px.data.tips()
fig=px.parallel_categories(df)

df = px.data.tips()
fig = px.box(df, x='time', y='total_bill')

df = px.data.iris()
fig = px.scatter_matrix(df)

df = px.data.tips()
fig = px.violin(df, y='total_bill', box=True)

df = px.data.tips()
fig = px.strip(df, x='total_bill', y='day')
```

Scientific Charts

```
df = px.data.medals_wide(indexed=True)
fig = px.imshow(df)

from skimage import data
img = data.astronaut()
fig = px.imshow(img, binary_format='jpeg',
binary_compression_level=0)

df = px.data.election()
fig = px.scatter_ternary(df, a='Joly',
b='Coderre', c='Bergeron')

df = px.data.wind()
fig = px.scatter_polar(df, r='frequency',
theta='direction')

df = pd.DataFrame(dict(r=[1, 5, 2, 2, 3],
theta=['processing cost', 'mechanical
properties', 'chemical stability', 'thermal
stability', 'device integration']))
fig = px.line_polar(df, r='r', theta='theta',
line_close=True)

df = px.data.iris()
fig = px.parallel_coordinates(df,
color='species_id')
```

Maps

```
df = px.data.election()
geojson = px.data.election_geojson()
fig = px.choropleth_mapbox(df,
geojson=geojson, color='Bergeron',
locations='district',
featureidkey='properties.district',center={'l
at': 45.5517, 'lon': -73.7073},
mapbox_style='carto-positron', zoom=9)

df = px.data.gapminder().query('year==2007')
fig = px.scatter_geo(df,
locations='iso_alpha', color='continent',
hover_name='country', size='pop',
projection='natural earth')

df = pd.read_csv(
'https://raw.githubusercontent.com/plotly/dat
asets/master/us-cities-top-1k.csv')
fig = px.scatter_mapbox(df, lat='lat',
lon='lon', hover_name='City',
hover_data=['State', 'Population'], zoom=3,
height=600)
fig.update_layout(mapbox_style='open-street-m
ap')

df = px.data.gapminder().query('year==2007')
fig = px.line_geo(df, locations='iso_alpha',
color='continent', projection='orthographic')

df = pd.read_csv(
'https://raw.githubusercontent.com/plotly/dat
asets/master/earthquakes-23k.csv')
fig = px.density_mapbox(df, lat='Latitude',
lon='Longitude', z='Magnitude', radius=10,
center=dict(lat=0, lon=180), zoom=0,
mapbox_style='stamen-terrain')
```