Create Dashboard using Plotly and Dash

Table of content

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.

Scenario:

The objective of this part of the Final Assignment is to analyze the historical trends in automobile sales during recession periods, as you did in the previous part. The goal is to provide insights into how the sales of XYZAutomotives, a company specializing in automotive sales, were affected during times of recession.

In this final assignment, you will have the opportunity to demonstrate the **Dashboarding** skills you have acquired in this course.

This lab aims to assess your abilities in creating various visualizations using *Plotly and Dash*. As a data scientist, you have been given a task to prepare a report on your finding from Automobile Sales data analysis.
You decided to develop a dashboard representing two main reports:-

- Yearly Automobile Sales Statistics
- Recession Period Statistics

NOTE: Year range is between 1980 and 2013.

Components of the report items

1. Yearly Automobile Sales Statistics

This report mainly consists of the following items:

- **Yearly Automobile Sales Using Line Chart for the Whole Period (line chart):**This line chart displays the average automobile sales for each year across the entire period.

- **Total Monthly Automobile Sales Using Line Chart (line chart):**This line chart displays the total monthly automobile sales for the selected year.

- **Average Vehicles Sold by Vehicle Type in the Selected Year (bar chart):**This bar chart shows the average number of vehicles sold for each vehicle type in the selected year.

- **Total Advertisement Expenditure for Each Vehicle Using Pie Chart (pie chart)**:
  This pie chart represents the total advertising expenditure for each vehicle type in the selected year.

2. Recession Period Statistics

- **Average Automobile Sales Fluctuation Over Recession Period (Year-wise):**This line chart shows the average automobile sales for each year during recession periods.

- **Average Number of Vehicles Sold by Vehicle Type:**This bar chart displays the average number of vehicles sold for each vehicle type during recession periods.

- **Total Expenditure Share by Vehicle Type During Recessions:**This pie chart represents the total advertising expenditure share by vehicle type during recession periods.

- **Effect of Unemployment Rate on Vehicle Type and Sales:**This bar chart shows the effect of unemployment rate on automobile sales by vehicle type during recession periods.

NOTE: You have worked creating a dashboard components in `Flight Delay Time Statistics Dashboard` section. You will be working on the similar lines for this Dashboard

Dataset Variables:

Dataset Variables for your reference

The dataset includes the following variables

- *Date*: The date of the observation.
- *Recession*: A binary variable indicating recession perion; 1 means it was recession, 0 means it was normal.
- *Automobile_Sales*: The number of vehicles sold during the period.
- *GDP*: The per capita GDP value in USD.

- *unemployment_rate*: The monthly unemployment rate.
- *Consumer_Confidence*: A synthetic index representing consumer confidence, which can impact consumer spending and automobile purchases.
- *Seasonality_Weight*: The weight representing the seasonality effect on automobile sales during the period.
- *Price*: The average vehicle price during the period.
- Advertising_Expenditure: The advertising expenditure of the company.
- *Vehicle_Type*: The type of vehicles sold; Supperminicar, Smallfamiliycar, Mediumfamilycar, Executivecar, Sports.
- *Competition*: The measure of competition in the market, such as the number of competitors or market share of major manufacturers.
- *Month*: Month of the observation extracted from Date.
- *Year*: Year of the observation extracted from Date.

Requirements to create the expected Dashboard

Requirements to create the expected Dashboard

You will be creating two dropdown menus:

- The first dropdown allows selection of the report type (Yearly Statistics or Recession Period Statistics).

- The second dropdown is for selecting the year. This should be enabled only when the user selects `Yearly Statistics report` and will be disabled if `Recession Period Statistics` is selected.

- Each dropdown will be placed within a separate division.
  You can refer to this [link](#) for understanding the concept of dropdowns in plotly dash.

- **App Layout**: This define the layout of the app, including titles, dropdown menus, and containers for displaying output.

  - **Callback functions for Interactivity**:

    - **Update Input Container:**

      Here we define a callback function to update the input container based on the selected statistics.

      The purpose of this function is to control the state (enabled or disabled) of the year dropdown based on the user's selection of the report type.

      Specifically:

      - Enabled (returns True) when `Yearly Statistics` is selected, allowing the user to choose a year.

      - Disabled (returns False) when `Recession Period Statistics` is selected, preventing the user from selecting a year because it is irrelevant in the context of recession statistics.

  - **Callback for Plotting:**

    Here we define a callback functions to update the output container based on selected statistics and year, creating various plots for the dashboard.

   The four plots have to be displayed in 2 rows, 2 column representation

NOTE:- For every task, you will required to save the screenshot/image, which you will be asked to submit for evaluation at the Final Submission stage.

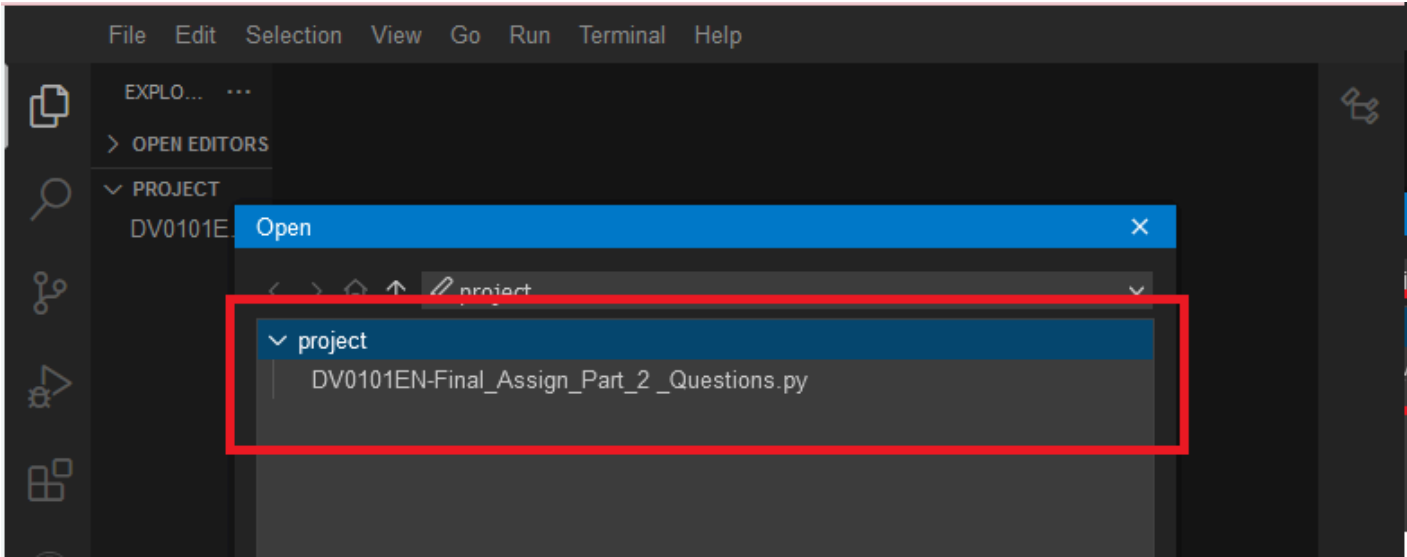# Get the application skeleton

- Copy and paste the below command in the terminal to download the skeleton.

  ```
  wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/InaFqKi-TlmTZwlzKvlNaQ/DV0101EN-Final-Assign-Part-2-Questions.py
  ```

- Click on File menu
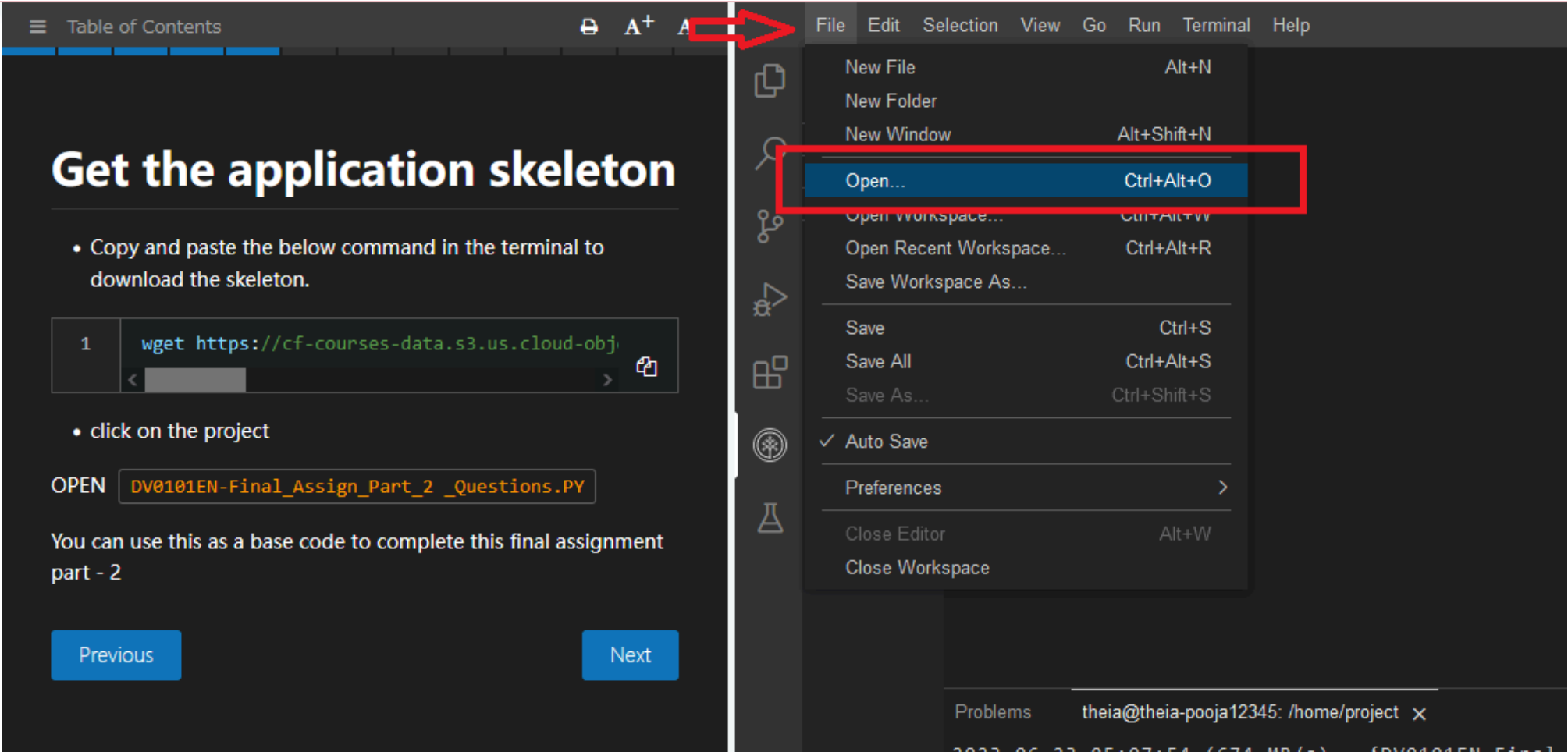
OPEN `DV0101EN-Final_Assign_Part_2 _Questions.PY`

You can use this as a base code to complete this final assignment part 2

Let's create the application

REVIEW

Search/Look for `REVIEW` word in the instructions to learn how commands are used and computations are carried out.

TASKS

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

TASK 2.1: Create a Dash application and give it a meaningful title

Note: Throughout this lab, you will see placeholder text like `'........'` or `groupby(...........)`. These dots should be replaced with appropriate values such as column names ('Year', 'Vehicle_Type', etc.), grouping keys, or chart titles.
Refer to the comments and hints above each task to determine what needs to be filled in.

```
html.H1("...............")
```

- Provide title of the dash application title as

```
Automobile Sales Statistics Dashboard
```

- Make the heading center aligned
- set color as `#503D36`
- font size as `24`

Sample: *style={'textAlign': 'left', 'color': '#000000', 'font-size': 0}*

REVIEW link for `html.H1` component

[Click here to view a sample example.](#)

NOTE: Once the application is up and running, take the screenshot representing the <u>title of the application</u> and save it as **'Title.png'**

TASK 2.2: Add drop-down menus to your dashboard with appropriate titles and options

Select Report-type Dropdown

Create a dropdown menu and add two ***Report*** options to it.

Below is the skeleton:

```
dcc.Dropdown(id='....',
            options=[
                    {'label': '....', 'value': '...'},
                    {'label': '....', 'value': '...'}
                    ],
            placeholder='....',
            style={....})
```

Parameters to be updated in `dcc.Dropdown`:

Set id to `dropdown-statistics`.

Set options to a **list containing dictionaries** with label and user-provided values for value.

Option List Parameters

1st option:

label: `Yearly Statistics`
value: `Yearly Statistics`

2nd option:

label: `Recession Period Statistics`
value: `Recession Period Statistics`

Set placeholder to `Select a report type`.

Set value to `Select Statistics`.

OPTIONAL: Set style to include width as 80%, padding as 3px, font size as 20px, and text-align-last as center.

Select Year Dropdown

Create a dropdown menu for selecting the ***YEAR*** options to it.
Below is the skeleton:

```
dcc.Dropdown(id='....',
             options=[{'label': i, 'value': i} for i in year_list],
             placeholder='....',
             style={....})
```

- You can pass a list of years as options in this dropdowwn
  [hint: year_list = [i for i in range(1980, 2024, 1)]]

- Set id to `select-year`

- Set value to `Select-year`

- Set placeholder to `Select-year`

For reference, observe how code under `REVIEW` has been structured.

REVIEW link for `dcc.Dropdown` component

Click here to view a sample example.

TASK 2.3: Add a division for output display with appropriate id and classname property

Add an inner division to display the output

```
html.Div([
    html.Div(id='........', className='..........., style={'display': 'flex'}),
])
```

- Set id to `output-container`.
- Set className to `chart-grid`
- Set style to be displayed as a `flex`

We will pass the plots as returned by the callback function into this `output-container` later refering to the class name of it.

TASK 2.4: Creating Callbacks; Define the callback function to update the input container based on the selected statistics and the output container

Update Input Container callback function

The purpose of this function is to enable or disable the year selection dropdown based on the user's choice of report type from another dropdown.Here we are having 2 parts:

Callback definition:

The parameters of the Callback definition are as follows:
Output: Specifies the output of the callback, which in this case is the `disabled` property of the dropdown with the ID `select-year`.
Input: Specifies the input that triggers the callback, which is the `value` property of the dropdown with the ID `dropdown-statistics`.

Callback Function:

- The function checks the value of selected statistics. If the selected statistics is `Yearly Statistics` the function returns False. This means the `disabled` property of the select-year dropdown will be set to False, enabling the dropdown so the user can select a year.

- If selected statistics is not `Yearly Statistics` (i.e., it is `Recession Period Statistics`), the function returns True. This means the `disabled` property of the select-year dropdown will be set to True, disabling the dropdown as selecting a year is not relevant in this context.

```
@app.callback(
    Output(component_id='....', component_property='disabled'),
    Input(component_id='....',component_property='....'))
def update_input_container(.....):
    if ..... =='Yearly Statistics':
        return False
    else:
        return True
```

Update Output Container callback function

Our layout has 1 output container, and we will be required to return the plots developed `dcc.Graph()` into this container as `divisions`.
For each report we need to display four plots.

The `update_output_container` function in this Dash application dynamically generates and displays graphs based on the user's selection from the dropdown menus. Specifically, it creates different sets of graphs for `Recession Period Statistics` and `Yearly Statistics`. We will be creating 4 plots for each report.

Here we have 2 parts:

Callback Definition:

Parameters:

- **Output**:This specifies that the `children` property of the component with the ID `output-container` will be updated.In Dash, children typically refers to the content within a container, such as graphs, text, or other HTML elements.

- **Input**:

- Input Property for Report type:This specifies that the `value` property of the dropdown menu with the ID `dropdown-statistics` (which allows the user to select between `Yearly Statistics` and `Recession Period Statistics`) will trigger the callback when it changes.

  - Input property for year selection: This specifies that the `value` property of the dropdown menu with the ID `select-year` (which allows the user to select a specific year) will trigger the callback when it changes.

```
@app.callback(
    Output(component_id='....', component_property='children'),
    [Input(component_id='....', component_property='...'), Input(component_id='....', component_property='...')])
```

Callback Function:

Here you will filter the data based on the `report type` selected.

When `Recession Period Statistics` is selected, the data is filtered to include only recession periods where `Recession equals 1`. Conversely, when `Yearly Statistics` is chosen, the data is filtered based on the selected year.

```
def update_output_container(...., ...):
    if .... == 'Recession Period Statistics':
        # Filter the data for recession periods
        recession_data = data[data['Recession'] == 1]
        ......................
```

NOTE: Take the screenshot representing the <u>the code snippet wherein you have created the two callback functions</u> and save it as **'Callbacks.png'**

TASK 2.5: Create and display graphs for Recession Report Statistics

You will be required to prepare the data as per the plot requirement (usually using `groupby()`)

   Note: Use `sum()` and `mean()` functions with `groupby()` depending on the question asked.

Then you need to Create a different plots using `plotly.express`.

```
#Plot 1 Automobile sales fluctuate over Recession Period (year wise) using line chart
     # grouping data for plotting
     yearly_rec=recession_data.groupby('Year')['Automobile_Sales'].mean().reset_index()
     # Plotting the line graph
     R_chart1 = dcc.Graph(
         figure=px.line(yearly_rec,
             x='........',
             y='........',
             title="........"))
.........
#Plot 2 Calculate the average number of vehicles sold by vehicle type and represent as a Bar chart
 # use groupby to create relevant data for plotting.
 #Hint:Use Vehicle_Type and Automobile_Sales columns
     average_sales  = recession_data.groupby(.............)
     R_chart2  = dcc.Graph(
         figure=px.bar(.............,
         x='.............',
         y='.............',
         title=".............."))
............
# Plot 3 : Pie chart for total expenditure share by vehicle type during recessions
         # grouping data for plotting
         # Hint:Use Vehicle_Type and Advertising_Expenditure columns
         exp_rec= recession_data.groupby(...............)
         R_chart3 = dcc.Graph(
                 figure=px.pie(.............,
                 values='............',
             names='.........',
             title="..........."
             )
     )
..........
# Plot 4 Develop a Bar chart for the effect of unemployment rate on vehicle type and sales
     #grouping data for plotting
     # Hint:Use unemployment_rate,Vehicle_Type and Automobile_Sales columns
     unemp_data= recession_data.groupby(['.......', '..............'])['.........'].mean().reset_index()
     R_chart4 = dcc.Graph(figure=px.bar(...............,
     x='...............',
     y='...............',
     color='.............',
     labels={'unemployment_rate': 'Unemployment Rate', 'Automobile_Sales': 'Average Automobile Sales'},
     title='Effect of Unemployment Rate on Vehicle Type and Sales'))
     return [
         html.Div(className='chart-item', children=[html.Div(children=R_chart1),html.Div(children=R_chart2)],style={'display': 'flex'}),
         html.Div(className='chart-item', children=[...........],style={'display': 'flex'})
         ]
```

REVIEW LINKS for Line charts, Bar charts and Pie charts:-

- Line Chart
- Bar Chart
- Pie Chart

NOTE: Label the plots properly.

NOTE: Once the application is up and running, take the screenshots representing the graphs for Recession report type , [each graph should be clearly captured] and save it as 'RecessionReportgraphs.png'

TASK 2.6: Create and display graphs for Yearly Report Statistics

Refer to the code snippet below:-

```
# Yearly Statistic Report Plots
    # Check for Yearly Statistics.
    elif (input_year and selected_statistics=='..............') :
        yearly_data = data[data['Year'] == ......]

..........
# Plot 1 :Yearly Automobile sales using line chart for the whole period.
............
        # grouping data for plotting.
        # Hint:Use the columns Year and Automobile_Sales.
        yas= data.groupby('Year')['Automobile_Sales'].mean().reset_index()
        Y_chart1 = dcc.Graph(figure=px.line(.................))

..........
# Plot 2 :Total Monthly Automobile sales using line chart.
..........
         # grouping data for plotting.
        # Hint:Use the columns Month and Automobile_Sales.
        mas=data.groupby(.............)
        Y_chart2 = dcc.Graph(figure=px.line(............,
            x='.....',
            y='......',
            title='Total Monthly Automobile Sales'))
 ..........
# Plot bar chart for average number of vehicles sold during the given year
............
         # grouping data for plotting.
         # Hint:Use the columns Year and Automobile_Sales
        avr_vdata=yearly_data.groupby........................
        Y_chart3 = dcc.Graph( figure=.................,title='Average Vehicles Sold by Vehicle Type in the year {}'.format(input_year)))
..........
# Plot 4 Total Advertisement Expenditure for each vehicle using pie chart
         # grouping data for plotting.
         # Hint:Use the columns Vehicle_Type and Advertising_Expenditure
        exp_data=yearly_data.groupby('.......')
        Y_chart4 = dcc.Graph(
            figure=px.pie(.........,
            values='......',
            names='......',
            title='Total Advertisment Expenditure for Each Vehicle'))
    return [
            html.Div(className='chart-item', children=[html.Div(children=Y_chart1),html.Div(children=Y_chart2)],style={'display':'flex'}),
            html.Div(className='chart-item', children=[...........],style={'display': 'flex'})
            ]
```

NOTE: Once the application is up and running, take the screenshots representing all four graphs for the Yearly report types and save it as'YearlyReportgraphs.png'

Run the Application

- Firstly, install setuptools, packaging, pandas and dash using the following command.

```
pip3.8 install setuptools
```

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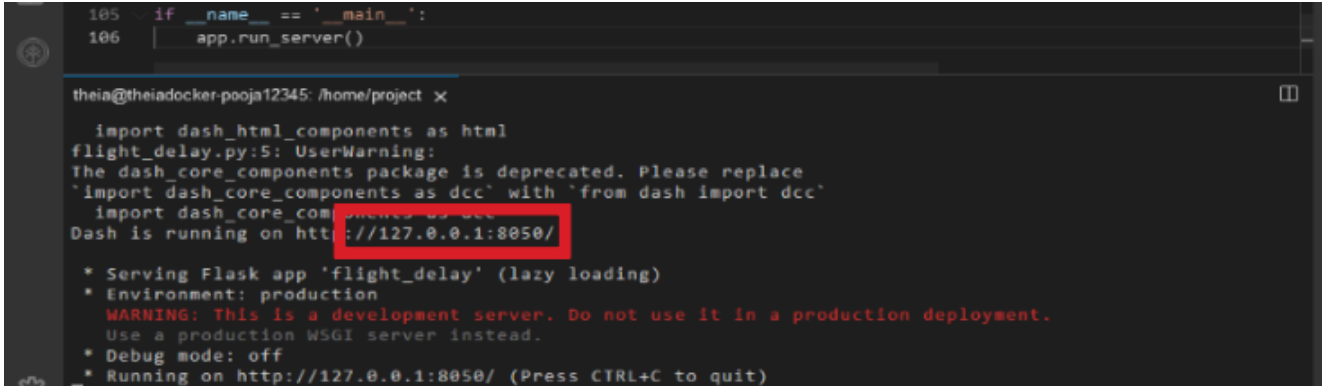
```
python3.8 -m pip install pandas dash
```

```
pip install more-itertools
```
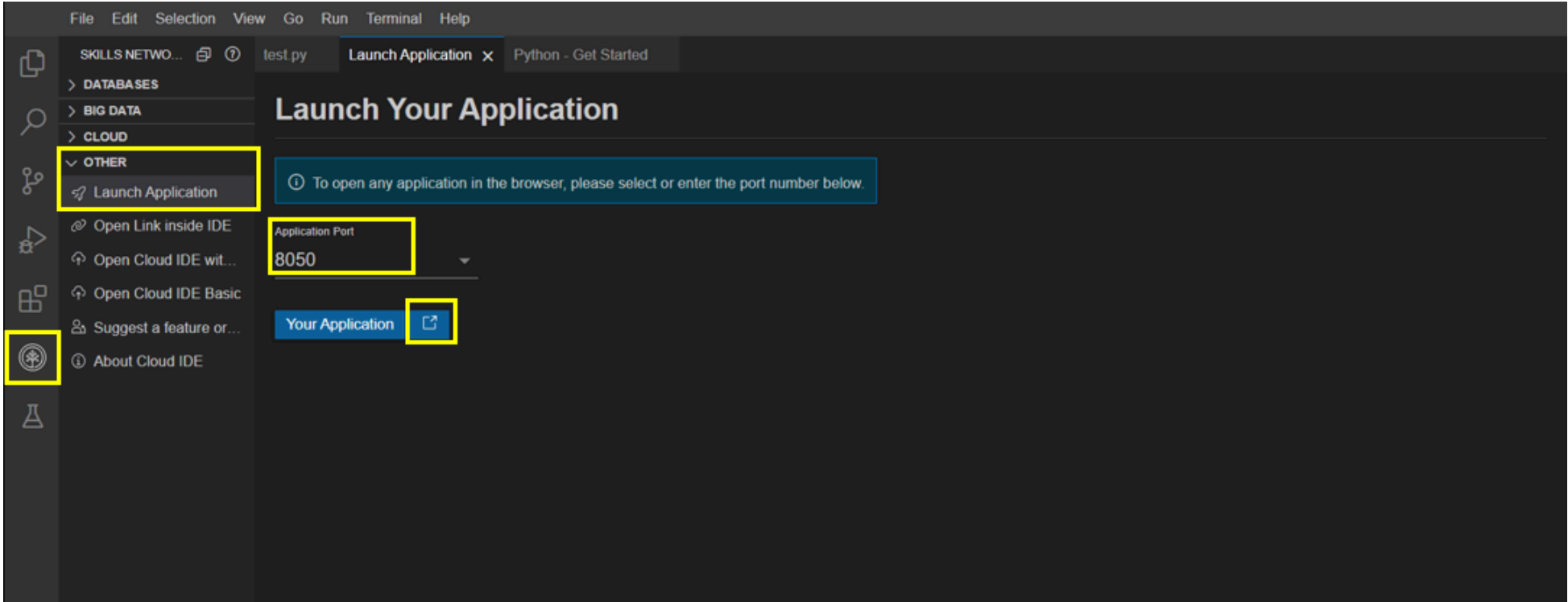
- Run the python file using the command

```
python3.8 DV0101EN-Final-Assign-Part-2-Questions.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`

Congratulations, you have successfully completed your application!

Author

Dr. Pooja