Student name: Hoang Long Tran

Student ID: s223128143

SIT225: Data Capture Technologies

Activity 6.1: Plotly data dashboard

Plotly Dash apps give a point-&-click interface to models written in Python, vastly expanding the notion of what's possible in a traditional "dashboard". With Dash apps, data scientists and engineers put complex Python analytics in the hands of business decision-makers and operators. In this activity, you will learn basic building blocks of Plotly to create Dash apps.

Hardware Required

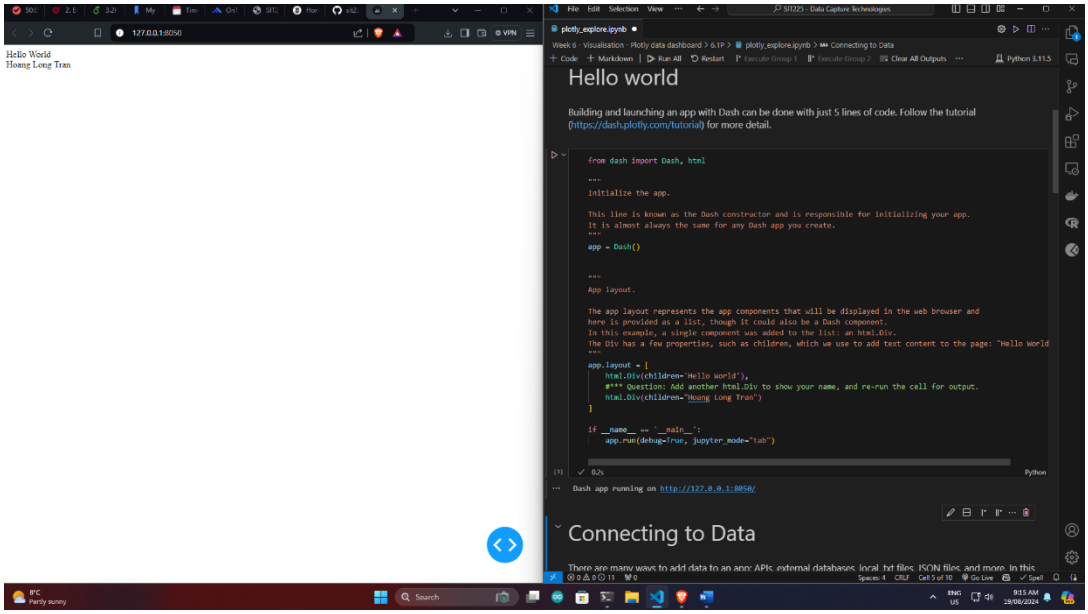
No hardware is required.

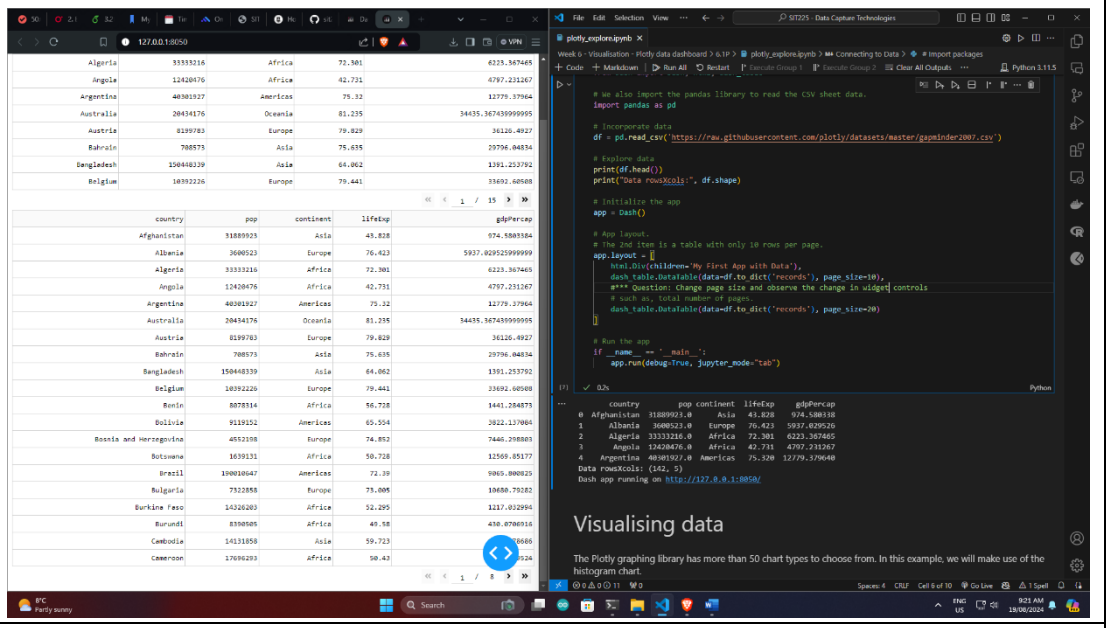
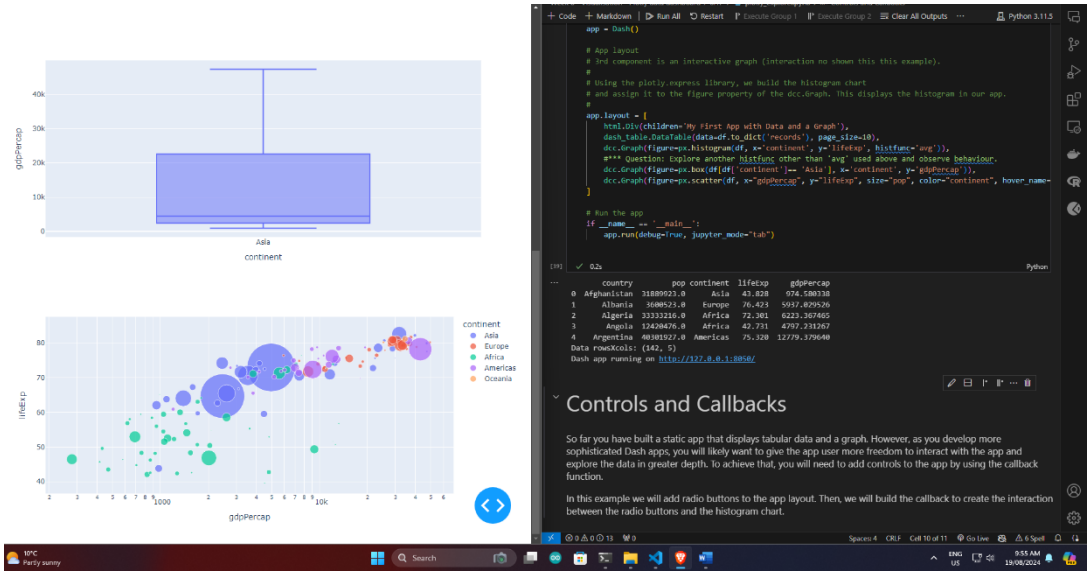
Software Required

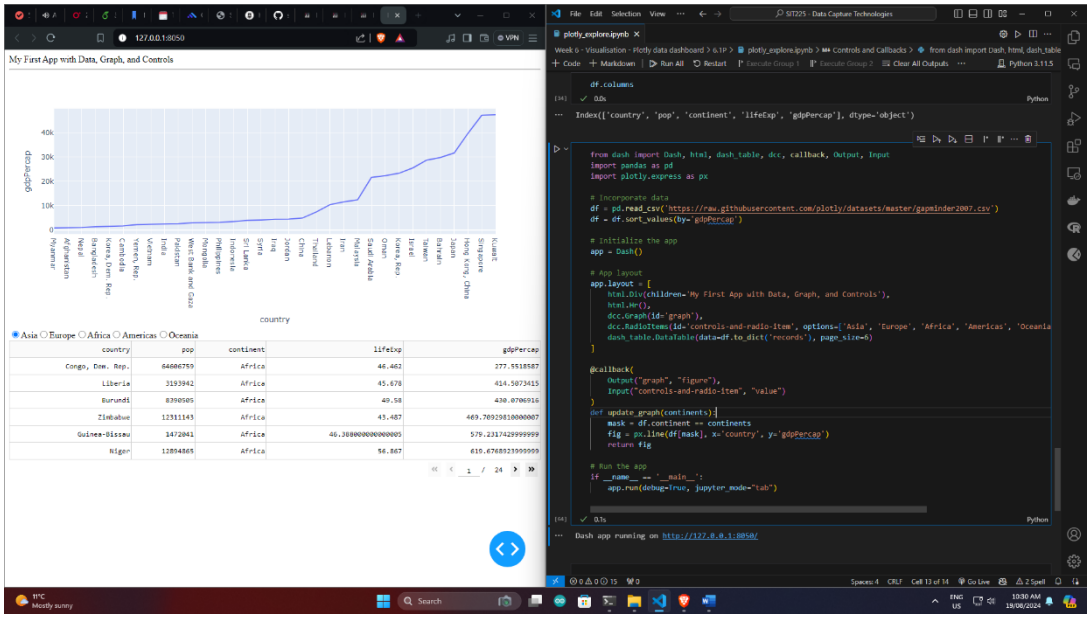
Plotly library and Dash module
Python 3

Steps

Step	Action
1	Install Plotly and dash using the command below in the command line. $ pip install plotly dash You can download Jupyter Notebook from here (https://github.com/deakin-deep-dreamer/sit225/blob/main/week_6/plotly_explore.ipynb) and run all the cells. The Notebook contains multiple sections such as Hello World which follows a sample code in a following cell. If you run the Hello world cell it will show Plotly Dash web page. The cell also includes a Question (#*** Question) which you will need to carry out to get a modified output. You will need to capture the output and share the screenshot in the following steps.

2	Question: **Hello World** cell has a question - add another html.Div to show your name, and re-run the cell for output. You will need to update the code, run the cell, capture the screenshot of the output and paste it here. Answer: 
3	Question: **Connecting to Data** cell has a question - change page size and observe the change in widget controls such as, total number of pages. You will need to update the code, run the cell, capture the screenshot of the output and paste it here.

	Answer: 
4	Question: **Visualising data** cell has a question - explore another histfunc other than 'avg' used above and observe behaviour. You will need to update the code, run the cell, capture the screenshot of the output and paste it here. Answer: 
5	Question: **Controls and Callbacks** cell has a question - use line graphs instead of histogram. You will need to update the code, run the cell, capture the screenshot of the output and paste it here.

	Answer: 
6	Question: Now you have learned how to use Plotly Dash for visualising your data, describe how you will be using this tool for your desired sensor monitoring dashboard with a number of sensors including DHT22 or accelerometer data. Answer: If I have enough data for DHT22, I could use a scatterplot and machine learning techniques to describe the relationship between temperature and humidity based on the season.
7	Question: Convert the Notebook to PDF and merge with this activity sheet PDF. You will need this merged PDF to combine with this week's OnTrack task for submission.

Answer:

Hello world

```
# Fill in student ID and name
#
student_id = "Hoang Long Tran"
student_first_last_name = "s223128143"
print(student_id, student_first_last_name)
```

```
Hoang Long Tran s223128143
```

```
# install plotly and dash, if not yet already
# ! pip install plotly dash

import plotly, dash
print(plotly.__version__)
print(dash.__version__)
```

```
5.22.0
2.17.1
```

Building and launching an app with Dash can be done with just 5 lines of code. Follow the tutorial (https://dash.plotly.com/tutorial) for more detail.

```
from dash import Dash, html

"""
Initialize the app.

This line is known as the Dash constructor and is responsible for initializing your app.
It is almost always the same for any Dash app you create.
"""
app = Dash()
```

6.1P. Plotly data dashboard

Q2.

```
from dash import Dash, html, dash_table, dcc, callback, Output, Input, State
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go

# Initialize the app
```

```
app = Dash()

# App layout
app.layout = html.Div([
    html.H1(children='Gyroscope data'),
    html.Hr(),

    # Drop down for graphs
    html.Label("Select Graph Type:"),
    dcc.Dropdown(
        id='graph-type',
        options=[
            {'label': 'Scatter Plot', 'value': 'scatter'},
            {'label': 'Line Chart', 'value': 'line'},
            {'label': 'Distribution Plot', 'value': 'distribution'},
        ],
        value = 'scatter'
    ),

    # Data Table
    dash_table.DataTable(data=df1.to_dict('records'), page_size=6),

    # Drop down for rotation selection
    dcc.Dropdown(id='rotation-type',
                 options=[ {'label': 'x', 'value': 'x'},
                           {'label': 'y', 'value': 'y'},
                           {'label': 'z', 'value': 'z'} ],
                 value=['x', 'y', 'z'], multi=True),

    # Input for the number of samples
    html.Label("Number of Samples: "),
    dcc.Input(id='num-samples', type='number', value=1020, min=100, step=20),

    # Container for the graph and summary table
    html.Div([
        dcc.Graph(figure={}, id='graph', style={'display': 'inline-block',
'width': '49%'}),
        dcc.Graph(id='data-summary', style={'display': 'inline-block', 'width':
'49%', 'vertical-align': 'top'}),
    ], style={'display': 'flex', 'flex-wrap': 'wrap'})
])

@callback(
    Output("graph", "figure"),
    Output('data-summary', 'figure'),
```

```python
    Input("rotation-type", "value"),
    Input('graph-type', 'value'),
    Input('num-samples', 'value'),
)
def update(rotation, graph_type, num_sample):
    df2 = df1.head(num_sample)

    # Create plot based on selected graph
    fig = {}
    if graph_type == 'scatter':
        fig = px.scatter(df2, x='Timestamp', y=rotation)
    elif graph_type == 'line':
        fig = px.line(df2, x='Timestamp', y=rotation)
    elif graph_type == 'distribution':
        fig = px.histogram(df2, x=rotation)

    # Data summary table
    try:
        df3 = df2[rotation]
        summary = df3.describe().reset_index()
        summary_table = go.Figure(data=[go.Table(
            header=dict(values=list(summary.columns),
                        fill_color='paleturquoise',
                        align='left'),
            cells=dict(values=[summary[col] for col in summary.columns],
                    fill_color='lavender',
                    align='left')

        )])
    except Exception as e:
        summary_table = go.Figure(data=[go.Table(
            header=dict(values=["No data selected"],
                        fill_color='paleturquoise',
                        align='left'),
            cells=dict(values=[["Please select at least one rotation axis"]],
                       fill_color='lavender',
                       align='left')
        )])

    return fig, summary_table

# Run the app
if __name__ == '__main__':
    app.run(debug=True, jupyter_mode="tab")
```

I have added some comments but here is a brief explanation. First, I initialize the app layout, then add a heading. Below the main heading is a drop-down menu of plots, with the default one being scatter plot. Below is the data table containing all the data in the data frame. Then it's the drop-down menu to select rotations, the default is all rotation is selected. There are options to choose the number of samples, default value is 1020 (all of the sample available), the step is 20 samples, if we go over 1020 then nothing changes because 1020 is the maximum of data, if we go below 1020, then the graph and summary statistics table will change because there is less data. If no rotation is selected, then the summary table notifies the user to select at least one rotation.

Q3.

https://www.youtube.com/watch?v=v0ceW-48gjg

Q4.

https://github.com/tomadonna1/SIT225_2024T2/tree/main/Pass%20Task%20Plotly%20data%20dashboard