In [1]:
```
# Fill in student ID and name
#
student_id = "221435713"
student_first_last_name = "Anish Basnsal"
print(student_id, student_first_last_name)
```

```
221435713 Anish Basnsal
```

In [2]:
```
# install plotly and dash, if not yet already
! pip install plotly dash

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

```
Requirement already satisfied: plotly in /Users/jazz/anaconda3/lib/python
3.12/site-packages (5.22.0)
Collecting dash
  Downloading dash-2.18.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: tenacity>=6.2.0 in /Users/jazz/anaconda3/li
b/python3.12/site-packages (from plotly) (8.2.2)
Requirement already satisfied: packaging in /Users/jazz/anaconda3/lib/pyth
on3.12/site-packages (from plotly) (23.2)
Requirement already satisfied: Flask<3.1,>=1.0.4 in /Users/jazz/anaconda3/
lib/python3.12/site-packages (from dash) (3.0.3)
Requirement already satisfied: Werkzeug<3.1 in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from dash) (3.0.3)
Collecting dash-html-components==2.0.0 (from dash)
  Downloading dash_html_components-2.0.0-py3-none-any.whl.metadata (3.8 k
B)
Collecting dash-core-components==2.0.0 (from dash)
  Downloading dash_core_components-2.0.0-py3-none-any.whl.metadata (2.9 k
B)
Collecting dash-table==5.0.0 (from dash)
  Downloading dash_table-5.0.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: importlib-metadata in /Users/jazz/anaconda
3/lib/python3.12/site-packages (from dash) (7.0.1)
Requirement already satisfied: typing-extensions>=4.1.1 in /Users/jazz/ana
conda3/lib/python3.12/site-packages (from dash) (4.11.0)
Requirement already satisfied: requests in /Users/jazz/anaconda3/lib/pytho
n3.12/site-packages (from dash) (2.32.2)
Collecting retrying (from dash)
  Downloading retrying-1.3.4-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: nest-asyncio in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from dash) (1.6.0)
Requirement already satisfied: setuptools in /Users/jazz/anaconda3/lib/pyt
hon3.12/site-packages (from dash) (69.5.1)
Requirement already satisfied: Jinja2>=3.1.2 in /Users/jazz/anaconda3/lib/
python3.12/site-packages (from Flask<3.1,>=1.0.4->dash) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in /Users/jazz/anaconda
3/lib/python3.12/site-packages (from Flask<3.1,>=1.0.4->dash) (2.2.0)
Requirement already satisfied: click>=8.1.3 in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from Flask<3.1,>=1.0.4->dash) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in /Users/jazz/anaconda3/li
b/python3.12/site-packages (from Flask<3.1,>=1.0.4->dash) (1.6.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Users/jazz/anaconda3/
lib/python3.12/site-packages (from Werkzeug<3.1->dash) (2.1.3)
Requirement already satisfied: zipp>=0.5 in /Users/jazz/anaconda3/lib/pyth
on3.12/site-packages (from importlib-metadata->dash) (3.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/jazz/ana
conda3/lib/python3.12/site-packages (from requests->dash) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from requests->dash) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/jazz/anaconda
3/lib/python3.12/site-packages (from requests->dash) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in /Users/jazz/anaconda
3/lib/python3.12/site-packages (from requests->dash) (2024.6.2)
Requirement already satisfied: six>=1.7.0 in /Users/jazz/anaconda3/lib/pyt
hon3.12/site-packages (from retrying->dash) (1.16.0)
Downloading dash-2.18.0-py3-none-any.whl (7.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.5/7.5 MB 7.8 MB/s eta 0:00:0
000:0100:01
Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
```

```
Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Installing collected packages: dash-table, dash-html-components, dash-core
-components, retrying, dash
Successfully installed dash-2.18.0 dash-core-components-2.0.0 dash-html-co
mponents-2.0.0 dash-table-5.0.0 retrying-1.3.4
5.22.0
2.18.0
```

# Hello world

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.

In [5]:
```
from dash import Dash, html

"""
Initialize the app.

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


"""
App layout.

The app layout represents the app components that will be displayed in th
here is provided as a list, though it could also be a Dash component.
In this example, a single component was added to the list: an html.Div.
The Div has a few properties, such as children, which we use to add text
"""
app.layout = [
    html.Div(children='Hello World'),
    #*** Question: Add another html.Div to show your name, and re-run the
    html.Div(children='Name: Anish Bansal')  # This is the added div to s
]

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

```
Dash app running on http://127.0.0.1:8050/
```

# Connecting to Data

There are many ways to add data to an app: APIs, external databases, local .txt files, JSON files, and more. In this example, we will highlight one of the most common ways of incorporating data from a CSV sheet.

In [10]:
```
# Import packages

# We import the dash_table module to display the data inside a Dash DataT
from dash import Dash, html, dash_table

# We also import the pandas library to read the CSV sheet data.
```

```
import pandas as pd

# Incorporate data
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/maste

# Explore data
print(df.head())
print("Data rowsXcols:", df.shape)

# Initialize the app
app = Dash()

# App layout.
# The 2nd item is a table with only 15 rows per page (changed from 10).
app.layout = [
    html.Div(children='My First App with Data'),
    dash_table.DataTable(data=df.to_dict('records'), page_size=15)  # Cha
]

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

```
       country         pop continent  lifeExp     gdpPercap
0  Afghanistan  31889923.0      Asia   43.828    974.580338
1      Albania   3600523.0    Europe   76.423   5937.029526
2      Algeria  33333216.0    Africa   72.301   6223.367465
3       Angola  12420476.0    Africa   42.731   4797.231267
4    Argentina  40301927.0  Americas   75.320  12779.379640
Data rowsXcols: (142, 5)
Dash app running on http://127.0.0.1:8050/
```

# Visualising data

The Plotly graphing library has more than 50 chart types to choose from. In this example, we will make use of the histogram chart.

In [14]:
```
# Import packages

# We import the dcc module (DCC stands for Dash Core Components).
# This module includes a Graph component called dcc.Graph, which is used
from dash import Dash, html, dash_table, dcc

# We also import the plotly.express library to build the interactive grap
import plotly.express as px

import pandas as pd

# Incorporate data
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/maste

# Explore data
print(df.head())
print("Data rowsXcols:", df.shape)

# Initialize the app
app = Dash()
```

```
# App layout
# 3rd component is an interactive graph (interaction no shown this this e
#
# Using the plotly.express library, we build the histogram chart
# and assign it to the figure property of the dcc.Graph. This displays th
#
app.layout = [
    html.Div(children='My First App with Data and a Graph'),
    dash_table.DataTable(data=df.to_dict('records'), page_size=10),
    dcc.Graph(figure=px.histogram(df, x='continent', y='lifeExp', histfun
]

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

```
       country         pop continent  lifeExp     gdpPercap
0  Afghanistan  31889923.0      Asia   43.828    974.580338
1      Albania   3600523.0    Europe   76.423   5937.029526
2      Algeria  33333216.0    Africa   72.301   6223.367465
3       Angola  12420476.0    Africa   42.731   4797.231267
4    Argentina  40301927.0  Americas   75.320  12779.379640
Data rowsXcols: (142, 5)
Dash app running on http://127.0.0.1:8050/
```

# Controls and Callbacks

So far you have built a static app that displays tabular data and a graph. However, as you develop more sophisticated Dash apps, you will likely want to give the app user more freedom to interact with the app and explore the data in greater depth. To achieve that, you will need to add controls to the app by using the callback function.

In this example we will add radio buttons to the app layout. Then, we will build the callback to create the interaction between the radio buttons and the histogram chart.

In [17]:
```
# Import packages

# We import dcc like we did in the previous section to use dcc.Graph.
# In this example, we need dcc for dcc.Graph as well as the radio buttons
#
# To work with the callback in a Dash app, we import the callback module
# commonly used within the callback: Output and Input.
#
from dash import Dash, html, dash_table, dcc, callback, Output, Input
import pandas as pd
import plotly.express as px

# Incorporate data
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/maste

# Initialize the app
app = Dash()

# App layout
app.layout = [
    html.Div(children='My First App with Data, Graph, and Controls'),
    html.Hr(),
```

```
    dcc.RadioItems(options=['pop', 'lifeExp', 'gdpPercap'], value='lifeEx
    dash_table.DataTable(data=df.to_dict('records'), page_size=6),
    dcc.Graph(figure={}, id='controls-and-graph')
    #*** Question: Use line graphs instead of histogram.
]

# Add controls to build the interaction
@callback(
    Output(component_id='controls-and-graph', component_property='figure'
    Input(component_id='controls-and-radio-item', component_property='val
)
def update_graph(col_chosen):
    # Using line graph instead of histogram
    fig = px.line(df, x='continent', y=col_chosen, title=f'Line graph of
    return fig

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

```
Dash app running on http://127.0.0.1:8050/
```