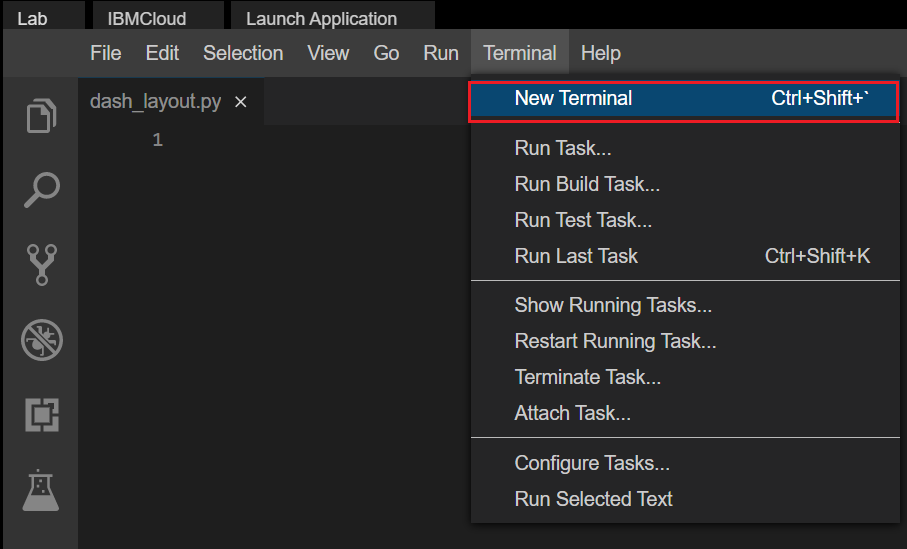
**Get the tool ready**

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



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

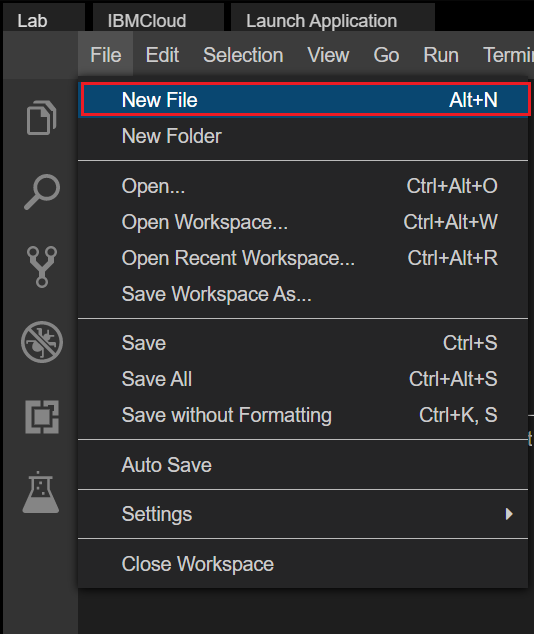
**pip3** install pandas dash

# TASK 1 - Dash Application layout

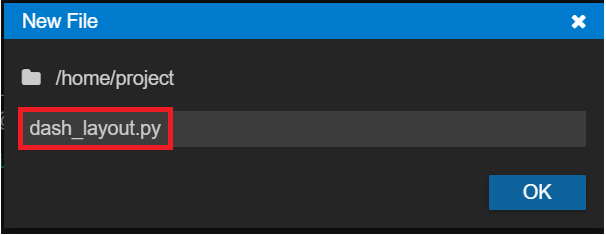
Let's start with

* Importing necessary libraries
* Title added using html.H1() tag

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



1. Provide the file name as dash\_layout.py



1. Copy the below code to the dash\_layout.py script and review the code.

# Import required packages

**import** pandas **as** pd

**import** dash

**import** dash\_core\_components **as** dcc

**import** dash\_html\_components **as** html

**import** plotly.express **as** px

**from** dash.dependencies **import** Input, Output

# Add Dataframe

# Add a bar graph figure

app = dash.Dash()

app.layout = html.Div(children=[

html.H1(

children='Dashboard',

style={

'textAlign': 'center'

}

)

# Create dropdown

# Bar graph

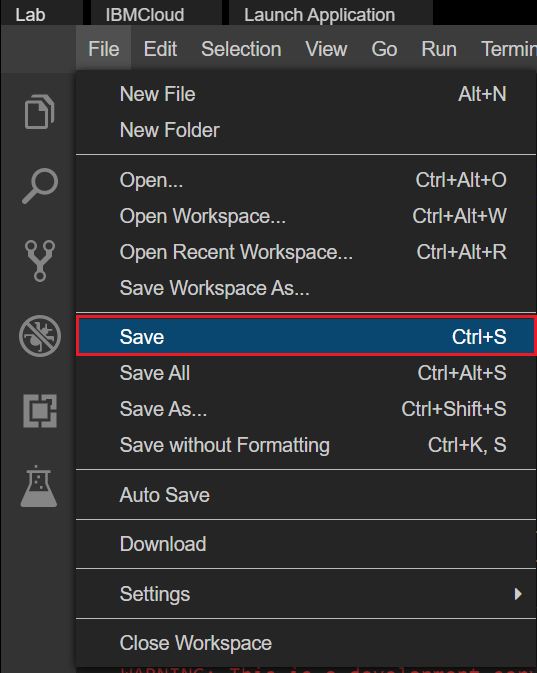
])

# Run Application

**if** \_\_name\_\_ == '\_\_main\_\_':

app.run\_server()

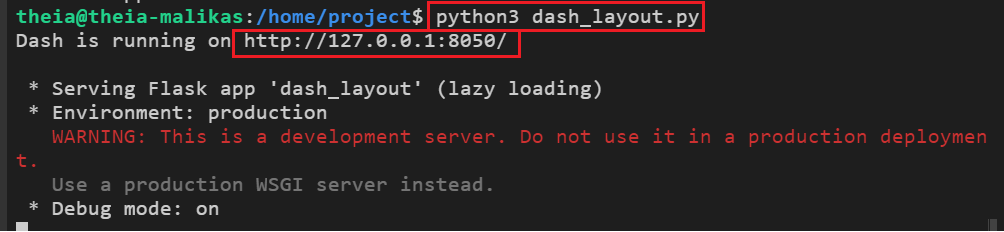
1. Save the application using Save option from File menu.



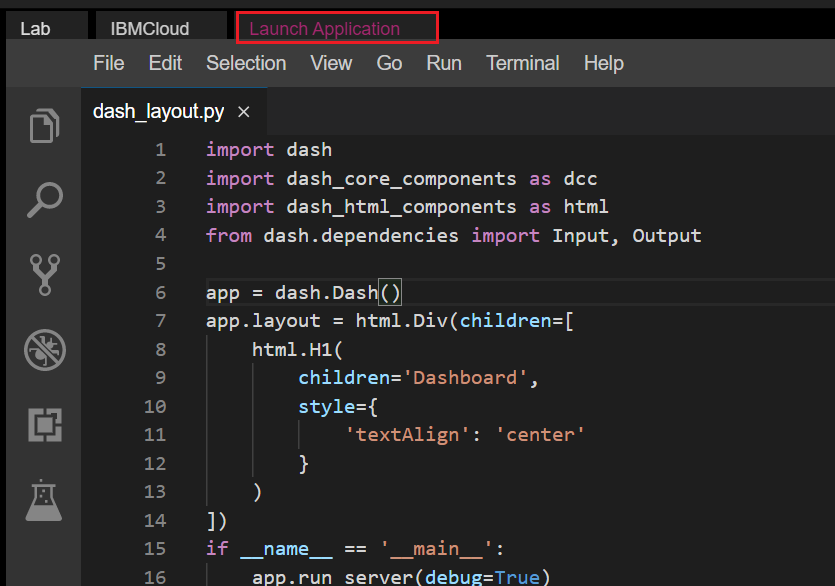
1. Run the python file using the following command in the terminal

**python3** **dash\_layout**.py

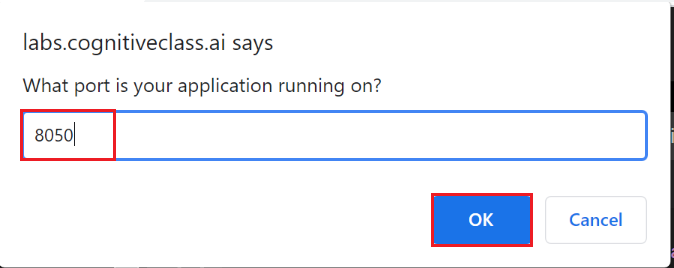
1. Observe the port number shown in the terminal.



1. Click on the Launch Application option from the menu bar.



1. Provide the port number and click OK



Note: If you are not able to see the application after launching just check the pop up window for your browser is enabled.

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



## Add dropdown

1. You can generate a drop down as shown below. You do by calling Dropdown off dash\_core\_components and passing the options as a list of dictionaries. You can set the default value using the value attribute and passing in the default option.

Note:

* Add a comma (,) before the placeholder in the skeleton file and then place the code.
* The placeholder here is "# Create dropdown " in the skeleton file.

# Create dropdown

dcc.Dropdown(options=[

{'label': 'New York City', 'value': 'NYC'},

{'label': u'Montréal', 'value': 'MTL'},

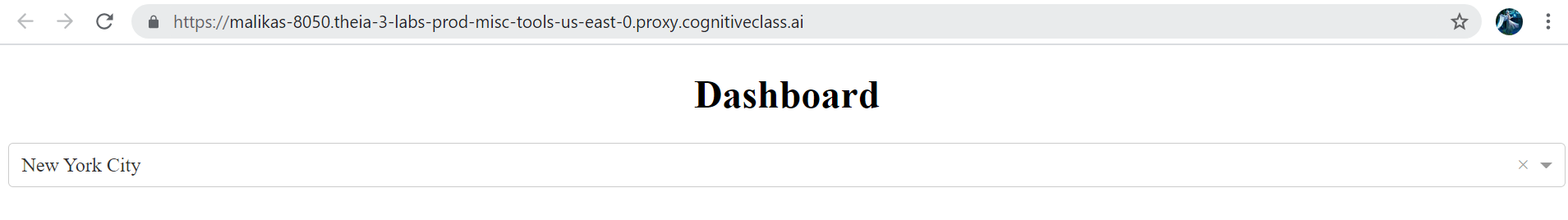
{'label': 'San Francisco', 'value': 'SF'}

],

value='NYC' # Providing a vallue to dropdown

)

1. After adding the dropdown the dashboard is displayed as below.



# Adding a dataframe

Assume you have a dataframe as:

Note: Place the code under the placeholder # Add Dataframe in the skeleton file copied before.

# Add Dataframe

df = pd.DataFrame({

"Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],

"Amount": [4, 1, 2, 2, 4, 5],

"City": ["SF", "SF", "SF", "NYC", "MTL", "NYC"]

})

# Task 2: Create Bar graph

The plotly.express module (usually imported as px) contains functions that can create entire figures at once, and is referred to as Plotly Express or PX. Plotly Express is a built-in part of the plotly library, and is the recommended starting point for creating most common figures

In order to create a graph on our layout, we use the Graph class from dashcorecomponents.

Note: Place the code under the placeholder # Add a bar graph figure in the skeleton file copied before.

# Add a bar graph figure

**fig** = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

Note: Place the code under the placeholder # Bar graph figure in the skeleton file copied before and also add a comma **,** before the placeholder.

# Bar graph

**dcc**.Graph(id='example-graph-2',figure=fig)

The dashboard with the dropdown and the bar graph is displayed as below.



Note: Here we are just creating the dropdown and bar chart without any functionality. Let's start with the real dataset to get the dropdown functionality with the graph. When you finish running the application press thekey Ctrl+C near the terminal window

to stop the running application and begin with the new application.

# Import required packages

import pandas as pd

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.express as px

from dash.dependencies import Input, Output

# Add Dataframe

df = pd.DataFrame({

"Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],

"Amount": [4, 1, 2, 2, 4, 5],

"City": ["SF", "SF", "SF", "NYC", "MTL", "NYC"]

})

# Add a bar graph figure

fig = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

app = dash.Dash()

# Create Layout

app.layout = html.Div(children=[

html.H1(

children='Dashboard',

style={

'textAlign': 'center'

}

),

# Create dropdown

dcc.Dropdown(options=[

{'label': 'New York City', 'value': 'NYC'},

{'label': u'MontrÃ©al', 'value': 'MTL'},

{'label': 'San Francisco', 'value': 'SF'}

],

value='NYC' # Providing a vallue to dropdown

),

# Bar graph

dcc.Graph(id='example-graph-2',figure=fig)

])

# Run Application

if \_\_name\_\_ == '\_\_main\_\_':

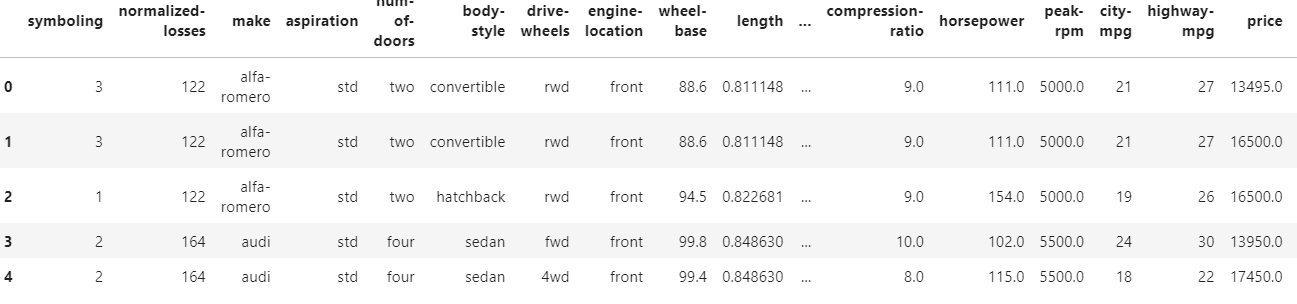
app.run\_server()

Sample

# Story:

Here we are looking into an **automobile dataset** which has various attributes like **drive-wheels,body-style and price**.

Lets view the snapshot of our selected dataset.



Here let's say we are selecting 3 important features **drive-wheels, body-style and Price**.

* The possible values of drive-wheels are **4 wheel Drive(4wd),Front WheelDrive(fwd) and Rear wheel Drive(rwd)**.
* The different body styles of the cars are **hardtop,sedan,convertible** and so on.
* There are 2 types of people here:
  + A customer who wants to purchase the cars with less price , different body styles and wants to look for the drive wheel with this arrangement.
  + A dealer who wants to showcase the prices for the cars with different body styles and drive wheels.
* As a data analyst, you have been given a task to visually show the **body-style and prices** with respect to each **drive wheel** selected.
* So ideally you want to showcase this in the form of 2 interactive charts such as **pie chart** and **bar chart** on selection of drive wheel.

Below is the key item,

* Drive wheels

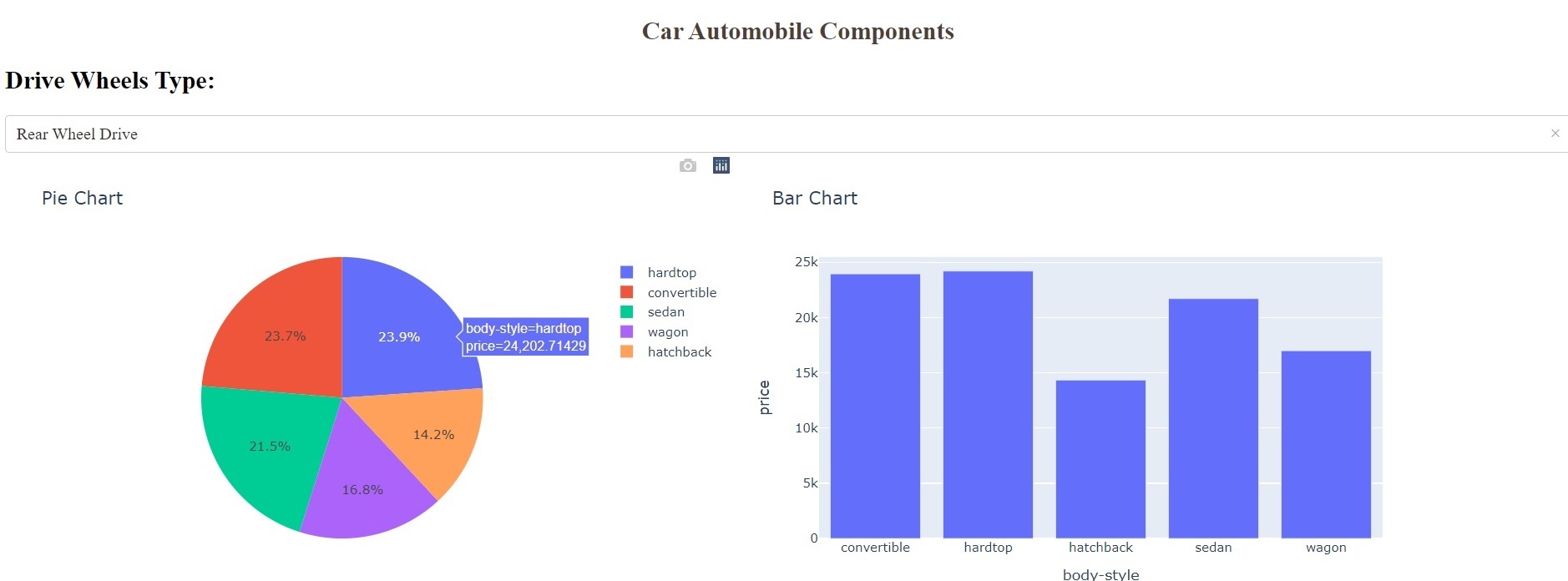
## Components of the item

1. Drive Wheel Type

For the chosen Drive wheel,

* + Pie Chart showing body style and price.
  + Bar Chart showing body style and price.

## Expected Layout



## Requirements to create the expected result

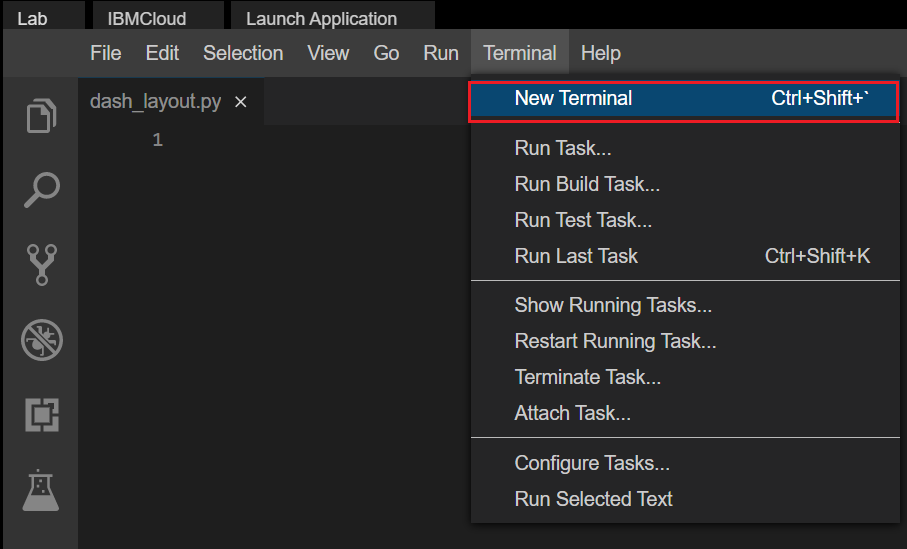
* A dropdown [**menu**](https://dash.plotly.com/dash-core-components/dropdown?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDV0101ENSkillsNetwork20297740-2021-01-01): For choosing Drive wheel type
* 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 dropdown(which is 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:

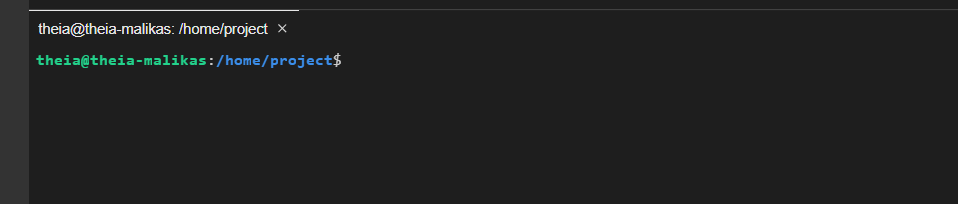
1. Import required libraries and read the dataset
2. Create an application layout
3. Add title to the dashboard using HTML H1 component
4. Add a dropdown using dcc.dropdown
5. Add the pie chart and bar chart core graph components.
6. Run the app

**Get the tool ready**

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



* Now, you have a terminal ready to start the lab.

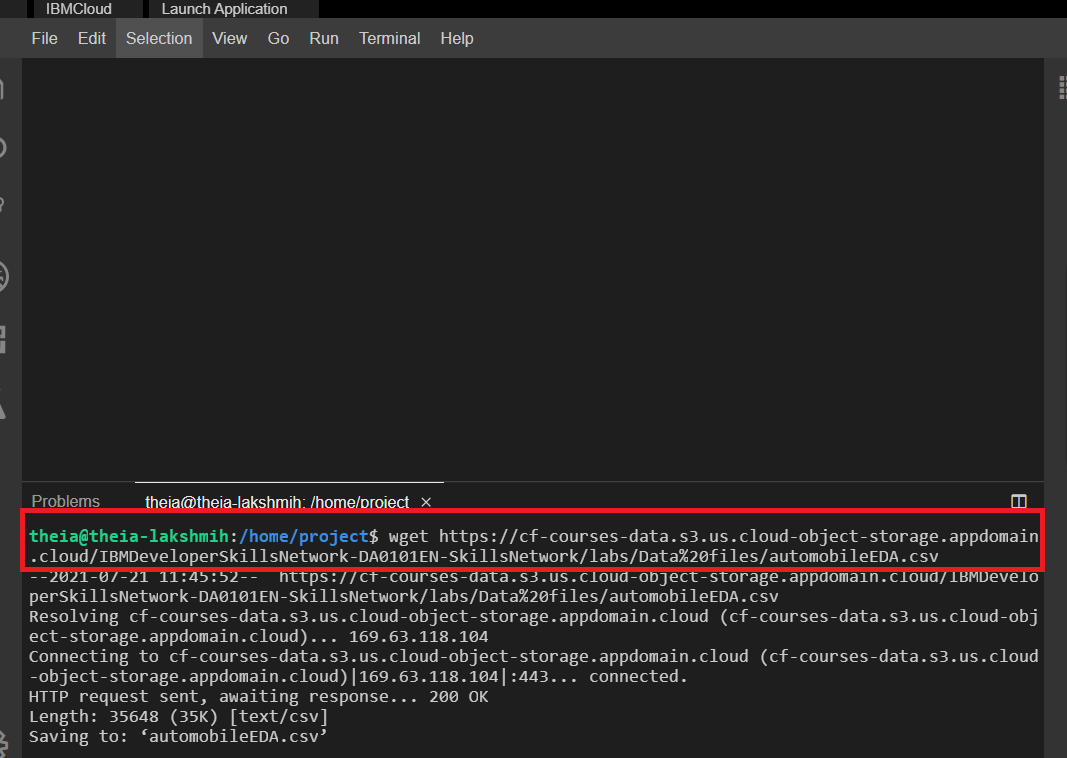


**Get the application skeleton**

* Copy and paste the command in the terminal to download the csv.

**wget** https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/automobileEDA.csv

The csv gets downloaded.



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

# Let's create the application

* Create a new file called Dash\_Auto.py
* Copy the code mentioned in the skeleton file and save it.

## Structure of the skeleton file

**import** pandas **as** pd

**import** dash

**import** dash\_html\_components **as** html

**import** dash\_core\_components **as** dcc

**from** dash.dependencies **import** Input, Output, State

**import** plotly.graph\_objects **as** go

**import** plotly.express **as** px

**from** dash **import** no\_update

app = dash.Dash(\_\_name\_\_)

# REVIEW1: Clear the layout and do not display exception till callback gets executed

app.config.suppress\_callback\_exceptions = True

# Read the automobiles data into pandas dataframe

auto\_data = pd.read\_csv('automobileEDA.csv',

encoding = "ISO-8859-1",

)

#Layout Section of Dash

app.layout = html.Div(children=[#TASK 3A

#outer division starts

html.Div([

# First inner divsion for adding dropdown helper text for Selected Drive wheels

html.Div(

#TASK 3B

),

#TASK 3C

#Second Inner division for adding 2 inner divisions for 2 output graphs

html.Div([

#TASK 3D

], style={'display': 'flex'}),

])

#outer division ends

])

#layout ends

#Place to add @app.callback Decorator

#TASK 3E

#Place to define the callback function .

#TASK 3F

**if** \_\_name\_\_ == '\_\_main\_\_':

app.run\_server()

# Hints to complete TASKS

Search/Look for TASK word in the script to identify places where you need to complete the code.

**TASK 3A: Add title to the dashboard**

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

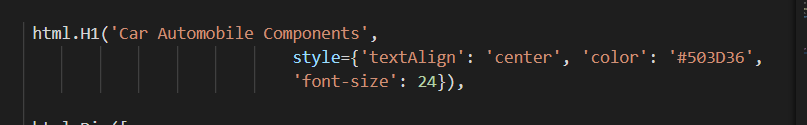
* Application title is Car Automobile Components
* Use style parameter provided below to make the title center aligned, with color code #503D36, and font-size as 24

html.H1('Car Automobile Components',

style={'textAlign': 'center', 'color': '#503D36',

'font-size': 24}),

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

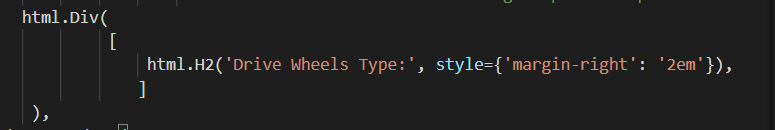


# TASK 3B: Add a Label to the dropdown

* Use the html.H2() tag to hold the label for the dropdown inside the first inner division
  + Label is Drive Wheels Type:
  + Use style parameter provided below to allign the label margin-right with value 2em which means 2 times the size of the current font.

html.H2('Drive Wheels Type:', style={'margin-right': '2em'}),

After updating the label the app.layout will now look like this



# TASK 3C: Next lets add the dropdown right below the first inner division.

* The dropdown has an id as demo-dropdown.
* These options have the labels as Rear Wheel Drive ,Front Wheel Drive and Four Wheel Drive
* The values allowed in the dropdown are rwd,fwd,4wd
* The default value when the dropdown is displayed is rwd.

dcc.Dropdown(

id='demo-dropdown',

options=[

{'label': 'Rear Wheel Drive', 'value': 'rwd'},

{'label': 'Front Wheel Drive', 'value': 'fwd'},

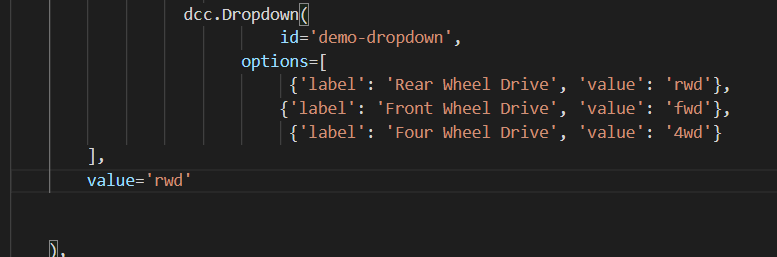
{'label': 'Four Wheel Drive', 'value': '4wd'}

],

**value**='rwd'

),

Once you add the dropdown the 'app.layout will appear as follows



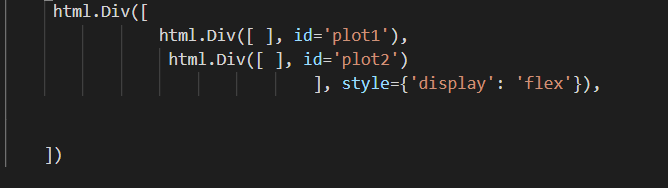
# TASK 3D: Add two empty divisions for output inside the next inner division .

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

**html**.Div([ ], id='plot1'),

**html**.Div([ ], id='plot2')

Once you add the divisions the 'app.layout will appear as follows



# TASK 3E: 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, our input is the value property of the component that has the ID demo-dropdown
* 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.

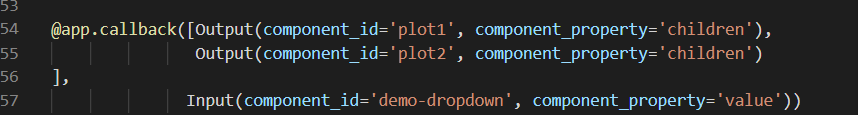
Component ids will be plot1 , plot2.

@app.callback([Output(component\_id='plot1', component\_property='children'),

Output(component\_id='plot2', component\_property='children')],

Input(component\_id='demo-dropdown', component\_property='value'))

Once you add the callback decorator the 'app.layout will appear as follows



# TASK 3F: 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 display\_selected\_drive\_charts() which will be wrapped by our decorator.
* The function first filters our dataframe auto\_data by the selected value of the drive-wheels from the dropdown as follows
* auto\_data[auto\_data['drive-wheels']==value] .
* Next we will group by the drive-wheels and body-style and calculate the mean price of the dataframe.
* Use the px.pie() and px.bar() function we will plot the pie chart and bar chart

**def** **display\_selected\_drive\_charts**(value):

filtered\_df = auto\_data[auto\_data['drive-wheels']==value].groupby(['drive-wheels','body-style'],as\_index=False). \

mean()

filtered\_df = filtered\_df

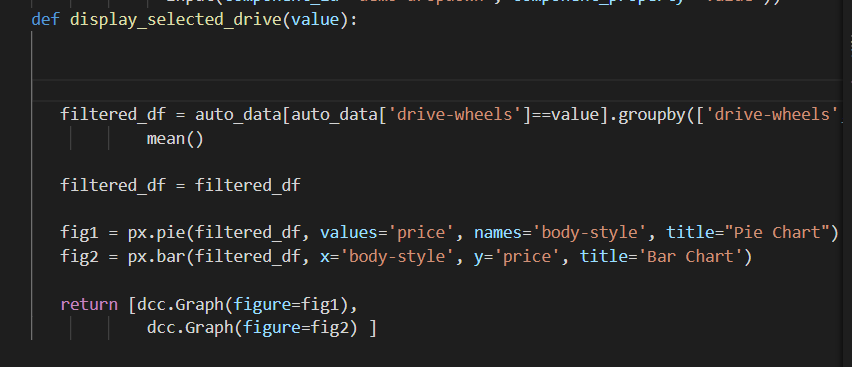
fig1 = px.pie(filtered\_df, values='price', names='body-style', title="Pie Chart")

fig2 = px.bar(filtered\_df, x='body-style', y='price', title='Bar Chart')

**return** [dcc.Graph(figure=fig1),

dcc.Graph(figure=fig2) ]

* Here for the pie chart we pass the filtered dataframe where values correspond to price and names will be body-style
* For the bar chart also we will pass the filtered dataframe where x-axis corresponds to body-style and y-axis as price.
* Finally we return the 2 figure objects fig1 and fig2 in dcc.Graph method and finally the plots are displayed as follows

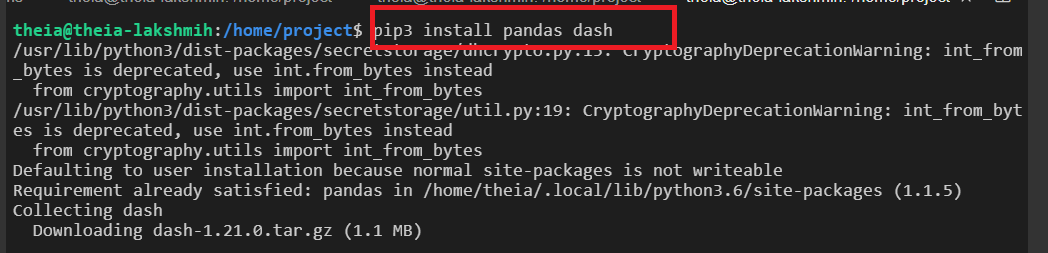


* Once you have finished coding save your code.

# Run the Application

* Firstly, install pandas and dash using the following command

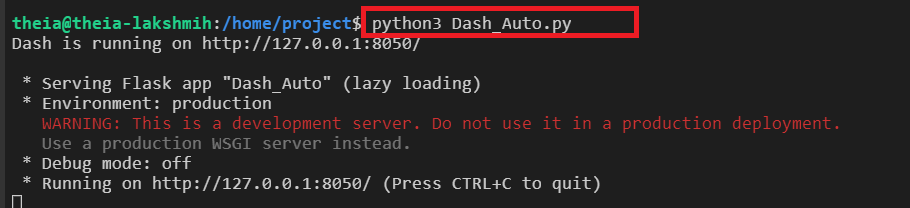
**pip3** install pandas dash



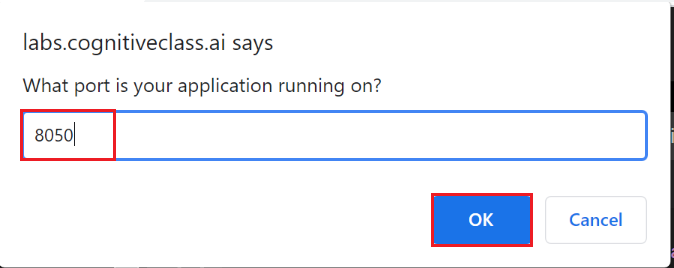
* Next Run the python file using the command

**python3** **Dash\_Auto**.py

* 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



* The graphs appear on selection of drive wheels.
* import pandas as pd
* import dash
* import dash\_html\_components as html
* import dash\_core\_components as dcc
* from dash.dependencies import Input, Output, State
* import plotly.graph\_objects as go
* import plotly.express as px
* from dash import no\_update
* app = dash.Dash(\_\_name\_\_)
* # REVIEW1: Clear the layout and do not display exception till callback gets executed
* app.config.suppress\_callback\_exceptions = True
* # Read the automobiles data into pandas dataframe
* auto\_data = pd.read\_csv('automobileEDA.csv',
* encoding = "ISO-8859-1",
* )
* #Layout Section of Dash
* app.layout = html.Div(children=[html.H1('Car Automobile Components',
* style={'textAlign': 'center', 'color': '#503D36',
* 'font-size': 24}),
* #outer division starts
* html.Div([
* # First inner divsion for adding dropdown helper text for Selected Drive wheels
* html.Div(
* html.H2('Drive Wheels Type:', style={'margin-right': '2em'}),
* ),
* #Second Inner division for adding 2 inner divisions for 2 output graphs
* dcc.Dropdown(
* id='demo-dropdown',
* options=[
* {'label': 'Rear Wheel Drive', 'value': 'rwd'},
* {'label': 'Front Wheel Drive', 'value': 'fwd'},
* {'label': 'Four Wheel Drive', 'value': '4wd'}
* ],
* value='rwd'
* ),
* #Second Inner division for adding 2 inner divisions for 2 output graphs
* html.Div([
* html.Div([ ], id='plot1'),
* html.Div([ ], id='plot2')
* ], style={'display': 'flex'}),
* ])
* #outer division ends
* ])
* #layout ends
* #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='demo-dropdown', component\_property='value'))
* #Place to define the callback function .
* def display\_selected\_drive\_charts(value):

* filtered\_df = auto\_data[auto\_data['drive-wheels']==value].groupby(['drive-wheels','body-style'],as\_index=False). \
* mean()
* filtered\_df = filtered\_df
* fig1 = px.pie(filtered\_df, values='price', names='body-style', title="Pie Chart")
* fig2 = px.bar(filtered\_df, x='body-style', y='price', title='Bar Chart')
* return [dcc.Graph(figure=fig1),
* dcc.Graph(figure=fig2) ]

* if \_\_name\_\_ == '\_\_main\_\_':
* app.run\_server()