**Dash Components**



**Objectives**

After completing the lab you will be able to:

* Know how to add multiple graphs to the dashboard
* Work with Dash Callbacks to handle multiple outputs

**Estimated time needed:** 30 minutes

**Dataset Used**

[Airline Reporting Carrier On-Time Performance](https://developer.ibm.com/exchanges/data/all/airline?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01&cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork-20297740&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newsletter.M12345678&cvo_campaign=000026UJ) dataset from [Data Asset eXchange](https://developer.ibm.com/exchanges/data?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01&cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork-20297740&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newsletter.M12345678&cvo_campaign=000026UJ)

# 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 course and project related labs. It utilizes Theia, an open-source IDE platform, that can be run on 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, dowloaded or installed. The next time you launch this lab, a new environment is 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 to your time accordingly and finish your labs in a single session.

# Let's start creating dash application

## Theme

Analyze flight delays in a dashboard.

## Dashboard Components

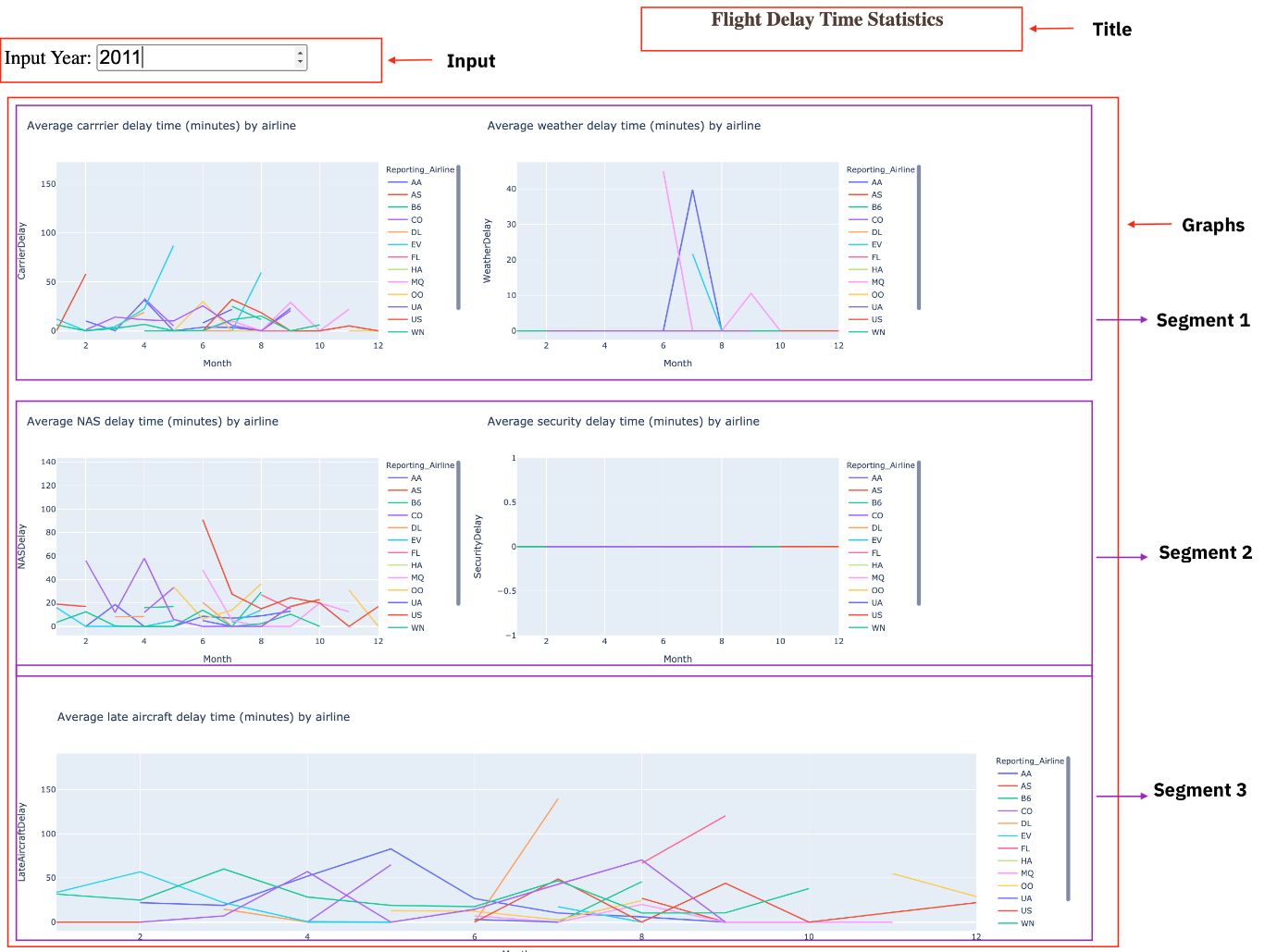
* Monthly average carrier delay by reporting airline for the given year.
* Monthly average weather delay by reporting airline for the given year.
* Monthly average national air system delay by reporting airline for the given year.
* Monthly average security delay by reporting airline for the given year.
* Monthly average late aircraft delay by reporting airline for the given year.

NOTE: Year range should be between 2010 and 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
* 5 Charts conveying the different types of flight delay. Chart section is divided into three segments.
  + Carrier and Weather delay in the first segment
  + National air system and Security delay in the second segment
  + Late aircraft delay in the third segment



### To do:

* Design layout for the application.
* Create a callback function. Add callback decorator, define inputs and outputs.
* Review the helper function that performs computation on the provided inputs.
* Create 5 line graphs.
* Run the application.

**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 -m pip install packaging

Copied!

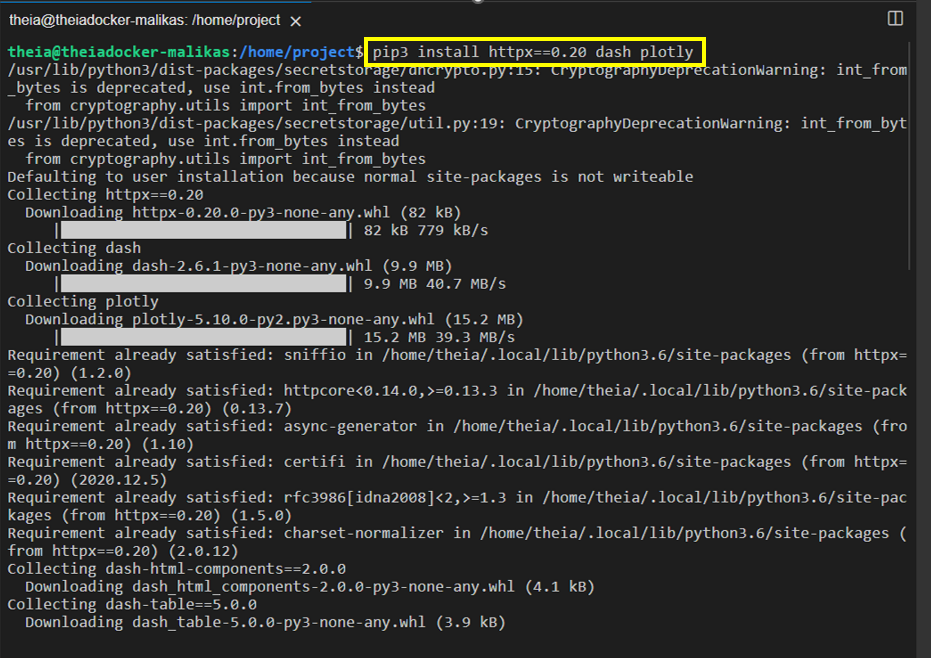
1. 1
2. python3 -m pip install pandas dash

Copied!

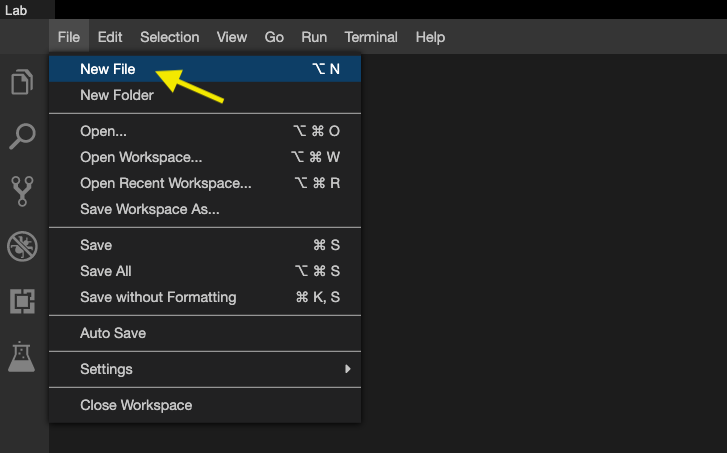


1. 1
2. pip3 install httpx==0.20 dash plotly

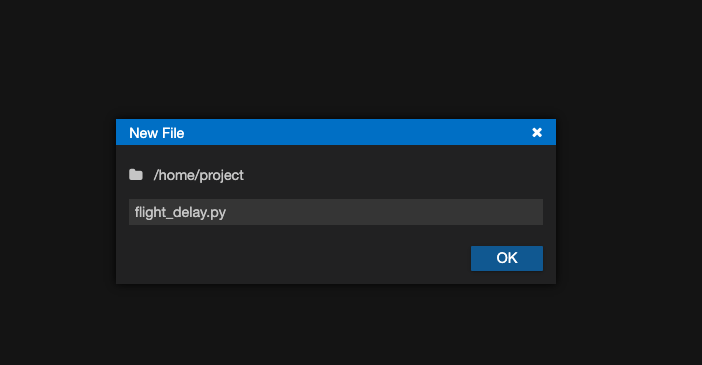
Copied!



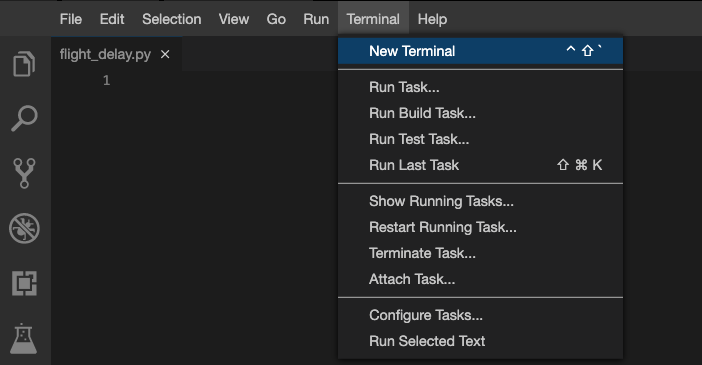
* Create a new python script, by clicking on the menu bar and selecting **File**->**New File**, as in the image below.



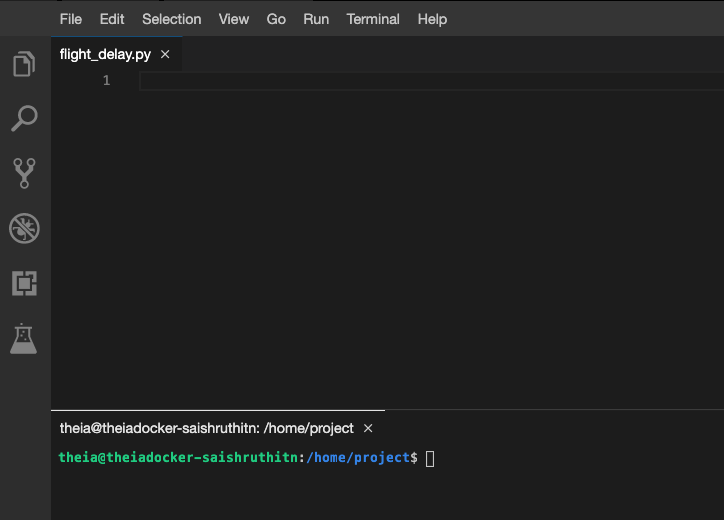
* Provide the file name as flight\_details.py



* 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**

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. import dash\_html\_components as html
20. import dash\_core\_components as dcc
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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01) 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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01) component id as carrier-plot.

##### Second inner division

* Add dcc.Graph() component.
* Update [dcc.Graph](https://dash.plotly.com/dash-core-components/graph?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01) 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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01) component id as nas-plot.

##### Second inner division

* Add dcc.Graph() component.
* Update [dcc.Graph](https://dash.plotly.com/dash-core-components/graph?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01) 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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01) component id as late-plot.

# 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

**TASK 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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01)
* 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. import dash\_html\_components as html
111. import dash\_core\_components as dcc
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()

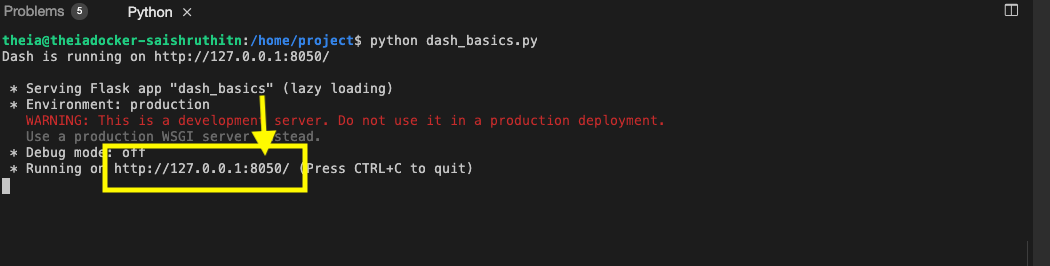
**TASK 6 - Run the application**

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

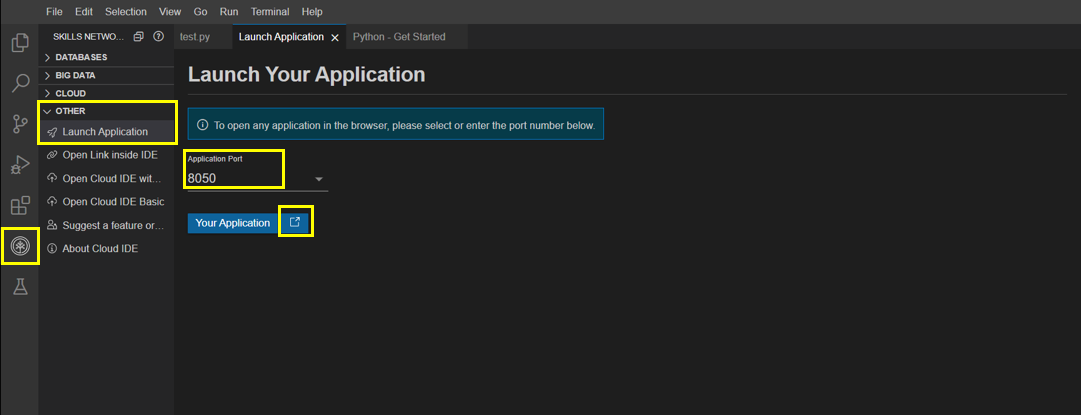
1. 1
2. python3 flight\_delay.py

Copied!

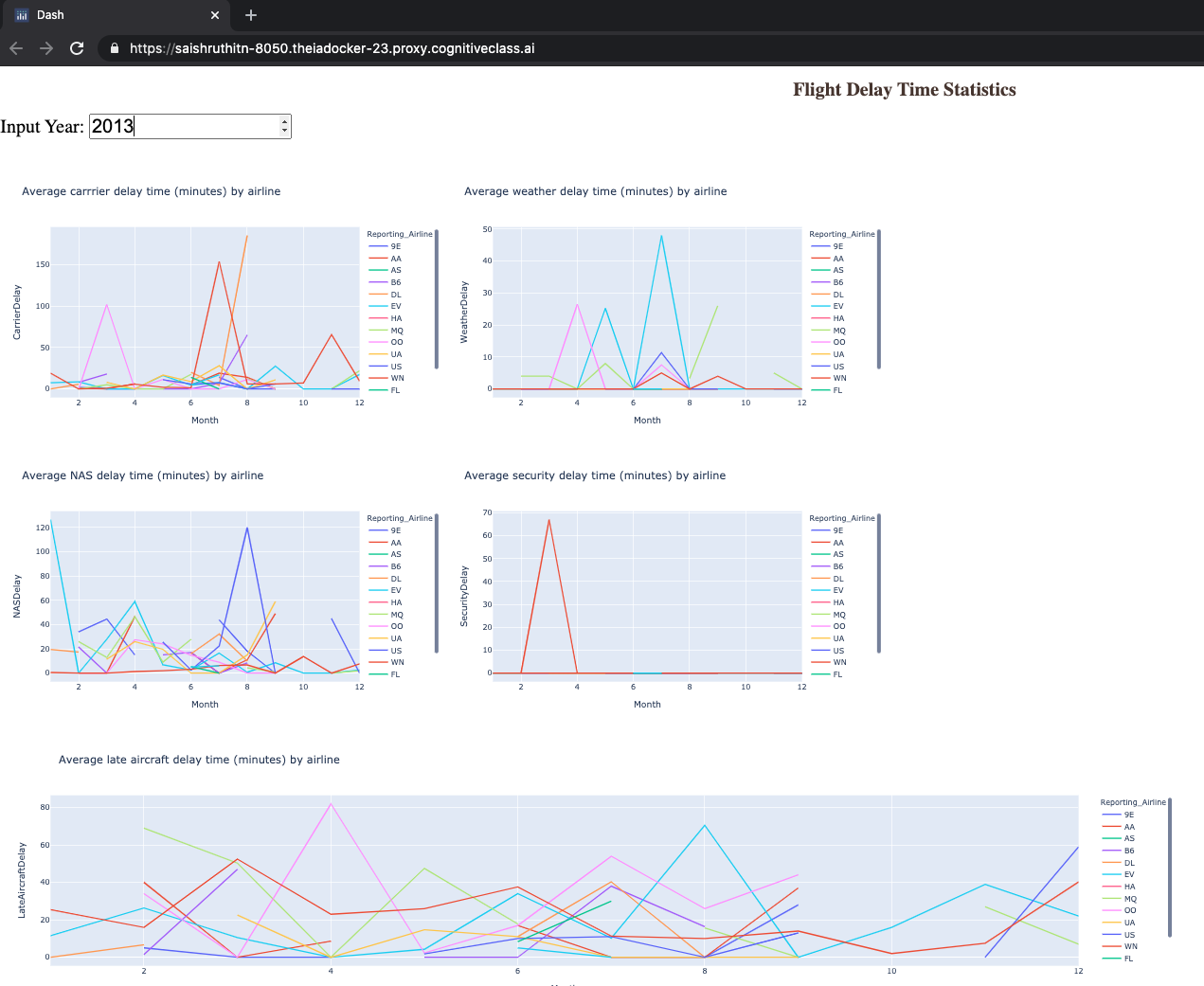
* Observe the port number shown in the terminal.



* 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:



**Congratulations, you have successfully created your dash application!**

# Exercise : Practice Tasks

You will practice some tasks to update the dashboard.

1. *Change the title to the dashboard from "Flight Delay Time Statistics" to "Flight Details Statistics Dashboard" using HTML H1 component and font-size as 35.*

Answer

*html.H1(‘Flight Details Statistics Dashboard’,style={‘textAlign’: ‘center’, ‘color’: ‘#503D36’, ‘font-size’: 35}),*

1. *Save the above changes and rename file as flight\_details.pyand relaunch the dashboard application to see the updated dashboard title.*

Answer

*Click on file –> save file as and write file name as flight\_details.py. Then go to terminal and Run the command python3 flight\_details.py to open the updated file again and relaunch the application by entering the port number.The updated dashboard title will be seen.*

1. *Write a command to stop the running app in the terminal*

Answer

*Press ctrl+c inside the terminal to stop the dash application.*

## Author

[Saishruthi Swaminathan](https://www.linkedin.com/in/saishruthi-swaminathan?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork970-2022-01-01&cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork-20297740&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newsletter.M12345678&cvo_campaign=000026UJ)

## Changelog

| **Date** | **Version** | **Changed by** | **Change Description** |
| --- | --- | --- | --- |
| 05-07-2021 | 1.0 | Saishruthi | Initial version created |
| 24-08-2022 | 1.1 | Pratiksha | Updated Instructions |
| 29-08-2022 | 1.2 | Pratiksha Verma | Updated Screenshot |

### © IBM Corporation 2020. All rights reserved.