Dash Components

Objectives

After completing the lab you will be able to:

- Work with Dash Callbacks

Estimated time needed: 30 minutes

Dataset Used

Airline Reporting Carrier On-Time Performance dataset from Data Asset eXchange

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

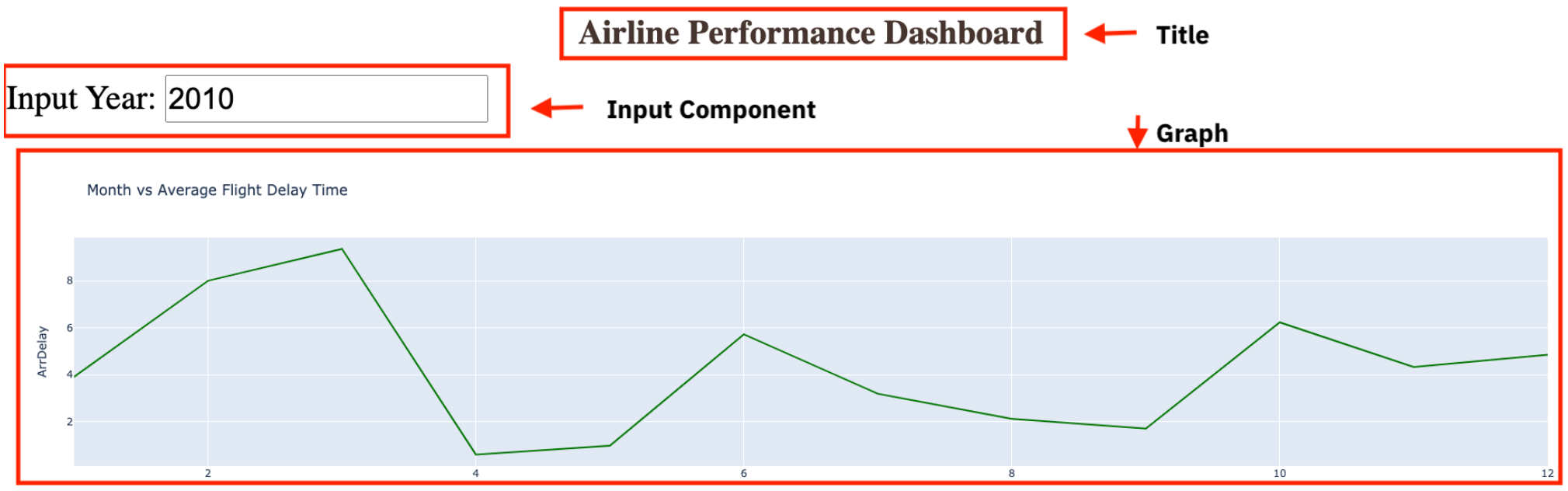
Theme

Extract average monthly arrival delay time and see how it changes over the year. Year range is from 2010 to 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
- Chart conveying the average monthly arrival delay



To do:

1. Import required libraries and read the dataset
2. Create an application layout
3. Add title to the dashboard application using HTML H1 component
4. Add an input text box using core input component
5. Add the line chart using core graph component
6. Run the app

Get the tool ready

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

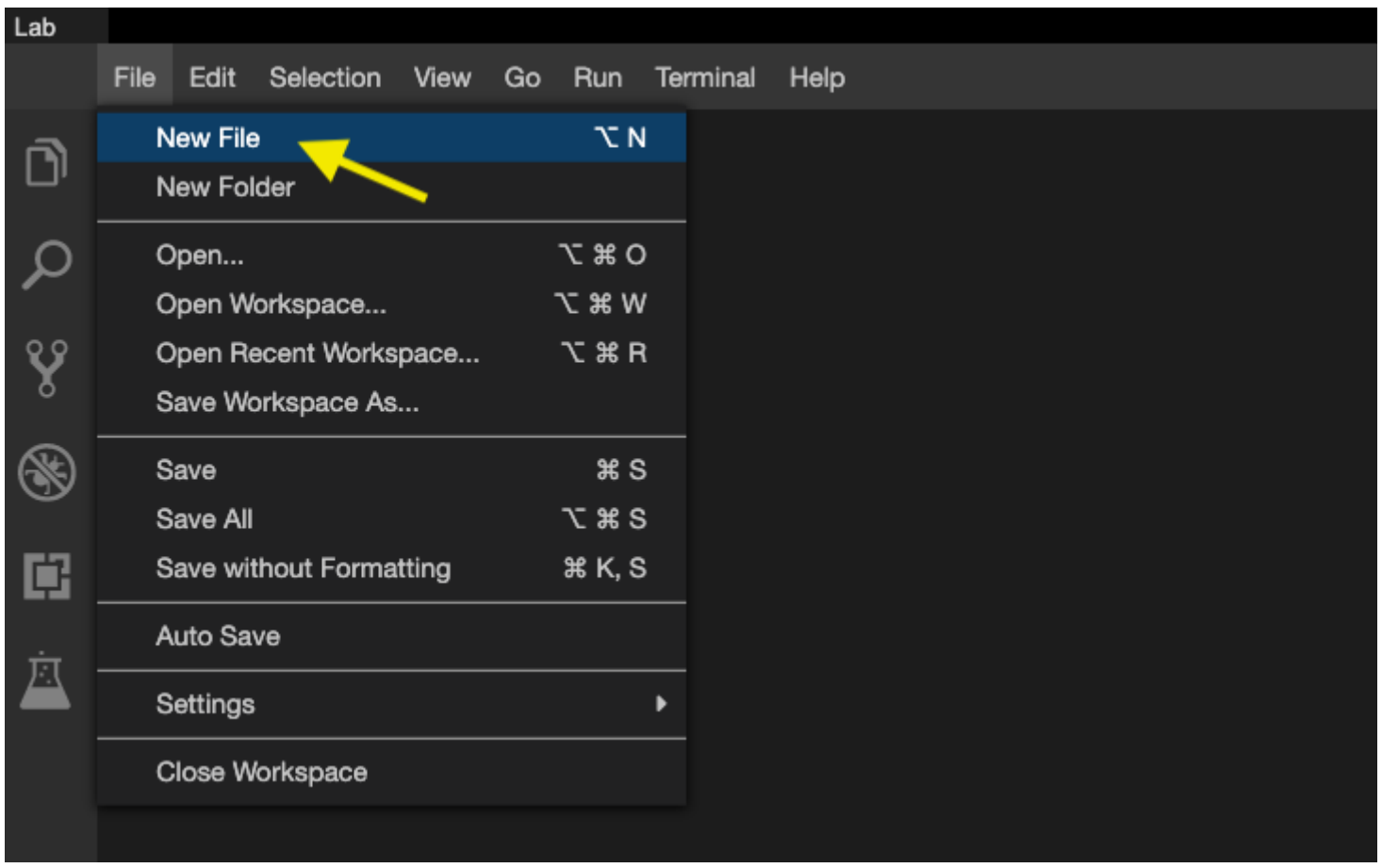
```
python3 -m pip install pandas dash
```

```
theia@theiadocker-malikas: /home/project ×

theia@theiadocker-malikas:/home/project$ python3 -m pip install pandas dash
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/c3/e2/00cacecafbab071c787019f00ad84ca3185952f6bb9bca9550ed83870d4d/pandas-1.1.5-cp36-cp36m-manylinux1_x8
hl (9.5MB)
    100% |████████████████████████████████| 9.5MB 163kB/s
Collecting dash
  Downloading https://files.pythonhosted.org/packages/cc/42/e1692b2d34e4135569db680efe3438e809a6b3f0ae607ad41aeff7741672/dash-2.6.1-py3-none-any.whl (9.9MB)
    100% |████████████████████████████████| 9.9MB 159kB/s
Collecting pytz>=2017.2 (from pandas)
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/d5/50/54451e88e3da4616286029a3a17fc377de817f66a0f50e1faaee90161724/pytz-2022.2.1-py2.py3-none-any.whl (5
    100% |████████████████████████████████| 501kB 3.2MB/s
Collecting python-dateutil>=2.7.3 (from pandas)
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/36/7a/87837f39d0296e723bb9b62bbb257d0355c7f6128853c78955f57342a56d/python_dateutil-2.8.2-py2.py3-none-an
(247kB)
    100% |████████████████████████████████| 256kB 5.8MB/s
Collecting numpy>=1.15.4 (from pandas)
  Downloading https://files.pythonhosted.org/packages/45/b2/6c7545bb7a38754d63048c7696804a0d947328125d81bf12beaa692c3ae3/numpy-1.19.5-cp36-cp36m-manylinux1_x8
hl (13.4MB)
    100% |████████████████████████████████| 13.4MB 111kB/s
Collecting contextvars==2.4; python_version < "3.7" (from dash)
  Downloading https://files.pythonhosted.org/packages/83/96/55b82d9f13763be9d672622e1b8106c85acb83edd7cc2fa5bc67cd9877e9/contextvars-2.4.tar.gz
Collecting dash-table==5.0.0 (from dash)
  Downloading https://files.pythonhosted.org/packages/da/ce/43f77dc8e7bbad02a9f88d07bf794eaf68359df756a28bb9f2f78e255bb1/dash_table-5.0.0-py3-none-any.whl
```

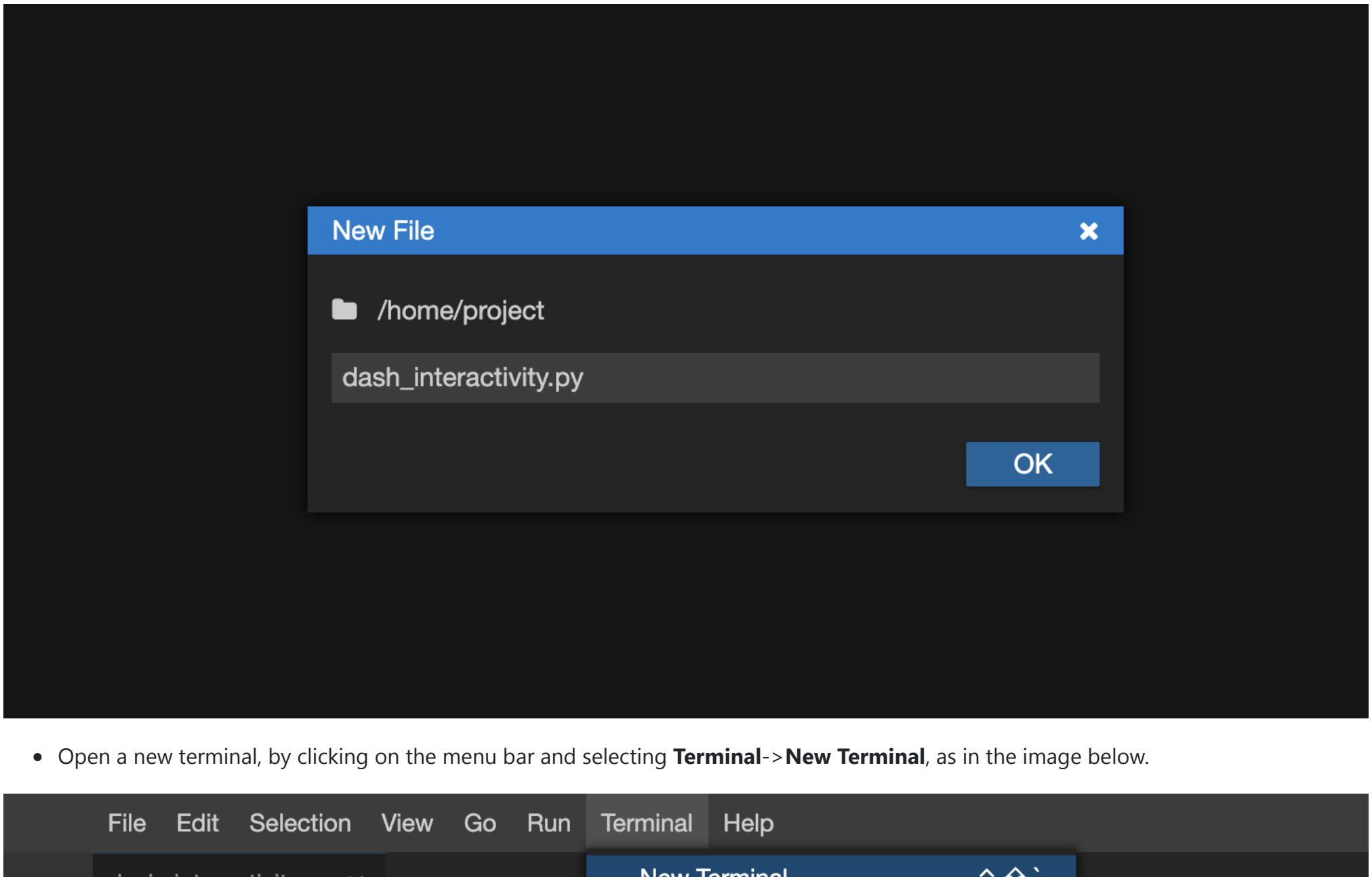
```
pip3 install httpx==0.20 dash plotly
```

```
theia@theiadocker-malikas: /home/project ×

theia@theiadocker-malikas:/home/project$ pip3 install httpx==0.20 dash plotly
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int_from
_bytes is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int_from_byt
es is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
Defaulting to user installation because normal site-packages is not writeable
Collecting httpx==0.20
  Downloading httpx-0.20.0-py3-none-any.whl (82 kB)
     |████████████████████████████████| 82 kB 779 kB/s
Collecting dash
  Downloading dash-2.6.1-py3-none-any.whl (9.9 MB)
     |████████████████████████████████| 9.9 MB 40.7 MB/s
Collecting plotly
  Downloading plotly-5.10.0-py2.py3-none-any.whl (15.2 MB)
     |████████████████████████████████| 15.2 MB 39.3 MB/s
Requirement already satisfied: sniffio in /home/theia/.local/lib/python3.6/site-packages (from httpx=
=0.20) (1.2.0)
Requirement already satisfied: httpcore<0.14.0,>=0.13.3 in /home/theia/.local/lib/python3.6/site-pack
ages (from httpx==0.20) (0.13.7)
Requirement already satisfied: async-generator in /home/theia/.local/lib/python3.6/site-packages (fro
m httpx==0.20) (1.10)
Requirement already satisfied: certifi in /home/theia/.local/lib/python3.6/site-packages (from httpx=
=0.20) (2020.12.5)
Requirement already satisfied: rfc3986[idna2008]<2,>=1.3 in /home/theia/.local/lib/python3.6/site-pac
kages (from httpx==0.20) (1.5.0)
Requirement already satisfied: charset-normalizer in /home/theia/.local/lib/python3.6/site-packages (
from httpx==0.20) (2.0.12)
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
```

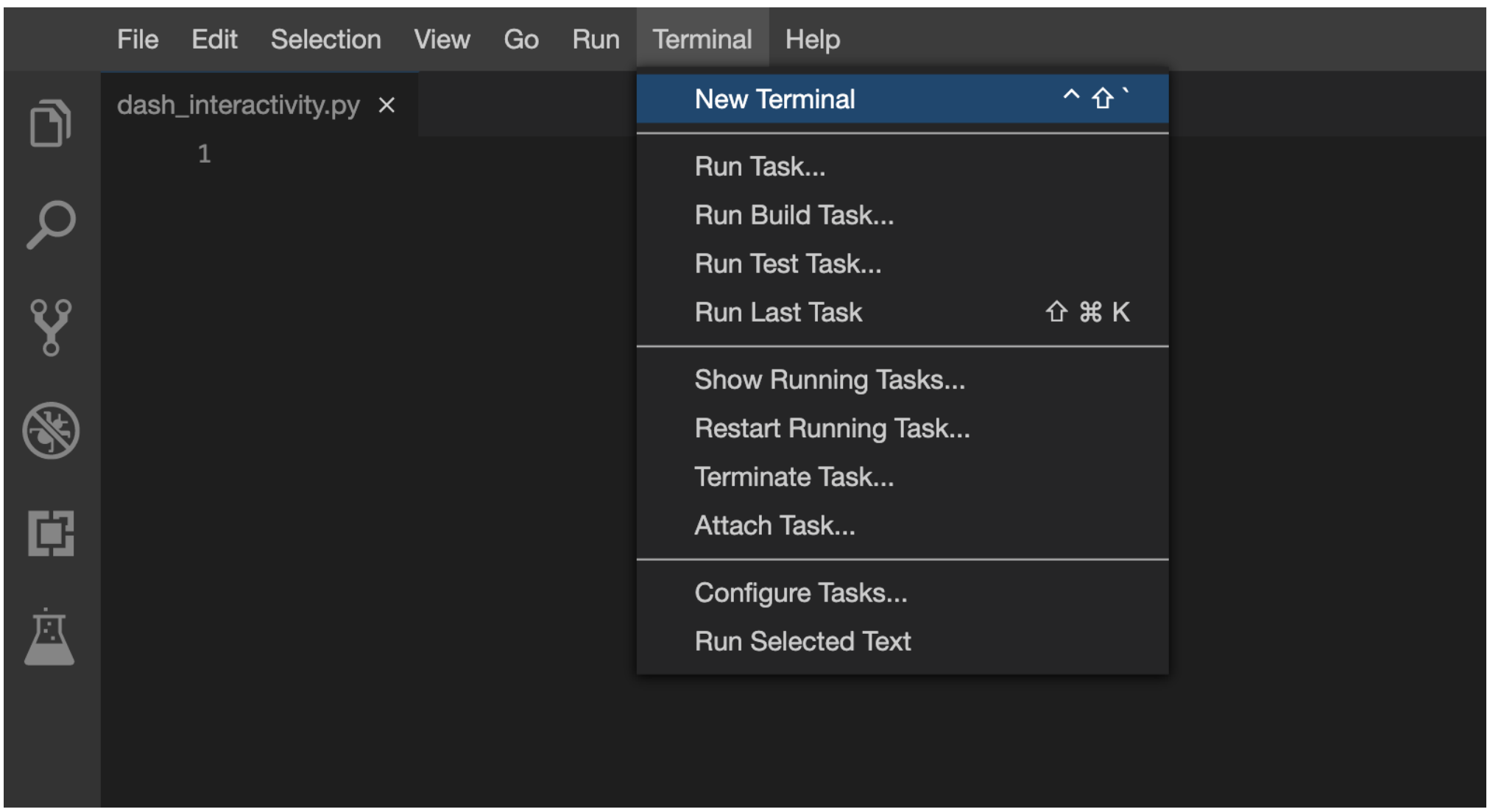
- Create a new python script, by clicking on the menu bar and selecting **File**->**New File**, as in the image below.



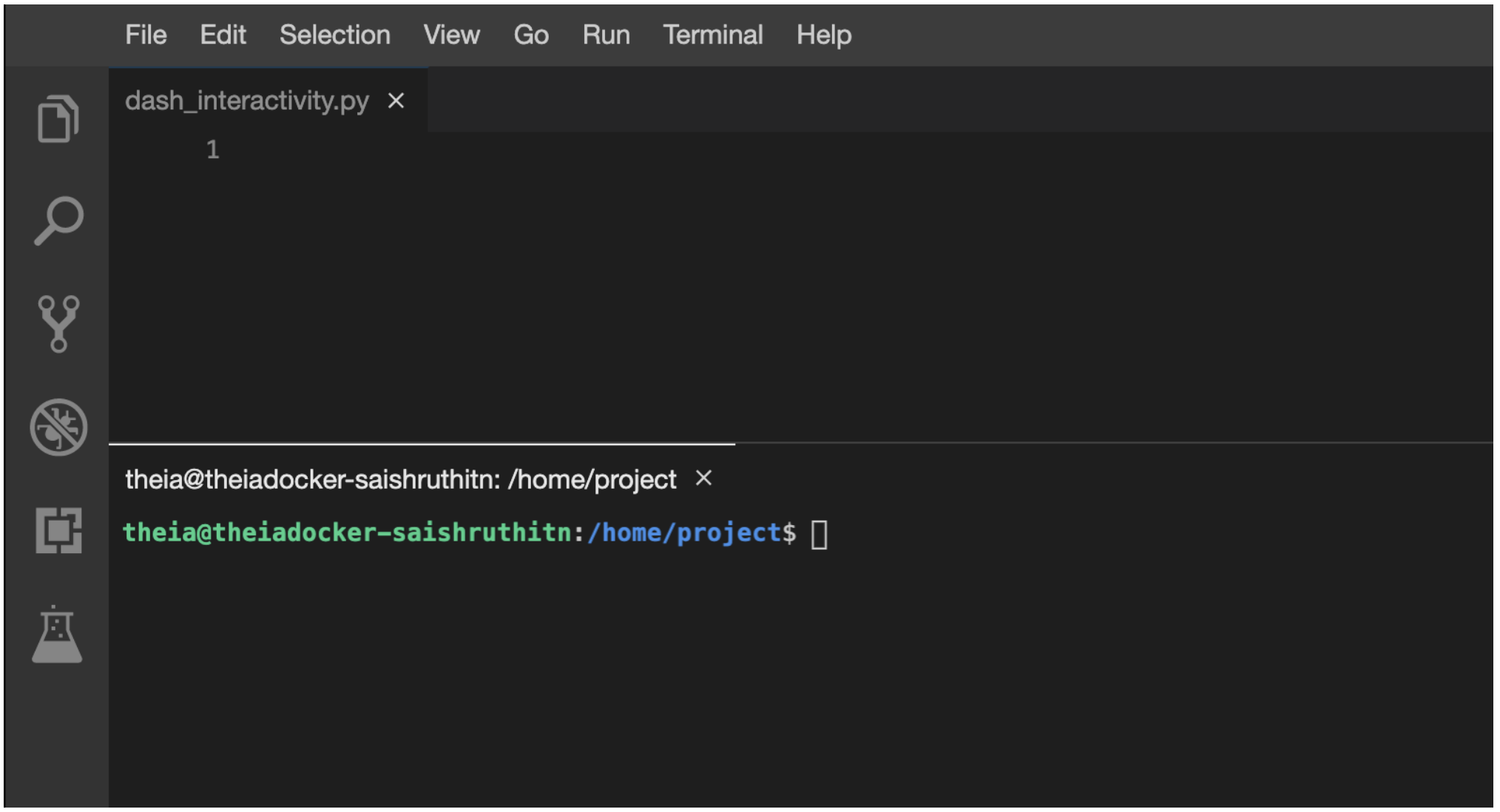
- Provide the file name as `dash_interactivity.py`



- 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

Let's start with

- Importing necessary libraries
- Reading the data

Copy the below code to the `dash_interactivity.py` script and review the code.

```
# Import required libraries
import pandas as pd
import plotly.graph_objects as go
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output

# Read the airline data into pandas dataframe
airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
                            encoding = "ISO-8859-1",
                            dtype={'Div1Airport': str, 'Div1TailNum': str,
                                   '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
- Chart conveying the average monthly arrival delay inside a layout division

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.
- Layout division added using `html.Div()` and chart added using `dcc.Graph()` tag inside the layout division.

Copy the below code to the `dash_interactivity.py` script and review the structure.

NOTE: Copy below the current code

```
# Create a dash application
app = dash.Dash(__name__)

# Get the layout of the application and adjust it.
# Create an outer division using html.Div and add title to the dashboard using html.H1 component
# Add a html.Div and core input text component
# Finally, add graph component.
app.layout = html.Div(children=[html.H1(),                                    html.Div(["Input Year", dcc.Input(),],
                                style={}),
                                html.Br(),
                                html.Br(),
                                html.Div(),
                                ])
```

TASK 3 - Update layout components

Application title

- Heading reference: Plotly H1 HTML Component
- Title as `Airline Performance Dashboard`
- Use `style` parameter and make the title `center` aligned, with color code `#503D36`, and font-size as `40`. Check `More about HTML` section here.

Input component

- Update dcc.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 `50px` and font-size to be `35`.
- Use `style` parameter and assign font-size as `40` for the whole division.

Output component

- Add `dcc.Graph()` component to the second division.
- Update dcc.Graph component `id` as `line-plot`.

TASK 4 - 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 `dash_interactivity.py` script and review the structure.

NOTE: Copy below the current code

```
# add callback decorator
@app.callback(Output(),
               Input())

# Add computation to callback function and return graph
def get_graph(entered_year):
    # Select data based on the entered year
    df =  airline_data[airline_data['Year']==int(entered_year)]

    # Group the data by Month and compute average over arrival delay time.
    line_data = df.groupby('Month')['ArrDelay'].mean().reset_index()

    #
    fig = go.Figure(data=)
    fig.update_layout()
    return fig

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

TASK 5 - Update the callback function

Callback decorator

- Refer examples provided here
- Update output component id parameter with the id provided in the `dcc.Graph()` component and component property as `figure`.
- Update input component id parameter with the id provided in the `dcc.Input()` component and component property as `value`.

Callback function

- Update `data` parameter of the `go.Figure()` with the scatter plot. Refer here. Sample syntax below:

```
go.Scatter(x='----', y='----', mode='-----', marker='----)
```

- Update `x` as `line_data['Month']`, `y` as `line_data['ArrDelay']`, `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 update layout function here.

Refer to the full python code of `dash_interactivity.py` below:

```
# Import required libraries
import pandas as pd
import plotly.graph_objects as go
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output

# Read the airline data into pandas dataframe
airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
                            encoding = "ISO-8859-1",
                            dtype={'Div1Airport': str, 'Div1TailNum': str,
                                   'Div2Airport': str, 'Div2TailNum': str})
# Create a dash application
app = dash.Dash(__name__)

app.layout = html.Div(children=[ html.H1('Airline Performance Dashboard',style={'textAlign': 'center', 'color':
'#503D36','font-size': 40}),
                                html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',
                                type='number', style={'height':'50px', 'font-size': 35}),],
                                style={'font-size': 40}),
                                html.Br(),
                                html.Br(),
                                html.Div(dcc.Graph(id='line-plot')),
                                ])

# add callback decorator
@app.callback( Output(component_id='line-plot', component_property='figure'),
               Input(component_id='input-year', component_property='value'))

# Add computation to callback function and return graph
def get_graph(entered_year):
    # Select 2019 data
    df =  airline_data[airline_data['Year']==int(entered_year)]

    # Group the data by Month and compute average over arrival delay time.
    line_data = df.groupby('Month')['ArrDelay'].mean().reset_index()

    fig = go.Figure(data=go.Scatter(x=line_data['Month'], y=line_data['ArrDelay'], mode='lines',
marker=dict(color='green')))
    fig.update_layout(title='Month vs Average Flight Delay Time', xaxis_title='Month', yaxis_title='ArrDelay')
    return fig

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

TASK 6 - Run the application

- Firstly, install pandas and dash using the following command in the terminal

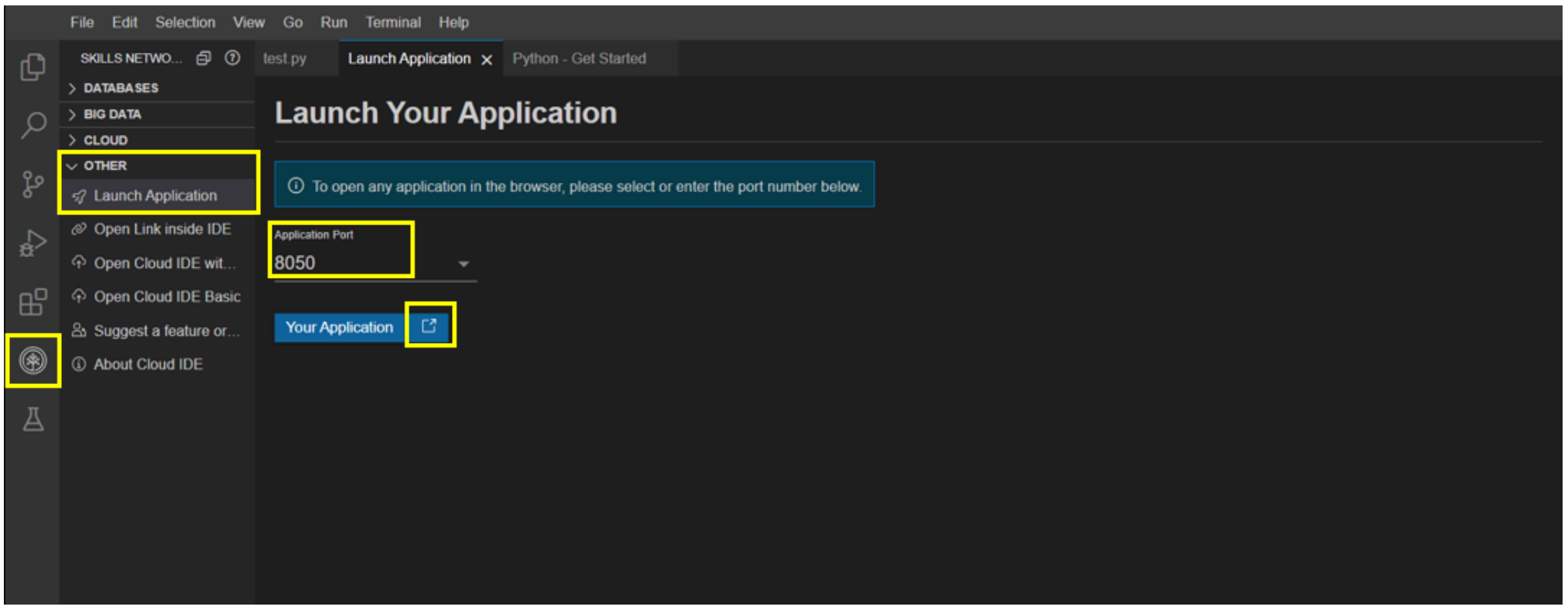
```
pip3 install pandas dash
```

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

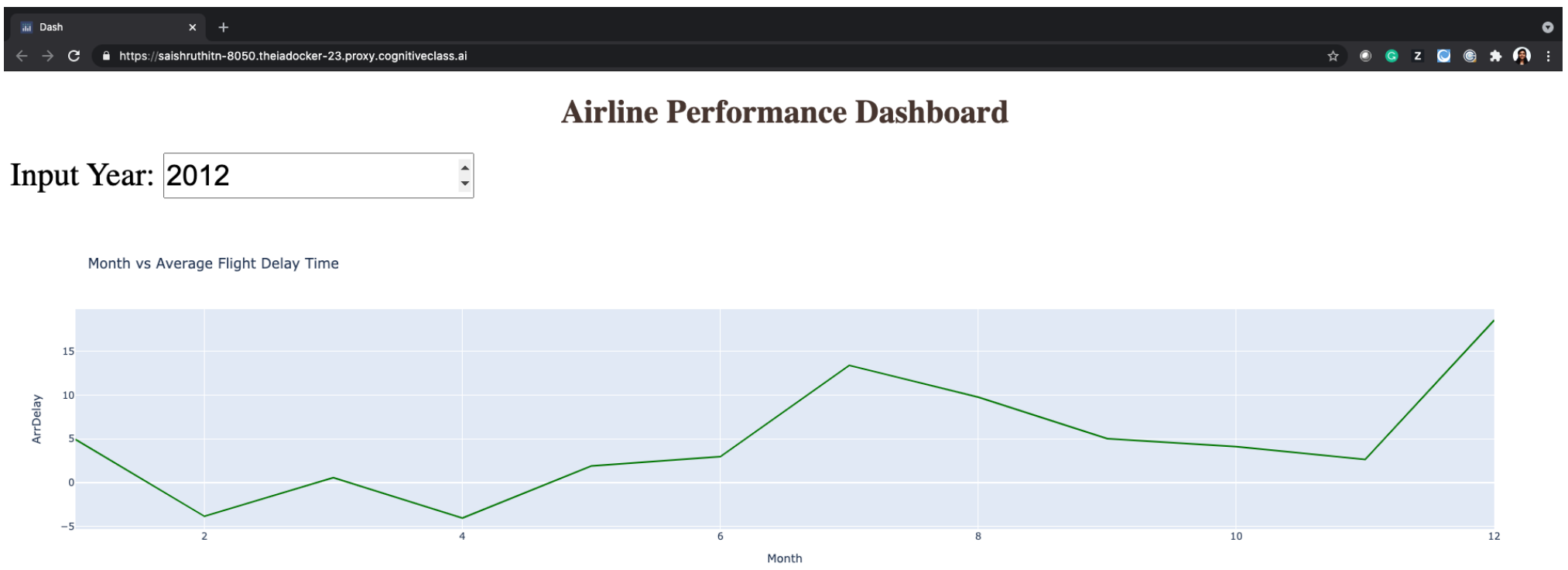
```
python3 dash_interactivity.py
```

- 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 "Airline Performance Dashboard" to "Airline Dash Interactivity" using HTML H1 component and font-size as 50.

▶ Answer

2. Update dcc.Input component `id` as `input-year`, default `value` as `2015`, and `type` as `number`. Use `style` parameter and assign height of the input box to be `40px` and font-size to be `40`. Use `style` parameter and assign font-size as `35` for the whole division.

▶ Answer

3. Save the above changes and relaunch the dashboard application to see the updated dashboard title.

▶ Answer

4. Write a command to stop the running app in the terminal

▶ Answer

Author

Saishruthi Swaminathan

Changelog

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