**Practice Assignment Part 2**



**Objectives**

After completing the lab you will be able to:

* Create a dash board layout with a RadioItem and a Dropdown
* Add Pie chart and Bar chart

**Estimated time needed:** 45 minutes

# 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 ᴥd computer on the cloud嬥ted 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.

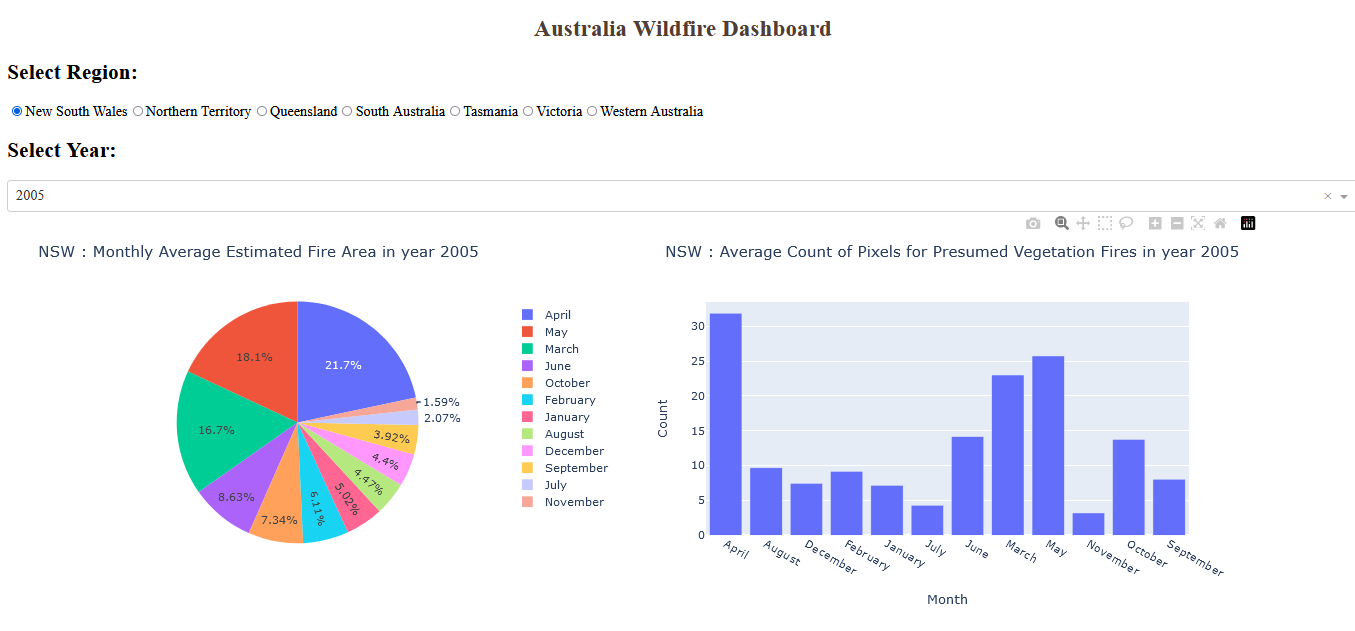
If you finish only part of the lab and return later, you may have to start from the beginning. So, it is a good idea to plan to your time accordingly and finish your labs in a single session.

# Components of Dashboard and Expected layout

## Components of the Dashboard

1. Select Region
2. Select Year
3. Divison to display
   * Pie Chart to display Monthly Average Estimated Fire Area for the selected Regions in the selected Year
   * Bar Chart to display Monthly Average Count of Pixels for Presumed Vegetation Fires for the selected Regions in the selected Year

## Expected Layout



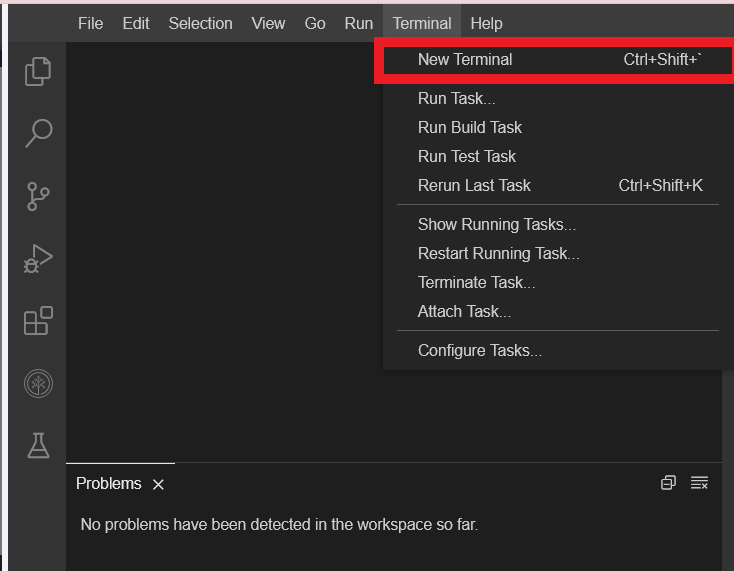
## Requirements to create the expected result

* A dropdown [menu](https://dash.plotly.com/dash-core-components/dropdown): For choosing year
* A radioitem for choosing the Region
* The layout will be designed as follows:
* An outer division with two inner divisions (as shown in the expected layout)
* One of the inner divisions will have information about the radioitem and dropdown (which are the input) and the other one is for adding graphs(the 2 output graphs).
* Callback function to compute data, create graph and return to the layout.

### To do:

1. Import required libraries and read the dataset
2. Create an application layout
3. Add title to the dashboard using HTML H1 component
4. Add a radioitem using dcc.RaioItems and dropdown using dcc.dropdown
5. Add the pie chart and bar chart core graph components.
6. Run the app

# Get the tool ready

1. Open a new terminal, by clicking on the menu bar and selecting **Terminal**->**New Terminal**, as in the image below.  
   
2. Install python packages required to run the application. Copy and paste the below command to the terminal.
3. 1
4. pip3.8 install setuptools

Copied!

1. 1
2. python3.8 -m pip install packaging

Copied!

1. 1
2. python3.8 -m pip install pandas dash

Copied!

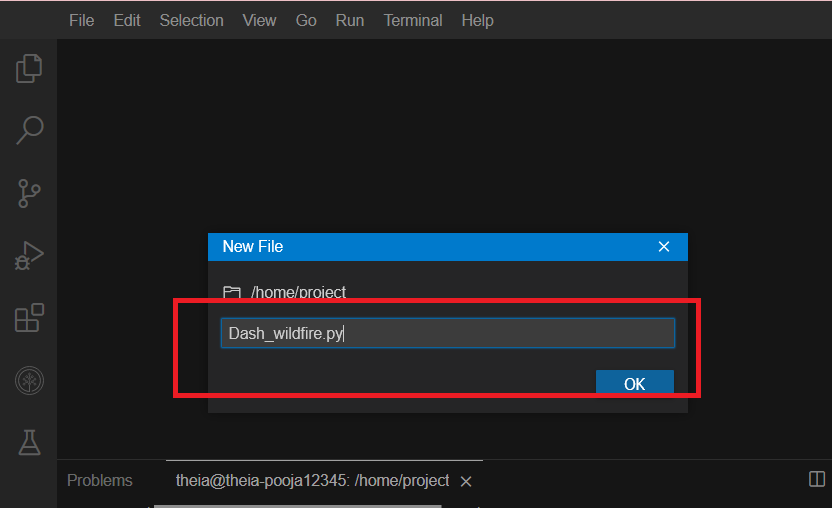
1. 1
2. pip3 install httpx==0.20 dash plotly

Copied!

## packageinstall

**Let's create the application**

* Create a new file called Dash\_wildfire.py



# Get the application skeleton

You can use this as a base code to complete the task below.

## Structure of the skeleton file

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. import pandas as pd
95. import dash
96. from dash import html, dcc
97. from dash.dependencies import Input, Output, State
98. import plotly.graph\_objects as go
99. import plotly.express as px
100. from dash import no\_update
101. import datetime as dt
102. #Create app
103. app = dash.Dash(\_\_name\_\_)
104. # Clear the layout and do not display exception till callback gets executed
105. app.config.suppress\_callback\_exceptions = True
106. # Read the wildfire data into pandas dataframe
107. df = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Historical\_Wildfires.csv')
108. #Extract year and month from the date column
109. df['Month'] = pd.to\_datetime(df['Date']).dt.month\_name() #used for the names of the months
110. df['Year'] = pd.to\_datetime(df['Date']).dt.year
111. #Layout Section of Dash
112. #Task 2.1 Add the Title to the Dashboard
113. app.layout = html.Div(children=[html.H1(..................),
114. # TASK 2.2: Add the radio items and a dropdown right below the first inner division
115. #outer division starts
116. html.Div([
117. # First inner divsion for adding dropdown helper text for Selected Drive wheels
118. html.Div([
119. html.H2(.........),
120. #Radio items to select the region
121. #dcc.RadioItems(['NSW',.....], value ='...', id='...',inline=True)]),
122. dcc.RadioItems([{"label":"New South Wales","value": "NSW"},
123. {..........},
124. {..........},
125. {..........},
126. {..........},
127. {..........},
128. {"label":"...","value": ..}], value = "...", id='.....,inline=True)]),
129. #Dropdown to select year
130. html.Div([
131. html.H2('.........', style={...........}),
132. dcc.Dropdown(.....................)
133. ]),
134. #Second Inner division for adding 2 inner divisions for 2 output graphs
135. #TASK 2.3: Add two empty divisions for output inside the next inner division.
136. html.Div([
138. html.Div([ ], id='........'),
139. html.Div([ ], id='.........')
140. ], style={'.........}),
141. ])
142. #outer division ends
143. ])
144. #layout ends
145. #TASK 2.4: Add the Ouput and input components inside the app.callback decorator.
146. #Place to add @app.callback Decorator
147. @app.callback([Output(component\_id=.........., component\_property=..........),
148. Output(component\_id=.........., component\_property=..........)],
149. [Input(component\_id=.........., component\_property=..........),
150. Input(component\_id=.........., component\_property=..........)])
152. #TASK 2.5: Add the callback function.
153. #Place to define the callback function .
154. def reg\_year\_display(input\_region,input\_year):
156. #data
157. region\_data = df[df['Region'] == input\_region]
158. y\_r\_data = region\_data[region\_data['Year']==input\_year]
159. #Plot one - Monthly Average Estimated Fire Area
161. est\_data = .........................
163. fig1 = px.pie(.............., title="{} : Monthly Average Estimated Fire Area in year {}".format(input\_region,input\_year))
165. #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
166. veg\_data = .............................
167. fig2 = px.bar(..............., title='{} : Average Count of Pixels for Presumed Vegetation Fires in year {}'.format(input\_region,input\_year))
169. return [.......,
170. ......... ]
171. if \_\_name\_\_ == '\_\_main\_\_':
172. app.run\_server()

**TASK 2.1: Add title to the dashboard**

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

* Application title is Australia Wildfire Dashboard
* Use style parameter provided below to make the title center aligned, with color code #503D36, and font-size as 26

1. 1
2. 2
3. 3
4. html.H1('Australia Wildfire Dashboard',
5. style={'textAlign': 'center', 'color': '#503D36',
6. 'font-size': 26}),

Copied!

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



Reference Links:  
[H1 component](https://dash.plotly.com/dash-html-components/h1)

[Dash HTML Components](https://dash.plotly.com/dash-html-components)

**TASK 2.2: Add the radio items and a dropdown right below the first inner division.**

Radio items to choose the Region

The radio items work similar to the dropdown, you need to call dcc.RadioItems and pass the list of items. Make use of inline=True property to display the radio items in a horizontal line

* You can extract the regions from the dataframe using df.Region.unque() or pass the list of all regions directly as ['NSW','QL','SA','TA','VI','WA','NT'] .
* Assign radioitems id as region
* Label as Select Region
* value as NSW

For your reference below are the abrivations used in the dataset for regions  
NSW - New South Wales  
NT - Northern Territory  
QL - Queensland  
SA - South Australia  
TA - Tasmania  
VI - Victoria  
WA - Western Australia  
Read more on [RadioItems](https://dash.plotly.com/dash-core-components/radioitems)

1. 1
2. 2
3. 3
4. 4
5. 5
6. html.Div([
7. html.H2('Select Region:', style={'margin-right': '2em'}),
8. #Radio items to select the region
9. dcc.RadioItems(['NSW','QL','SA','TA','VI','WA'], 'NSW', id='region',inline=True)]),

Copied!

* or you can use labels:value pair a well in raioditems as below

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. #OR you can use labels:value pair a well in raioditems as below
11. #Radio items to select the region
12. dcc.RadioItems([{"label":"New South Wales","value": "NSW"},
13. {"label":"Northern Territory","value": "NT"},
14. {"label":"Queensland","value": "QL"},
15. {"label":"South Australia","value": "SA"},
16. {"label":"Tasmania","value": "TA"},
17. {"label":"Victoria","value": "VI"},
18. {"label":"Western Australia","value": "WA"}],"NSW", id='region',inline=True)]),

Copied!

Dropdown to choose the Year

* The dropdown has an id as year.
* The label as Select Year
* The values allowed in the dropdown are years from 2005 to 2020
* The default value when the dropdown is displayed is 2005.

1. 1
2. 2
3. 3
4. 4
5. 5
6. html.Div([
7. html.H2('Select Year:', style={'margin-right': '2em'}),
8. dcc.Dropdown(df.Year.unique(), value = 2005,id='year')
9. #notice the use of unique() from pandas to fetch the values of year from the dataframe for dropdown
10. ]),

Copied!

Reference [link](https://dash.plotly.com/dash-core-components/dropdown)

# TASK 2.3: Add two empty divisions for output inside the next inner division.

* Use 2 html.Div() tags .
* Provide division ids as plot1 and plot2.

1. 1
2. 2
3. html.Div([ ], id='plot1'),
4. html.Div([ ], id='plot2')

**TASK 2.4: Add the Ouput and input components inside the app.callback decorator.**

* The inputs and outputs of our application’s interface are described declaratively as the arguments of @app.callback decorator.

-In Dash, the inputs and outputs of our application are simply the properties of a particular component.

* In this example, we have two inputs:-  
  - input for Region is the value property of the component that has the ID region  
  - input for Year is the value property of the component that has the ID year
* Our layout has 2 outputs so we need to create 2 output components.

It is a list with 2 output parameters with component id and property. Here, the component property will be children as we have created empty division and passing in dcc.Graph (figure) after computation.

Component ids will be plot1 , plot2.

1. 1
2. 2
3. 3
4. 4
5. @app.callback([Output(component\_id='plot1', component\_property='children'),
6. Output(component\_id='plot2', component\_property='children')],
7. [Input(component\_id='region', component\_property='value'),
8. Input(component\_id='year', component\_property='value')])

**TASK 2.5: Add the callback function.**

* Whenever an input property changes, the function that the callback decorator wraps will get called automatically.
* In this case let us define a function reg\_year\_display() which will be wrapped by our decorator.
* The function first filters our dataframe df by the selected value of the region from the radio items and year from the dropdown as follows
* region\_data = df[df['Region'] == input\_region]
* y\_r\_data = region\_data[region\_data['Year']==input\_year]
* For pie chart on Monthly Average Estimated Fire Area: -
  + Next we will group by the Month and calculate the mean Estimated\_fire\_area of the dataframe.
  + Use the px.pie() function to plot the pie chart
* For bar chart on Monthly Average Count of Pixels for Presumed Vegetation Fires: -
  + Next we will group by the Month and calculate the mean Count of the dataframe.
  + Use the px.bar() function to plot the bar chart

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. def reg\_year\_display(input\_region,input\_year):
21. #data
22. region\_data = df[df['Region'] == input\_region]
23. y\_r\_data = region\_data[region\_data['Year']==input\_year]
24. #Plot one - Monthly Average Estimated Fire Area
26. est\_data = y\_r\_data.groupby('Month')['Estimated\_fire\_area'].mean().reset\_index()
28. fig1 = px.pie(est\_data, values='Estimated\_fire\_area', names='Month', title="{} : Monthly Average Estimated Fire Area in year {}".format(input\_region,input\_year))
30. #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
31. veg\_data = y\_r\_data.groupby('Month')['Count'].mean().reset\_index()
32. fig2 = px.bar(veg\_data, x='Month', y='Count', title='{} : Average Count of Pixels for Presumed Vegetation Fires in year {}'.format(input\_region,input\_year))
34. return [dcc.Graph(figure=fig1),
35. dcc.Graph(figure=fig2) ]

Copied!

* Finally we return the 2 figure objects fig1 and fig2 in dcc.Graph method.
* Once you have finished coding save your code.

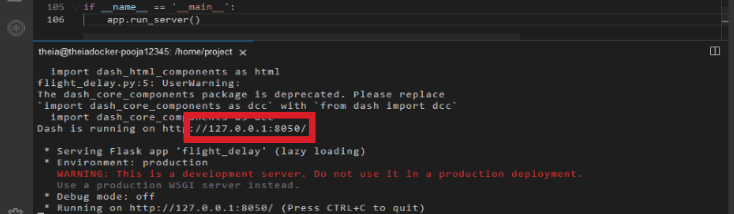
# Run the Application

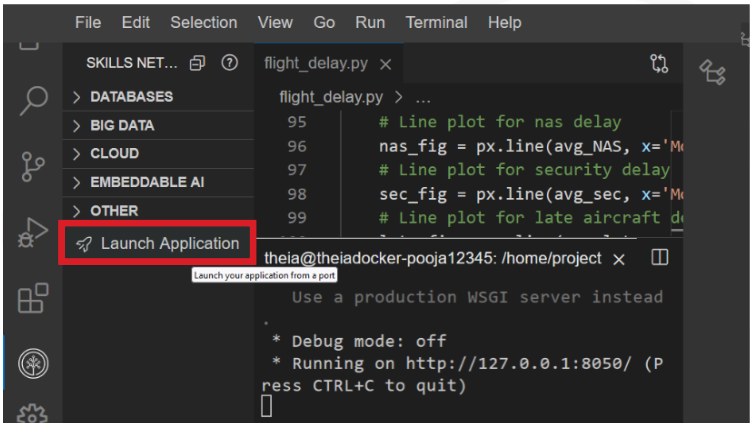
* Next Run the python file using the command

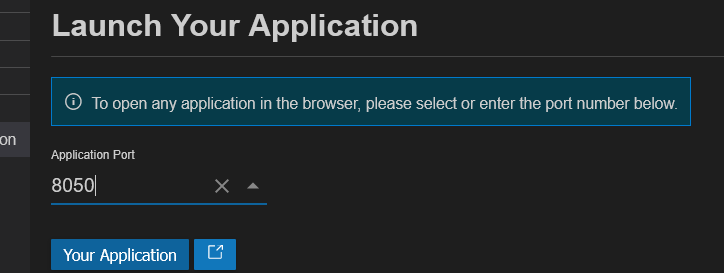
1. 1
2. python3.8 Dash\_wildfire.py

Copied!

* Observe the port number shown in the terminal.



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



Refer to the complete code Dash\_wildfire.py 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. 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. import pandas as pd
81. import dash
82. from dash import html, dcc
83. from dash.dependencies import Input, Output, State
84. import plotly.graph\_objects as go
85. import plotly.express as px
86. from dash import no\_update
87. import datetime as dt
88. #Create app
89. app = dash.Dash(\_\_name\_\_)
90. #Clear the layout and do not display exception till callback gets executed
91. app.config.suppress\_callback\_exceptions = True
92. # Read the wildfire data into pandas dataframe
93. df = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Historical\_Wildfires.csv')
94. #Extract year and month from the date column
95. df['Month'] = pd.to\_datetime(df['Date']).dt.month\_name() #used for the names of the months
96. df['Year'] = pd.to\_datetime(df['Date']).dt.year
97. #Layout Section of Dash
98. #Task 1 Add the Title to the Dashboard
99. app.layout = html.Div(children=[html.H1('Australia Wildfire Dashboard',
100. style={'textAlign': 'center', 'color': '#503D36',
101. 'font-size': 26}),
102. # TASK 2: Add the radio items and a dropdown right below the first inner division
103. #outer division starts
104. html.Div([
105. # First inner divsion for adding dropdown helper text for Selected Drive wheels
106. html.Div([
107. html.H2('Select Region:', style={'margin-right': '2em'}),
108. #Radio items to select the region
109. #dcc.RadioItems(['NSW','QL','SA','TA','VI','WA'], 'NSW', id='region',inline=True)]),
110. dcc.RadioItems([{"label":"New South Wales","value": "NSW"},
111. {"label":"Northern Territory","value": "NT"},
112. {"label":"Queensland","value": "QL"},
113. {"label":"South Australia","value": "SA"},
114. {"label":"Tasmania","value": "TA"},
115. {"label":"Victoria","value": "VI"},
116. {"label":"Western Australia","value": "WA"}],"NSW", id='region',inline=True)]),
117. #Dropdown to select year
118. html.Div([
119. html.H2('Select Year:', style={'margin-right': '2em'}),
120. dcc.Dropdown(df.Year.unique(), value = 2005,id='year')
121. ]),
122. #TASK 3: Add two empty divisions for output inside the next inner division.
123. #Second Inner division for adding 2 inner divisions for 2 output graphs
124. html.Div([
126. html.Div([ ], id='plot1'),
127. html.Div([ ], id='plot2')
128. ], style={'display': 'flex'}),
129. ])
130. #outer division ends
131. ])
132. #layout ends
133. #TASK 4: Add the Ouput and input components inside the app.callback decorator.
134. #Place to add @app.callback Decorator
135. @app.callback([Output(component\_id='plot1', component\_property='children'),
136. Output(component\_id='plot2', component\_property='children')],
137. [Input(component\_id='region', component\_property='value'),
138. Input(component\_id='year', component\_property='value')])
139. #TASK 5: Add the callback function.
140. #Place to define the callback function .
141. def reg\_year\_display(input\_region,input\_year):
142. #data
143. region\_data = df[df['Region'] == input\_region]
144. y\_r\_data = region\_data[region\_data['Year']==input\_year]
145. #Plot one - Monthly Average Estimated Fire Area
146. est\_data = y\_r\_data.groupby('Month')['Estimated\_fire\_area'].mean().reset\_index()
147. fig1 = px.pie(est\_data, values='Estimated\_fire\_area', names='Month', title="{} : Monthly Average Estimated Fire Area in year {}".format(input\_region,input\_year))
148. #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
149. veg\_data = y\_r\_data.groupby('Month')['Count'].mean().reset\_index()
150. fig2 = px.bar(veg\_data, x='Month', y='Count', title='{} : Average Count of Pixels for Presumed Vegetation Fires in year {}'.format(input\_region,input\_year))
151. return [dcc.Graph(figure=fig1),
152. dcc.Graph(figure=fig2) ]
153. if \_\_name\_\_ == '\_\_main\_\_':
154. app.run\_server()

Copied!

### Congratulations, you have successfully created dash application!

## Author

[Dr. Pooja](https://www.linkedin.com/in/p-b28802262/)

## Changelog

| **Date** | **Version** | **Changed by** | **Change Description** |
| --- | --- | --- | --- |
| 2023-07-06 | 0.1 | Dr. Pooja | Initial Lab Version |

### Oporation 2021. All rights reserved.