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 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.

Components of Dashboard and Expected layout

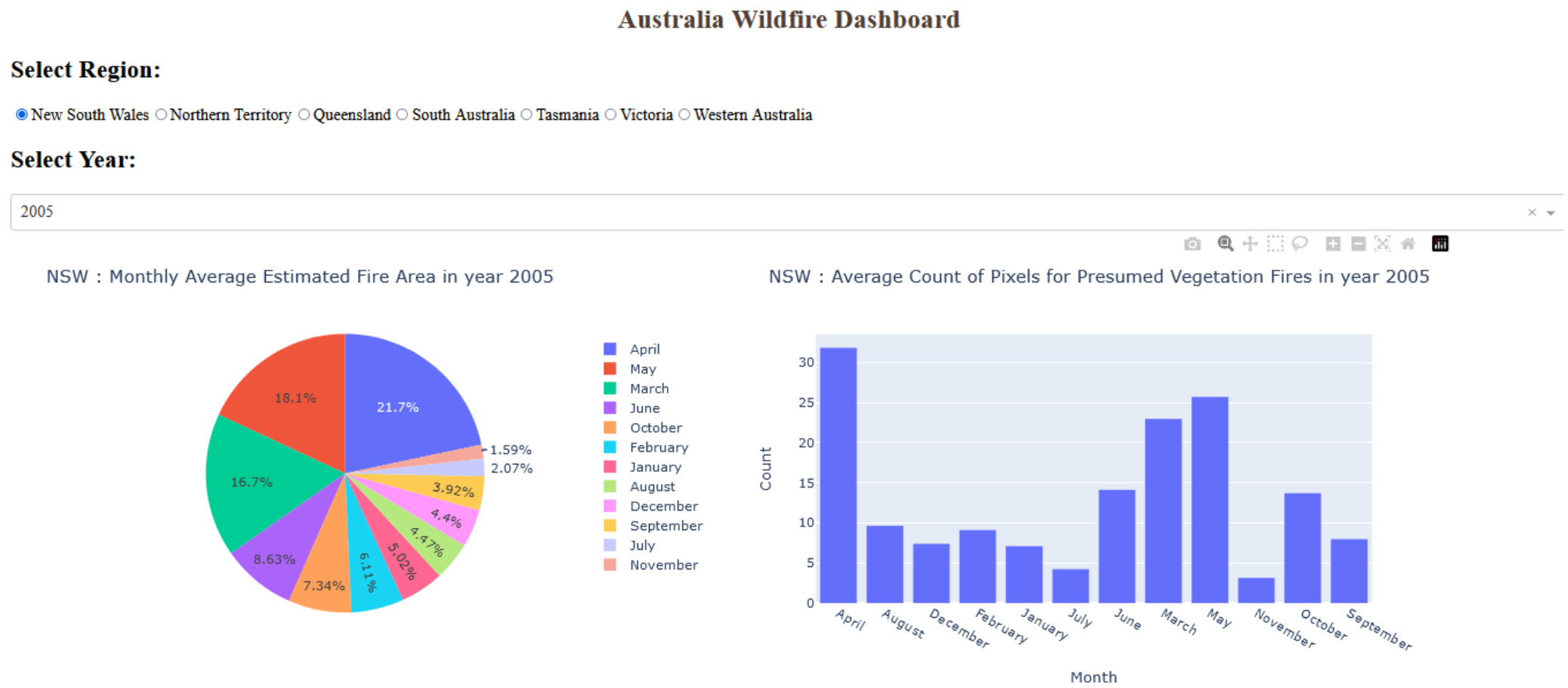
Components of the Dashboard

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: 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).

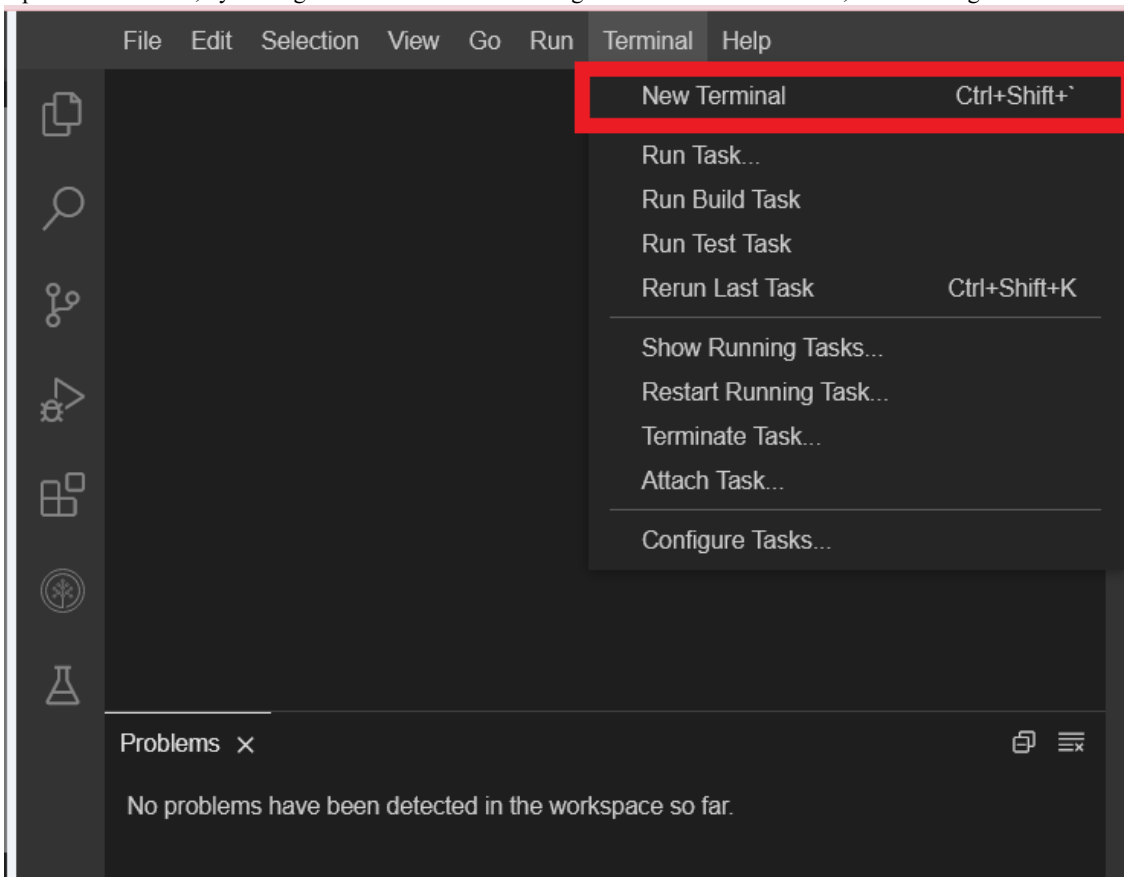
- Callback function to compute data, create graph and return to the layout.

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.

```
1. 1
```

```
1. pip3.8 install setuptools
```

Copied!

```
1. 1
```

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

Copied!

```
1. 1
```

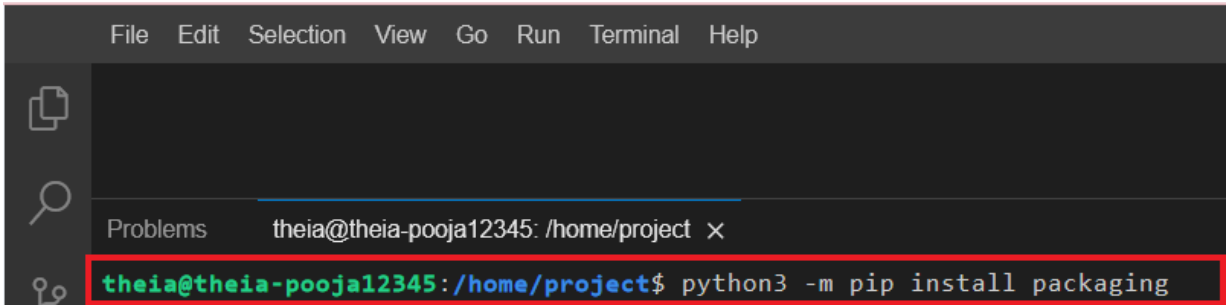
```
1. python3.8 -m pip install pandas dash
```

Copied!

```
1. 1
```

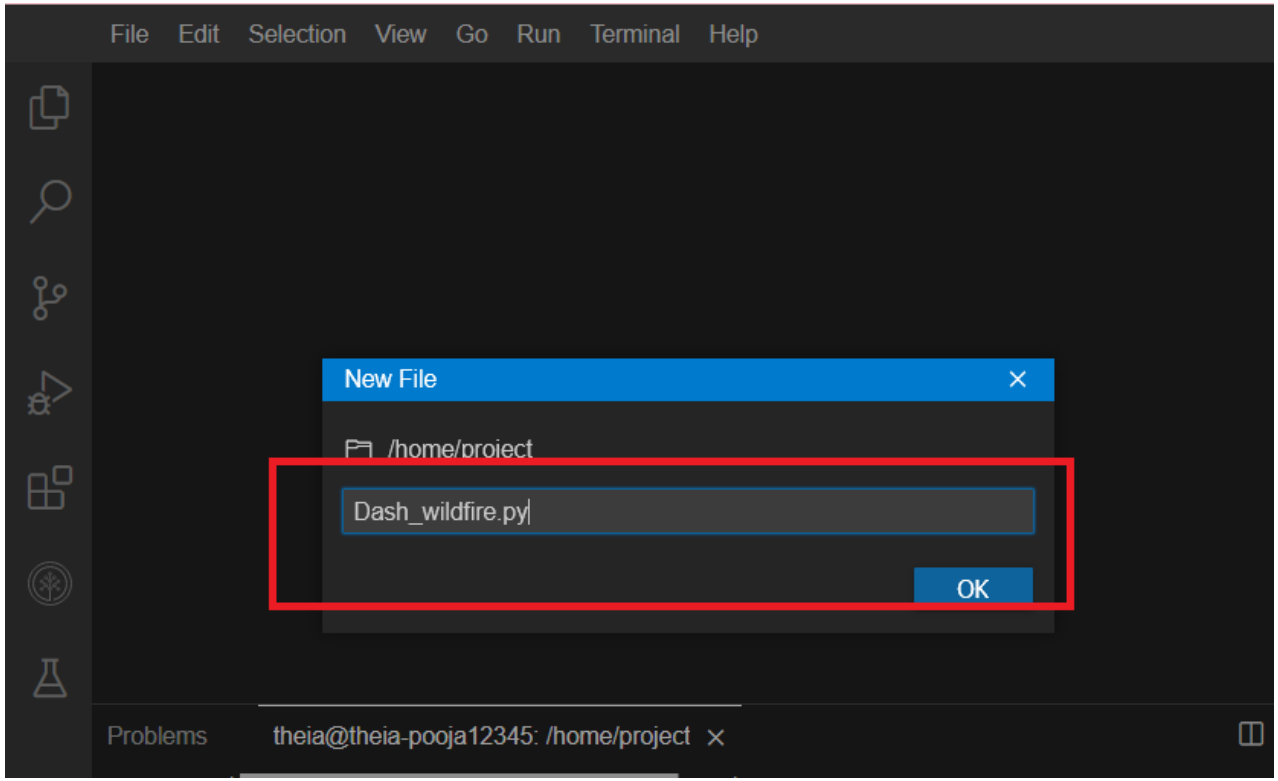
```
1. pip3 install httpx==0.20 dash plotly
```

Copied!



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
```

```
 1. import pandas as pd
 2. import dash
 3. from dash import html, dcc
 4. from dash.dependencies import Input, Output, State
 5. import plotly.graph_objects as go
 6. import plotly.express as px
 7. from dash import no_update
 8. import datetime as dt
 9.
10. #Create app
11.
12. app = dash.Dash(__name__)
13.
14. # Clear the layout and do not display exception till callback gets executed
15. app.config.suppress_callback_exceptions = True
16.
17. # Read the wildfire data into pandas dataframe
18. df =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Historical_Wildfires.csv')
19.
20. #Extract year and month from the date column
21.
22. df['Month'] = pd.to_datetime(df['Date']).dt.month_name() #used for the names of the months
23. df['Year'] = pd.to_datetime(df['Date']).dt.year
24.
25. #Layout Section of Dash
26.
27. #Task 2.1 Add the Title to the Dashboard
28. app.layout = html.Div(children=[html.H1(.................),
29.
30. # TASK 2.2: Add the radio items and a dropdown right below the first inner division
31. #outer division starts
32.      html.Div([
33.                     # First inner divsion for  adding dropdown helper text for Selected Drive wheels
34.                      html.Div([
35.                              html.H2(.........),
36.
37.                      #Radio items to select the region
38.                      #dcc.RadioItems(['NSW',.....], value ='...', id='...',inline=True)]),
39.                      dcc.RadioItems([{"label":"New South Wales","value": "NSW"},
40.                                      {.........},
41.                                      {.........},
42.                                      {.........},
43.                                      {.........},
44.                                      {.........},
45.                                      {"label":"...","value": ..}], value = "...", id='.....,inline=True)]),
46.                      #Dropdown to select year
47.                      html.Div([
```

```
                            html.H2('.........', style={...........}),
                        dcc.Dropdown(.....................)
                    ]),
#Second Inner division for adding 2 inner divisions for 2 output graphs
#TASK 2.3: Add two empty divisions for output inside the next inner division.
                    html.Div([

                        html.Div([ ], id='........'),
                        html.Div([ ], id='.........')
                    ], style={'.........}),

    ])
    #outer division ends

])
#layout ends
#TASK 2.4: Add the Ouput and input components inside the app.callback decorator.
#Place to add @app.callback Decorator
@app.callback([Output(component_id=.........., component_property=..........),
               Output(component_id=.........., component_property=..........)],
               [Input(component_id=.........., component_property=..........),
                Input(component_id=.........., component_property=..........)])


#TASK 2.5: Add the callback function.
#Place to define the callback function .
def reg_year_display(input_region,input_year):

    #data
   region_data = df[df['Region'] == input_region]
   y_r_data = region_data[region_data['Year']==input_year]
    #Plot one - Monthly Average Estimated Fire Area

   est_data = .......................

   fig1 = px.pie(............., title="{} : Monthly Average Estimated Fire Area in year {}".format(input_region,input_year))

     #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
   veg_data = ............................
   fig2 = px.bar(............., title='{} : Average Count of Pixels for Presumed Vegetation Fires in year {}'.format(input_region,input_year))

   return [.......,
            ......... ]

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

Copied!

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
2
3
```

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

Copied!

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

Australia Wildfire Dashboard

Reference Links:
H1 component

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

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

```
4. 4
5. 5
```

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

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
```

```
1. #OR you can use labels:value pair a well in raioditems as below
2.              #Radio items to select the region
3.                     dcc.RadioItems([{"label":"New South Wales","value": "NSW"},
4.                                     {"label":"Northern Territory","value": "NT"},
5.                                     {"label":"Queensland","value": "QL"},
6.                                     {"label":"South Australia","value": "SA"},
7.                                     {"label":"Tasmania","value": "TA"},
8.                                     {"label":"Victoria","value": "VI"},
9.                                     {"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
```

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

Copied!

Reference link

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
```

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

Copied!

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
```

```
1. @app.callback([Output(component_id='plot1', component_property='children'),
2.                Output(component_id='plot2', component_property='children')],
3.                [Input(component_id='region', component_property='value'),
4.                 Input(component_id='year', component_property='value')])
```

Copied!

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
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

```
def reg_year_display(input_region,input_year):

    #data
   region_data = df[df['Region'] == input_region]
   y_r_data = region_data[region_data['Year']==input_year]
    #Plot one - Monthly Average Estimated Fire Area

   est_data = y_r_data.groupby('Month')['Estimated_fire_area'].mean().reset_index()

   fig1 = px.pie(est_data, values='Estimated_fire_area', names='Month', title="{} : Monthly Average Estimated Fire Area in year {}".format(input_region,input_year))

     #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
   veg_data = y_r_data.groupby('Month')['Count'].mean().reset_index()

   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))

   return [dcc.Graph(figure=fig1),
           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

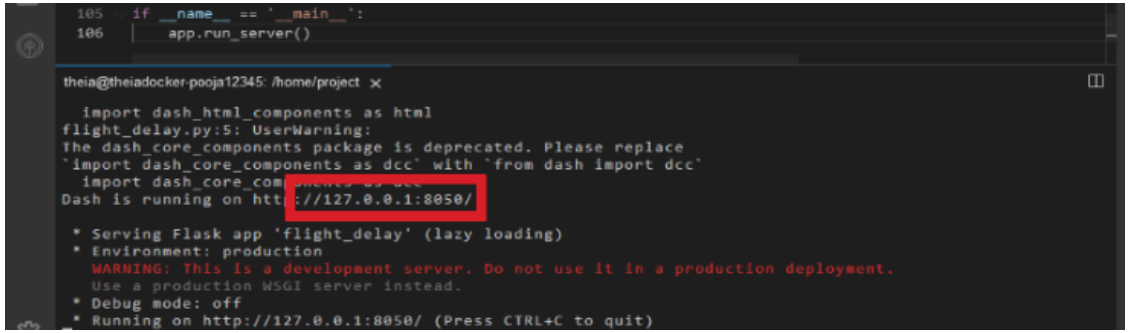
- Next Run the python file using the command

```
1
```

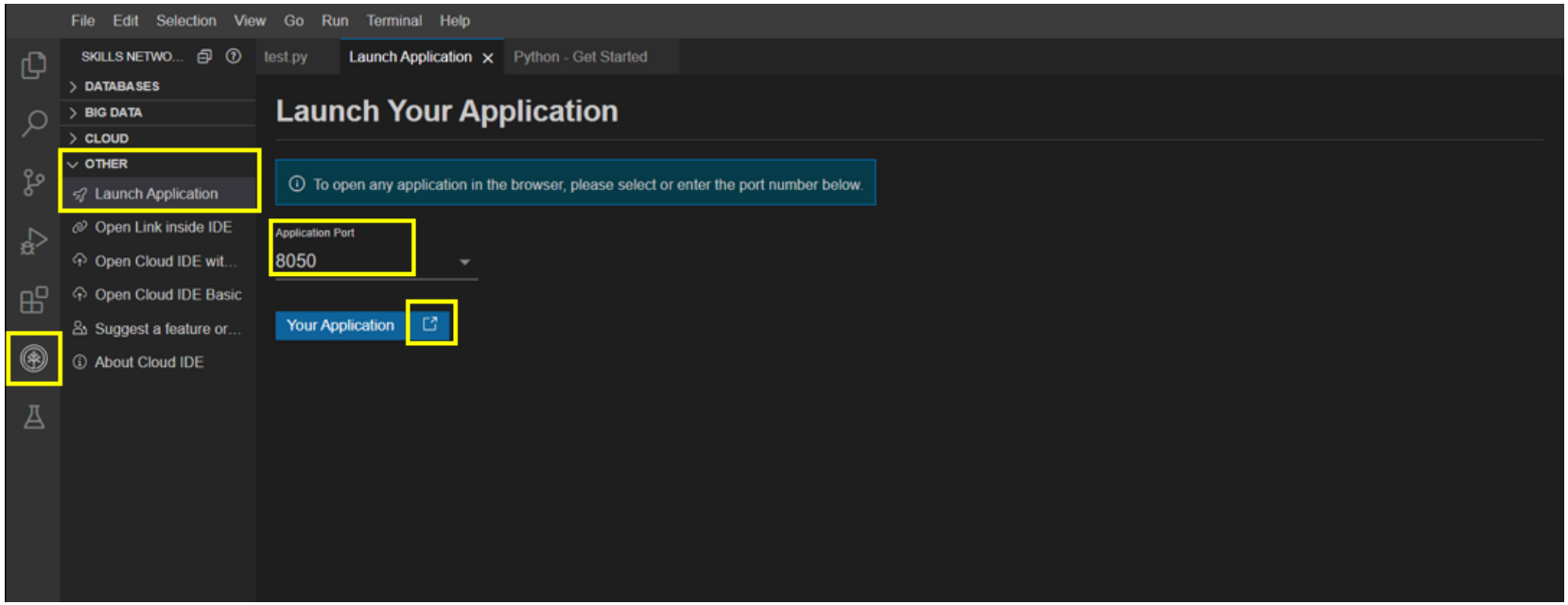
```
python3.8 Dash_wildfire.py
```

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
```

```
 1. import pandas as pd
 2. import dash
 3. from dash import html, dcc
 4. from dash.dependencies import Input, Output, State
 5. import plotly.graph_objects as go
 6. import plotly.express as px
 7. from dash import no_update
 8. import datetime as dt
 9. #Create app
10. app = dash.Dash(__name__)
11. #Clear the layout and do not display exception till callback gets executed
12. app.config.suppress_callback_exceptions = True
13. # Read the wildfire data into pandas dataframe
14. df =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Historical_Wildfires.csv')
15. #Extract year and month from the date column
16. df['Month'] = pd.to_datetime(df['Date']).dt.month_name() #used for the names of the months
17. df['Year'] = pd.to_datetime(df['Date']).dt.year
18. #Layout Section of Dash
19. #Task 1 Add the Title to the Dashboard
20. app.layout = html.Div(children=[html.H1('Australia Wildfire Dashboard',
21.                                 style={'textAlign': 'center', 'color': '#503D36',
22.                                 'font-size': 26}),
23. # TASK 2: Add the radio items and a dropdown right below the first inner division
24.      #outer division starts
25.      html.Div([
26.                     # First inner divsion for  adding dropdown helper text for Selected Drive wheels
27.                      html.Div([
28.                              html.H2('Select Region:', style={'margin-right': '2em'}),
29.
30.                      #Radio items to select the region
31.                      #dcc.RadioItems(['NSW','QL','SA','TA','VI','WA'], 'NSW', id='region',inline=True)]),
32.                      dcc.RadioItems([{"label":"New South Wales","value": "NSW"},
33.                                      {"label":"Northern Territory","value": "NT"},
34.                                      {"label":"Queensland","value": "QL"},
35.                                      {"label":"South Australia","value": "SA"},
36.                                      {"label":"Tasmania","value": "TA"},
37.                                      {"label":"Victoria","value": "VI"},
38.                                      {"label":"Western Australia","value": "WA"}],"NSW", id='region',inline=True)]),
39.                      #Dropdown to select year
40.                      html.Div([
41.                              html.H2('Select Year:', style={'margin-right': '2em'}),
42.                          dcc.Dropdown(df.Year.unique(), value = 2005,id='year')
43.                      ]),
44. #TASK 3: Add two empty divisions for output inside the next inner division.
45.          #Second Inner division for adding 2 inner divisions for 2 output graphs
46.                      html.Div([
47.
48.                          html.Div([ ], id='plot1'),
49.                          html.Div([ ], id='plot2')
50.                      ], style={'display': 'flex'}),
51.
```

```
    ])
    #outer division ends

])
#layout ends
#TASK 4: Add the Ouput and input components inside the app.callback decorator.
#Place to add @app.callback Decorator
@app.callback([Output(component_id='plot1', component_property='children'),
               Output(component_id='plot2', component_property='children')],
               [Input(component_id='region', component_property='value'),
                Input(component_id='year', component_property='value')])
#TASK 5: Add the callback function.
#Place to define the callback function .
def reg_year_display(input_region,input_year):
    #data
   region_data = df[df['Region'] == input_region]
   y_r_data = region_data[region_data['Year']==input_year]
    #Plot one - Monthly Average Estimated Fire Area
   est_data = y_r_data.groupby('Month')['Estimated_fire_area'].mean().reset_index()
    fig1 = px.pie(est_data, values='Estimated_fire_area', names='Month', title="{} : Monthly Average Estimated Fire Area in year {}".format(input_region,input_year))
     #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
   veg_data = y_r_data.groupby('Month')['Count'].mean().reset_index()
   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))
   return [dcc.Graph(figure=fig1),
           dcc.Graph(figure=fig2) ]
if __name__ == '__main__':
    app.run_server()

```

Copied!

Congratulations, you have successfully created dash application!

Author

Dr. Pooja

Changelog

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