In [1]:
```
# Fill in student ID and name
#
student_id = "223212228"
student_first_last_name = "Krystal_Nguyen"
print(student_id, student_first_last_name)
```

```
223212228 Krystal_Nguyen
```

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/d.o.npat/anaconda3/lib/python3.10/site-packages (5.22.0)
Requirement already satisfied: dash in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (2.17.1)
Requirement already satisfied: tenacity>=6.2.0 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from plotl
y) (8.3.0)
Requirement already satisfied: packaging in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from plotly) (2
3.2)
Requirement already satisfied: setuptools in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from dash) (65.
6.3)
Requirement already satisfied: dash-html-components==2.0.0 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages
(from dash) (2.0.0)
Requirement already satisfied: importlib-metadata in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from da
sh) (7.2.0)
Requirement already satisfied: Werkzeug<3.1 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from dash)
(3.0.3)
Requirement already satisfied: Flask<3.1,>=1.0.4 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from das
h) (3.0.3)
Requirement already satisfied: requests in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from dash) (2.32.
3)
Requirement already satisfied: retrying in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from dash) (1.3.
4)
Requirement already satisfied: nest-asyncio in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from dash)
(1.6.0)
Requirement already satisfied: typing-extensions>=4.1.1 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (f
rom dash) (4.12.2)
Requirement already satisfied: dash-table==5.0.0 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from das
h) (5.0.0)
Requirement already satisfied: dash-core-components==2.0.0 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages
(from dash) (2.0.0)
Requirement already satisfied: itsdangerous>=2.1.2 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from F
lask<3.1,>=1.0.4->dash) (2.2.0)
Requirement already satisfied: click>=8.1.3 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from Flask<3.
1,>=1.0.4->dash) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from Flask<
3.1,>=1.0.4->dash) (1.8.2)
Requirement already satisfied: Jinja2>=3.1.2 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from Flask<
3.1,>=1.0.4->dash) (3.1.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from Wer
kzeug<3.1->dash) (2.1.3)
Requirement already satisfied: zipp>=0.5 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from importlib-m
etadata->dash) (3.19.2)
Requirement already satisfied: idna<4,>=2.5 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from requests
```

```
->dash) (3.7)
Requirement already satisfied: certifi>=2017.4.17 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from re
quests->dash) (2024.6.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (f
rom requests->dash) (2.0.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from re
quests->dash) (2.2.1)
Requirement already satisfied: six>=1.7.0 in /Users/d.o.npat/anaconda3/lib/python3.10/site-packages (from retrying->
dash) (1.16.0)
5.22.0
2.17.1
```

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 [3]:
```python
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()

"""
App layout.
The app layout represents the app components that will be displayed in the web browser and
here is provided as a list, though it could also be a Dash component.
In this example, we have two components added to the list: two html.Div elements.
The Divs have a few properties, such as children, which we use to add text content to the page.
"""
app.layout = [
    html.Div(children='Hello World'),
    html.Div(children='My name is Krystal Nguyen')
]
```

```
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 [4]:
```
# Import packages
import ssl
import certifi
import urllib.request
from dash import Dash, html, dash_table
import pandas as pd
import io

# Create a custom SSL context using certifi
ssl_context = ssl.create_default_context(cafile=certifi.where())

# Use the custom SSL context when opening the URL
url = 'https://raw.githubusercontent.com/plotly/datasets/master/gapminder2007.csv'
with urllib.request.urlopen(url, context=ssl_context) as response:
    csv_data = response.read().decode('utf-8')

# Incorporate data
df = pd.read_csv(io.StringIO(csv_data))

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

# Initialize the app
app = Dash()

# App layout with multiple DataTables of different page sizes
app.layout = html.Div([
```

```
    html.H1(children='My First App with Data - Page Size Comparison'),

    html.H2(children='Table with 5 rows per page'),
    dash_table.DataTable(
        id='table-5-rows',
        data=df.to_dict('records'),
        columns=[{"name": i, "id": i} for i in df.columns],
        page_size=5
    ),

    html.H2(children='Table with 10 rows per page'),
    dash_table.DataTable(
        id='table-10-rows',
        data=df.to_dict('records'),
        columns=[{"name": i, "id": i} for i in df.columns],
        page_size=10
    ),

    html.H2(children='Table with 20 rows per page'),
    dash_table.DataTable(
        id='table-20-rows',
        data=df.to_dict('records'),
        columns=[{"name": i, "id": i} for i in df.columns],
        page_size=20
    )
])

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

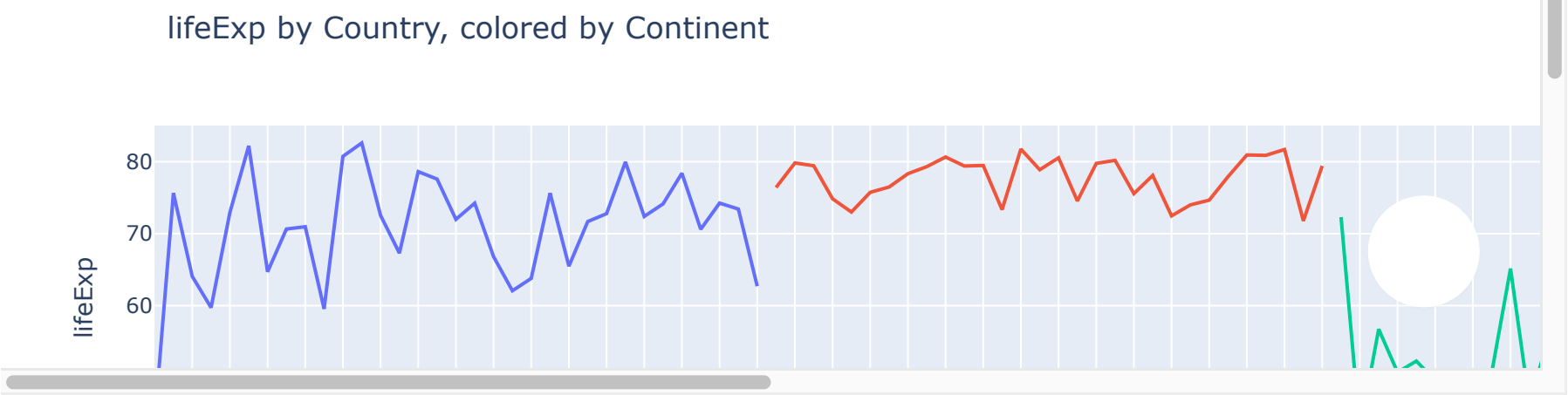
```
       country         pop continent  lifeExp     gdpPercap
0  Afghanistan  31889923.0      Asia   43.828    974.580338
1      Albania   3600523.0    Europe   76.423   5937.029526
2      Algeria  33333216.0    Africa   72.301   6223.367465
3       Angola  12420476.0    Africa   42.731   4797.231267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

My First App with Data, Graph, and Controls

- ○ pop
- ◉ lifeExp
- ○ gdpPercap

country	pop	continent	lifeExp	gdpPercap
Afghanistan	31889923	Asia	43.828	974.5803384
Albania	3600523	Europe	76.423	5937.029525999999
Algeria	33333216	Africa	72.301	6223.367465
Angola	12420476	Africa	42.731	4797.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995

« ‹ 1 / 24 › »



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 [5]:
```
import ssl
import certifi
import urllib.request
from dash import Dash, html, dash_table, dcc
import plotly.express as px
import pandas as pd
import io

# Create a custom SSL context using certifi
ssl_context = ssl.create_default_context(cafile=certifi.where())

# Use the custom SSL context when opening the URL
url = 'https://raw.githubusercontent.com/plotly/datasets/master/gapminder2007.csv'
with urllib.request.urlopen(url, context=ssl_context) as response:
    csv_data = response.read().decode('utf-8')

# Read the CSV data
df = pd.read_csv(io.StringIO(csv_data))

# Initialize the app
app = Dash()

# App layout
app.layout = html.Div([
    html.H1(children='Histogram Function Comparison'),
    dash_table.DataTable(data=df.to_dict('records'), page_size=10),
    html.H2(children='Average Life Expectancy by Continent'),
    dcc.Graph(figure=px.histogram(df, x='continent', y='lifeExp', histfunc='avg', title='Average (default)')),
    html.H2(children='Sum of Life Expectancy by Continent'),
    dcc.Graph(figure=px.histogram(df, x='continent', y='lifeExp', histfunc='sum', title='Sum')),
    html.H2(children='Maximum Life Expectancy by Continent'),
    dcc.Graph(figure=px.histogram(df, x='continent', y='lifeExp', histfunc='max', title='Maximum')),
    html.H2(children='Minimum Life Expectancy by Continent'),
```

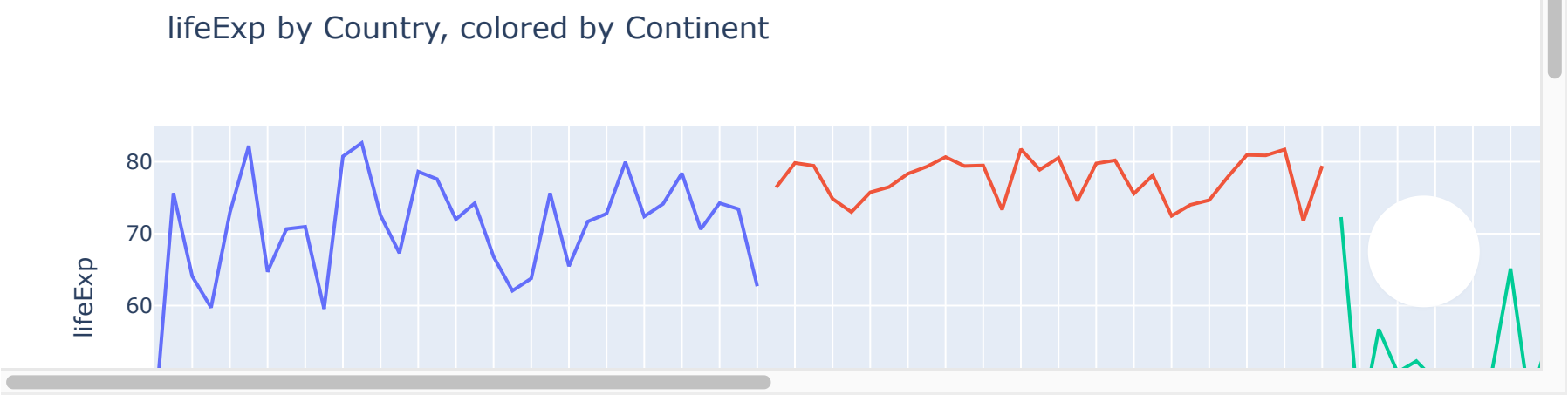
```
    dcc.Graph(figure=px.histogram(df, x='continent', y='lifeExp', histfunc='min', title='Minimum')),
    html.H2(children='Count of Countries by Continent'),
    dcc.Graph(figure=px.histogram(df, x='continent', histfunc='count', title='Count'))
])

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

My First App with Data, Graph, and Controls

- ○ pop
- ◉ lifeExp
- ○ gdpPercap

country	pop	continent	lifeExp	gdpPercap
Afghanistan	31889923	Asia	43.828	974.5803384
Albania	3600523	Europe	76.423	5937.029525999999
Algeria	33333216	Africa	72.301	6223.367465
Angola	12420476	Africa	42.731	4797.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995

« ‹ 1 / 24 › »



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 [6]:
```
import ssl
import certifi
import urllib.request
from dash import Dash, html, dash_table, dcc, callback, Output, Input
import plotly.express as px
import pandas as pd
import io

# Create a custom SSL context using certifi
ssl_context = ssl.create_default_context(cafile=certifi.where())

# Use the custom SSL context when opening the URL
url = 'https://raw.githubusercontent.com/plotly/datasets/master/gapminder2007.csv'
with urllib.request.urlopen(url, context=ssl_context) as response:
    csv_data = response.read().decode('utf-8')

# Incorporate data
df = pd.read_csv(io.StringIO(csv_data))

# Initialize the app
app = Dash()

# App layout
app.layout = html.Div([
    html.H1(children='My First App with Data, Graph, and Controls'),
    html.Hr(),
    dcc.RadioItems(options=['pop', 'lifeExp', 'gdpPercap'], value='lifeExp', id='controls-and-radio-item'),
```

```
    dash_table.DataTable(data=df.to_dict('records'), page_size=6),
    dcc.Graph(figure={}, id='controls-and-graph')
])

# 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='value')
)
def update_graph(col_chosen):
    fig = px.line(df, x='country', y=col_chosen, color='continent',
                  title=f'{col_chosen} by Country, colored by Continent')
    fig.update_xaxes(tickangle=45)
    return fig

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

My First App with Data, Graph, and Controls

- ○ pop
- ◉ lifeExp
- ○ gdpPercap

country	pop	continent	lifeExp	gdpPercap
Afghanistan	31889923	Asia	43.828	974.5803384
Albania	3600523	Europe	76.423	5937.029525999999
Algeria	33333216	Africa	72.301	6223.367465
Angola	12420476	Africa	42.731	4797.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995

« ‹ 1 / 24 › »