Dash Components

Objectives

After completing the lab, you will be able to:

- Work with Dash Callbacks

Estimated time needed: 45 minutes

Dataset Used

Airline Reporting Carrier On-Time Performance dataset from Data Asset eXchange

About Skills Network Cloud IDE

This Skills Network Labs Cloud IDE (Integrated Development Environment) provides a hands-on environment in your web browser for completing the course and project-related labs. It utilizes Theia, an open-source IDE platform that can be run on a desktop or on the cloud.
So far in the course, you have been using Jupyter Notebooks to run your Python code. This IDE provides an alternative for editing and running your Python code. In this lab, you will be using this alternative Python runtime to create and launch your Dash applications.

Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. When you launch the Cloud IDE, you are presented with a 'dedicated computer on the cloud' exclusively for you. This is available to you as long as you are actively working on the labs.

Once you close your session or it is timed out due to inactivity, you are logged off, and this 'dedicated computer on the cloud' is deleted along with any files you may have created, downloaded or installed. The next time you launch this lab, a new environment will be created for you.

If you finish only part of the lab and return later, you may have to start from the beginning. So, it is a good idea to plan your time accordingly and finish your labs in a single session.

Let's start creating dash application

Theme

Extract the average monthly arrival delay time and see how it changes over the year. The year range is from 2010 to 2020.

Expected Output

Below is the expected result from the lab. Our dashboard application consists of three components:

- Title of the application
- Component to enter input year
- Chart conveying the average monthly arrival delay

Tasks To do:

1. Task 1
   - Import required libraries
   - Read the dataset
2. Task 2
   - Create an application layout
   - Add title to the dashboard application using HTML H1 component
3. Task 3
   - Add input text box using core input and output components
   - Add the callback decorator and generate the line chart within the callback function.
4. Task 4
   - Practice Exercise

Get the tool ready

- Install python packages required to run the application. Copy and paste the below command to the terminal.

1. 1

1. `python3 -m pip install packaging`

Copied!

1. 1

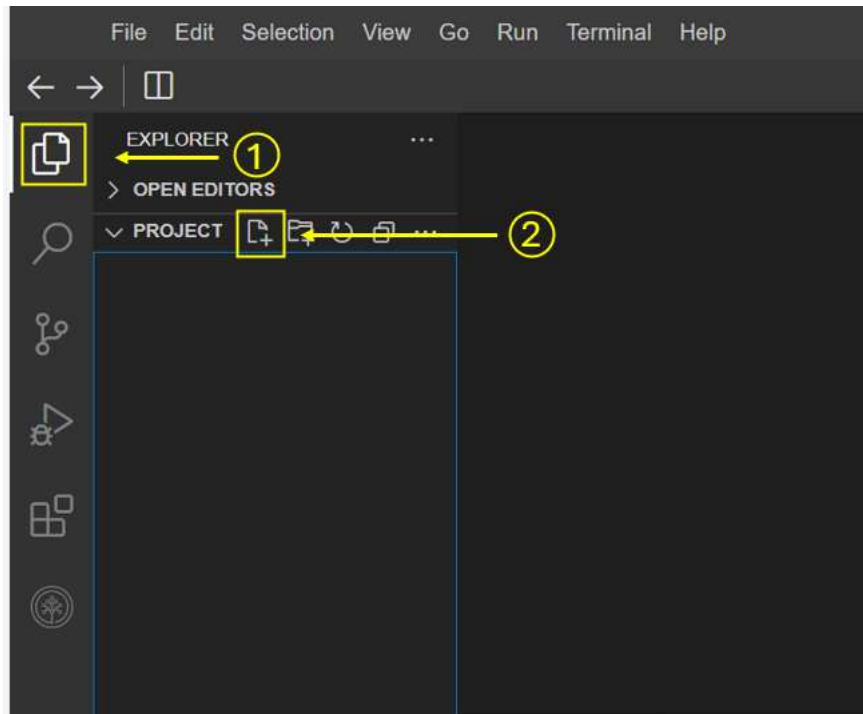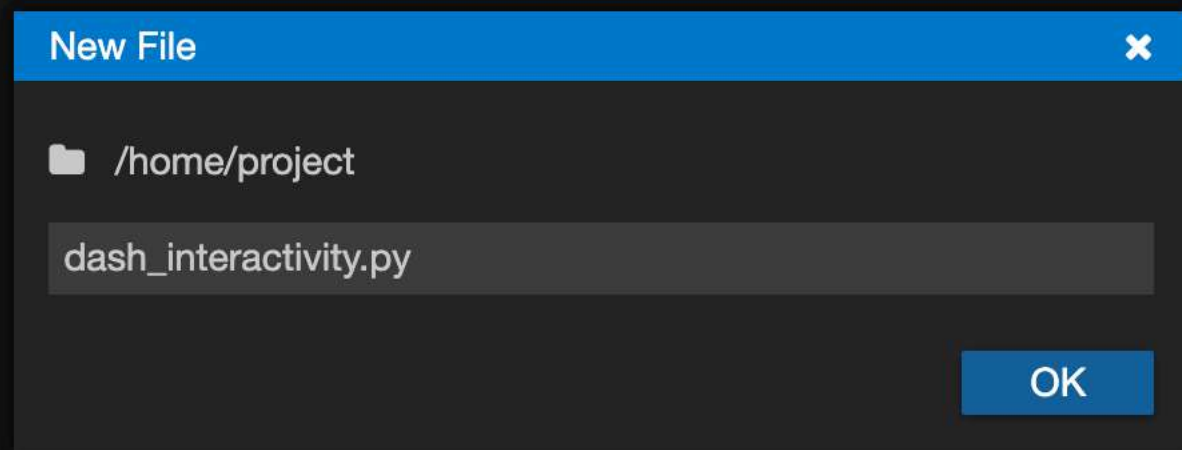1. `python3 -m pip install pandas dash`

Copied!

1. 1

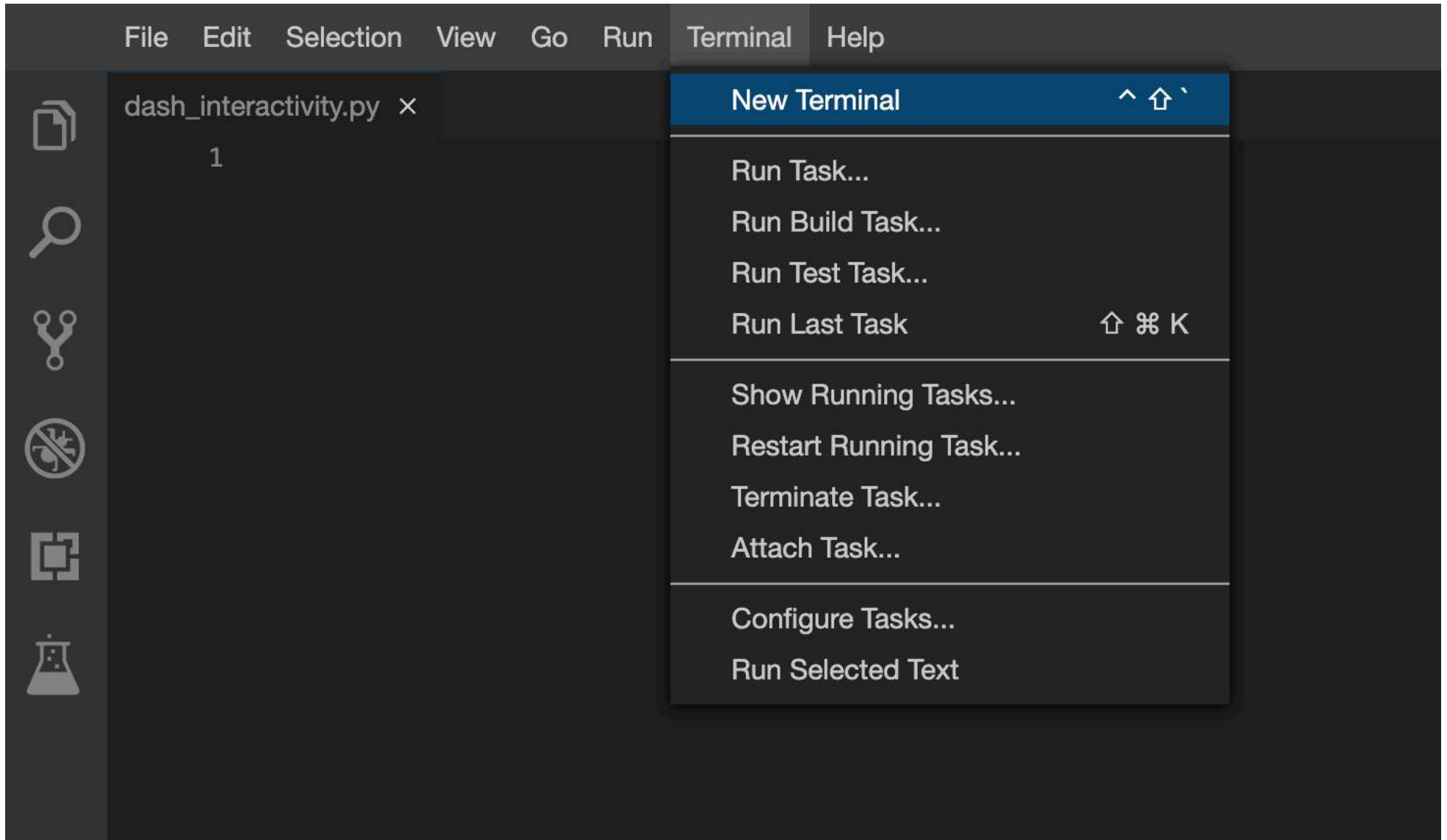1. `pip3 install httpx==0.20 dash plotly`

Copied!

- Create a new python script, by clicking on the side tool bar **explorer** icon and selecting **new file** icon, as shown in the image below.
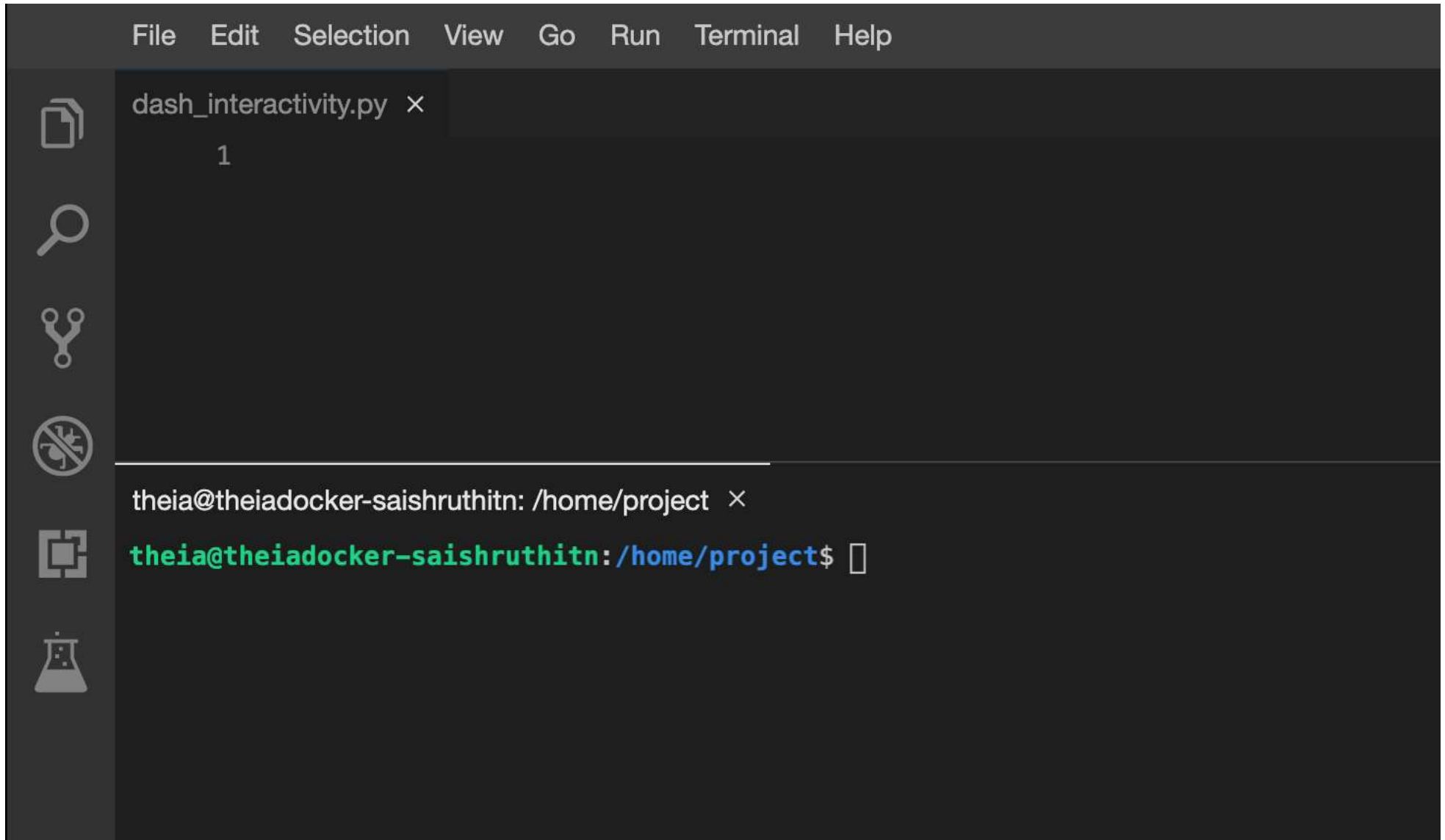

- Provide the file name as `dash_interactivity.py`

- Open a new terminal by clicking on the menu bar and selecting **Terminal->New Terminal**, as in the image below.

- Now, you have script and terminal ready to start the lab.

```
File    Edit    Selection    View    Go    Run    Terminal    Help

dash_interactivity.py  ×

        1
```

```
theia@theiadocker-saishruthitn: /home/project  ×

theia@theiadocker-saishruthitn:/home/project$ ▯
```

TASK 1 - Read the data

Here you will be:

- Importing necessary libraries
- Reading the data from a CSV file

In this exercise we require the following libraries :

- `pandas`
- `plotly`
- `dash`

- `dash_html_components`
- `dash_core_components`
- `dash.dependencies`

We will first import these libraries

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
```

```
1. import pandas as pd
2. import plotly.graph_objects as go
3. import dash
4. from dash import dcc
5. from dash import html
6. from dash.dependencies import Input, Output
```

Copied!

Now we will read the dataset using the `pd.read_csv()` function.

```
1. 1
2. 2
3. 3
4. 4
```

```
1. airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
2.                             encoding = "ISO-8859-1",
3.                             dtype={'Div1Airport': str, 'Div1TailNum': str,
4.                                    'Div2Airport': str, 'Div2TailNum': str})
```

Copied!

- The above code reads a CSV file called **airline_data.csv** from a URL using pandas, a popular data analysis library in Python.
- The file is encoded using `ISO-8859-1 character encoding`, which is a standard way of representing characters in the file.
- We defined data type of specific columns such as (`Div1Airport, Div1TailNum, Div2Airport, and Div2TailNum`) to be strings, which ensures that these columns are read as text instead of numbers.
- The resulting data is stored in a pandas dataframe object called `airline_data`, which can be used for further analysis.

Copy the below code to the `dash_interactivity.py` script and review the code.

```
 1. 1
 2. 2
 3. 3
 4. 4
 5. 5
 6. 6
 7. 7
 8. 8
 9. 9
10. 10
11. 11
12. 12
13. 13
```

```
 1. # Import required libraries
 2. import pandas as pd
 3. import plotly.graph_objects as go
 4. import dash
 5. from dash import dcc
 6. from dash import html
 7. from dash.dependencies import Input, Output
 8.
 9. # Read the airline data into the pandas dataframe
10. airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
11.                             encoding = "ISO-8859-1",
12.                             dtype={'Div1Airport': str, 'Div1TailNum': str,
13.                                    'Div2Airport': str, 'Div2TailNum': str})
```

Copied!

Now save and run this code.

Copy and paste the below command in the terminal to run the code.

1. 1

1. `python3 dash_interactivity.py`

Copied!

After running the above code you can see the below-expected output:

TASK 2 - Create dash application and get the layout

Next, we create a skeleton for our dash application. Overall this layout creates a simple container with a heading, an input field, and some empty space.

In the upcoming tasks, you can modify and add new components and styles to the basic layout provided. This will allow you to customize and enhance the user interface of your Dash app to meet your specific needs and requirements as follows:

- First we will define an application `app.layout`.

- Create a heading using html.h1() and add style information within the divison `html.Div()`.

- create a inner division using `html.Div()` function for adding input and output components such as:

  - Input: `label`, dropdown `input-year` and style parameters

- Output: type of Graph `line-plot`

```
1.  1
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
```

```
1.  # Create a dash application layout
2.  app = dash.Dash(__name__)
3.
4.  # Get the layout of the application and adjust it.
5.  # Create an outer division using html.Div and add title to the dashboard using html.H1 component
6.  # Add a html.Div and core input text component
7.  # Finally, add graph component.
8.  app.layout = html.Div(children=[html.H1(),
9.                                  html.Div(["Input Year", dcc.Input(),],
10.                                 style={}),
11.                                 html.Br(),
12.                                 html.Br(),
13.                                 html.Div(),
14.                                 ])
```

Copied!

Mapping to the respective Dash HTML tags:

Application title add using html.H1() tag

- Heading reference: [Plotly H1 HTML Component](#)
- Title as `Airline Performance Dashboard`
- Use `style` parameter for the title and make it `center` aligned, with color code `#503D36`, and font-size as `40`. Check `More about HTML` section [here](#).

NOTE: After adding the components, you code will look like the below code.(You can copy dash application code to `dash_interactivity.py` script and run)

```
1.  1
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
```

27. 27

```
 1. # Import required libraries
 2. import pandas as pd
 3. import plotly.graph_objects as go
 4. import dash
 5. from dash import dcc
 6. from dash import html
 7. from dash.dependencies import Input, Output
 8. # Read the airline data into pandas dataframe
 9. airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
10.                            encoding = "ISO-8859-1",
11.                            dtype={'Div1Airport': str, 'Div1TailNum': str,
12.                                   'Div2Airport': str, 'Div2TailNum': str})
13.
14. # Create a dash application
15. app = dash.Dash(__name__)
16.
17. app.layout = html.Div(children=[html.H1('Airline Performance Dashboard',style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
18.                                 html.Div(["Input Year", dcc.Input(),],
19.                                 style={}),
20.                                 html.Br(),
21.                                 html.Br(),
22.                                 html.Div(),
23.                                 ])
24.
25. # Run the app
26. if __name__ == '__main__':
27.     app.run_server()
```

Copied!

To run the Dash app follow the below steps

- First, install pandas and dash using the following command in the terminal

1. 1

1. `pip3 install pandas dash`

Copied!

- Copy and paste the below command in the terminal to run the application.

1. 1

1. `python3 dash_interactivity.py`

Copied!

- Observe the port number shown in the terminal.

- Click on the `Launch Application` option from the side menu bar.Provide the port number and click `OK`

The app will open in a new browser tab like below:

After running the above code, launch the app in a new tab and below is the expected result from the code:
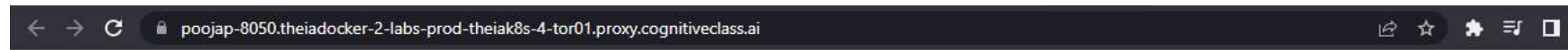
Input and Output components of layout

Input component

- As our input is a dropdown showing a list of years we will use the [dcc.Input()](#) function. We define the following parameters
    - id: `input-year`, which is a unique identifier for this specific input field. The default `value` for this input field will be set to `2010`, and the type of `input` will be a `number`.
- `style` parameter for the dropdown: Here within it we define'height`of the input box to be50px`and`font-size`to be35`` to make the text larger and more readable.
- `style` parameter for the whole division: Now assign `font-size` as `40` .

Output component

- Add `dcc.Graph()` component to the second division.
- Update dcc.Graph component `id` as `line-plot`.

NOTE: After adding the componenets you code will look like below code.(You can copy dash application code to `dash_interactivity.py` script and run).

To terminate a currently running program in the Python terminal (also known as the Python REPL), you can use the KeyboardInterrupt shortcut. This can be done by pressing the CTRL and C keys simultaneously.

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27

```
1.  # Import required libraries
2.  import pandas as pd
3.  import plotly.graph_objects as go
4.  import dash
5.  from dash import dcc
6.  from dash import html
7.  from dash.dependencies import Input, Output
8.  # Read the airline data into pandas dataframe
9.  airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
10.                             encoding = "ISO-8859-1",
11.                             dtype={'Div1Airport': str, 'Div1TailNum': str,
12.                                    'Div2Airport': str, 'Div2TailNum': str})
13. # Create a dash application
14. app = dash.Dash(__name__)
15.
16. app.layout = html.Div(children=[ html.H1('Airline Performance Dashboard',style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
17.                                 html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',
18.                                 type='number', style={'height':'50px', 'font-size': 35}),],
19.                                 style={'font-size': 40}),
20.                                 html.Br(),
21.                                 html.Br(),
22.                                 html.Div(dcc.Graph(id='line-plot')),
23.                                 ])
24.
25. # Run the app
26. if __name__ == '__main__':
27.     app.run_server()
```

Copied!

After running the above code, launch the app in a new tab and below is the expected result from the code:
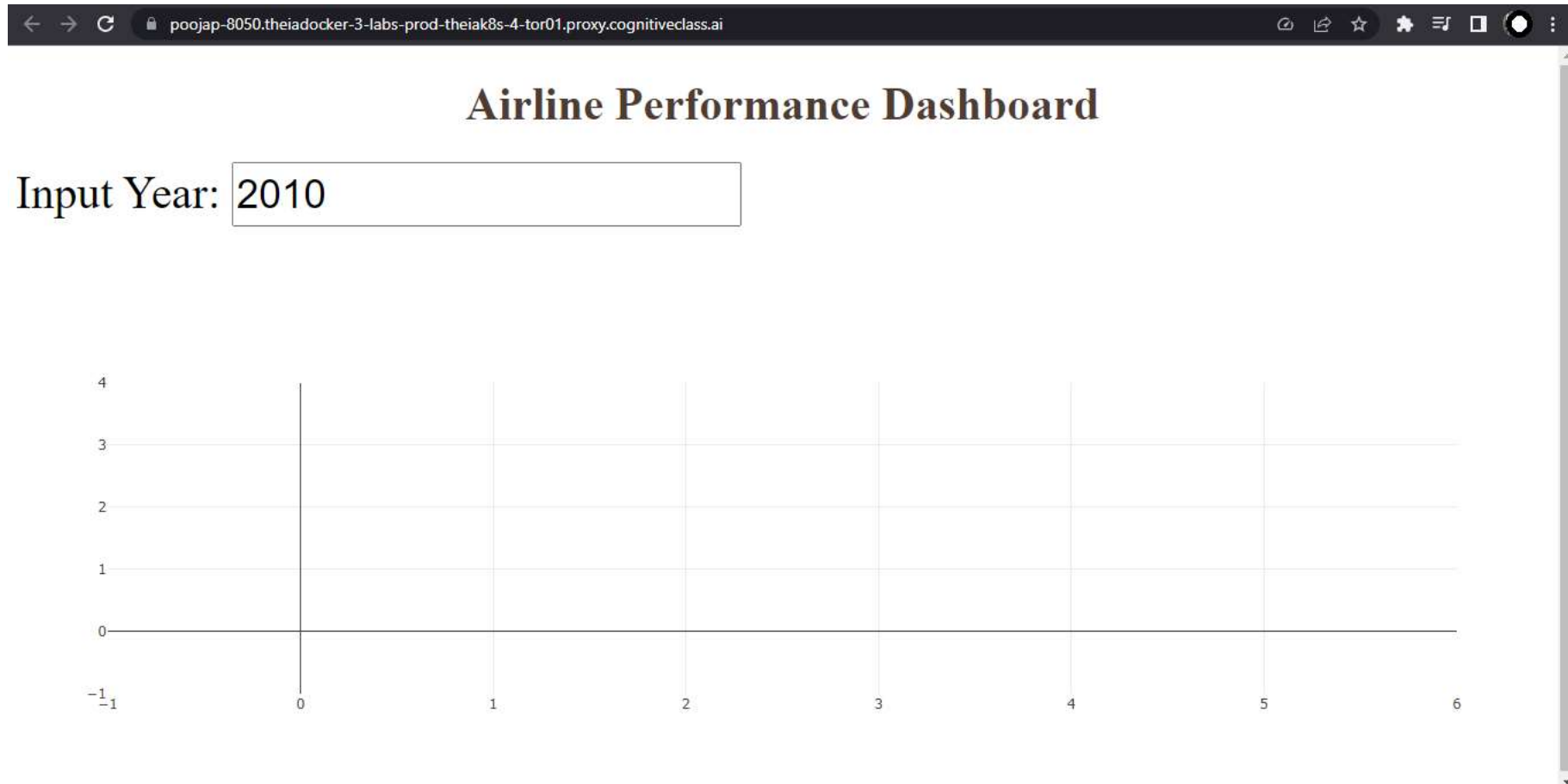
TASK 3 - Add the application callback function

Callback

In Python, **@app.callback** is a decorator used in the Dash framework to specify that a function should be called when an input component changes its value.The Input and Output functions are used to define the inputs and outputs of a callback function.

The core idea of this application is to get year as `user input`(input function) and update the dashboard(output function) in real-time with the help of `callback` function.

Steps:

- Define the callback decorator
- Define the callback function that uses the input provided to perform the computation
- Create graph and return it as an output

The below code is base structure for calback decorator and function graph.

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
 6.  6
 7.  7
 8.  8
 9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
```

```
 1. # add callback decorator
 2. @app.callback(Output(),
 3.               Input())
 4.
 5. # Add computation to callback function and return graph
 6. def get_graph(entered_year):
 7.     # Select data based on the entered year
 8.     df =  airline_data[airline_data['Year']==int(entered_year)]
 9.
10.     # Group the data by Month and compute the average over arrival delay time.
11.     line_data = df.groupby('Month')['ArrDelay'].mean().reset_index()
12.
13.     #
14.     fig = go.Figure(data=)
15.     fig.update_layout()
16.     return fig
17.
18. # Run the app
19. if __name__ == '__main__':
20.     app.run_server()
```

Copied!

Update the callback function

Callback decorator

- Refer to examples provided [here](#)

- `Input()` function takes two parameters:

  - `component-id` with the value `input-year`, which is the ID of the input dropdown.
  - `component_property` being accessed is the `value` property, which represents the year entered by the user.

- `Output()` function takes two parameters:

  - `component-id` with the value `line-plot`, which is the id of the output.
  - `component_property` being modified is the `figure` property, which specifies the data and layout of the line plot.

Callback function

- Update `data` parameter of the `go.Figure()` with the scatter plot. Refer [here](#). Sample syntax below:

1. 1

```
1. go.Scatter(x='----', y='----', mode='-----', marker='----')
```

Copied!

In the go.Scatter() update the parameter as below:

- Update x as `line_data['Month']`

- Update y as `line_data['ArrDelay']`

- Update `mode` as `lines`, and `marker` as `dict(color='green')`

- Update `fig.update_layout` with title, xaxis_title, and yaxis_title parameters.

    - Title as `Month vs Average Flight Delay Time`
    - `xaxis_title` as `Month`
    - `yaxis_title` as `ArrDelay`
      Refer the updated layout function here.

Refer to the full python code of `dash_interactivity.py` below:

```
1.  1
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
```

```
1.
2.  # Import required libraries
3.  import pandas as pd
4.  import plotly.graph_objects as go
5.  import dash
6.  from dash import dcc
7.  from dash import html
8.  from dash.dependencies import Input, Output
9.
10. # Read the airline data into the pandas dataframe
11. airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
```

```
12.                              encoding = "ISO-8859-1",
13.                              dtype={'Div1Airport': str, 'Div1TailNum': str,
14.                                     'Div2Airport': str, 'Div2TailNum': str})
15. # Create a dash application
16. app = dash.Dash(__name__)
17.
18. app.layout = html.Div(children=[ html.H1('Airline Performance Dashboard',style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
19.                              html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',
20.                              type='number', style={'height':'50px', 'font-size': 35}),],
21.                              style={'font-size': 40}),
22.                              html.Br(),
23.                              html.Br(),
24.                              html.Div(dcc.Graph(id='line-plot')),
25.                              ])
26.
27. # add callback decorator
28. @app.callback( Output(component_id='line-plot', component_property='figure'),
29.                Input(component_id='input-year', component_property='value'))
30.
31. # Add computation to callback function and return graph
32. def get_graph(entered_year):
33.     # Select 2019 data
34.     df =  airline_data[airline_data['Year']==int(entered_year)]
35.
36.     # Group the data by Month and compute average over arrival delay time.
37.     line_data = df.groupby('Month')['ArrDelay'].mean().reset_index()
38.
39.     fig = go.Figure(data=go.Scatter(x=line_data['Month'], y=line_data['ArrDelay'], mode='lines', marker=dict(color='green')))
40.     fig.update_layout(title='Month vs Average Flight Delay Time', xaxis_title='Month', yaxis_title='ArrDelay')
41.     return fig
42.
43. # Run the app
44. if __name__ == '__main__':
45.     app.run_server()
```

Copied!

After running the above code, launch the app in a new tab and below is the expected final result from the code:

TASK 4 : Practice Exercise

You will practice some tasks to create the new dashboard.

Note: Please refer the Bar chart exercise of [Plotly basics: scatter, line, bar, bubble, histogram, pie, sunburst](#) lab for this exercise.

Theme

Let us use a bar chart to extract the number of flights from a specific airline that goes to a destination.
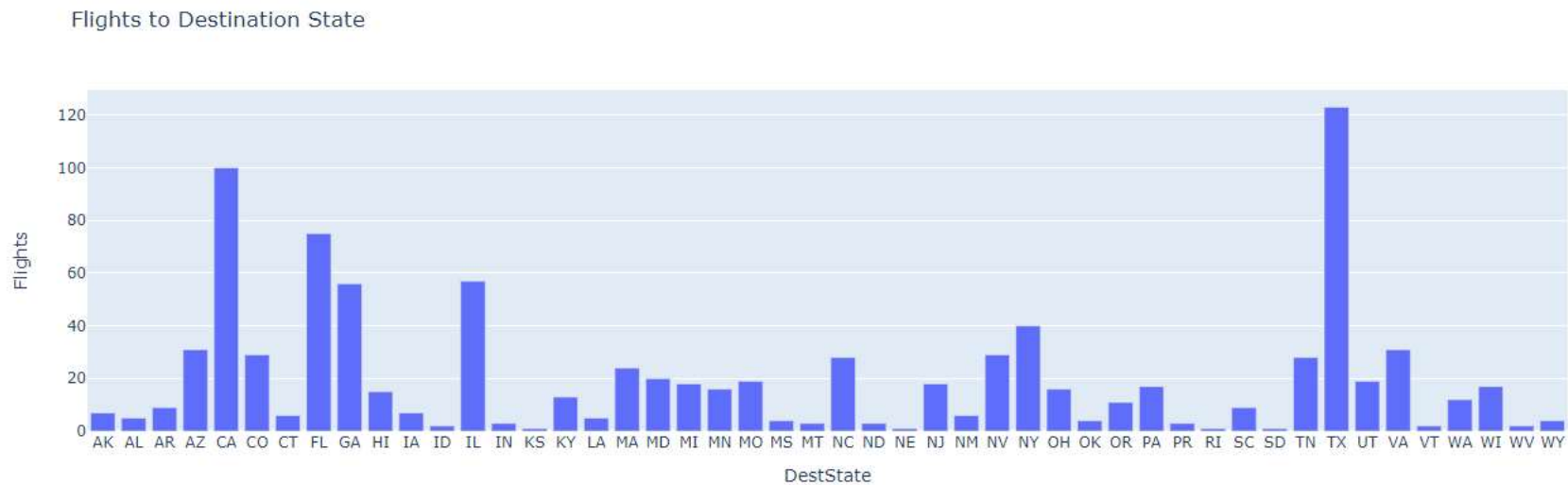
Expected Output

Below is the expected result from the lab. Our dashboard application consists of three components:

- Title of the application
- Component to enter input year
- A chart conveying Total number of flights to the destination state.

Total number of flights to the destination state split by reporting airline

Input Year: 2010

Create a new python script and provide the file name as `dash_interactivity_barplot.py`.

1.   Import all the required libraries and read the [Airline Reporting Carrier On-Time Performance](#) dataset.

    ▼ Answer

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13

    1.   # Import required libraries

```
2.      import pandas as pd
3.      import plotly.graph_objects as go
4.      import plotly.express as px
5.      import dash
6.      from dash import dcc
7.      from dash import html
8.      from dash.dependencies import Input, Output
9.      # Read the airline data into pandas dataframe
10.     airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
11.                     encoding = "ISO-8859-1",
12.                     dtype={'Div1Airport': str, 'Div1TailNum': str,
13.                            'Div2Airport': str, 'Div2TailNum': str})
```
Copied!

2.   Give the title to the dashboard to "Total number of flights to the destination state split by reporting air" using HTML H1 component and font-size as 50.

- **Input component** Make changes to a component called dcc.Input in a tool called Dash. We are updating its id to be input-year, which is a unique identifier for this specific input field. The default value for this input field will be set to 2010, and the type of input will be a number.

  To make the text larger and more readable, use the style parameter and assign the height of the input box to 50px and font-size to be 35. Use style parameter again and assign font-size of 40 for the whole division

- **Output component**Add dcc.Graph() component to the second division. Update dcc.Graph component id as `bar-plot`.

▼ Answer

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
1. app = dash.Dash(__name__)
2. app.layout = html.Div(children=[ html.H1('Total number of flights to the destination state split by reporting airline',
3.                      style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
4.                      html.Div(["Input Year: ", dcc. Input(id='input-year',value='2010',
5.                      type='number', style={'height':'50px', 'font-size': 35}),],
6.                      style={'font-size': 40}),html.Br(), html.Br(),
7.                      html.Div(dcc.Graph(id='bar-plot')),])
```
Copied!

3.   **Add Callback decorator**

- Update **output** `component id` parameter with `bar-plot` same as the id provided in the `dcc.Graph()` component and `component property` as `figure`.
- Update **input** `component id` parameter with `input-year` same as the id provided in the `dcc.Input()` component and `component property` as `value`.

  Use `style` parameter and assign font-size as `35` for the whole division.

▼ Answer

```
1. 1
2. 2
1. @app.callback( Output(component_id='bar-plot',component_property='figure'),
2.                Input(component_id='input-year', component_property='value'))
```
Copied!

4.   **Define Callback graph function**

- Group the data by destination state and reporting airline. Compute the total number of flights in each combination
- Use plotly express bar chart function px.bar. Provide input data, x and y axis variables, and a chart title. This will give the total number of flights to the destination state

▼ Answer

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
1. def get_graph(entered_year):
```

```
2.      df =  airline_data[airline_data['Year']==int(entered_year)]
3.      bar_data = df.groupby('DestState')['Flights'].sum().reset_index()
4.      fig = px.bar(bar_data, x= "DestState", y= "Flights", title='Total number of flights to the destination state split by reporting airline')
5.      fig.update_layout(title='Flights to Destination State', xaxis_title='DestState', yaxis_title='Flights')
6.      return fig
```
Copied!

5.   Add the code for running the app, save the above changes, and relaunch the dashboard application to see the new dashboard.

▼ Answer

```
1. 1
2. 2
1. if __name__ == '__main__':
2.      app.run_server()
```
Copied!

Congratulations, you have successfully created your dash application!

Author

[Saishruthi Swaminathan](#)

Changelog

Date	Version	Changed by	Change Description
05-07-2021	1.0	Saishruthi	Initial version created
24-08-2022	1.1	Pratiksha	Instructions updated
29-08-2022	1.2	Pratiksha Verma	Updated Screenshot
12-04-2023	1.3	Pooja Patel	Updated instructions and Screenshot
06-07-2023	1.4	Dr. Pooja	Updated code