Data Visualization with Dash

(IE481, Lab #3)

•••

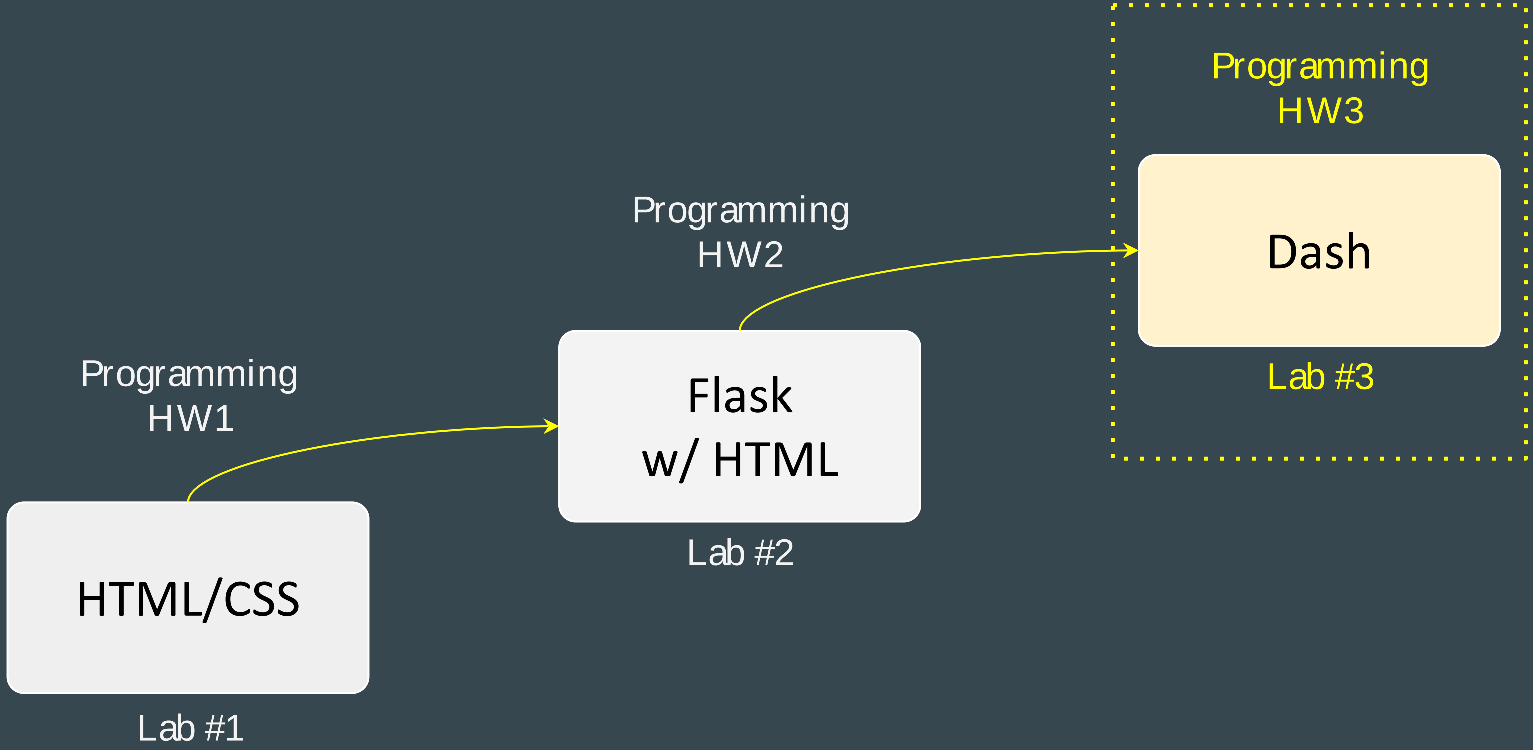
Joonyoung Park

2020. 04. 02

Lab#3 Announcement

- All Lab#3 practice source codes (link)
- Lab#3 practices will proceed as below:
    - Editing tools: PyCharm IDE (only for editing) // using any others is okay
    - Executions: Using CLI (Command Line Interface)
    - Testing: using local server, the last practice will be hosted via Heroku
- For progress reporting: after each practice, upload your practice result
    - Please upload your screenshot of each practice result to this document (Link)
- All of the practice answer codes will be shared after finishing the Lab#3

This week..


Programming
HW1
HTML/CSS
Lab #1
Programming
HW2
Flask
w/ HTML
Lab #2
Programming
HW3
Dash
Lab #3

Dash

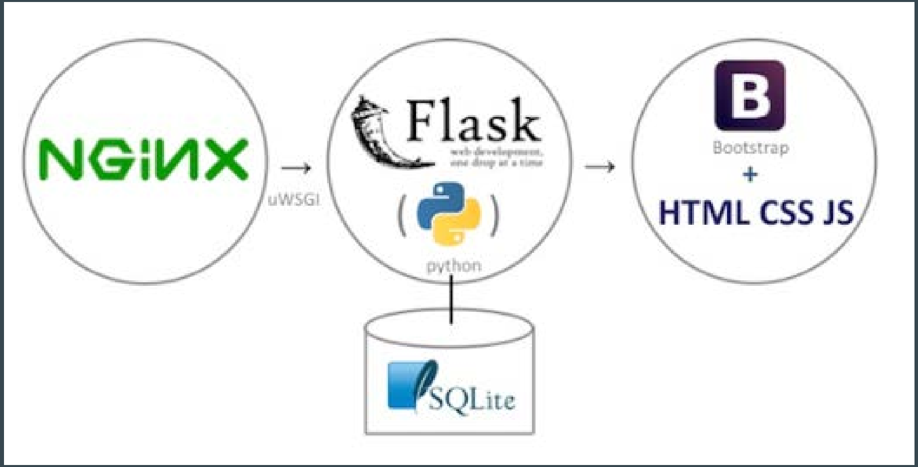
What is Dash?

Python framework
for building web applications

What is the difference between Dash and Flask?

Flask is generic web framework based on Jinja2 and Werkzeug

(Werkzeug: WSGI library)



https://www.quora.com/When-should-I-use-Flask-vs-Dash
http://oliverelliott.org/article/computing/notes_labsite/

Dash is specialized
for building analytical web applications
(E.g., Data visualization)

⬆

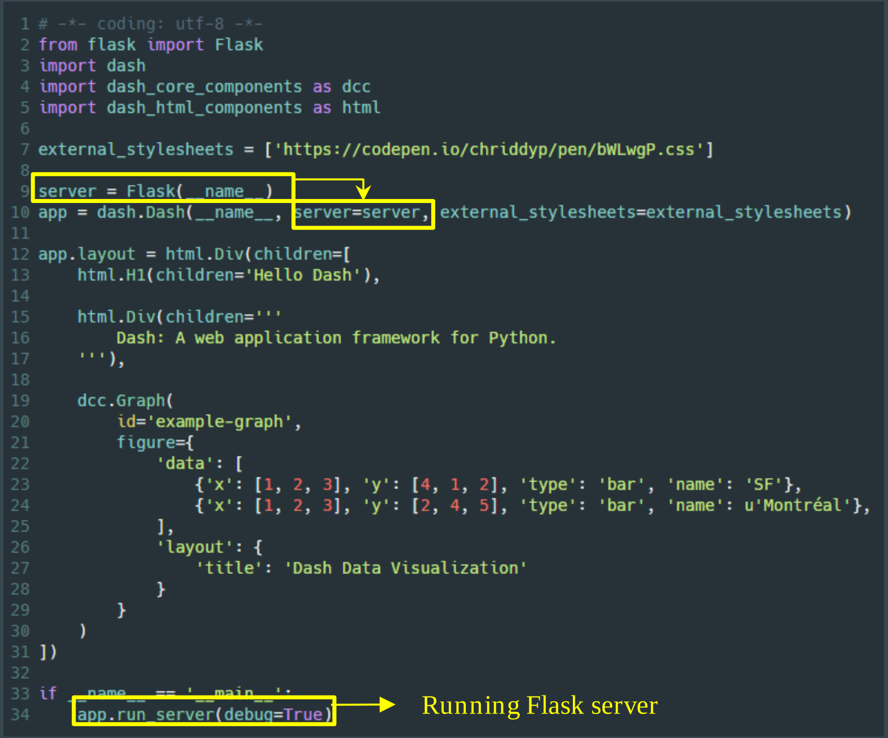
Built on
Plotly, React.js, and Flask

https://www.quora.com/When-should-I-use-Flask-vs-Dash

How to use Dash with Flask?

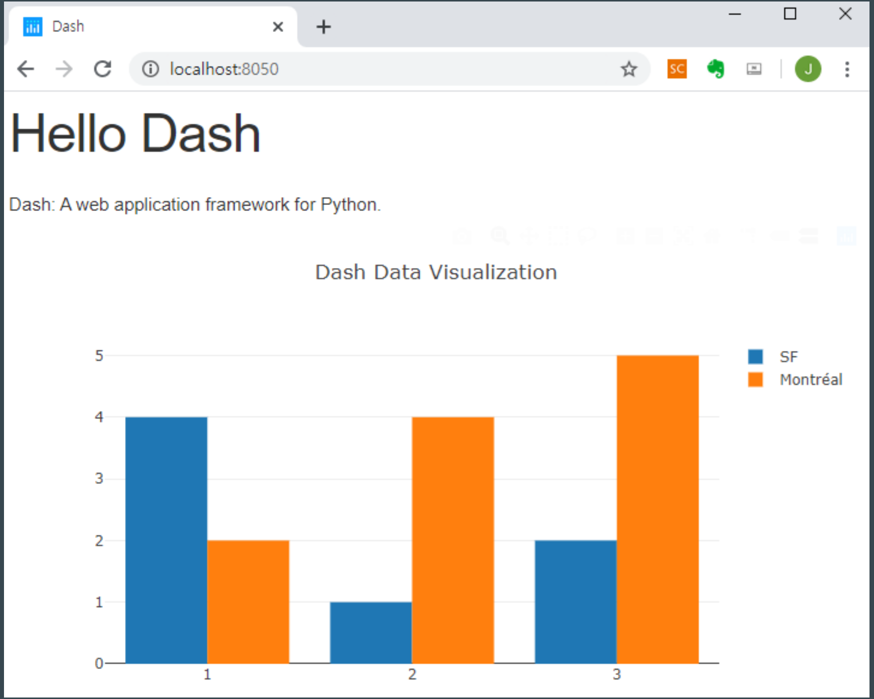
An example of using Dash with Flask

- Dash uses Flask for the server



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

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

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

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={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'title': 'Dash Data Visualization'
            }
        }
    )
])

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

Running Flask server



Dash

localhost:8050

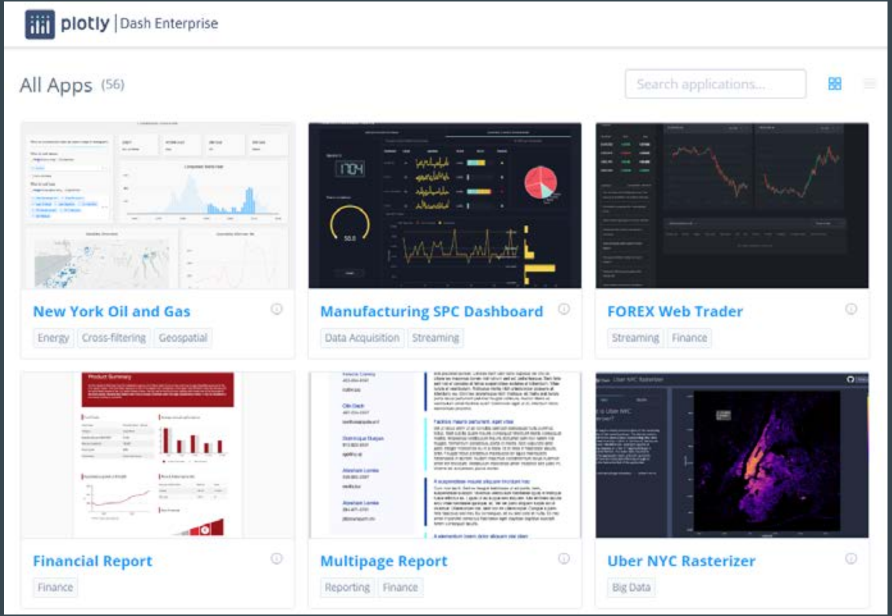
Hello Dash

Dash: A web application framework for Python.

https://dash.plotly.com/integrating-dash

Using Dash for data visualization

- Open source python framework for creating 'analytical' web application
- Integrating with Flask for building a variety of data-driven web applications
- Usually used with Plotly library and React.js for data analytics app (e.g., dashboard application)



A variety of examples of Dashboard applications using Dash with Plotly

https://dash-gallery.plotly.host/Portal/
https://medium.com/plotly/introducing-dash-5ecf7191b503

Goal

- Understand a basic operation of Dash framework
- Learn how to use Dash for developing interactive web applications
- Learn how to process and visualize data using Dash and Plotly
- Practice how to use Dash and Plotly for data visualization

Dash Basic

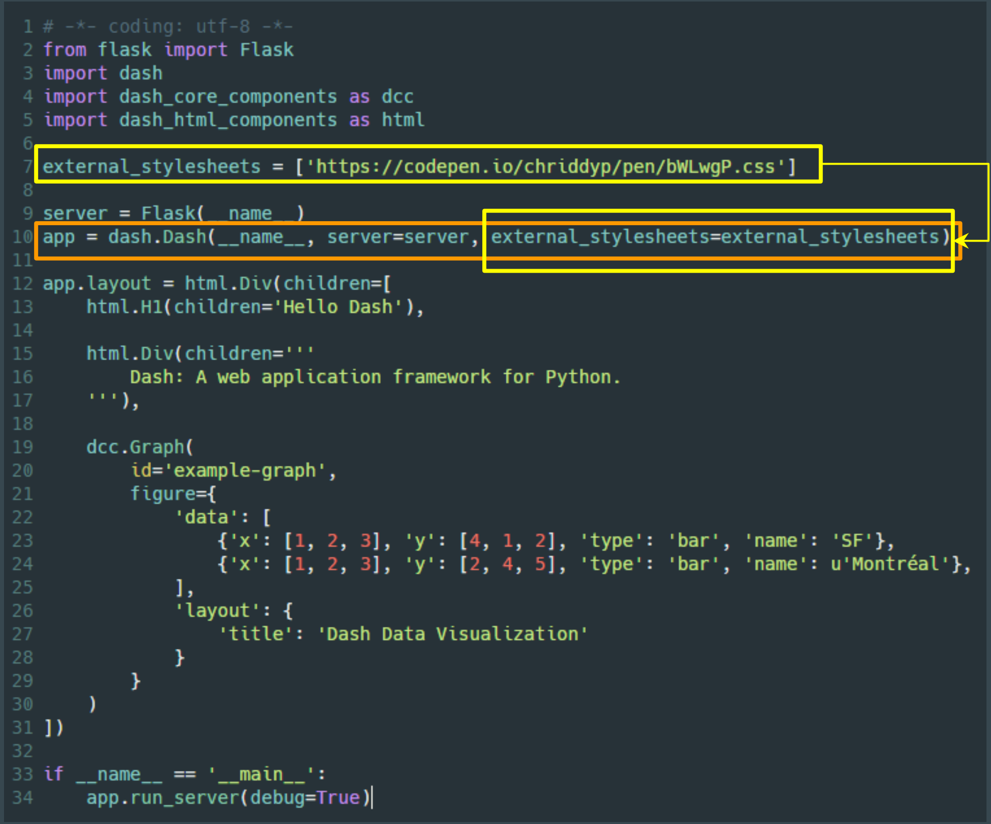
Practice scope

- Practice 1: Dash basic operation (displaying a static graph)
- Practice 2: Dash layout (Color change, resize, and align)
- Practice 3: Dash callback
- Practice 4: Interactive Graph with callback and data imports

https://dash.plotly.com/introduction

An example of Dash application

- Import an external CSS file to style web page
- Create a Dash app with importing the CSS file

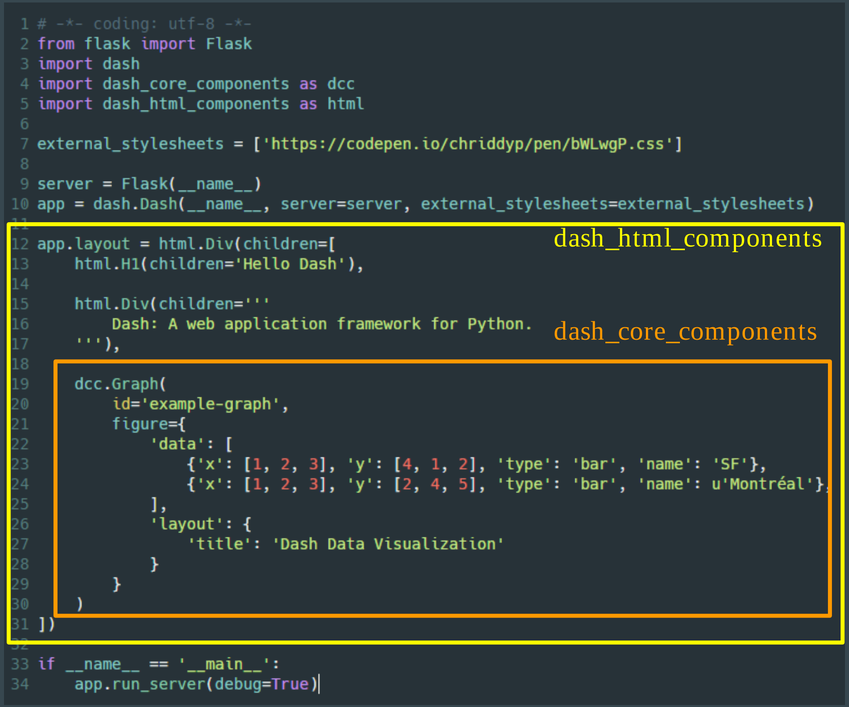


```python
# -*- coding: utf-8 -*-
from flask import Flask
import dash
import dash_core_components as dcc
import dash_html_components as html

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

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

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={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'title': 'Dash Data Visualization'
            }
        }
    )
])

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

https://dash.plotly.com/layout

An example of Dash application

- Using 'dash_core_components' to include and use Dash components (e.g., Graph) in a web page
- Using 'dash_html_components' to provide page layout information



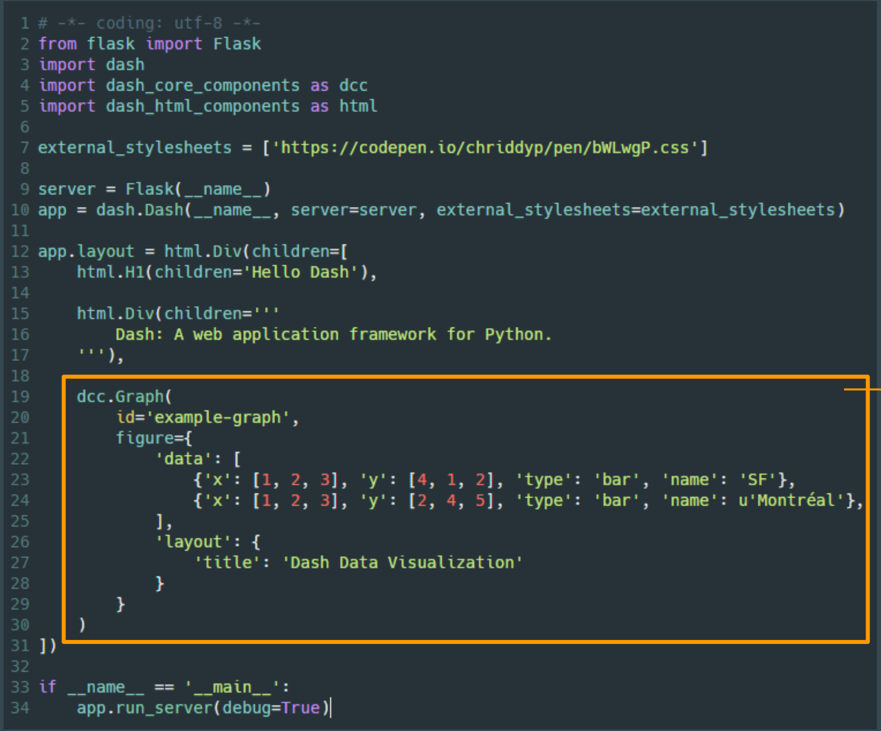
```python
# -*- coding: utf-8 -*-
from flask import Flask
import dash
import dash_core_components as dcc
import dash_html_components as html

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

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

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={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'title': 'Dash Data Visualization'
            }
        }
    )
])

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

```
<div>
  <h1> Hello Dash </h1>
   <div>
      Dash: A Web application framework for Python
    </div>
     {Graph}
</div>
```

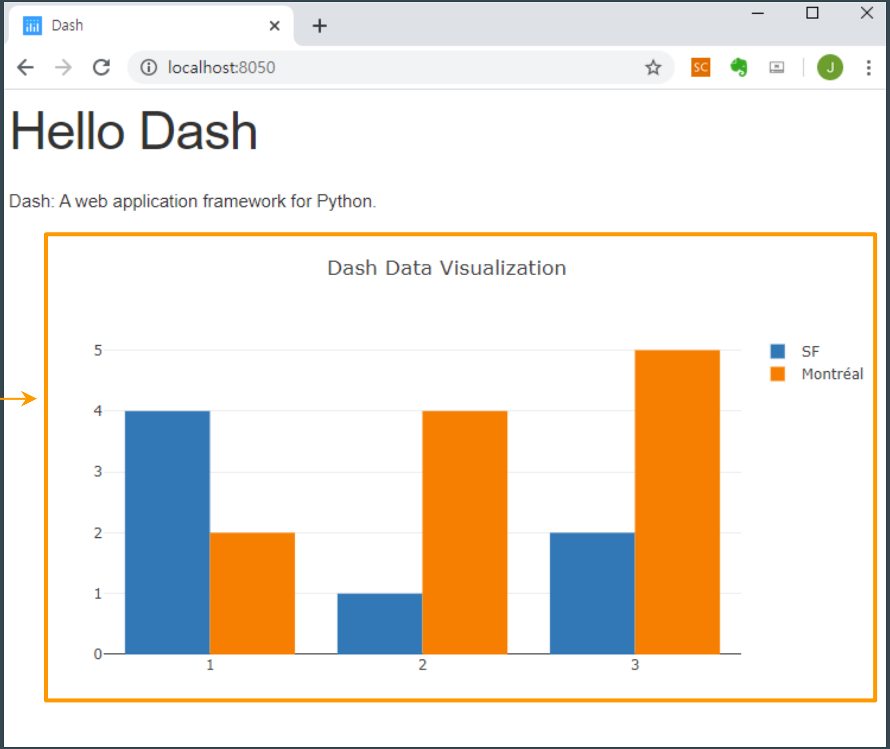
https://dash.plotly.com/layout

An example of Dash application

- Code result



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

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

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

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={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'title': 'Dash Data Visualization'
            }
        }
    )
])

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



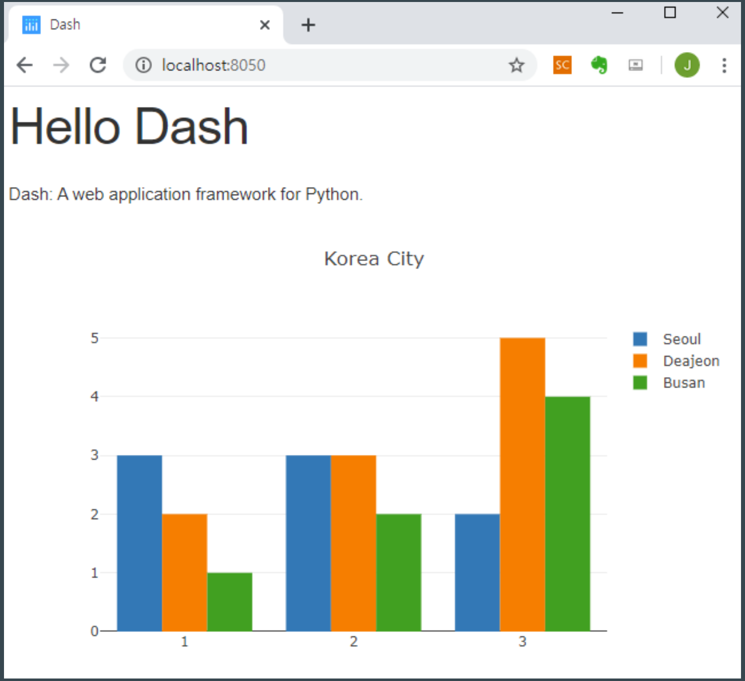
Hello Dash

Dash: A web application framework for Python.

https://dash.plotly.com/layout

Practice #1: Displaying 3 bar charts

- Display 3 bar charts
- Each chart name is the Korea city: Seoul(blue), Daejeon(orange), Busan(green)
- You may use this skeleton source code: link
- Check and upload your screenshot for grading: here



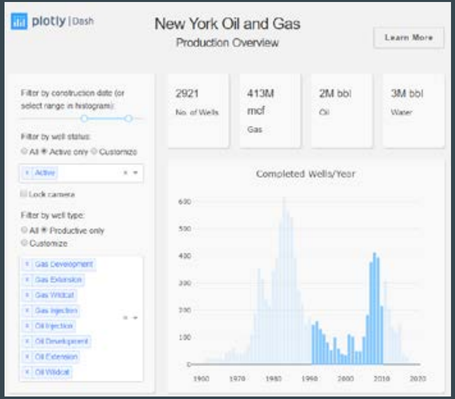
Hello Dash

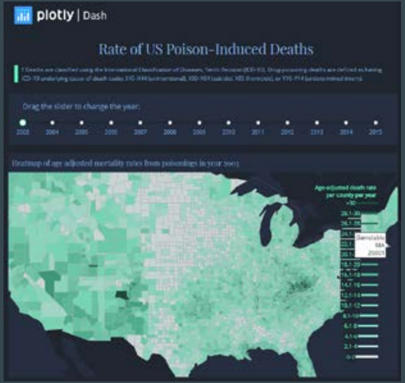
Dash: A web application framework for Python.

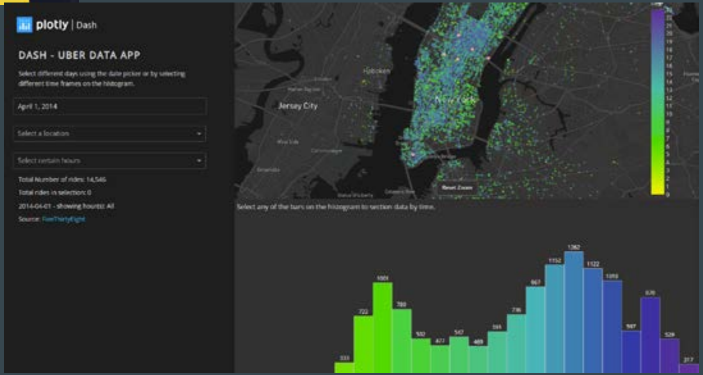
Dash Layout

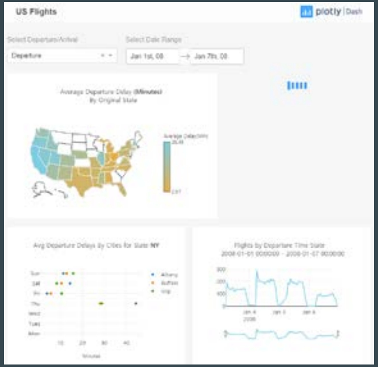
Before learning..

Many Dash layouts and styles available....




plotly | Dash
Rate of US Poison-Induced Deaths
Drag the slider to change the year:
Heatmap of age adjusted mortality rates from poisonings in year 2003


plotly | Dash
DASH - UBER DATA APP
Select different days using the date picker or by selecting different time frames on the histogram.
April 1, 2014
Select a location
Select certain hours
Total Number of rides: 14,546
Total rides in selection: 0
2014-04-01 - showing hour(s): All
Source: FiveThirtyEight
Jersey City
Select any of the bars on the histogram to section data by time.


US Flights
plotly | Dash
Departure
Jan 1st, 08
Jan 7th, 08
Average Departure Delay (Minutes) By Original State
Flights by Departure Time State

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

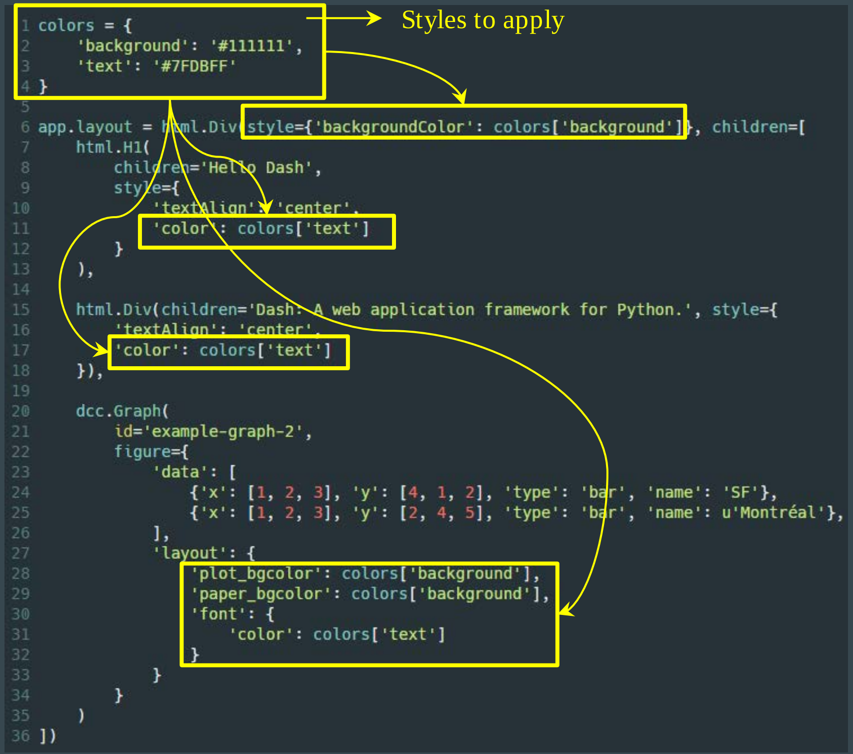
Practice scope

- Practice 1: Dash basic operation (displaying a static graph)
- Practice 2: Dash layout (Color change, resize, and align)
- Practice 3: Dash callback
- Practice 4: Interactive Graph with callback and data imports

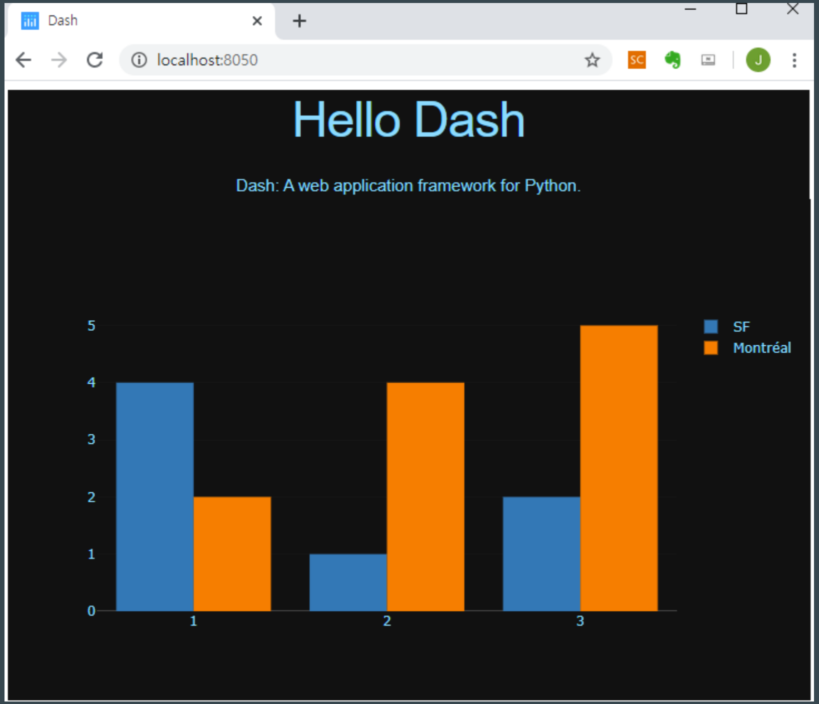
https://dash.plotly.com/introduction

Dash layout: Change color

- Applying color styles to Dash components



```python
colors = {
    'background': '#111111',
    'text': '#7FDBFF'
}

app.layout = html.Div(style={'backgroundColor': colors['background']}, children=[
    html.H1(
        children='Hello Dash',
        style={
            'textAlign': 'center',
            'color': colors['text']
        }
    ),

    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': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'plot_bgcolor': colors['background'],
                'paper_bgcolor': colors['background'],
                'font': {
                    'color': colors['text']
                }
            }
        }
    )
])
```



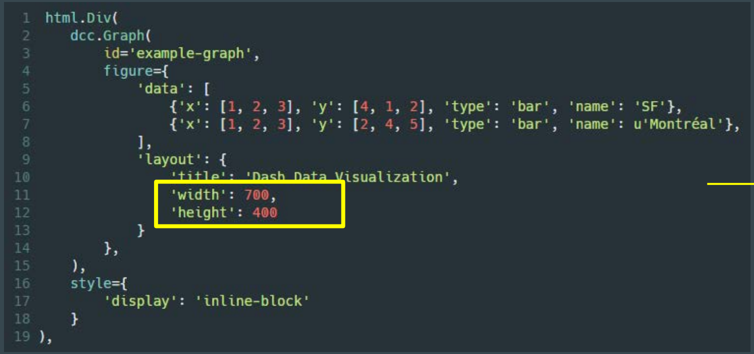
Hello Dash

Dash: A web application framework for Python.

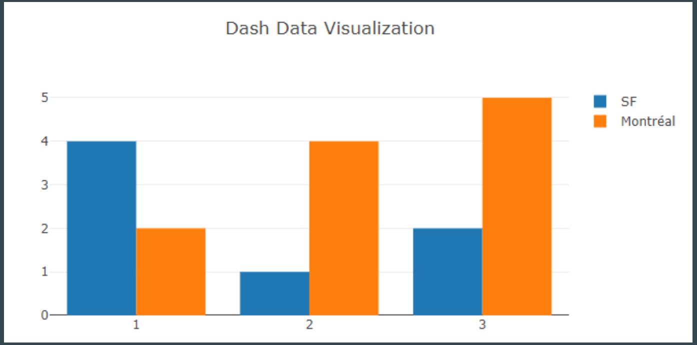
https://dash.plotly.com/layout

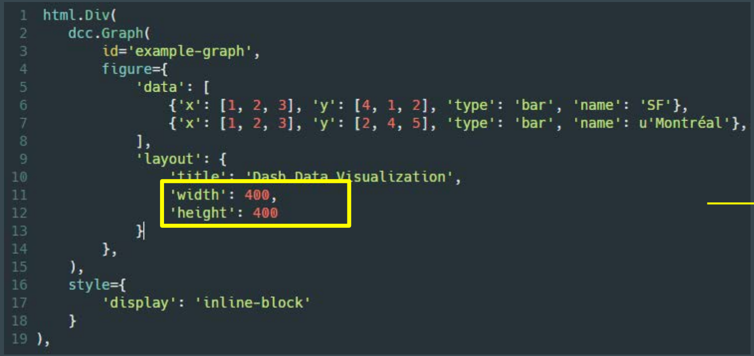
Dash layout: Resizing layout

- Using width, height of ‘layout’

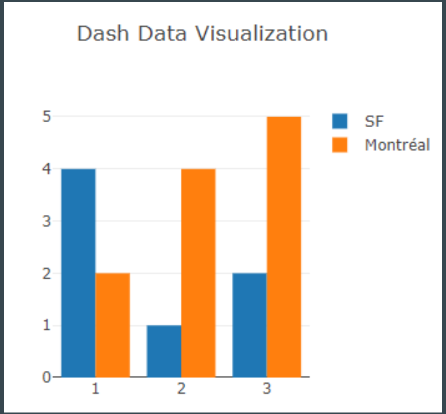


```
html.Div(
    dcc.Graph(
        id='example-graph',
        figure={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'title': 'Dash Data Visualization',
                'width': 700,
                'height': 400
            }
        },
    ),
    style={
        'display': 'inline-block'
    }
),
```





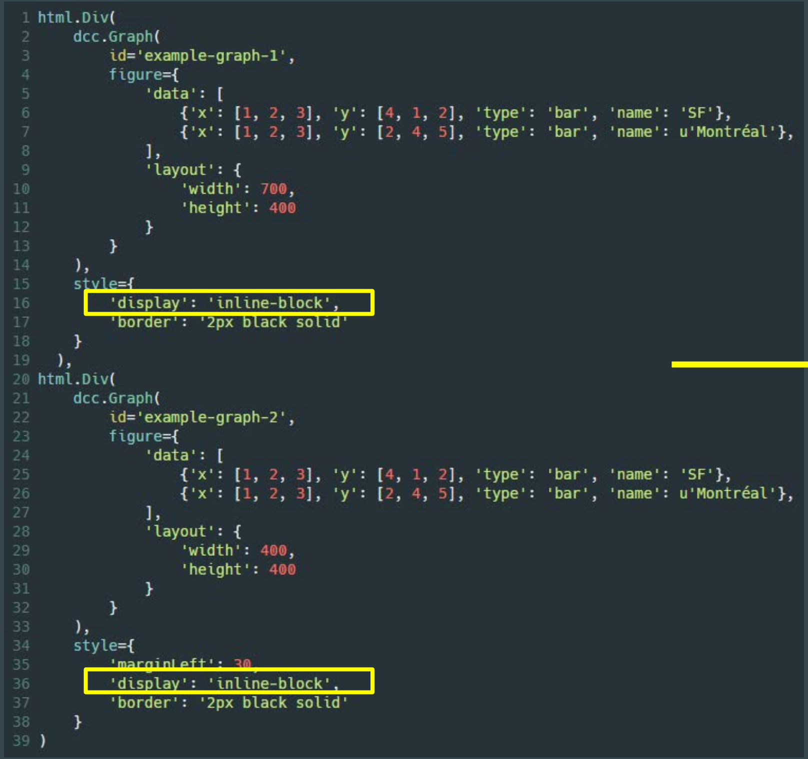
```
html.Div(
    dcc.Graph(
        id='example-graph',
        figure={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'title': 'Dash Data Visualization',
                'width': 400,
                'height': 400
            }
        },
    ),
    style={
        'display': 'inline-block'
    }
),
```



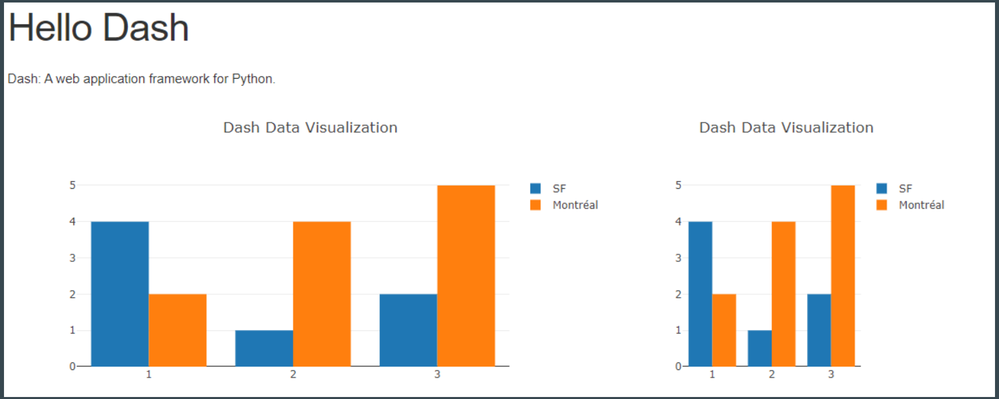
https://dash.plotly.com/layout

Dash layout: Aligning layout

- To arrange layout horizontally, apply ‘inline-block’ style to ‘<div>’



```
html.Div(
    dcc.Graph(
        id='example-graph-1',
        figure={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'width': 700,
                'height': 400
            }
        }
    ),
    style={
        'display': 'inline-block',
        'border': '2px black solid'
    }
  ),
html.Div(
    dcc.Graph(
        id='example-graph-2',
        figure={
            'data': [
                {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
            ],
            'layout': {
                'width': 400,
                'height': 400
            }
        }
    ),
    style={
        'marginLeft': 30,
        'display': 'inline-block',
        'border': '2px black solid'
    }
)
```

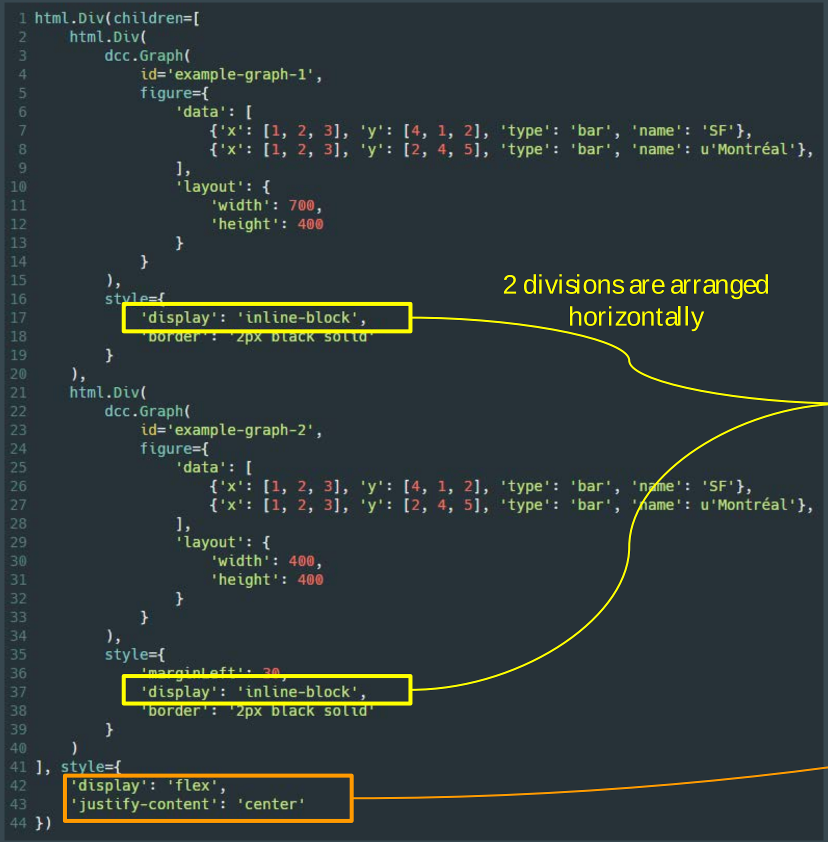


Hello Dash

Dash: A web application framework for Python.

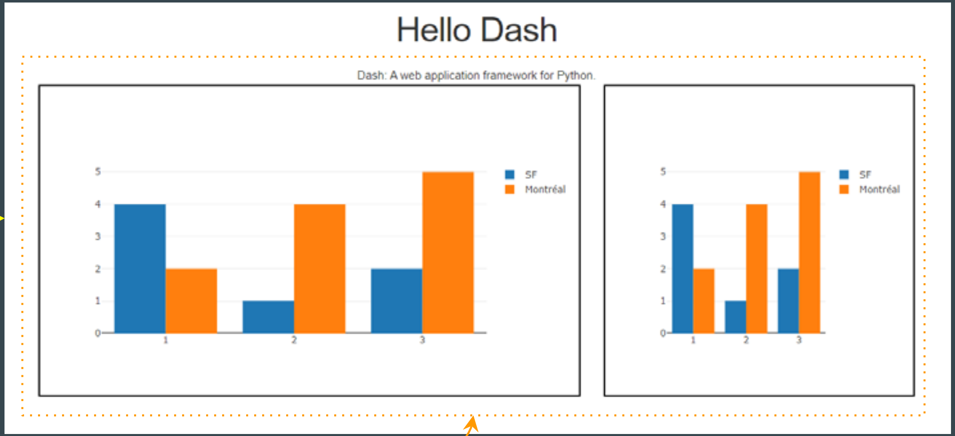
https://dash.plotly.com/layout

Dash layout: Aligning layout



```
html.Div(children=[
    html.Div(
        dcc.Graph(
            id='example-graph-1',
            figure={
                'data': [
                    {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                    {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
                ],
                'layout': {
                    'width': 700,
                    'height': 400
                }
            }
        ),
        style={
            'display': 'inline-block',
            'border': '2px black solid'
        }
    ),
    html.Div(
        dcc.Graph(
            id='example-graph-2',
            figure={
                'data': [
                    {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                    {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
                ],
                'layout': {
                    'width': 400,
                    'height': 400
                }
            }
        ),
        style={
            'marginLeft': 30,
            'display': 'inline-block',
            'border': '2px black solid'
        }
    )
], style={
    'display': 'flex',
    'justify-content': 'center'
})
```

2 divisions are arranged horizontally

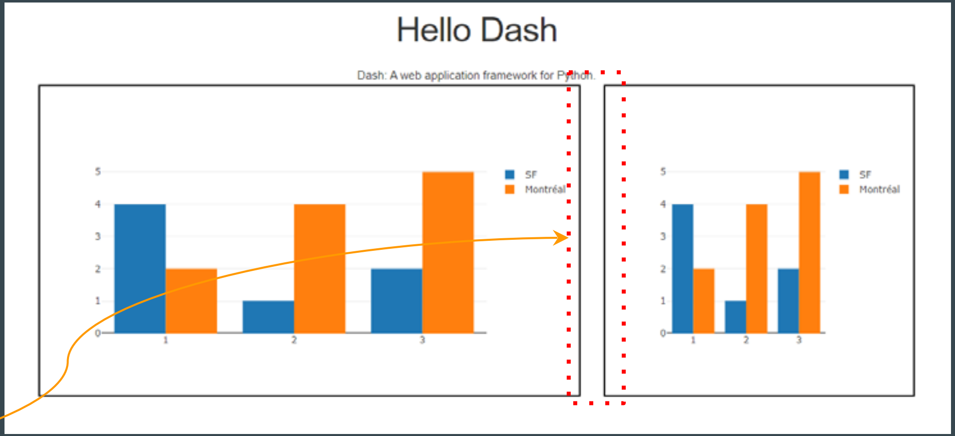


2 divisions are aligned center

https://dash.plotly.com/layout

Dash layout: Aligning layout

```python
html.Div(children=[
    html.Div(
        dcc.Graph(
            id='example-graph-1',
            figure={
                'data': [
                    {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                    {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
                ],
                'layout': {
                    'width': 700,
                    'height': 400
                }
            }
        ),
        style={
            'display': 'inline-block',
            'border': '2px black solid'
        }
    ),
    html.Div(
        dcc.Graph(
            id='example-graph-2',
            figure={
                'data': [
                    {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
                    {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': u'Montréal'},
                ],
                'layout': {
                    'width': 400,
                    'height': 400
                }
            }
        ),
        style={
            'marginLeft': 30,
            'display': 'inline-block',
            'border': '2px black solid'
        }
    )
], style={
    'display': 'flex',
    'justify-content': 'center'
})
```



You can have space with margin options (e.g., 'marginLeft': 30)

https://dash.plotly.com/layout

Practice #2: Changing Dash component styles

- Styling your Dash layout as below (Fig. result screenshot)
- Using color options, ‘inline-block’, and flex options (see pp.24)
- You may use this skeleton source code: link
- Check and upload your screenshot for grading: here

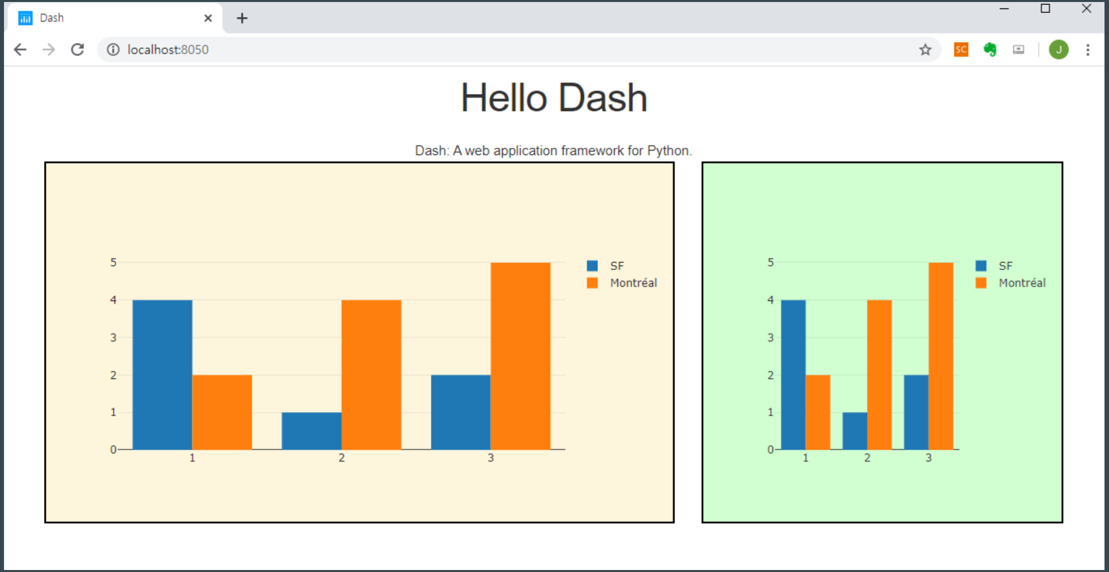
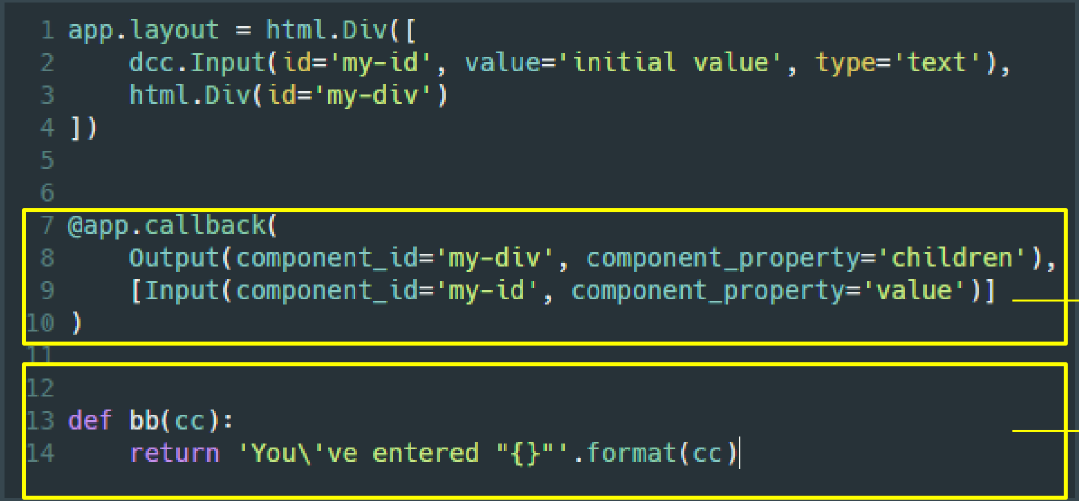


Fig. result screenshot

Dash Callback

Dash Callback: Basic callback

- There are 2 types of callback: **Input**, **Output**
- **Input** callback is used to receive and process a user's input data
- **Output** callback is used to display results of the process



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


@app.callback(
    Output(component_id='my-div', component_property='children'),
    [Input(component_id='my-id', component_property='value')]
)


def bb(cc):
    return 'You\'ve entered "{}"'.format(cc)
```

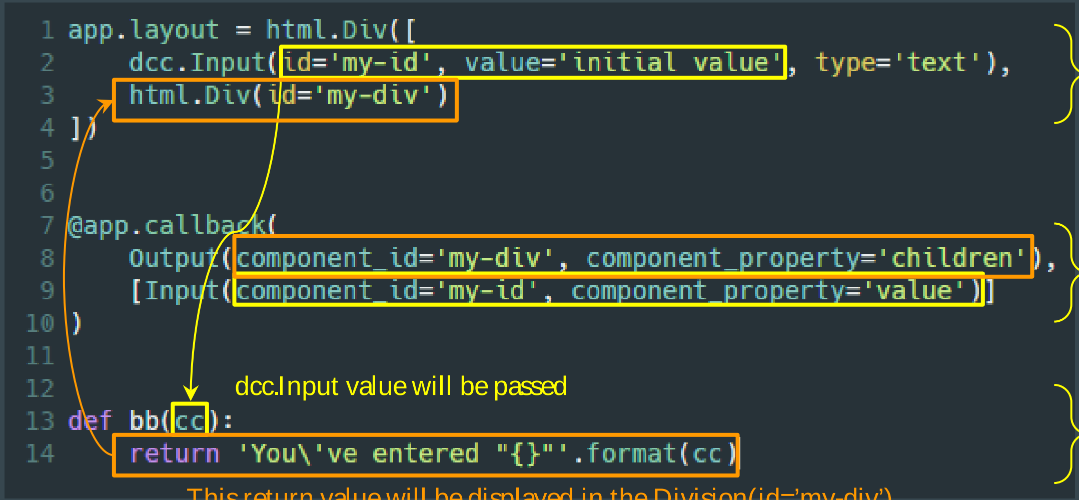
Callback module consists of descriptions of Input and Output parts

Callback function

https://dash.plotly.com/basic-callbacks

Dash Callback: Basic callback

- In the callback,
  - we declared that Input is the ‘value’ property of the component that has the ID ‘my-id’
  - our output is the ‘children’ property of the component with the ID ‘my-div’



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


@app.callback(
    Output(component_id='my-div', component_property='children'),
    [Input(component_id='my-id', component_property='value')]
)

def bb(cc):
    return 'You\'ve entered "{}"'.format(cc)
```

User interface parts to receive input and display output on the web page

Descriptions for input and output

dcc.Input value will be passed

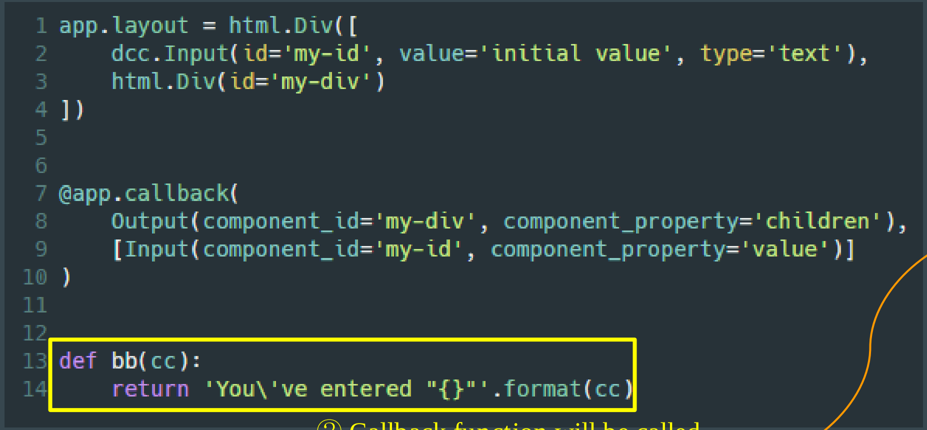
Whenever input occurs, it will be called!

This return value will be displayed in the Division(id=’my-div’)

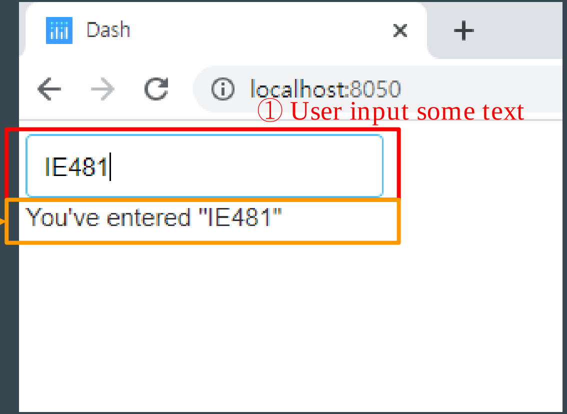
https://dash.plotly.com/basic-callbacks

Dash Callback: Basic callback

- In the callback,
  - we declared that Input is the 'value' property of the component that has the ID 'my-id'
  - our output is the 'children' property of the component with the ID 'my-div'



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


@app.callback(
    Output(component_id='my-div', component_property='children'),
    [Input(component_id='my-id', component_property='value')]
)


def bb(cc):
    return 'You\'ve entered "{}"'.format(cc)
```



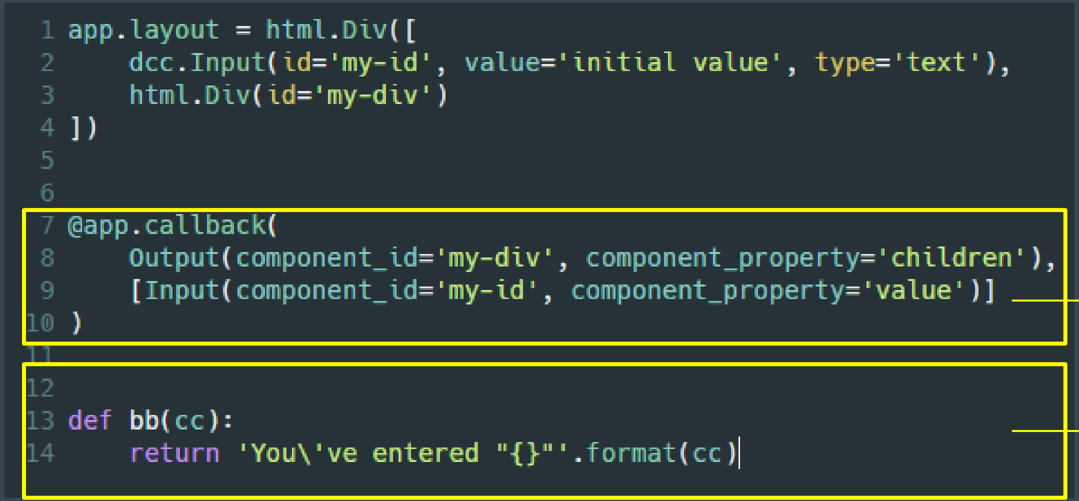
You've entered "IE481"

② Callback function will be called
③ Callback function returns results

https://dash.plotly.com/basic-callbacks

Dash Callback: Basic callback

- There are 2 types of callback: **Input**, **Output**
- **Input** callback is used to receive and process user's input data
- **Output** callback is used to display results of the process



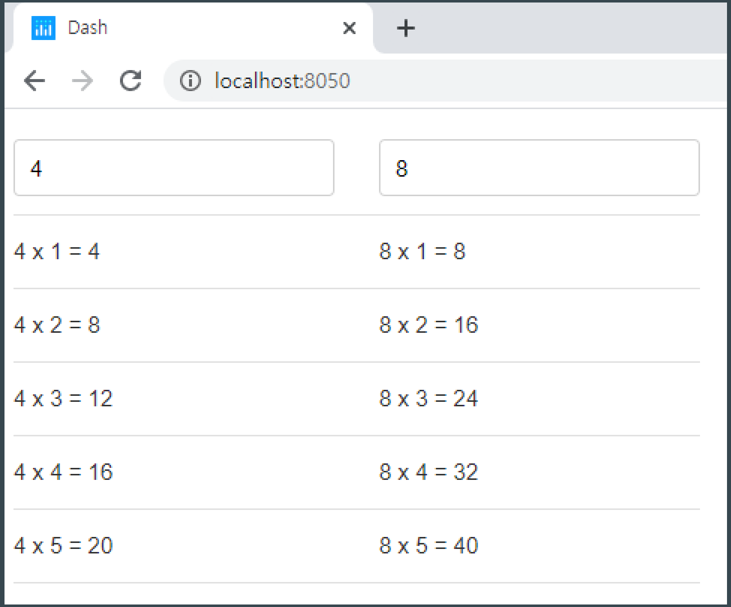
```
app.layout = html.Div([
    dcc.Input(id='my-id', value='initial value', type='text'),
    html.Div(id='my-div')
])


@app.callback(
    Output(component_id='my-div', component_property='children'),
    [Input(component_id='my-id', component_property='value')]
)


def bb(cc):
    return 'You\'ve entered "{}"'.format(cc)
```

Callback module consists of descriptions of Input and Output parts
Callback function

https://dash.plotly.com/basic-callbacks

Practice #3: Multiple input/output callback

- Design and build multiple input/output callback for multiplication table
- Just display 5 lines only (see Fig.)
- You may use this skeleton source code: link
- Check and upload your screenshot for grading: here



4	8
4 x 1 = 4	8 x 1 = 8
4 x 2 = 8	8 x 2 = 16
4 x 3 = 12	8 x 3 = 24
4 x 4 = 16	8 x 4 = 32
4 x 5 = 20	8 x 5 = 40

Fig. result screenshot

Interactive Graph

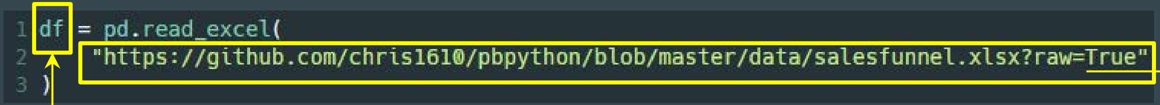
Import data

+

Callback

Import data (.xlsx data)

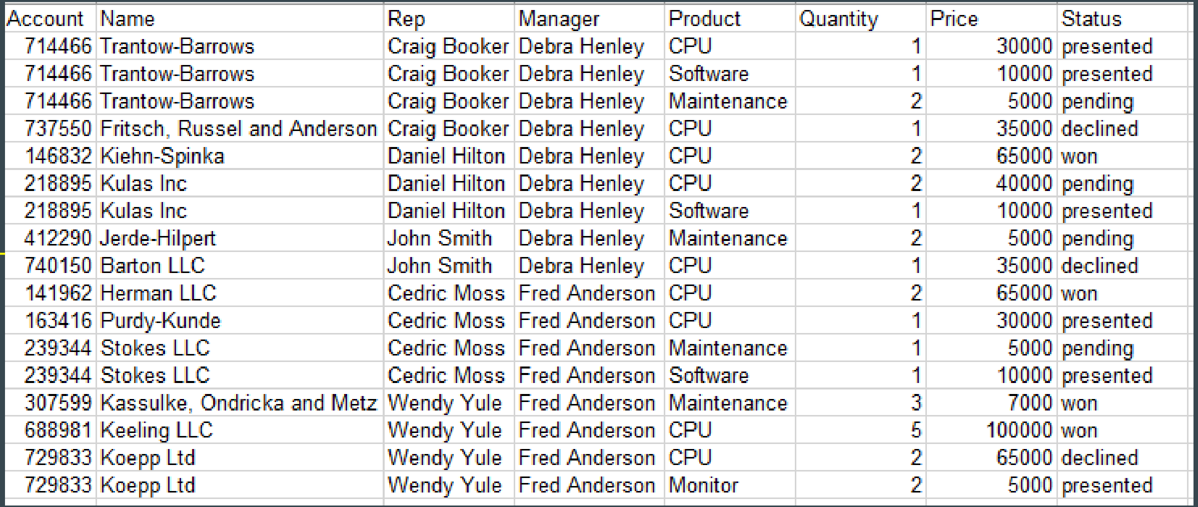
- Step 1: Install ‘xlrd’ python package (command ‘pip install xlrd’)
- Step 2: Read excel file using ‘pandas’ module



```
df = pd.read_excel(
  "https://github.com/chris1610/pbpython/blob/master/data/salesfunnel.xlsx?raw=True"
)
```

Remote file

Later, we can access the file like
df[‘Manager’]
df[‘Status’]
df[‘Name’]

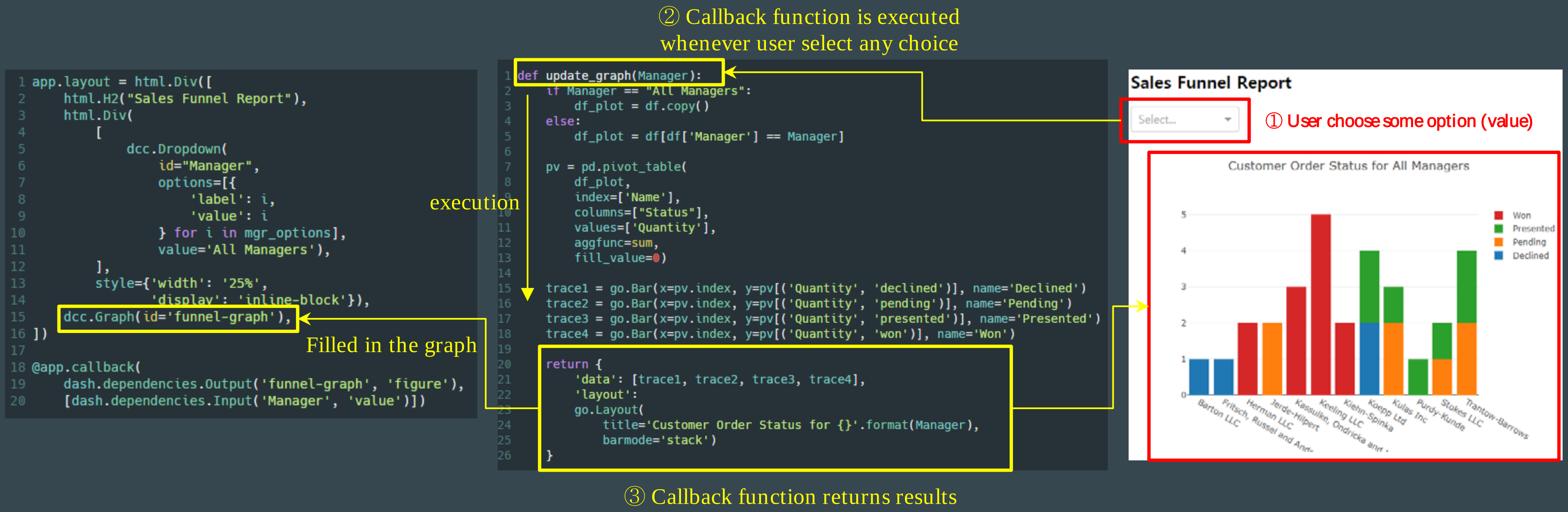


Account	Name	Rep	Manager	Product	Quantity	Price	Status
714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won
218895	Kulas Inc	Daniel Hilton	Debra Henley	CPU	2	40000	pending
218895	Kulas Inc	Daniel Hilton	Debra Henley	Software	1	10000	presented
412290	Jerde-Hilpert	John Smith	Debra Henley	Maintenance	2	5000	pending
740150	Barton LLC	John Smith	Debra Henley	CPU	1	35000	declined
141962	Herman LLC	Cedric Moss	Fred Anderson	CPU	2	65000	won
163416	Purdy-Kunde	Cedric Moss	Fred Anderson	CPU	1	30000	presented
239344	Stokes LLC	Cedric Moss	Fred Anderson	Maintenance	1	5000	pending
239344	Stokes LLC	Cedric Moss	Fred Anderson	Software	1	10000	presented
307599	Kassulke, Ondricka and Metz	Wendy Yule	Fred Anderson	Maintenance	3	7000	won
688981	Keeling LLC	Wendy Yule	Fred Anderson	CPU	5	100000	won
729833	Koepp Ltd	Wendy Yule	Fred Anderson	CPU	2	65000	declined
729833	Koepp Ltd	Wendy Yule	Fred Anderson	Monitor	2	5000	presented

https://github.com/chris1610/pbpython

Callback operation

- Step 1: User choose some value in the Dropdown interface
- Step 2: Callback function is executed → Executing callback body → Return results (display outputs)



② Callback function is executed whenever user select any choice

```python
app.layout = html.Div([
    html.H2("Sales Funnel Report"),
    html.Div(
        [
            dcc.Dropdown(
                id="Manager",
                options=[{
                    'label': i,
                    'value': i
                } for i in mgr_options],
                value='All Managers'),
        ],
        style={'width': '25%',
               'display': 'inline-block'}),
    dcc.Graph(id='funnel-graph'),
])

@app.callback(
    dash.dependencies.Output('funnel-graph', 'figure'),
    [dash.dependencies.Input('Manager', 'value')])
```

```python
def update_graph(Manager):
    if Manager == "All Managers":
        df_plot = df.copy()
    else:
        df_plot = df[df['Manager'] == Manager]

    pv = pd.pivot_table(
        df_plot,
        index=['Name'],
        columns=["Status"],
        values=['Quantity'],
        aggfunc=sum,
        fill_value=0)

    trace1 = go.Bar(x=pv.index, y=pv[('Quantity', 'declined')], name='Declined')
    trace2 = go.Bar(x=pv.index, y=pv[('Quantity', 'pending')], name='Pending')
    trace3 = go.Bar(x=pv.index, y=pv[('Quantity', 'presented')], name='Presented')
    trace4 = go.Bar(x=pv.index, y=pv[('Quantity', 'won')], name='Won')

    return {
        'data': [trace1, trace2, trace3, trace4],
        'layout':
        go.Layout(
            title='Customer Order Status for {}'.format(Manager),
            barmode='stack')
    }
```

execution

Filled in the graph

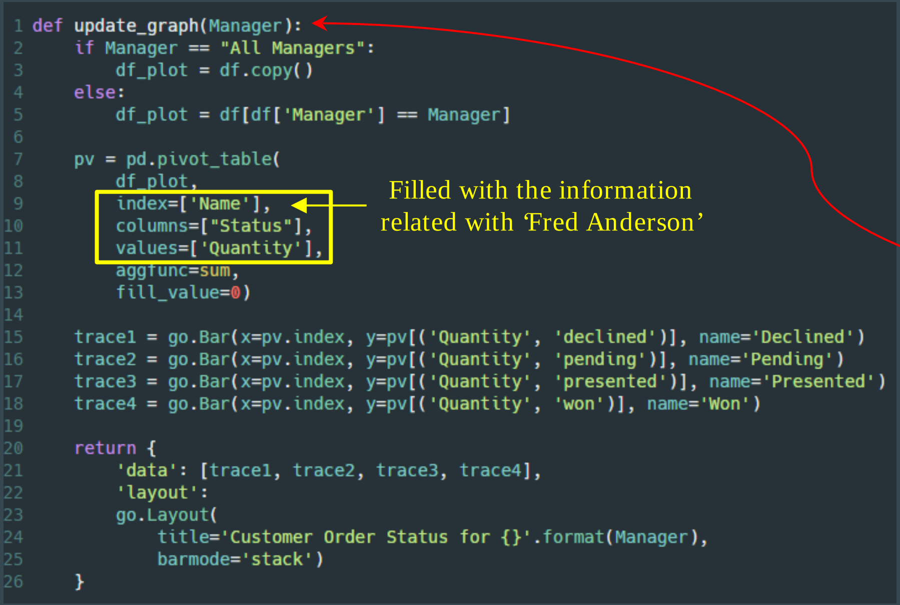
① User choose some option (value)

③ Callback function returns results

https://github.com/chris1610/pbpython

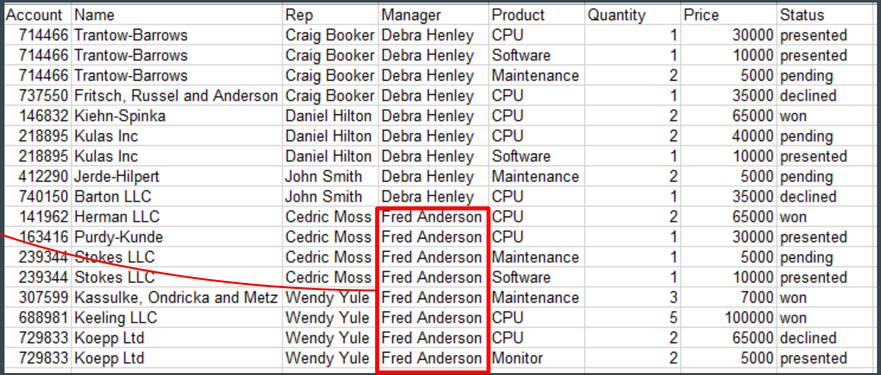
Detailed callback operation (in this example)

- Step 1: User choose one of manager (one of between 'Debra Henley' and 'Fred Anderson') (Callback Input)
- Step 2: Based on the 'Manager' information from step 1, a pivot_table will be filled with related information
  - E.g., df['Name'] → index, df['Status'] → columns



```
def update_graph(Manager):
    if Manager == "All Managers":
        df_plot = df.copy()
    else:
        df_plot = df[df['Manager'] == Manager]

    pv = pd.pivot_table(
        df_plot,
        index=['Name'],
        columns=["Status"],
        values=['Quantity'],
        aggfunc=sum,
        fill_value=0)

    trace1 = go.Bar(x=pv.index, y=pv[('Quantity', 'declined')], name='Declined')
    trace2 = go.Bar(x=pv.index, y=pv[('Quantity', 'pending')], name='Pending')
    trace3 = go.Bar(x=pv.index, y=pv[('Quantity', 'presented')], name='Presented')
    trace4 = go.Bar(x=pv.index, y=pv[('Quantity', 'won')], name='Won')

    return {
        'data': [trace1, trace2, trace3, trace4],
        'layout':
        go.Layout(
            title='Customer Order Status for {}'.format(Manager),
            barmode='stack')
    }
```

Filled with the information related with 'Fred Anderson'



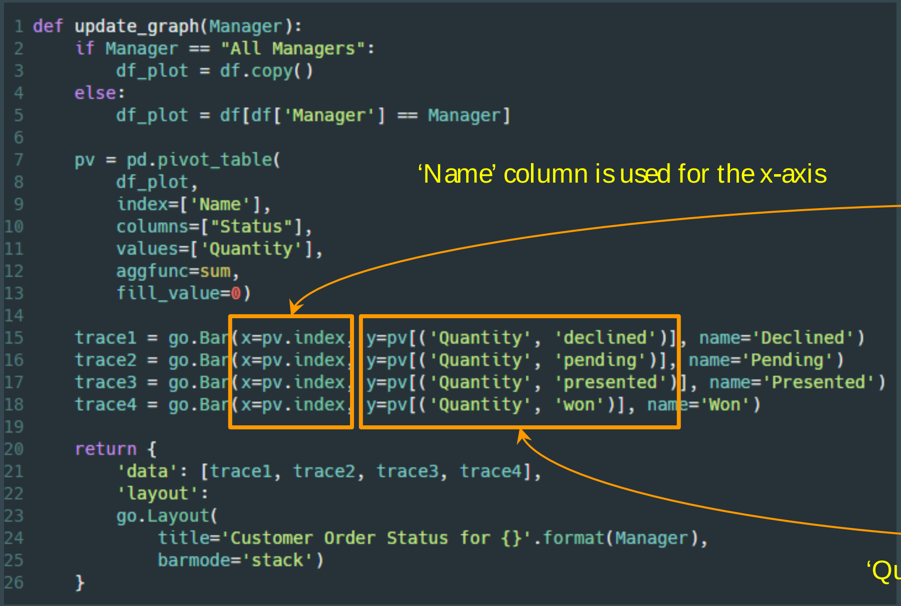
Account	Name	Rep	Manager	Product	Quantity	Price	Status
714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won
218895	Kulas Inc	Daniel Hilton	Debra Henley	CPU	2	40000	pending
218895	Kulas Inc	Daniel Hilton	Debra Henley	Software	1	10000	presented
412290	Jerde-Hilpert	John Smith	Debra Henley	Maintenance	2	5000	pending
740150	Barton LLC	John Smith	Debra Henley	CPU	1	35000	declined
141962	Herman LLC	Cedric Moss	Fred Anderson	CPU	2	65000	won
163416	Purdy-Kunde	Cedric Moss	Fred Anderson	CPU	1	30000	presented
239344	Stokes LLC	Cedric Moss	Fred Anderson	Maintenance	1	5000	pending
239344	Stokes LLC	Cedric Moss	Fred Anderson	Software	1	10000	presented
307599	Kassulke, Ondricka and Metz	Wendy Yule	Fred Anderson	Maintenance	3	7000	won
688981	Keeling LLC	Wendy Yule	Fred Anderson	CPU	5	100000	won
729833	Koepp Ltd	Wendy Yule	Fred Anderson	CPU	2	65000	declined
729833	Koepp Ltd	Wendy Yule	Fred Anderson	Monitor	2	5000	presented

Assuming that a user choose the 'Fred Anderson' in the Dropdown interface

https://github.com/chris1610/pbpython

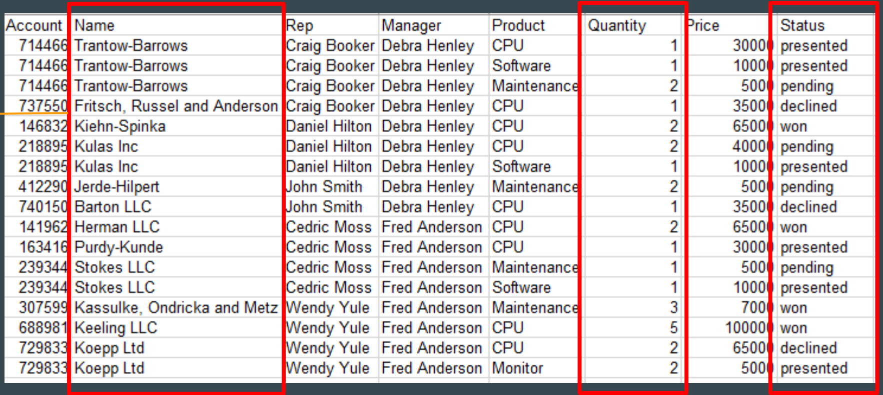
Detailed callback operation (in this example)

- Step 3: Using Plotly graph objects, information of x-axis and y-axis is filled
- Step 4: Return the completed Plotly graph objects (here, ‘trace1’ ~ ‘trace2’) (Callback Output)



```python
def update_graph(Manager):
    if Manager == "All Managers":
        df_plot = df.copy()
    else:
        df_plot = df[df['Manager'] == Manager]

    pv = pd.pivot_table(
        df_plot,
        index=['Name'],
        columns=["Status"],
        values=['Quantity'],
        aggfunc=sum,
        fill_value=0)

    trace1 = go.Bar(x=pv.index, y=pv[('Quantity', 'declined')], name='Declined')
    trace2 = go.Bar(x=pv.index, y=pv[('Quantity', 'pending')], name='Pending')
    trace3 = go.Bar(x=pv.index, y=pv[('Quantity', 'presented')], name='Presented')
    trace4 = go.Bar(x=pv.index, y=pv[('Quantity', 'won')], name='Won')

    return {
        'data': [trace1, trace2, trace3, trace4],
        'layout':
        go.Layout(
            title='Customer Order Status for {}'.format(Manager),
            barmode='stack')
    }
```

‘Name’ column is used for the x-axis



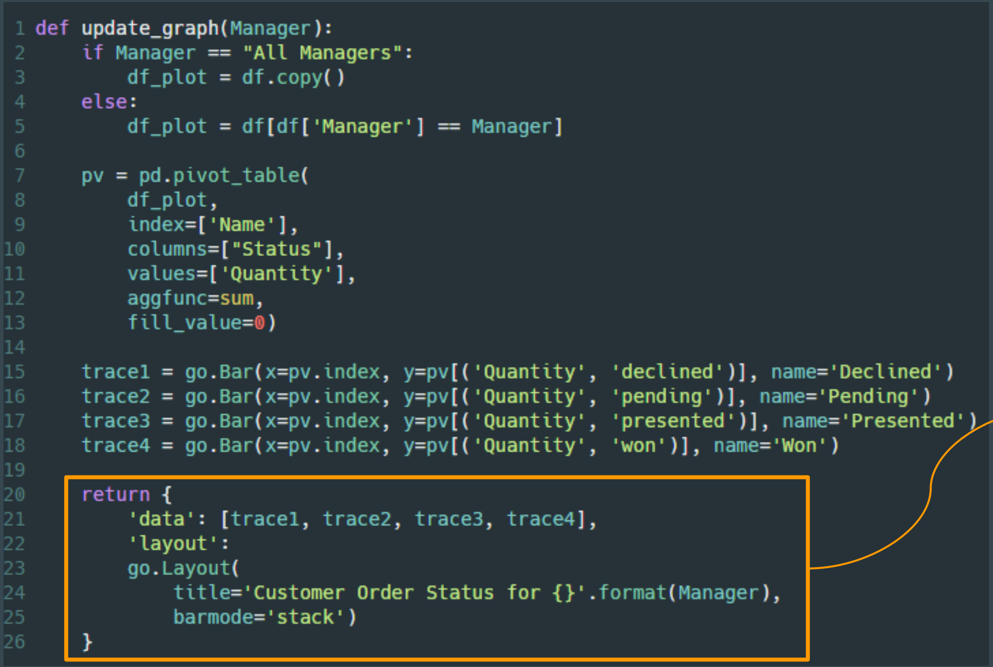
Account	Name	Rep	Manager	Product	Quantity	Price	Status
714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won
218895	Kulas Inc	Daniel Hilton	Debra Henley	CPU	2	40000	pending
218895	Kulas Inc	Daniel Hilton	Debra Henley	Software	1	10000	presented
412290	Jerde-Hilpert	John Smith	Debra Henley	Maintenance	2	5000	pending
740150	Barton LLC	John Smith	Debra Henley	CPU	1	35000	declined
141962	Herman LLC	Cedric Moss	Fred Anderson	CPU	2	65000	won
163416	Purdy-Kunde	Cedric Moss	Fred Anderson	CPU	1	30000	presented
239344	Stokes LLC	Cedric Moss	Fred Anderson	Maintenance	1	5000	pending
239344	Stokes LLC	Cedric Moss	Fred Anderson	Software	1	10000	presented
307599	Kassulke, Ondricka and Metz	Wendy Yule	Fred Anderson	Maintenance	3	7000	won
688981	Keeling LLC	Wendy Yule	Fred Anderson	CPU	5	100000	won
729833	Koepp Ltd	Wendy Yule	Fred Anderson	CPU	2	65000	declined
729833	Koepp Ltd	Wendy Yule	Fred Anderson	Monitor	2	5000	presented

‘Quantity’ of each ‘Status’ put together in y-axis

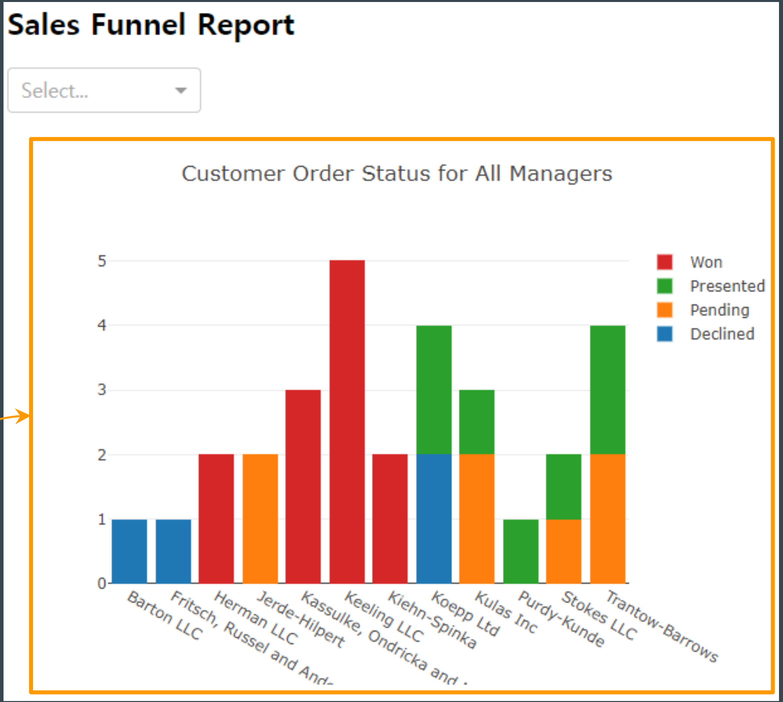
https://github.com/chris1610/pbpython

Detailed callback operation (in this example)

- Step 5: Display the Plotly graph objects to the web page



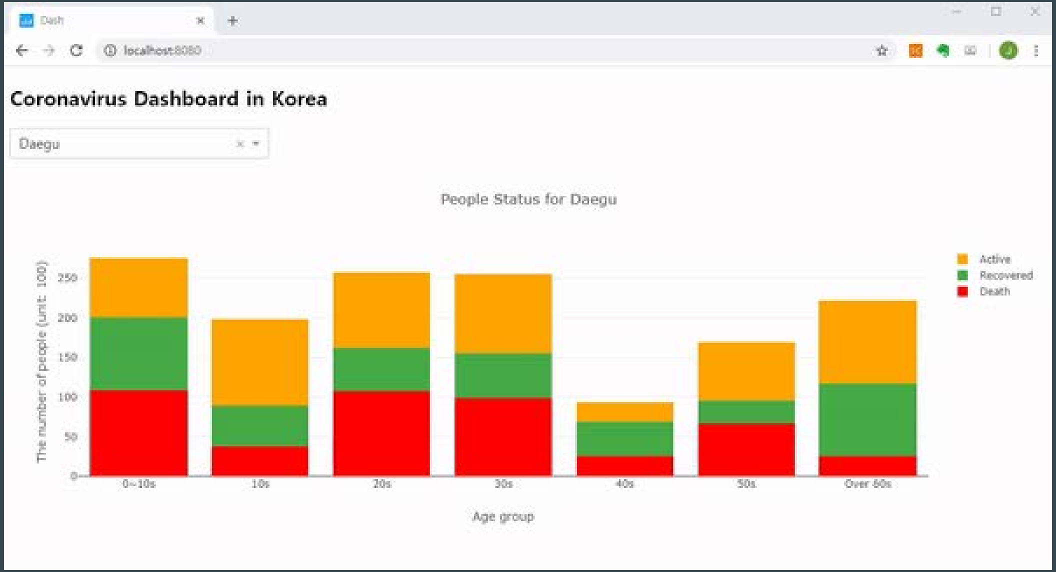
```python
def update_graph(Manager):
    if Manager == "All Managers":
        df_plot = df.copy()
    else:
        df_plot = df[df['Manager'] == Manager]

    pv = pd.pivot_table(
        df_plot,
        index=['Name'],
        columns=["Status"],
        values=['Quantity'],
        aggfunc=sum,
        fill_value=0)

    trace1 = go.Bar(x=pv.index, y=pv[('Quantity', 'declined')], name='Declined')
    trace2 = go.Bar(x=pv.index, y=pv[('Quantity', 'pending')], name='Pending')
    trace3 = go.Bar(x=pv.index, y=pv[('Quantity', 'presented')], name='Presented')
    trace4 = go.Bar(x=pv.index, y=pv[('Quantity', 'won')], name='Won')

    return {
        'data': [trace1, trace2, trace3, trace4],
        'layout':
        go.Layout(
            title='Customer Order Status for {}'.format(Manager),
            barmode='stack')
    }
```



https://github.com/chris1610/pbpython

Programming assignment #3: COVID-19 status in Korea

- Design dropdown interface to select a city of Korea
- Display the status of disease of each city according to choosing the city in the dropdown
- Insert this COVID-19 status dashboard in your homepage and host it in Heroku
- More detailed instructions will be delivered with dataset after class


Dash
localhost:8080
Coronavirus Dashboard in Korea
Daegu
People Status for Daegu
Active
Recovered
Death
The number of people (unit: 100)
0
50
100
150
200
250
0~10s
10s
20s
30s
40s
50s
Over 60s
Age group