**Get the tool ready**

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

1. 1
2. python3.11 -m pip install packaging

Copied!

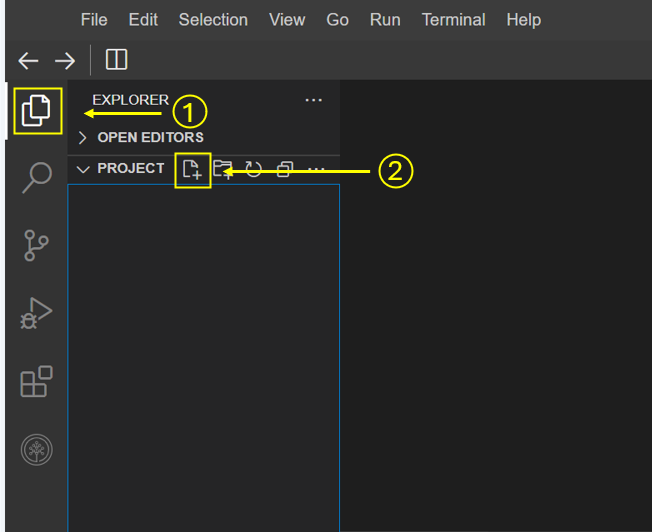
1. 1
2. python3.11 -m pip install pandas dash

Copied!

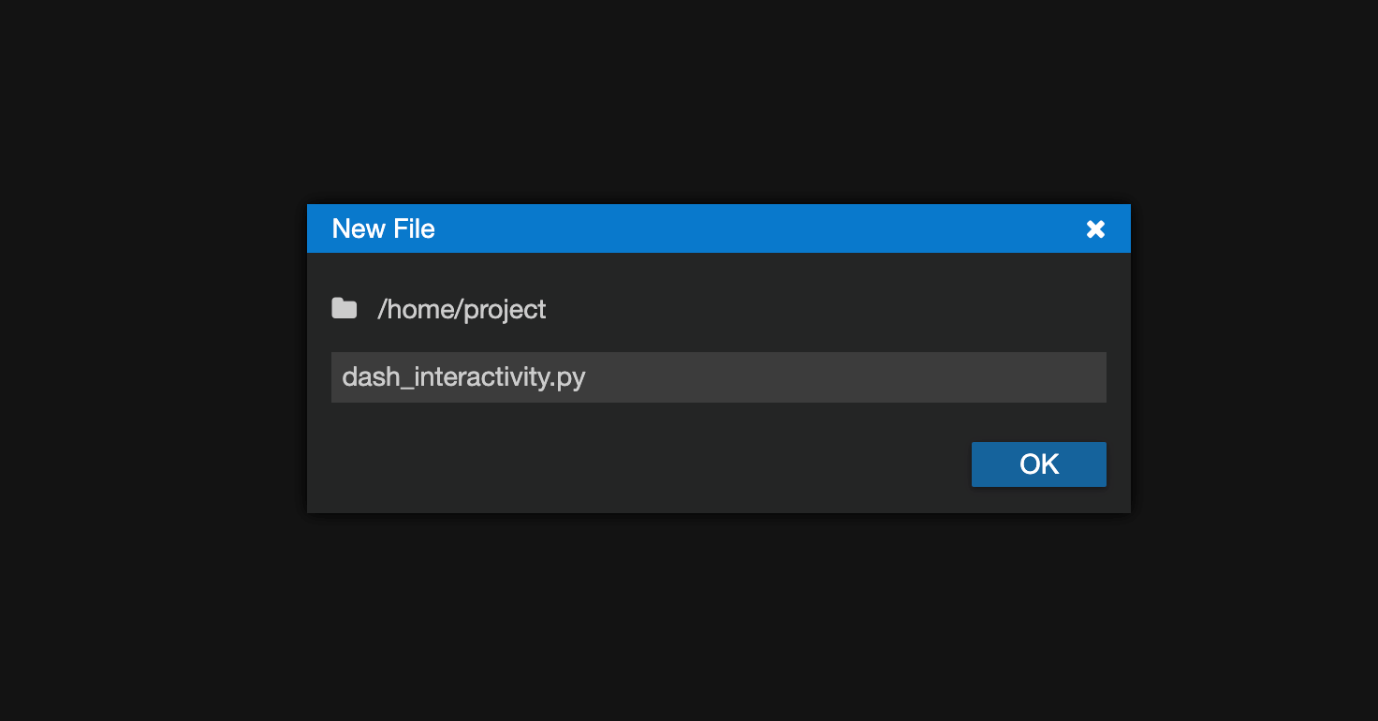
1. 1
2. pip3 install httpx==0.20 dash plotly

Copied!

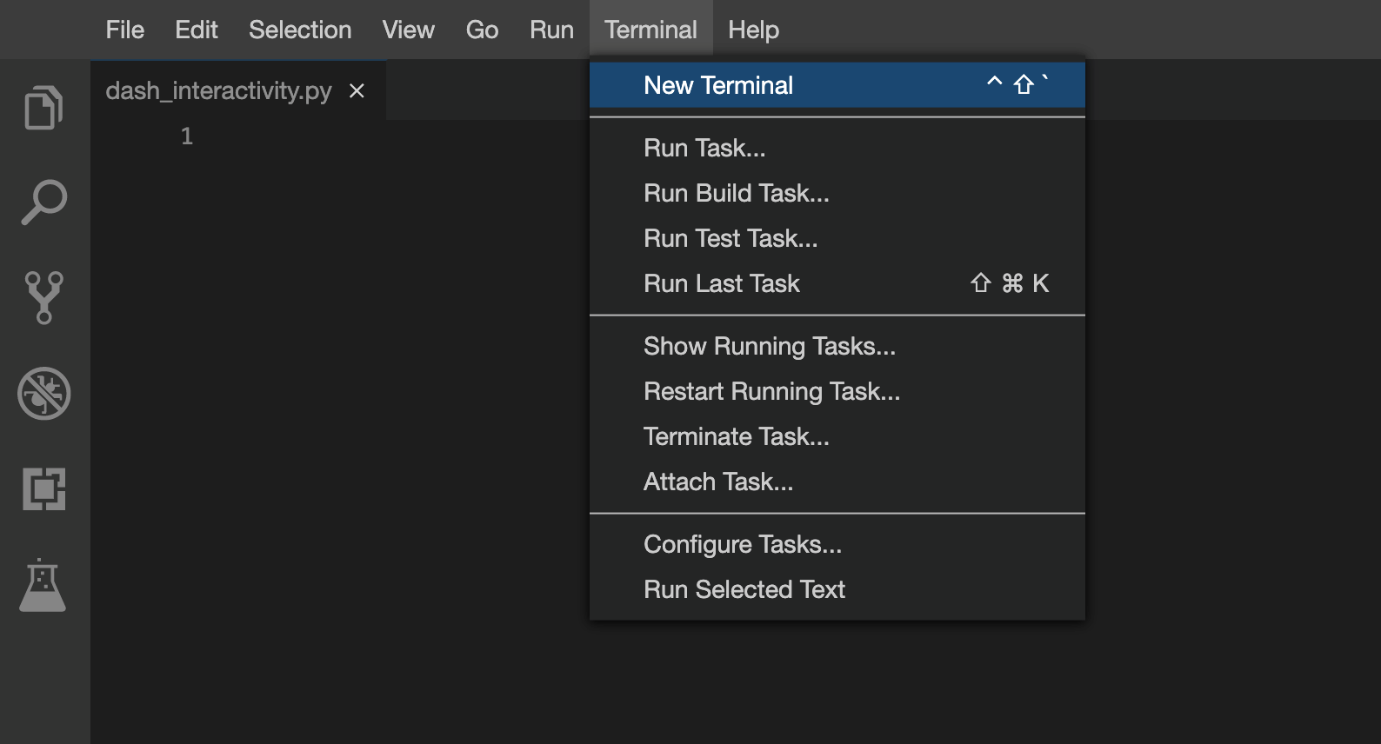
* Create a new python script, by clicking on the side tool bar **explorer** icon and selecting **new file** icon, as shown in the image below.



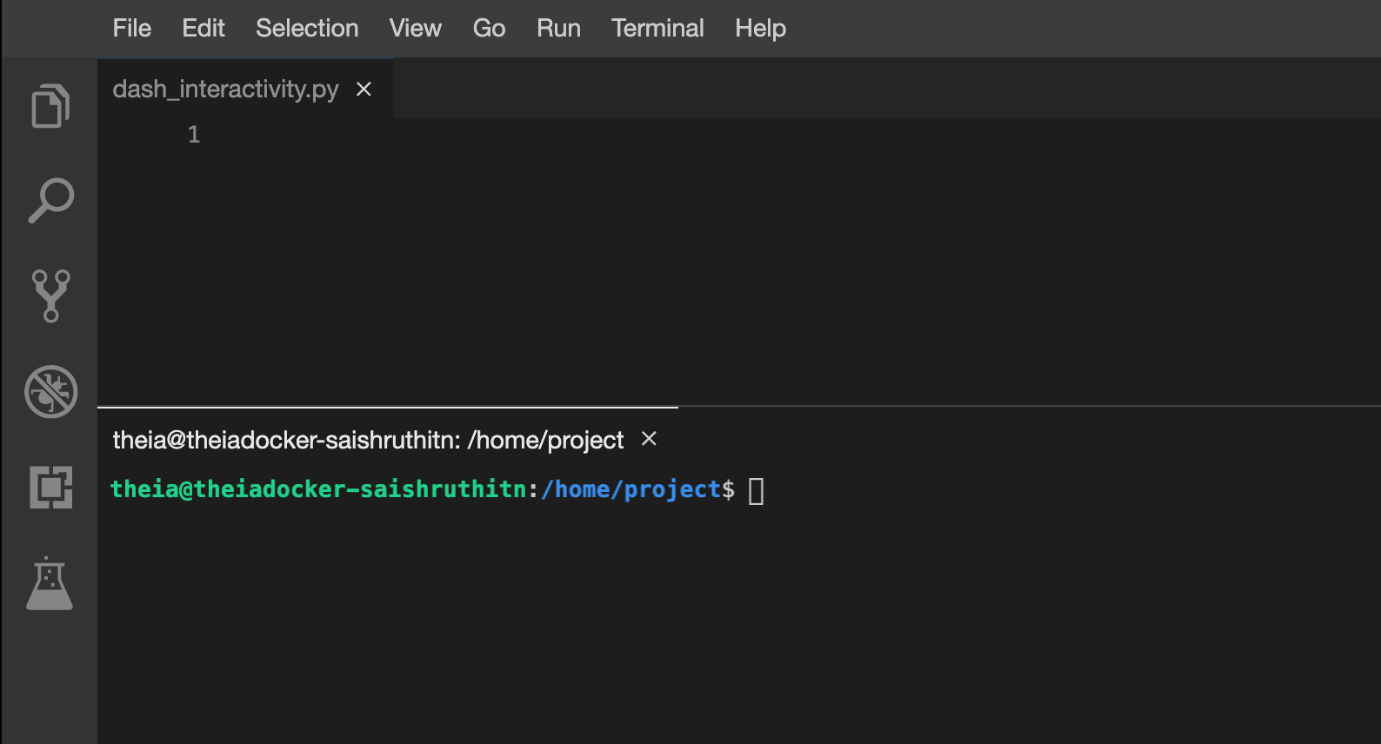
* Provide the file name as dash\_interactivity.py



* 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.



**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
7. import pandas as pd
8. import plotly.graph\_objects as go
9. import dash
10. from dash import dcc
11. from dash import html
12. 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
5. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
6. encoding = "ISO-8859-1",
7. dtype={'Div1Airport': str, 'Div1TailNum': str,
8. '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
14. # Import required libraries
15. import pandas as pd
16. import plotly.graph\_objects as go
17. import dash
18. from dash import dcc
19. from dash import html
20. from dash.dependencies import Input, Output
21. # Read the airline data into the pandas dataframe
22. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
23. encoding = "ISO-8859-1",
24. dtype={'Div1Airport': str, 'Div1TailNum': str,
25. 'Div2Airport': str, 'Div2TailNum': str})

Copied!

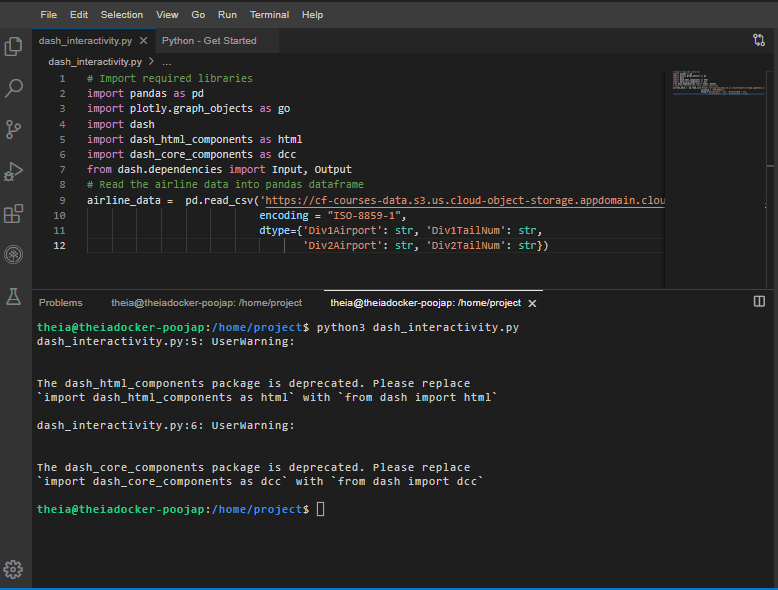
Now save and run this code.

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

1. 1
2. python3.11 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
15. # Create a dash application layout
16. app = dash.Dash(\_\_name\_\_)
17. # Get the layout of the application and adjust it.
18. # Create an outer division using html.Div and add title to the dashboard using html.H1 component
19. # Add a html.Div and core input text component
20. # Finally, add graph component.
21. app.layout = html.Div(children=[html.H1(),
22. html.Div(["Input Year", dcc.Input(),],
23. style={}),
24. html.Br(),
25. html.Br(),
26. html.Div(),
27. ])

Copied!

## Mapping to the respective Dash HTML tags:

### Application title add using html.H1() tag

* Heading reference: [Plotly H1 HTML Component](https://dash.plotly.com/dash-html-components/h1)
* 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](https://dash.plotly.com/layout).

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
28. # Import required libraries
29. import pandas as pd
30. import plotly.graph\_objects as go
31. import dash
32. from dash import dcc
33. from dash import html
34. from dash.dependencies import Input, Output
35. # Read the airline data into pandas dataframe
36. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
37. encoding = "ISO-8859-1",
38. dtype={'Div1Airport': str, 'Div1TailNum': str,
39. 'Div2Airport': str, 'Div2TailNum': str})
41. # Create a dash application
42. app = dash.Dash(\_\_name\_\_)
43. app.layout = html.Div(children=[html.H1('Airline Performance Dashboard',style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
44. html.Div(["Input Year", dcc.Input(),],
45. style={}),
46. html.Br(),
47. html.Br(),
48. html.Div(),
49. ])
50. # Run the app
51. if \_\_name\_\_ == '\_\_main\_\_':
52. 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
2. pip3.11 install pandas dash

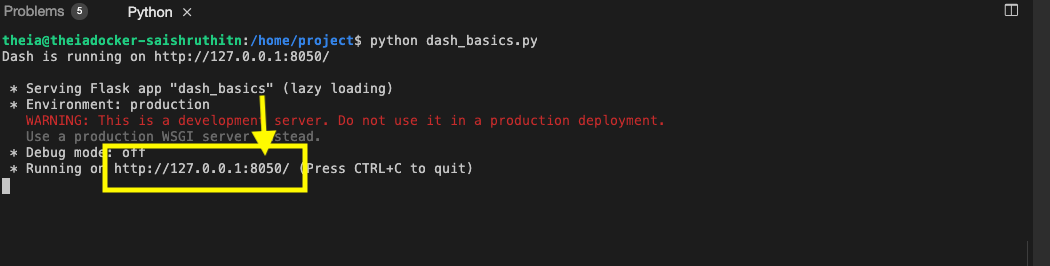
Copied!

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

1. 1
2. python3.11 dash\_interactivity.py

Copied!

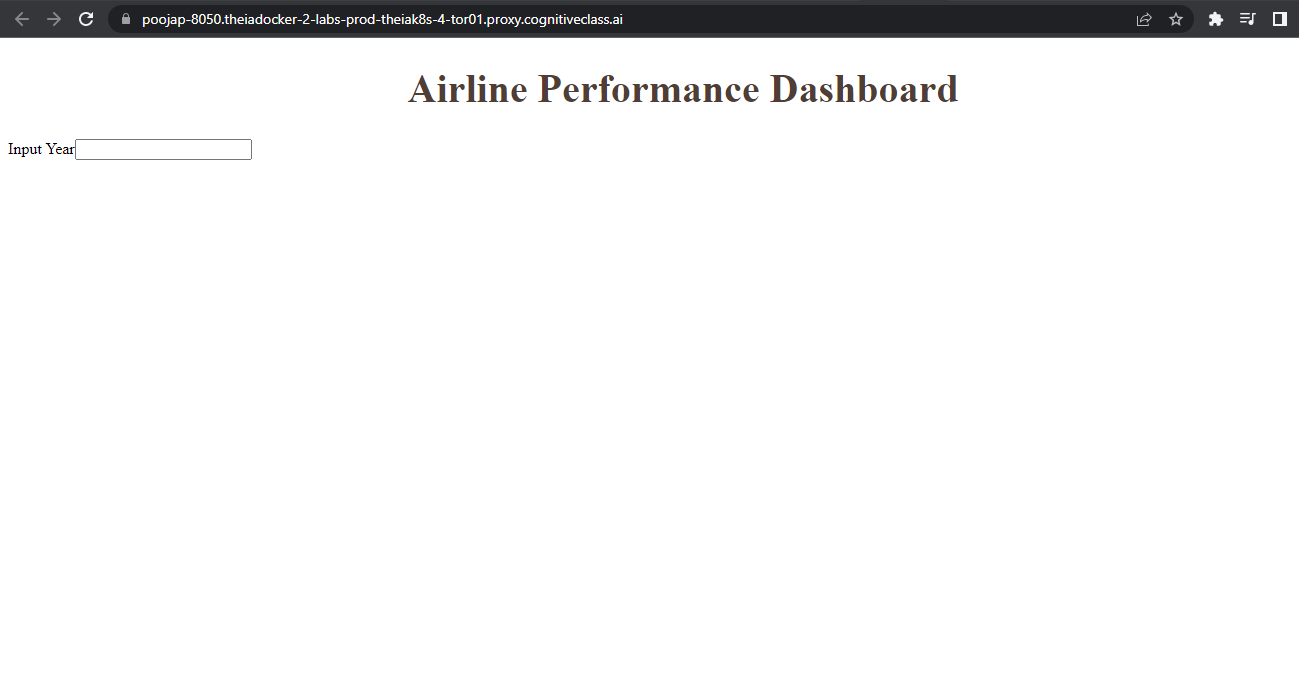
* Observe the port number shown in the terminal.



* 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()](https://dash.plotly.com/dash-core-components/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’heightof the input box to be50pxandfont-sizeto 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](https://dash.plotly.com/dash-core-components/graph) component id as line-plot.

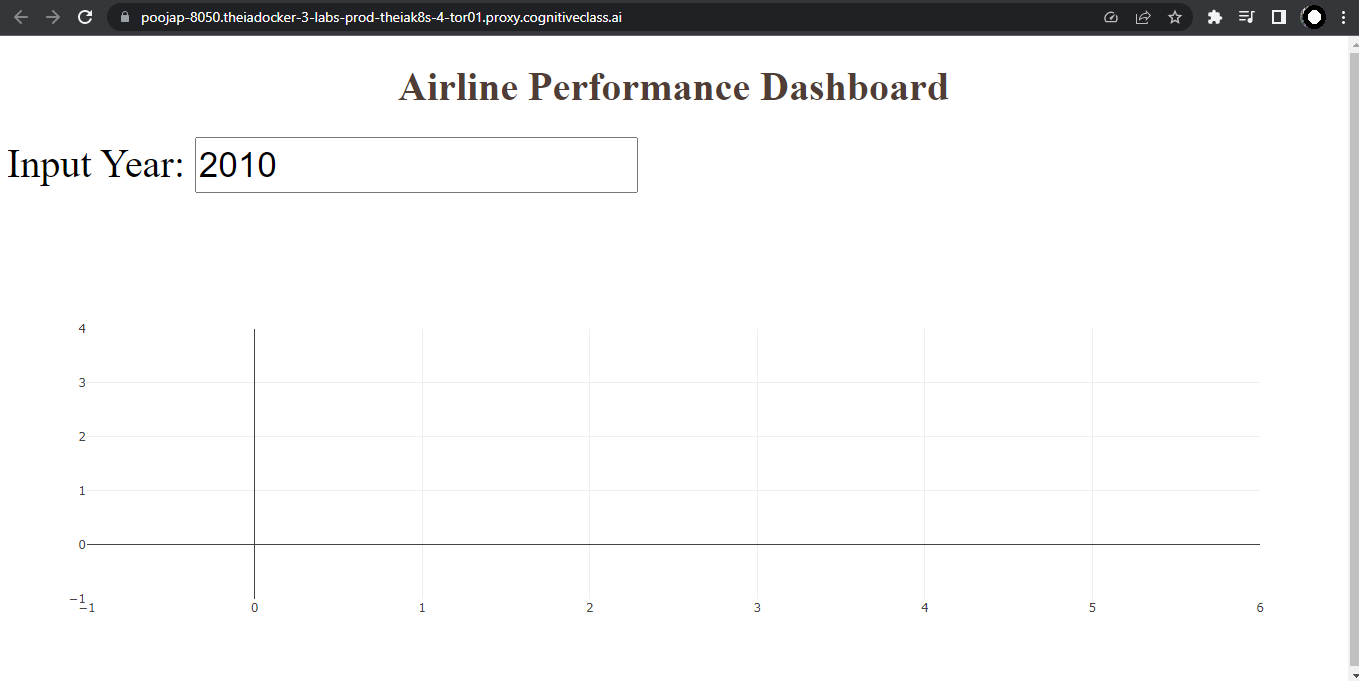
NOTE: After adding the componenets you code will look like below code.(You can copy dash application code to dash\_interactivity.py script and run).

**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
28. # Import required libraries
29. import pandas as pd
30. import plotly.graph\_objects as go
31. import dash
32. from dash import dcc
33. from dash import html
34. from dash.dependencies import Input, Output
35. # Read the airline data into pandas dataframe
36. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
37. encoding = "ISO-8859-1",
38. dtype={'Div1Airport': str, 'Div1TailNum': str,
39. 'Div2Airport': str, 'Div2TailNum': str})
40. # Create a dash application
41. app = dash.Dash(\_\_name\_\_)
43. app.layout = html.Div(children=[ html.H1('Airline Performance Dashboard',style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
44. html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',
45. type='number', style={'height':'50px', 'font-size': 35}),],
46. style={'font-size': 40}),
47. html.Br(),
48. html.Br(),
49. html.Div(dcc.Graph(id='line-plot')),
50. ])
51. # Run the app
52. if \_\_name\_\_ == '\_\_main\_\_':
53. 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
21. # add callback decorator
22. @app.callback(Output(),
23. Input())
24. # Add computation to callback function and return graph
25. def get\_graph(entered\_year):
26. # Select data based on the entered year
27. df = airline\_data[airline\_data['Year']==int(entered\_year)]
29. # Group the data by Month and compute the average over arrival delay time.
30. line\_data = df.groupby('Month')['ArrDelay'].mean().reset\_index()
32. #
33. fig = go.Figure(data=)
34. fig.update\_layout()
35. return fig
36. # Run the app
37. if \_\_name\_\_ == '\_\_main\_\_':
38. app.run\_server()

# Update the callback function

### Callback decorator

* Refer to examples provided [here](https://dash.plotly.com/basic-callbacks)
* 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](https://plotly.com/python/line-and-scatter/#scatter-and-line-plot-with-goscatter). Sample syntax below:

1. 1
2. 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](https://plotly.com/python/line-and-scatter/#style-scatter-plots).

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
46. # Import required libraries
47. import pandas as pd
48. import plotly.graph\_objects as go
49. import dash
50. from dash import dcc
51. from dash import html
52. from dash.dependencies import Input, Output
53. # Read the airline data into the pandas dataframe
54. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
55. encoding = "ISO-8859-1",
56. dtype={'Div1Airport': str, 'Div1TailNum': str,
57. 'Div2Airport': str, 'Div2TailNum': str})
58. # Create a dash application
59. app = dash.Dash(\_\_name\_\_)
61. app.layout = html.Div(children=[ html.H1('Airline Performance Dashboard',style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
62. html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',
63. type='number', style={'height':'50px', 'font-size': 35}),],
64. style={'font-size': 40}),
65. html.Br(),
66. html.Br(),
67. html.Div(dcc.Graph(id='line-plot')),
68. ])
69. # add callback decorator
70. @app.callback( Output(component\_id='line-plot', component\_property='figure'),
71. Input(component\_id='input-year', component\_property='value'))
72. # Add computation to callback function and return graph
73. def get\_graph(entered\_year):
74. # Select 2019 data
75. df = airline\_data[airline\_data['Year']==int(entered\_year)]
77. # Group the data by Month and compute average over arrival delay time.
78. line\_data = df.groupby('Month')['ArrDelay'].mean().reset\_index()
79. fig = go.Figure(data=go.Scatter(x=line\_data['Month'], y=line\_data['ArrDelay'], mode='lines', marker=dict(color='green')))
80. fig.update\_layout(title='Month vs Average Flight Delay Time', xaxis\_title='Month', yaxis\_title='ArrDelay')
81. return fig
82. # Run the app
83. if \_\_name\_\_ == '\_\_main\_\_':
84. 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 1 - Data Preparation**

Let’s start with

* Importing necessary libraries
* Reading and sampling 500 random data points
* Get the chart ready

Copy the below code to the dash\_basics.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
14. 14
15. 15
16. 16
17. 17
18. 18
19. # Import required packages
20. import pandas as pd
21. import plotly.express as px
22. import dash
23. from dash import dcc
24. from dash import html
25. # Read the airline data into pandas dataframe
26. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
27. encoding = "ISO-8859-1",
28. dtype={'Div1Airport': str, 'Div1TailNum': str,
29. 'Div2Airport': str, 'Div2TailNum': str})
30. # Randomly sample 500 data points. Setting the random state to be 42 so that we get same result.
31. data = airline\_data.sample(n=500, random\_state=42)
32. # Pie Chart Creation
33. fig = px.pie(data, values='Flights', names='DistanceGroup', title='Distance group

**TASK 2 - Create dash application and get the layout skeleton**

Next, we create a skeleton for our dash application. Our dashboard application has three components as seen before:

* Title of the application
* Description of the application
* Chart conveying the proportion of distance group by month

Mapping to the respective Dash HTML tags:

* Title added using html.H1() tag
* Description added using html.P() tag
* Chart added using dcc.Graph() tag

Copy the below code to the dash\_basics.py script and review the structure.

*NOTE*: Copy below the current 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
14. 14
15. 15
16. 16
17. # Create a dash application
18. app = dash.Dash(\_\_name\_\_)
19. # Get the layout of the application and adjust it.
20. # Create an outer division using html.Div and add title to the dashboard using html.H1 component
21. # Add description about the graph using HTML P (paragraph) component
22. # Finally, add graph component.
23. app.layout = html.Div(children=[html.H1(),
24. html.P(),
25. dcc.Graph(),
27. ])
28. # Run the application
29. if \_\_name\_\_ == '\_\_main\_\_':
30. app.run\_server()

Copied!

PreviousNext

**TASK 3 - Add the application title**

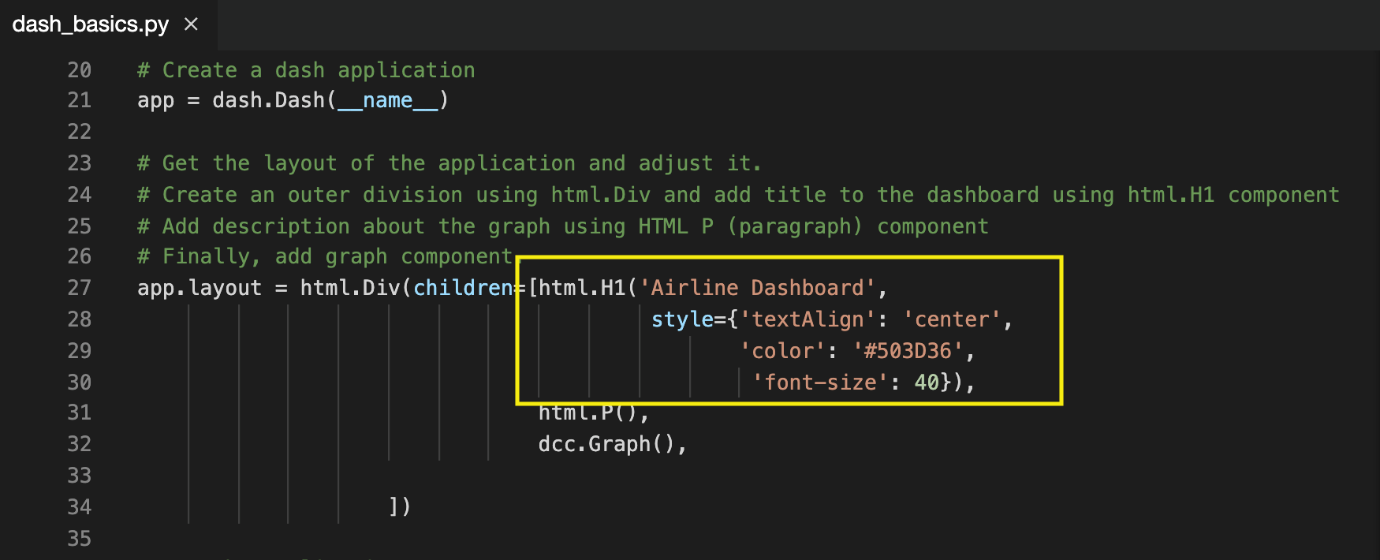
Update the html.H1() tag to hold the application title.

* Application title is Airline Dashboard
* Use style parameter provided below to make the title center aligned, with color code #503D36, and font-size as 40

1. 1
2. 'Airline Dashboard',style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}

Copied!

After updating the html.H1() with the application title, the app.layout will look like:



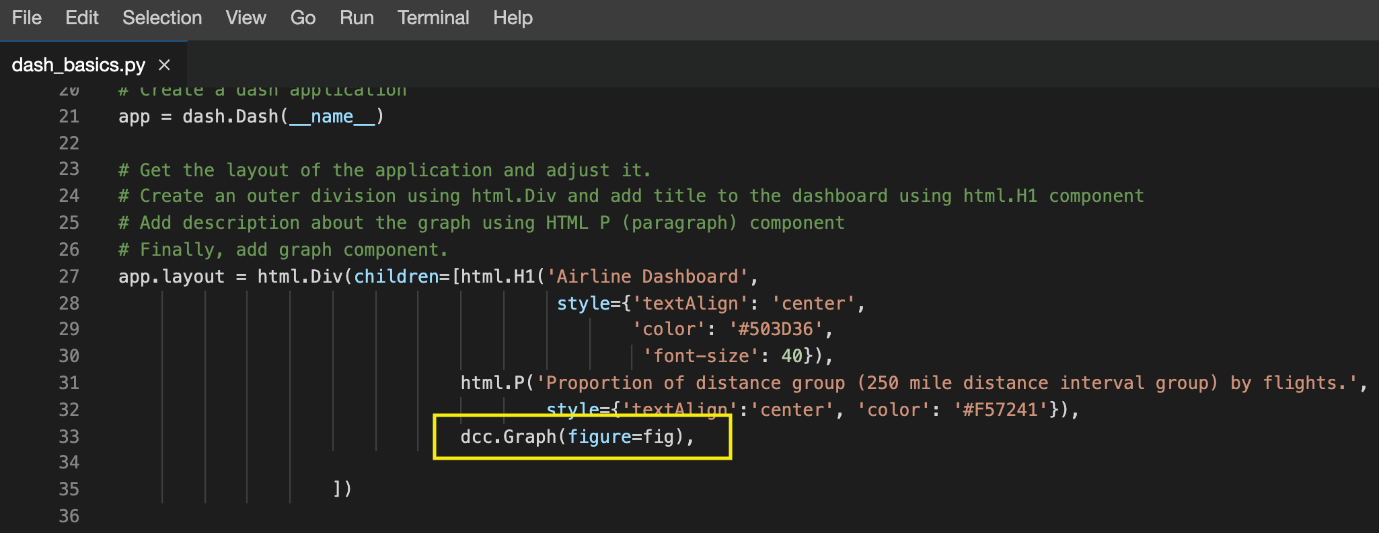
# TASK 5 - Update the graph

Update figure parameter of dcc.Graph() component to add the pie chart. We have created pie chart and assigned it to fig. Let’s use that to update the figure parameter.

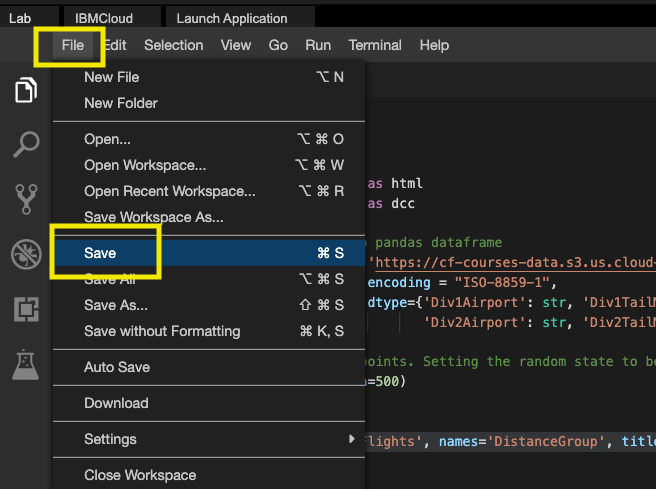
1. 1
2. figure=fig

Copied!

After updating the dcc.Graph() with the application title, the app.layout will look like:



Before running the application, save the file by clicking on **File -> Save** from the menu bar.



You can Refer to the entire python code here

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. # Import required packages
38. import pandas as pd
39. import plotly.express as px
40. import dash
41. from dash import dcc
42. from dash import html
43. # Read the airline data into pandas dataframe
44. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
45. encoding = "ISO-8859-1",
46. dtype={'Div1Airport': str, 'Div1TailNum': str,
47. 'Div2Airport': str, 'Div2TailNum': str})
48. # Randomly sample 500 data points. Setting the random state to be 42 so that we get same result.
49. data = airline\_data.sample(n=500, random\_state=42)
50. # Pie Chart Creation
51. fig = px.pie(data, values='Flights', names='DistanceGroup', title='Distance group proportion by flights')
52. # Create a dash application
53. app = dash.Dash(\_\_name\_\_)
54. # Get the layout of the application and adjust it.
55. # Create an outer division using html.Div and add title to the dashboard using html.H1 component
56. # Add description about the graph using HTML P (paragraph) component
57. # Finally, add graph component.
58. app.layout = html.Div(children=[html.H1('Airline Dashboard', style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
59. html.P('Proportion of distance group (250 mile distance interval group) by flights.', style={'textAlign':'center', 'color': '#F57241'}),
60. dcc.Graph(figure=fig),
62. ])
63. # Run the application
64. if \_\_name\_\_ == '\_\_main\_\_':
65. app.run\_server()

**TASK 1 - Read the data**

Let’s start with

* Importing necessary libraries
* Reading the data

Copy the below code to the flight\_delay.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
14. 14
15. # Import required libraries
16. import pandas as pd
17. import plotly.graph\_objects as go
18. import dash
19. from dash import dcc
20. from dash import html
21. from dash.dependencies import Input, Output
22. import plotly.express as px
23. # Read the airline data into pandas dataframe
24. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
25. encoding = "ISO-8859-1",
26. dtype={'Div1Airport': str, 'Div1TailNum': str,
27. 'Div2Airport': str, 'Div2TailNum': str})

**TASK 2 - Create dash application and get the layout skeleton**

Next, we create a skeleton for our dash application. Our dashboard application layout has three components as seen before:

* Title of the application
* Component to enter input year inside a layout division
* 5 Charts conveying the different types of flight delay

Mapping to the respective Dash HTML tags:

* Title added using html.H1() tag
* Layout division added using html.Div() and input component added using dcc.Input() tag inside the layout division.
* 5 charts split into three segments. Each segment has a layout division added using html.Div() and chart added using dcc.Graph() tag inside the layout division.

Copy the below code to the flight\_delay.py script and review the structure.

*NOTE*: Copy below the current 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
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. # Create a dash application
23. app = dash.Dash(\_\_name\_\_)
24. # Build dash app layout
25. app.layout = html.Div(children=[ html.H1(),
26. html.Div(["Input Year: ", dcc.Input()],
27. style={'font-size': 30}),
28. html.Br(),
29. html.Br(),
30. html.Div([
31. html.Div(),
32. html.Div()
33. ], style={'display': 'flex'}),
35. html.Div([
36. html.Div(),
37. html.Div()
38. ], style={'display': 'flex'}),
40. html.Div(, style={'width':'65%'})
41. ])

Copied!

*NOTE*: We are using display as flex for two outer divisions to get graphs side by side in a row.

# TASK 3 - Update layout components

### Application title

* Title as Flight Delay Time Statistics, align text as center, color as #503D36, and font size as 30.

### Input component

* Update [dcc.Input](https://dash.plotly.com/dash-core-components/input) component id as input-year, default value as 2010, and type as number. Use style parameter and assign height of the input box to be 35px and font-size to be 30.

### Output component - Segment 1

Segment 1 is the first html.Div(). We have two inner division where first two graphs will be placed.

#### Skeleton

1. 1
2. 2
3. 3
4. 4
5. html.Div([
6. html.Div(),
7. html.Div()
8. ], style={'display': 'flex'}),

Copied!

#### First inner division

* Add dcc.Graph() component.
* Update [dcc.Graph](https://dash.plotly.com/dash-core-components/graph) component id as carrier-plot.

##### Second inner division

* Add dcc.Graph() component.
* Update [dcc.Graph](https://dash.plotly.com/dash-core-components/graph) component id as weather-plot.

### Output component - Segment 2

Segment 2 is the second html.Div(). We have two inner division where the next two graphs will be placed.

#### Skeleon

1. 1
2. 2
3. 3
4. 4
5. html.Div([
6. html.Div(),
7. html.Div()
8. ], style={'display': 'flex'}),

Copied!

#### First inner division

* Add dcc.Graph() component.
* Update [dcc.Graph](https://dash.plotly.com/dash-core-components/graph) component id as nas-plot.

##### Second inner division

* Add dcc.Graph() component.
* Update [dcc.Graph](https://dash.plotly.com/dash-core-components/graph) component id as security-plot.

### Output component - Segment 3

Segment 3 is the last html.Div().

#### Skeleon

1. 1
2. html.Div(, style={'width':'65%'})

Copied!

* Add dcc.Graph() component to the first inner division.
* Update [dcc.Graph](https://dash.plotly.com/dash-core-components/graph) component id as late-plot.

PreviousNext

**TASK 4 - Review and add supporting function**

Below is the function that gets input year and data, perform computation for creating charts and plots.

Copy the below code to the flight\_delay.py script and review the structure.

*NOTE*: Copy below the current 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
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. """ Compute\_info function description
24. This function takes in airline data and selected year as an input and performs computation for creating charts and plots.
25. Arguments:
26. airline\_data: Input airline data.
27. entered\_year: Input year for which computation needs to be performed.
29. Returns:
30. Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.
31. """
32. def compute\_info(airline\_data, entered\_year):
33. # Select data
34. df = airline\_data[airline\_data['Year']==int(entered\_year)]
35. # Compute delay averages
36. avg\_car = df.groupby(['Month','Reporting\_Airline'])['CarrierDelay'].mean().reset\_index()
37. avg\_weather = df.groupby(['Month','Reporting\_Airline'])['WeatherDelay'].mean().reset\_index()
38. avg\_NAS = df.groupby(['Month','Reporting\_Airline'])['NASDelay'].mean().reset\_index()
39. avg\_sec = df.groupby(['Month','Reporting\_Airline'])['SecurityDelay'].mean().reset\_index()
40. avg\_late = df.groupby(['Month','Reporting\_Airline'])['LateAircraftDelay'].mean().reset\_index()
41. return avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late

Copied!

# PreviousTASK 5 - Add the application callback function

The core idea of this application is to get year as user input and update the dashboard in real-time. We will be using callback function for the same.

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
* Run the application

Copy the below code to the flight\_delay.py script and review the structure.

*NOTE*: Copy below the current 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
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. # Callback decorator
33. @app.callback( [
34. Output(component\_id='carrier-plot', component\_property='figure'),
35. ---
36. ---
37. ---
38. ---
39. ],
40. Input(....))
41. # Computation to callback function and return graph
42. def get\_graph(entered\_year):
44. # Compute required information for creating graph from the data
45. avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late = compute\_info(airline\_data, entered\_year)
47. # Line plot for carrier delay
48. carrier\_fig = px.line(avg\_car, x='Month', y='CarrierDelay', color='Reporting\_Airline', title='Average carrier delay time (minutes) by airline')
49. # Line plot for weather delay
50. weather\_fig = ------
51. # Line plot for nas delay
52. nas\_fig = ------
53. # Line plot for security delay
54. sec\_fig = ------
55. # Line plot for late aircraft delay
56. late\_fig = ------
58. return[carrier\_fig, weather\_fig, nas\_fig, sec\_fig, late\_fig]
59. # Run the app
60. if \_\_name\_\_ == '\_\_main\_\_':
61. app.run\_server()

# TASK 6 - Update the callback function

### Callback decorator

* Refer examples provided [here](https://dash.plotly.com/basic-callbacks)
* We have 5 output components added in a list. Update output component id parameter with the ids provided in the dcc.Graph() component and set the component property as figure. One sample has been added to the skeleton.
* Update input component id parameter with the id provided in the dcc.Input() component and component property as value.

### Callback function

Next is to update the get\_graph function. We have already added a function compute\_info that will perform computation on the data using the input.

Mapping the returned value from the function compute\_info to graph:

* avg\_car - input for carrier delay
* avg\_weather - input for weather delay
* avg\_NAS - input for NAS delay
* avg\_sec - input for security delay
* avg\_late - input for late aircraft delay

Code has been provided for plotting carrier delay. Follow the same process and use the above mapping to get plots for other 4 delays.

Refer to the full code of 4.8\_Flight\_Delay\_Time\_Statistics\_Dashboard.py

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
46. 46
47. 47
48. 48
49. 49
50. 50
51. 51
52. 52
53. 53
54. 54
55. 55
56. 56
57. 57
58. 58
59. 59
60. 60
61. 61
62. 62
63. 63
64. 64
65. 65
66. 66
67. 67
68. 68
69. 69
70. 70
71. 71
72. 72
73. 73
74. 74
75. 75
76. 76
77. 77
78. 78
79. 79
80. 80
81. 81
82. 82
83. 83
84. 84
85. 85
86. 86
87. 87
88. 88
89. 89
90. 90
91. 91
92. 92
93. 93
94. 94
95. 95
96. 96
97. 97
98. 98
99. 99
100. 100
101. 101
102. 102
103. 103
104. 104
105. 105
106. 106
107. # Import required libraries
108. import pandas as pd
109. import dash
110. from dash import dcc
111. from dash import html
112. from dash.dependencies import Input, Output
113. import plotly.express as px
114. # Read the airline data into pandas dataframe
115. airline\_data = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline\_data.csv',
116. encoding = "ISO-8859-1",
117. dtype={'Div1Airport': str, 'Div1TailNum': str,
118. 'Div2Airport': str, 'Div2TailNum': str})
119. # Create a dash application
120. app = dash.Dash(\_\_name\_\_)
121. # Build dash app layout
122. app.layout = html.Div(children=[ html.H1('Flight Delay Time Statistics',
123. style={'textAlign': 'center', 'color': '#503D36',
124. 'font-size': 30}),
125. html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',
126. type='number', style={'height':'35px', 'font-size': 30}),],
127. style={'font-size': 30}),
128. html.Br(),
129. html.Br(),
130. # Segment 1
131. html.Div([
132. html.Div(dcc.Graph(id='carrier-plot')),
133. html.Div(dcc.Graph(id='weather-plot'))
134. ], style={'display': 'flex'}),
135. # Segment 2
136. html.Div([
137. html.Div(dcc.Graph(id='nas-plot')),
138. html.Div(dcc.Graph(id='security-plot'))
139. ], style={'display': 'flex'}),
140. # Segment 3
141. html.Div(dcc.Graph(id='late-plot'), style={'width':'65%'})
142. ])
143. """ Compute\_info function description
144. This function takes in airline data and selected year as an input and performs computation for creating charts and plots.
145. Arguments:
146. airline\_data: Input airline data.
147. entered\_year: Input year for which computation needs to be performed.
149. Returns:
150. Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.
151. """
152. def compute\_info(airline\_data, entered\_year):
153. # Select data
154. df = airline\_data[airline\_data['Year']==int(entered\_year)]
155. # Compute delay averages
156. avg\_car = df.groupby(['Month','Reporting\_Airline'])['CarrierDelay'].mean().reset\_index()
157. avg\_weather = df.groupby(['Month','Reporting\_Airline'])['WeatherDelay'].mean().reset\_index()
158. avg\_NAS = df.groupby(['Month','Reporting\_Airline'])['NASDelay'].mean().reset\_index()
159. avg\_sec = df.groupby(['Month','Reporting\_Airline'])['SecurityDelay'].mean().reset\_index()
160. avg\_late = df.groupby(['Month','Reporting\_Airline'])['LateAircraftDelay'].mean().reset\_index()
161. return avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late
162. """Callback Function
163. Function that returns fugures using the provided input year.
164. Arguments:
165. entered\_year: Input year provided by the user.
167. Returns:
168. List of figures computed using the provided helper function `compute\_info`.
169. """
170. # Callback decorator
171. @app.callback( [
172. Output(component\_id='carrier-plot', component\_property='figure'),
173. Output(component\_id='weather-plot', component\_property='figure'),
174. Output(component\_id='nas-plot', component\_property='figure'),
175. Output(component\_id='security-plot', component\_property='figure'),
176. Output(component\_id='late-plot', component\_property='figure')
177. ],
178. Input(component\_id='input-year', component\_property='value'))
179. # Computation to callback function and return graph
180. def get\_graph(entered\_year):
182. # Compute required information for creating graph from the data
183. avg\_car, avg\_weather, avg\_NAS, avg\_sec, avg\_late = compute\_info(airline\_data, entered\_year)
185. # Line plot for carrier delay
186. carrier\_fig = px.line(avg\_car, x='Month', y='CarrierDelay', color='Reporting\_Airline', title='Average carrrier delay time (minutes) by airline')
187. # Line plot for weather delay
188. weather\_fig = px.line(avg\_weather, x='Month', y='WeatherDelay', color='Reporting\_Airline', title='Average weather delay time (minutes) by airline')
189. # Line plot for nas delay
190. nas\_fig = px.line(avg\_NAS, x='Month', y='NASDelay', color='Reporting\_Airline', title='Average NAS delay time (minutes) by airline')
191. # Line plot for security delay
192. sec\_fig = px.line(avg\_sec, x='Month', y='SecurityDelay', color='Reporting\_Airline', title='Average security delay time (minutes) by airline')
193. # Line plot for late aircraft delay
194. late\_fig = px.line(avg\_late, x='Month', y='LateAircraftDelay', color='Reporting\_Airline', title='Average late aircraft delay time (minutes) by airline')
196. return[carrier\_fig, weather\_fig, nas\_fig, sec\_fig, late\_fig]
197. # Run the app
198. if \_\_name\_\_ == '\_\_main\_\_':
199. app.run\_server()