Dash Framework

In this chapter, we will discuss about the Dash framework in detail.

Dash is an open-source Python framework used for building analytical web applications. It is a powerful library that simplifies the development of data-driven applications. Its especially useful for Python data scientists who arent very familiar with web development. Users can create amazing dashboards in their browser using dash.

Built on top of Plotly.js, React, and Flask, Dash ties modern UI elements like dropdowns, sliders and graphs directly to your analytical python code.

Dash apps consist of a Flask server that communicates with front-end React components using JSON packets over HTTP requests.

Dash applications are written purely in python, so NO HTML or JavaScript is necessary.

Dash Setup

If Dash is not already installed in your terminal, then install the below mentioned Dash libraries. As these libraries are under active development, install and upgrade then frequently. Python 2 and 3 are also supported.

- pip install dash==0.23.1 # The core dash backend
- pip install dash-renderer==0.13.0 # The dash front-end

- ...0.11.0 # HTML components
- ...0.26.0 # Supercharged components
- ...phing library

In ...ng properly, here, we created a simple
da...

Dash or App Layout

Dash apps are composed of two parts. The first part is the layout of the app which basically describes how the application looks like. The second part describes the interactivity of the application.

Core Components

We can build the layout with the **dash_html_components** and the **dash_core_components** library. Dash provides python classes for all the visual components of the application. We can also customize our own components with JavaScript and React.js.

import dash_core_components as dcc

import dash_html_components as html

The dash_html_components is for all HTML tags where the dash_core_components is for interactivity built with React.js.

Using above two libraries, let us write a code as given below −

Advertisement

k: A web application framework for Python.'

An   like this −

```html
<div>
    <h1> Hello Dash </h1>
    <div> Dash Framework: A web application framework for Python. </div>
</div>
```

Writing Simple Dash app

We will learn how to write a simple example on dash using above mentioned library in a file **dashApp.py**.

```python
# -*- coding: utf-8 -*-
import dash
import dash_core_components as dcc
import dash_html_components as html

app = dash.Dash()
app.layout = html.Div(children=[
    html.H1(children='Hello Dash'),
    html.Div(children='''Dash Framework: A web application framework for
Python.'''),

    dcc.Graph(
        id='example-graph',
        figure={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'Delhi'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name':
u'Mumbai'},
            ],
            'layout': {
                'title': 'Dash Data Visualization'
            }
```

R
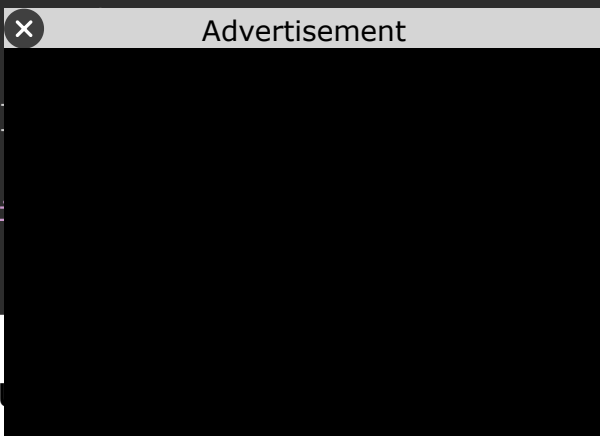
Note the following points while running the Dash app.

(MyDjangoEnv) C:\Users\rajesh\Desktop\MyDjango\dash>python dashApp1.py

- Serving Flask app "dashApp1" (lazy loading)

- Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.

- Debug mode: on

- Restarting with stat

- Debugger is active!

- Debugger PIN: 130-303-947

- Running on **http://127.0.0.1:8050/** (Press CTRL+C to quit)

```
127.0.0.1 - - [12/Aug/2018 09:32:39] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2018 09:32:42] "GET /_dash-layout HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2018 09:32:42] "GET /_dash-dependencies HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2018 09:32:42] "GET /favicon.ico HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2018 09:39:52] "GET /favicon.ico HTTP/1.1" 200 -
```

Visit **http:127.0.0.1:8050/** in your web browser. You should see an app that looks like this.
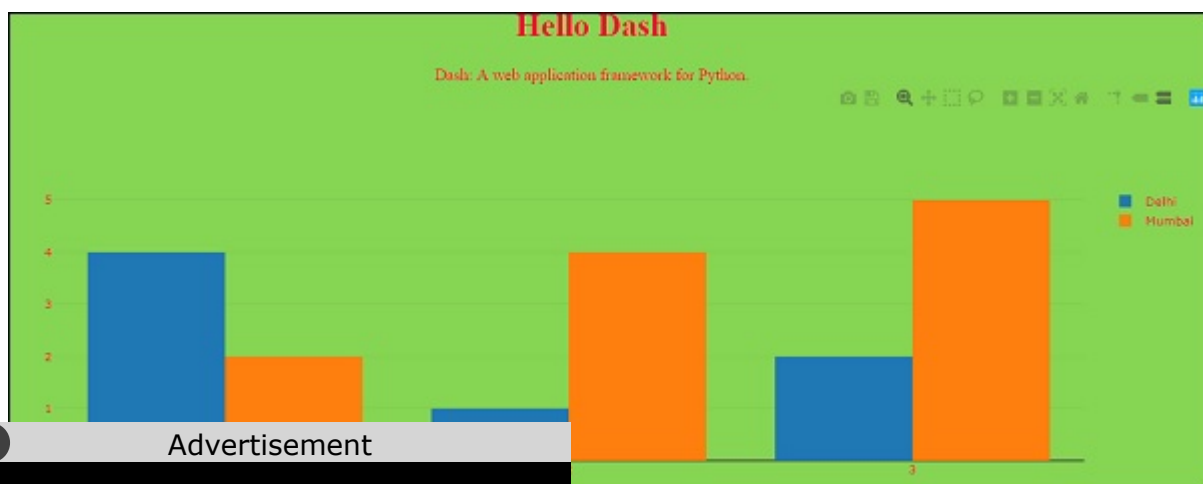
In [...] be noted are as follows −

[...] of components like html.Div and dcc.Graph.

[...] has a component for every HTML tag. The [...]ponent generates a <h1> Hello Dash </h1>

- Not all components are pure HTML. The dash_core_components describe higher-level components that are interactive and are generated with JavaScript, HTML, and CSS through the React.js library.

- Each component is described entirely through keyword attributes. Dash is declarative: you will primarily describe your application through these attributes.

- The children property is special. By convention, its always the first attribute which means that you can omit it.

- Html.H1 (children=Hello Dash) is the same as html.H1 (Hello Dash).

- The fonts in your application will look a little bit different than what is displayed here. This application is using a custom CSS stylesheet to modify the default styles of the elements. Custom font style is permissible, but as of now, we can add the below URL or any URL of your choice −
  app.css.append_css
  ({external_url:**https://codepen.io/chriddyp/pen/bwLwgP.css**}) to get your file to get the same look and feel of these examples.

More about HTML

The dash_html_components library contains a component class for every HTML tag as well as keyword arguments for all of the HTML arguments.

Let us add the inline style of the components in our previous app text −

```python
# -*- coding: utf-8 -*-
import dash
import dash_core_components as dcc
import dash_html_components as html

app = dash.Dash()
colors = {
    'background': '#87D653',
    'text': '#ff0033'
}
```

```
                                    undColor': colors['background']},
```

```python
        ),

        html.Div(children='Dash: A web application framework for Python.', style={
            'textAlign': 'center',
            'color': colors['text']
        }),

        dcc.Graph(
            id='example-graph-2',

            figure={
                'data': [
                    {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'Delhi'},
                    {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name':
 u'Mumbai'},
                ],
                'layout': {
                    'plot_bgcolor': colors['background'],
                    'paper_bgcolor': colors['background'],
                    'font': {
                        'color': colors['text']
                    }
                }
            }
        )
    ])

    if __name__ == '__main__':
        app.run_server(debug=True)
```
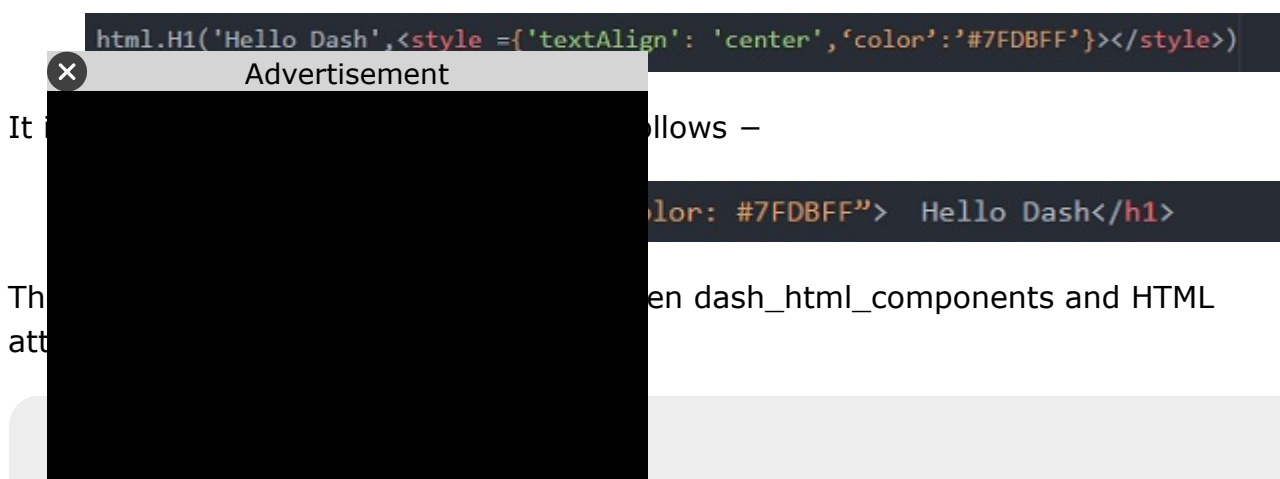
In the above example, we modified the inline styles of the html.Div and html.H1 components with the style property.

```
html.H1('Hello Dash',<style ={'textAlign': 'center','color':'#7FDBFF'}></style>)
```

It ⬛⬛⬛⬛⬛⬛⬛⬛⬛⬛⬛⬛ llows −

⬛⬛lor: #7FDBFF"> Hello Dash</h1>

Th⬛⬛⬛⬛⬛⬛⬛⬛⬛⬛⬛⬛en dash_html_components and HTML
att⬛⬛⬛

- For style property in Dash, you can just supply a dictionary, whereas in HTML, it is semicolon-separated string.
- Style dictionary keys are **camelCased**, so text-align changes to **textalign**.
- ClassName in Dash is similar to HTML class attribute.
- The first argument is the children of the HTML tag which is specified through the children keyword argument.

Reusable Components

By writing our markup in Python, we can create complex reusable components like tables without switching contexts or languages –

Below is a quick example that generates a Table from pandas dataframe.

```python
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd

df = pd.read_csv(
    'https://gist.githubusercontent.com/chriddyp/'
    'c78bf172206ce24f77d6363a2d754b59/raw/'
    'c353e8ef842413cae56ae3920b8fd78468aa4cb2/'
    'usa-agricultural-exports-2011.csv')

def generate_table(dataframe, max_rows=10):
    return html.Table(
        # Header
        [html.Tr([html.Th(col) for col in dataframe.columns])] +
        # Body
        [html.Tr([
            html.Td(dataframe.iloc[i][col]) for col in dataframe.columns
        ]) for i in range(min(len(dataframe), max_rows))]
```

```
xports (2011)'),
```

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

Our output will be something like −

US Agriculture Exports (2011)

Unnamed: 0	state	total exports	beef	pork	poultry	dairy	fruits fresh	fruits proc	total fruits	veggies fresh	veggies proc	total veggies	corn	wheat	cotton
0	Alabama	1390.63	34.4	10.6	481	4.06	8	17.1	25.11	5.5	8.9	14.33	34.9	70	317.61
1	Alaska	13.31	0.2	0.1	0	0.19	0	0	0	0.6	1	1.56	0	0	0
2	Arizona	1463.17	71.3	17.9	0	105.48	19.3	41	60.27	147.5	239.4	386.91	7.3	48.7	423.95
3	Arkansas	3586.02	53.2	29.4	562.9	3.53	2.2	4.7	6.88	4.4	7.1	11.45	69.5	114.5	665.44
4	California	16472.88	228.7	11.1	225.4	929.95	2791.8	5944.6	8736.4	803.2	1303.5	2106.79	34.6	249.3	1064.95
5	Colorado	1851.33	261.4	66	14	71.94	5.7	12.2	17.99	45.1	73.2	118.27	183.2	400.5	0
6	Connecticut	259.62	1.1	0.1	6.9	9.49	4.2	8.9	13.1	4.3	6.9	11.16	0	0	0
7	Delaware	282.19	0.4	0.6	114.7	2.3	0.5	1	1.53	7.6	12.4	20.03	26.9	22.9	0
8	Florida	3764.09	42.6	0.9	56.9	66.31	438.2	933.1	1371.36	171.9	279	450.86	3.5	1.8	78.24
9	Georgia	2860.84	31	18.9	630.4	38.38	74.6	158.9	233.51	59	95.8	154.77	57.8	65.4	1154.07

More about Visualization

The dash_core_components library includes a component called **Graph**.

Graph renders interactive data visualizations using the open source plotly.js JavaScript graphing library. Plotly.js support around 35 chart types and renders charts in both vector-quality SVG and high-performance WebGL.

Below is an example that creates a scatter plot from a Pandas dataframe −

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import plotly.graph_objs as go

app = dash.Dash()

df = pd.read_csv(
    'https://gist.githubusercontent.com/chriddyp/' +
    '5d1ea79569ed194d432e56108a04d188/raw/' +
    'a9f9e8076b837d541398e999dcbac2b2826a81f8/'+
```

```
                                = i]['gdp per capita'],
```
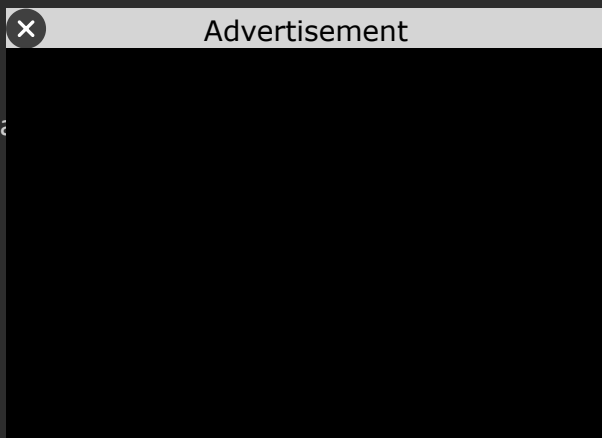
```python
                y=df[df['continent'] == i]['life expectancy'],
                text=df[df['continent'] == i]['country'],
                mode='markers',
                opacity=0.7,
                marker={
                    'size': 15,
                    'line': {'width': 0.5, 'color': 'white'}
                },
                name=i
            ) for i in df.continent.unique()
        ],
        'layout': go.Layout(
            xaxis={'type': 'log', 'title': 'GDP Per Capita'},
            yaxis={'title': 'Life Expectancy'},
            margin={'l': 40, 'b': 40, 't': 10, 'r': 10},
            legend={'x': 0, 'y': 1},
            hovermode='closest'
        )
    }
)
])


if __name__ == '__main__':
    app.run_server()
```
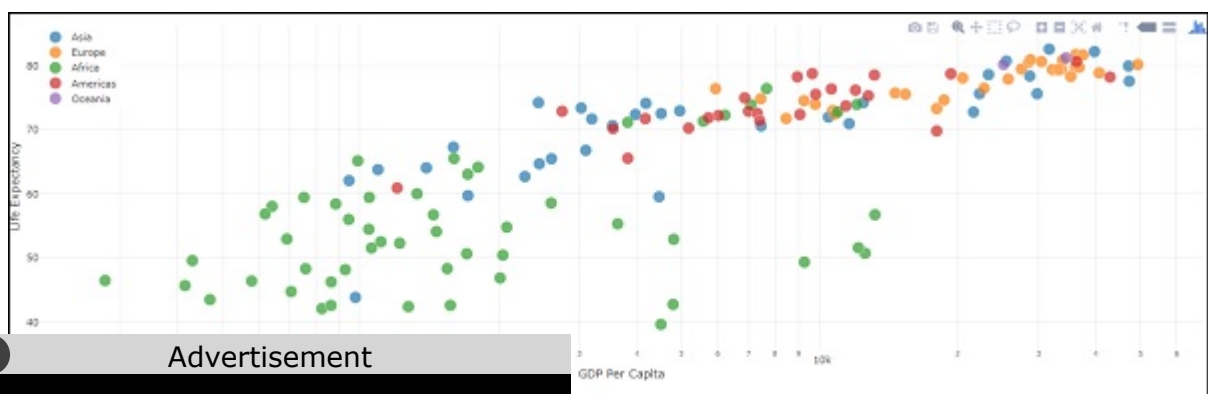
The output of the above code is as follows −

Th[...]ve. You can hover over points to see their
va[...]ces, click and drag to zoom, hold down shift,
an[...]

M[...]

While dash exposes HTML flavours through the dash_html_components library, it can be tedious to write your copy in HTML. For writing blocks of texts, you can use the Markdown component in the dash_core_components library.

Core Components

The dash_core_components includes a set of higher-level components like dropdowns, graphs, markdown, blocks and many more.

Like all other Dash components, they are described entirely declaratively. Every option that is configurable is available as a keyword argument of the component.

Below is the example, using some of the available components −

```python
# -*- coding: utf-8 -*-
import dash
import dash_core_components as dcc
import dash_html_components as html


app = dash.Dash()

app.layout = html.Div([
    html.Label('Dropdown'),
    dcc.Dropdown(
        options=[
            {'label': 'New York City', 'value': 'NYC'},
            {'label': u'Montral', 'value': 'MTL'},
            {'label': 'San Francisco', 'value': 'SF'}
        ],
        value='MTL'
    ),

    html.Label('Multi-Select Dropdown'),
    dcc.Dropdown(
        options=[
            value': 'NYC'},
            ': 'MTL'},
            value': 'SF'}
```

```
      html.Label('Radio Items'),
      dcc.RadioItems(
         options=[
            {'label': 'New York City', 'value': 'NYC'},
            {'label': u'Montral', 'value': 'MTL'},
            {'label': 'San Francisco', 'value': 'SF'}
         ],
         value='MTL'
      ),

      html.Label('Checkboxes'),
      dcc.Checklist(
         options=[
            {'label': 'New York City', 'value': 'NYC'},
            {'label': u'Montral', 'value': 'MTL'},
            {'label': 'San Francisco', 'value': 'SF'}
         ],
         values=['MTL', 'SF']
      ),

      html.Label('Text Input'),
      dcc.Input(value='MTL', type='text'),
```

Chapters ∨   ⊞ Categories ☰

```
      dcc.Slider(
         min=0,
         max=9,
         marks={i: 'Label {}'.format(i) if i == 1 else str(i) for i in range(1,
6)},
         value=5,
      ),
], style={'columnCount': 2})

if __name__ == '__main__':
```
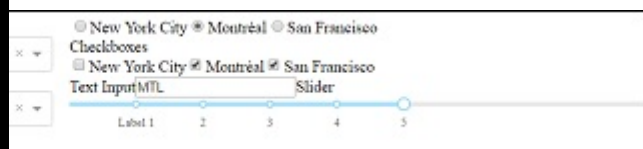
Advertisement

Ou...ws —

Calling Help

Dash components are declarative. Every configurable aspect of these components is set during installation as a keyword argument. You can call help in your python console on any of the components to learn more about a component and its available arguments. Some of them are given below −

```
>>> help(dcc.Dropdown)
Help on class Dropdown in module builtins:
class Dropdown(dash.development.base_component.Component)
| A Dropdown component.
| Dropdown is an interactive dropdown element for selecting one or more

| items.
| The values and labels of the dropdown items are specified in the `options`
| property and the selected item(s) are specified with the `value` property.
|
| Use a dropdown when you have many options (more than 5) or when you are
| constrained for space. Otherwise, you can use RadioItems or a Checklist,
| which have the benefit of showing the users all of the items at once.
|
| Keyword arguments:
| - id (string; optional)
| - options (list; optional): An array of options
| - value (string | list; optional): The value of the input. If `multi` is false
-- More --
```

To summarize, the layout of a Dash app describes what the app looks like. The layout is a hierarchical tree of components. The dash_html_components library provides classes for all the HTML tags and the keyword arguments, and describes the HTML attributes like style, className, and id. The dash_core_components library generates higher-level components like controls and graphs.

C# Tutorial

PHP Tutorial

R Tutorial

HTML Tutorial

CSS Tutorial

JavaScript Tutorial

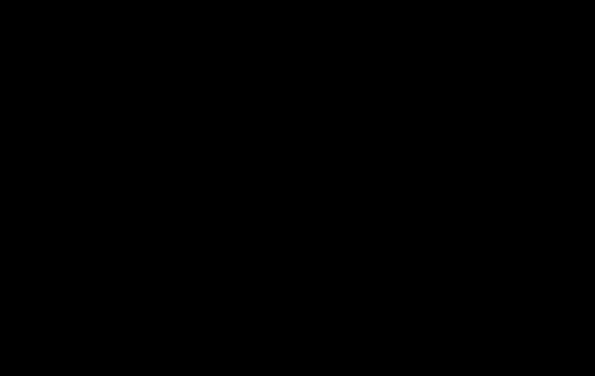SQL Tutorial

TRENDING TECHNOLOGIES

Cloud Computing Tutorial

Amazon Web Services Tutorial

Microsoft Azure Tutorial

Git Tutorial

Ethical Hacking Tutorial

Docker Tutorial

Kubernetes Tutorial

DSA Tutorial

Spring Boot Tutorial

SDLC Tutorial

Unix Tutorial

CERTIFICATIONS

Business Analytics Certification

Java & Spring Boot Advanced Certification

Data Science Advanced Certification

Cloud Computing And DevOps

Advanced Certification In Business Analytics

Artificial Intelligence And Machine Learning

Online Python Compiler

Online Go Compiler

Online C Compiler

Online C++ Compiler

Online C# Compiler

Online PHP Compiler

Online MATLAB Compiler

Online Bash Terminal

Online SQL Compiler

Online Html Editor

ABOUT US | OUR TEAM | CAREERS | JOBS | CONTACT US | TERMS OF USE |

PRIVACY POLICY | REFUND POLICY | COOKIES POLICY | FAQ'S

tutorialspoint

Tutorials Point is a leading Ed Tech company striving to provide the best learning material on technical and non-technical subjects.

© Copyright 2025. All Rights Reserved.