

# Dash

Learn how to build dashboards in Python using Dash.

Dash is Python framework for building web applications. Dash is open source, and its apps run on the web browser. A Dash application is usually composed of two parts. The first part is the layout and describes how the app will look like and the second part describes the interactivity of the application. Dash provides HTML classes that enable us to generate HTML content with Python. To use these classes, we need to import `dash_core_components` and `dash_html_components`.

## Step1: Dash Installation

In order to start using Dash, we have to install several packages.

The core dash backend.

Dash front-end

Dash HTML components

Dash core components

Plotly

```
pip install dash==0.21.1
pip install dash-renderer==0.13.0
pip install dash-html-components==0.11.0
pip install dash-core-components==0.23.0
pip install plotly --upgrade
```

## Step2: Import Packages (here are the packages used for SDG)

```
import dash
```

```
import dash_core_components as dcc
```

```
import dash_html_components as html
from dash.dependencies import Input, Output, State
from datetime import datetime
import pandas as pd
import numpy as np
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import pymongo
import dns
import json
import dash_bootstrap_components as dbc
import plotly
from plotly.offline import plot
import random
```

### Step3: Interactivity

This is one of the main parts in our dashboard which used in SDG dashboard. I will explain the details step by step as following.

It is important to remind ourselves that at its core a Dash application is also a Flask application. For purposes of deployment, we need to access the Flask application instance. Dash enables us to do this using `app.server`.

```
app = dash.Dash()

app.layout = html.Div([
    dcc.Input(id='my-id', value='Dash App', type='text'),
    html.Div(id='my-div')
])
```

**3.1** After import the libraries we need, we need to build a frame for dashboard, which is an

adjustable part for different purposes in different projects. For example, we coded for SDG as following, where **## header and logo** is specific for GlobalAI and could be fixed for future projects. The SDG includes polar plot, bar plot, heatmap and also word cloud for different companies and 17 SDG. Additionally, we have a table for portfolio, which is used as template.

```
server=app.server

app.layout = html.Div([
    html.Br(),
    html.Br(),
    ## header and logo
    html.Div([
        html.H1('SDG-Index', className = 'ten columns', style = {'margin-top': 10, 'margin-left': 15, 'color': colors
            html.Img(
                src = 'https://images.squarespace-cdn.com/content/5c036cd54eddec1d4ff1c1eb/1557908564936-YSBRPFCGYV2CE43',
                style = {
                    'height': '11%',
                    'width': '11%',
                    'float': 'right',
                    'position': 'relative',
                    'margin-top': 11,
                    'margin-right': 0
                },
                className = 'two columns'
            )
        ], className = 'row'),
```

It will show as following:



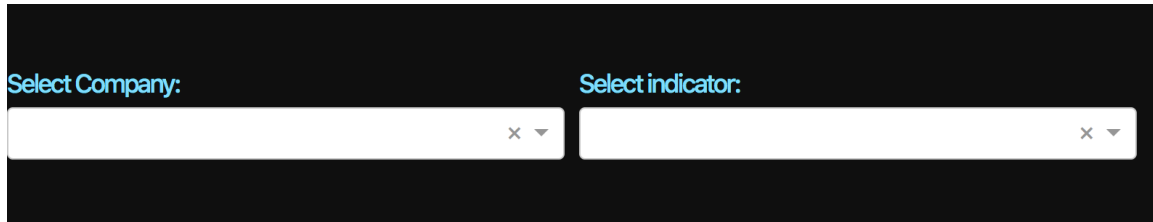
**3.2** The next import part for building dash is `@app.callback`, which includes **Output**, **Input** and also could have **State**, which is used for **submit button** in SDG, shown as following:

```
@app.callback(
    Output('my_polar', 'figure'),
    [Input('submit-button', 'n_clicks')],
    [State('company_ticker', 'value'),
     State('my_indicator_symbol', 'value')])
```

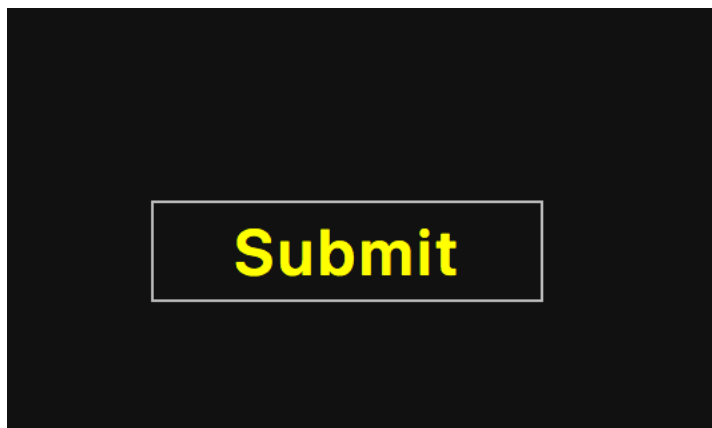
The inputs and outputs of a Dash application interface are described declaratively through

the `app.callback` decorator. In Dash, the inputs and outputs of the above application are simply the properties of a particular component. In previous example, before 3.1, input is the value property of the component that has the ID `my-id`. The output is the children property of the component with the ID `my-div`.

Select bar:



Submit Button:



**3.3** Finally, we could define the functions we need to fill the frame we built earlier at 3.1. For SDG, we make the content for companies separately and also need a part when we select the option: **All**, which is an option for including all the 17 SDGs and comparison between the selected company and its sector in bar plot and scatter plot.



#### Step4: Layout

Laying out a dashboard page is a great step to build a better looking of our plots, which could change the size, color, pattern and direction etc. for various requirements. When creating a dashboard layout, use the column grid system. For SDG, we could find adding titles, changing colors and other steps during our processes.

```
fig.update_layout(  
    xaxis = dict(range=[-1,1]),  
    showlegend=True,  
    legend=go.layout.Legend(  
        x=0,  
        y=1.0  
    ),  
)  
  
fig.update_layout(  
    plot_bgcolor='rgba(0,0,0,0)',  
)
```

```
fig.update_layout(  
    polar=dict(  
        bgcolor = "#111111",  
        radialaxis=dict(  
            color="white",  
            visible=True,  
            range=[-5,5]  
        ),  
    ),  
    showlegend=False  
)  
  
fig.update_layout(barmode='group', xaxis_tickangle=-45,height=800)
```

#### Step5: Start and Run the Dash dashboard

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