Q0: import dash standard libraries

In [6]:

```
from dash import Dash, dcc, html, Input, Output
from jupyter_dash import JupyterDash
import pandas as pd
!pip install --upgrade plotly
import plotly.express as px
import dash
```

```
Requirement already satisfied: plotly in c:\users\hp\anaconda3\lib\site-p
ackages (5.9.0)
Collecting plotly
  Downloading plotly-5.16.1-py2.py3-none-any.whl (15.6 MB)
     ---------------------------------------- 15.6/15.6 MB 4.5 MB/s eta
0:00:00
Requirement already satisfied: packaging in c:\users\hp\anaconda3\lib\sit
e-packages (from plotly) (21.3)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\hp\anaconda3\l
ib\site-packages (from plotly) (8.0.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\hp\an
aconda3\lib\site-packages (from packaging->plotly) (3.0.9)
Installing collected packages: plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 5.9.0
    Uninstalling plotly-5.9.0:
      Successfully uninstalled plotly-5.9.0
Successfully installed plotly-5.16.1
```

In [ ]:

```
titanic = pd.read_csv('titanic passenger list.csv')
titanic.head()
```

Q1: Make dash that prints "Data Science"

In [ ]:

```
app = JupyterDash()
```

In [ ]:

```
app.layout = html.Div([html.H1('Data Science')])
```

In [ ]:

```
app.run_server()
```

Q2: Make dash that takes name input and outpt "Welcome to Data Science dashboard {name}"

In [ ]:

```
app = JupyterDash()

app.layout = html.Div([
    html.H1("Data Science Dashboard"),
    dcc.Input(id="name-input", type="text", placeholder="Enter your name"),
    html.Div(id="output-message")
])

@app.callback(
    Output("output-message", "children"),
    Input("name-input", "value")
)
def update_output(name):
    if name:
        return f"Welcome to Data Science dashboard, {name}!"
    else:
        return "Enter your name above."

app.run_server()
```

Q3: Make Plotly-Dash that takes two inputs (numbers) and output 3 numbers x, y , z= x+y

- hint: use html.Div, dcc.Input(type='number')

In [ ]:

```
app = JupyterDash()

app.layout = html.Div([
    html.H1("Sum Calculator", style={"textAlign": "center", "marginBottom": "20px"}),

    html.Div([
        html.Label("Enter the first number:"),
        dcc.Input(id="number1", type="number", value=0, style={"marginRight": "10px"}),
    ], style={"marginBottom": "15px"}),

    html.Div([
        html.Label("Enter the second number:"),
        dcc.Input(id="number2", type="number", value=0, style={"marginRight": "10px"}),
    ], style={"marginBottom": "20px"}),

    html.Div(id="result", style={"fontSize": "18px", "fontWeight": "bold"})
], style={"maxWidth": "400px", "margin": "auto", "padding": "20px", "border": "1px solic


@app.callback(
    Output("result", "children"),
    Input("number1", "value"),
    Input("number2", "value")
)
def calculate_sum(number1, number2):
    x = number1
    y = number2
    z = x + y
    return f"The sum of {x} and {y} is {z}"

app.run_server()
```

Q4.1: Make JubyterDash with Rangeslider from 1-15 with 6 option

In [ ]:

```
app = JupyterDash()

app.layout = html.Div([
    html.H1("RangeSlider App"),

    dcc.RangeSlider(
        id="range-slider",
        min=1,
        max=15,
        step=1,
        marks={i: str(i) for i in range(1, 16)},
        value=[3, 9]
    ),

    html.Div(id="output")
])

@app.callback(
    Output("output", "children"),
    Input("range-slider", "value")
)
def update_output(value):
    return f"Selected range: {value[0]} - {value[1]}"

app.run_server()
```

Q4.2: add to the previous dash: Label, set default value on rangeslider as 7-10

In [ ]:

```
from dash import Dash, dcc, html, Input, Output
from jupyter_dash import JupyterDash
app = JupyterDash()

app.layout = html.Div([
    html.H1("RangeSlider App"),

    dcc.RangeSlider(
        id="range-slider",
        min=1,
        max=15,
        step=1,
        marks={i: str(i) for i in range(1, 16)},
        value=[7, 10]
    ),

    html.Div(id="output")
])

@app.callback(
    Output("output", "children"),
    Input("range-slider", "value")
)
def update_output(value):
    return f"Selected range: {value[0]} - {value[1]}"

app.run_server()
```

Q5: Make 3 different Dcc types (Dropdown - Checkbox...) in one dash

In [2]:

```
app = JupyterDash()

def create_dropdown():
    return html.Div([
        html.Label("Select an option:"),
        dcc.Dropdown(
            id="dropdown",
            options=[
                {'label': 'Option 1', 'value': 'option1'},
                {'label': 'Option 2', 'value': 'option2'},
                {'label': 'Option 3', 'value': 'option3'}
            ],
            value='option1'
        ),
    ], style={'margin': '20px'})

def create_checkboxes():
    return html.Div([
        html.Label("Select options:"),
        dcc.Checklist(
            id="checkboxes",
            options=[
                {'label': 'Checkbox 1', 'value': 'checkbox1'},
                {'label': 'Checkbox 2', 'value': 'checkbox2'},
                {'label': 'Checkbox 3', 'value': 'checkbox3'}
            ],
            value=['checkbox1']
        ),
    ], style={'margin': '20px'})

def create_radioitems():
    return html.Div([
        html.Label("Select one option:"),
        dcc.RadioItems(
            id="radioitems",
            options=[
                {'label': 'Radio 1', 'value': 'radio1'},
                {'label': 'Radio 2', 'value': 'radio2'},
                {'label': 'Radio 3', 'value': 'radio3'}
            ],
            value='radio1'
        ),
    ], style={'margin': '20px'})

app.layout = html.Div([
    html.H1("Dash Components Example", style={'textAlign': 'center'}),
    create_dropdown(),
    create_checkboxes(),
    create_radioitems(),

    html.Div(id="output", style={'textAlign': 'center', 'margin': '40px'})
])

@app.callback(
    Output("output", "children"),
    [
        Input("dropdown", "value"),
        Input("checkboxes", "value"),
        Input("radioitems", "value")
```

```
    ]
)
def update_output(dropdown_value, checkboxes_value, radioitems_value):
    return html.Div([
        html.P(f"Dropdown selected: {dropdown_value}"),
        html.P(f"Checkboxes selected: {', '.join(checkboxes_value)}"),
        html.P(f"RadioItem selected: {radioitems_value}")
    ])

app.run_server()
```

```
C:\Users\HP\anaconda3\lib\site-packages\dash\dash.py:516: UserWarning: Ju
pyterDash is deprecated, use Dash instead.
See https://dash.plotly.com/dash-in-jupyter (https://dash.plotly.com/dash
-in-jupyter) for more details.
  warnings.warn(
```

Dash Components Example

Select an option:

Option 3 ×

Select options:

- [x] Checkbox 1
- [x] Checkbox 2
- [ ] Checkbox 3

Select one option:

- (•) Radio 1
- ( ) Radio 2
- ( ) Radio 3

Dropdown selected: option3

Checkboxes selected: checkbox1, checkbox2

RadioItem selected: radio1

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

Q6: Make Plotly chart, then add it to a plotly_dash

In [ ]:

```
df = titanic

# Create a Plotly chart using Plotly Express
fig = px.line(df, x='x', y='y', title='Sample Line Chart')

# Create the Plotly Dash app
app = JupyterDash(__name__)

# Define the app layout
app.layout = html.Div([
    html.H1("Plotly Dash Chart Example", style={'textAlign': 'center'}),
    dcc.Graph(figure=fig)
])

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

In [ ]:

```
fig = px.scatter(titanic, x='age', y='fare', color='survived', title='Titanic Dataset')

app = JupyterDash()

app.layout = html.Div([
    html.H1("Titanic Dataset Scatter Plot", style={'textAlign': 'center'}),
    dcc.Graph(figure=fig)
])

app.run_server()
```

Q7: From our data make plotly-dash with suitble chart, and Dropdown.

- Dropdown should contain at least 3 options, each one represents different column.
- Add label to dropdown.
- The chart data should change depending on the selected dropdown option.

In [7]:

```
import pandas as pd
df = pd.read_csv("titanic passenger list.csv")
```

In [ ]:

```
app = JupyterDash()

dropdown_options = [
    {'label': 'Age', 'value': 'age'},
    {'label': 'Fare', 'value': 'fare'},
    {'label': 'Number of Siblings/Spouses Aboard', 'value': 'sibsp'},
]

app.layout = html.Div([
    html.H1("Titanic Dataset Chart with Dropdown", style={'textAlign': 'center'}),
    dcc.Dropdown(id='column-dropdown', options=dropdown_options, value='age'),
    dcc.Graph(id='chart-graph')
])

@app.callback(
    Output('chart-graph', 'figure'),
    Input('column-dropdown', 'value')
)
def update_chart(selected_column):
    fig = px.histogram(df, x=selected_column, title=f'Distribution of {selected_column.c
    return fig

app.run_server()
```

Q8: add 2 more charts to the prevoius Ploty-Dash

In [ ]:

```
app = JupyterDash()

dropdown_options = [
    {'label': 'Age', 'value': 'age'},
    {'label': 'Fare', 'value': 'fare'},
    {'label': 'Number of Siblings/Spouses Aboard', 'value': 'sibsp'},
]

app.layout = html.Div([
    html.H1("Titanic Dataset Charts with Dropdown", style={'textAlign': 'center'}),
    dcc.Dropdown(id='column-dropdown', options=dropdown_options, value='age'),
    dcc.Graph(id='histogram-graph'),
    dcc.Graph(id='boxplot-graph'),
    dcc.Graph(id='scatter-graph')
])

@app.callback(
    Output('histogram-graph', 'figure'),
    Input('column-dropdown', 'value')
)
def update_histogram(selected_column):
    fig = px.histogram(df, x=selected_column, title=f'Distribution of {selected_column.c
    return fig

@app.callback(
    Output('boxplot-graph', 'figure'),
    Input('column-dropdown', 'value')
)
def update_boxplot(selected_column):
    fig = px.box(df, x='survived', y=selected_column, points="all", title=f'Box Plot of
    return fig

@app.callback(
    Output('scatter-graph', 'figure'),
    Input('column-dropdown', 'value')
)
def update_scatter(selected_column):
    fig = px.scatter(df, x='age', y=selected_column, color='survived', title=f'Scatter P
    return fig

app.run_server()
```

Q9: Search for "Plotly Gallery" and fetch for a graph and use it with our data.

- Note: You need to put the resourse addres of fetched your graph.

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   pclass     1305 non-null   float64
 1   survived   1305 non-null   float64
 2   name       1305 non-null   object 
 3   sex        1305 non-null   object 
 4   age        1046 non-null   float64
 5   sibsp      1305 non-null   float64
 6   parch      1305 non-null   float64
 7   ticket     1305 non-null   object 
 8   fare       1304 non-null   float64
 9   cabin      292 non-null    object 
 10  embarked   1303 non-null   object 
 11  boat       484 non-null    object 
 12  body       121 non-null    float64
 13  home.dest  741 non-null    object 
dtypes: float64(7), object(7)
memory usage: 143.3+ KB
```

In [13]:

```
df  = df.dropna(subset=['pclass', 'sex'])
```

box plots [Read more (https://plotly.com/python/box-plots/)](https://plotly.com/python/box-plots/)

In [16]:

```
df.tail(5)
```

Out[16]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	em
1301	3.0	0.0	Youseff, Mr. Gerious	male	46.0	0.0	0.0	2628	7.2250	NaN	
1304	3.0	0.0	Zabour, Miss. Hileni	female	15.0	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	male	27.0	0.0	0.0	2656	7.2250	NaN	
1307	3.0	0.0	Zakarian, Mr. Ortin	male	27.0	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	male	29.0	0.0	0.0	315082	7.8750	NaN	

In [ ]:

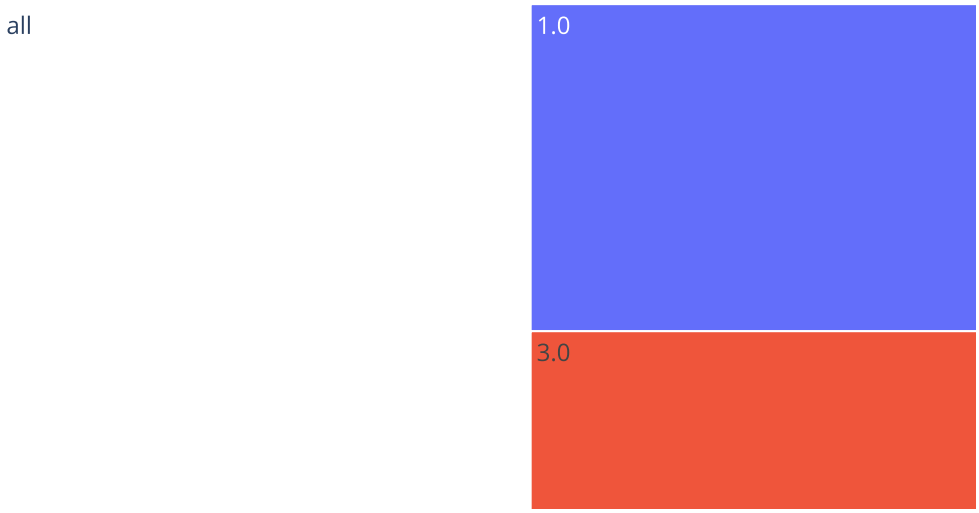
```
#from dash import jupyter_dash

#jupyter_dash.default_mode="external"
```

In [ ]:

```
#app.run(jupyter_server_url="<your-url>")
```

In [22]:

```
fig = px.icicle(df, path=[px.Constant("all"),'pclass','sex'],
                values= 'survived')
fig.show()
```



Insight

Females had a higher survival rate compared to males

Source for icicle chart [Read more (https://plotly.com/python/icicle-charts/)](https://plotly.com/python/icicle-charts/)

Challenge 1.1: Create Plotly dash with two charts depends on different columns of the data

- First Chart : Pie
- Secound Chart: is by your Choice (Not Pie)

In [ ]:

```
import matplotlib.pyplot as plt
survival_freq = df['survived'].value_counts()

# Create a bar chart
survival_freq.plot(kind='bar')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.xticks([0, 1], ['Did Not Survive', 'Survived'])
plt.title('Survival Count')
plt.show()
```

Challenge 1.2: Make Hovering over a slice of the pie changes the data for the second chart depending on what the slice is.

In [ ]:

```
app = JupyterDash()

app.layout = html.Div([
    html.H1("Titanic Dataset Analysis"),

    dcc.Graph(id='pie-chart'),

    dcc.Graph(id='bar-chart'),
])

@app.callback(
    [Output('pie-chart', 'figure'), Output('bar-chart', 'figure')],
    [Input('pie-chart', 'clickData')]  # Use 'clickData' to get the selected slice
)
def update_charts(click_data):
    if click_data is None:
        selected_class = None
    else:
        selected_class = click_data['points'][0]['label']

    pie_fig = px.pie(
        titanic_df,
        names='pclass',
        title='Distribution of Passenger Classes'
    )

    if selected_class is None:
        filtered_df = titanic_df
        bar_title = 'Distribution of Ages'
    else:
        filtered_df = titanic_df[titanic_df['pclass'] == selected_class]
        bar_title = f'Distribution of Ages for Class {selected_class}'

    bar_fig = px.histogram(
        filtered_df,
        x='age',
        nbins=10,
        title=bar_title
    )

    return pie_fig, bar_fig

app.run_server()
```