Matplotlib-Seaborn Handbook for Ai

Ashraful Islam Mahi

Full Stack Python Developer | ML & Robotics Enthusiast | Founder & CEO, PytronLab

Introduction to Matplotlib for Plotting

What is Matplotlib?

- It's a foundational python library for creating static, dynamic & animated blocks. It allows us for the detailed customization the blocks.

0. Install *Matplotlib*:

In [ ]:
```
%pip install matplotlib
```

1. Import *Matplotlib*:

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
```

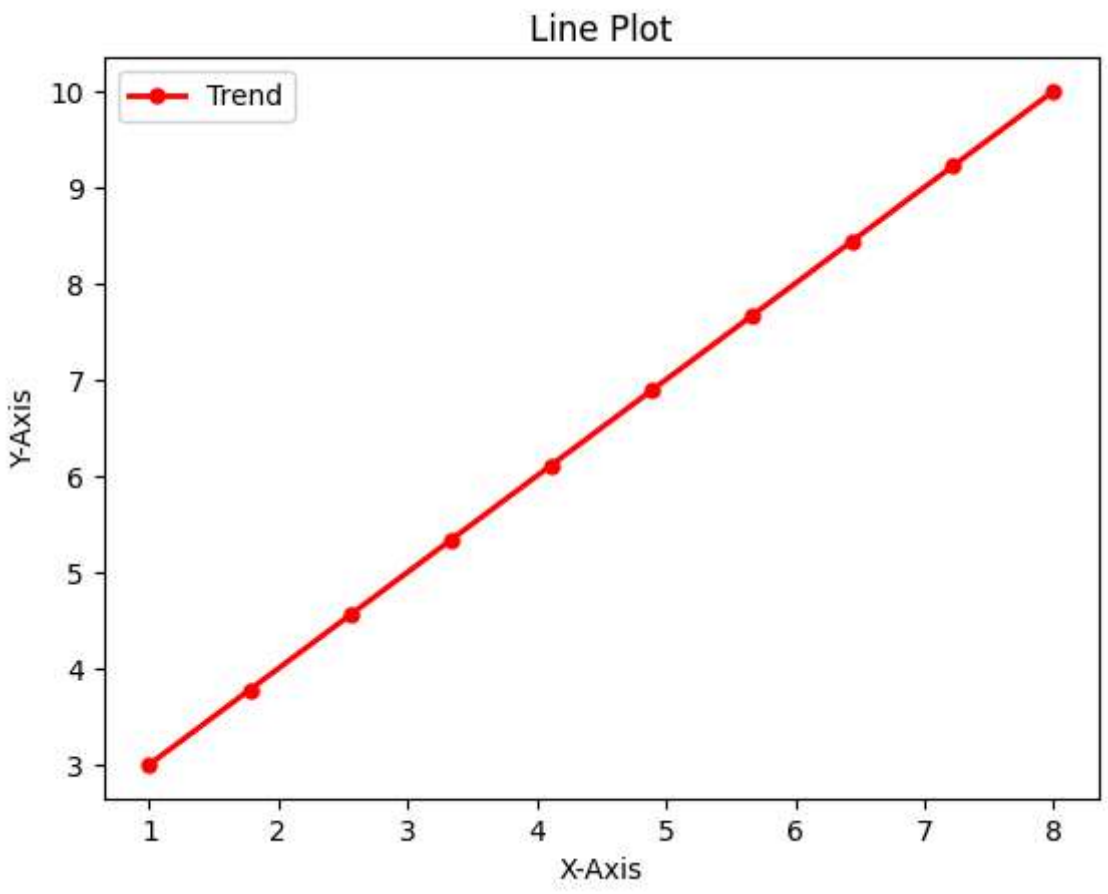
Mostly Used Plots:

- Line Plot
- Bar Chart
- Pie Chart
- Histogram
- Scatter Plot
- Box Plot

1. Ploting *line* curve:

Example:

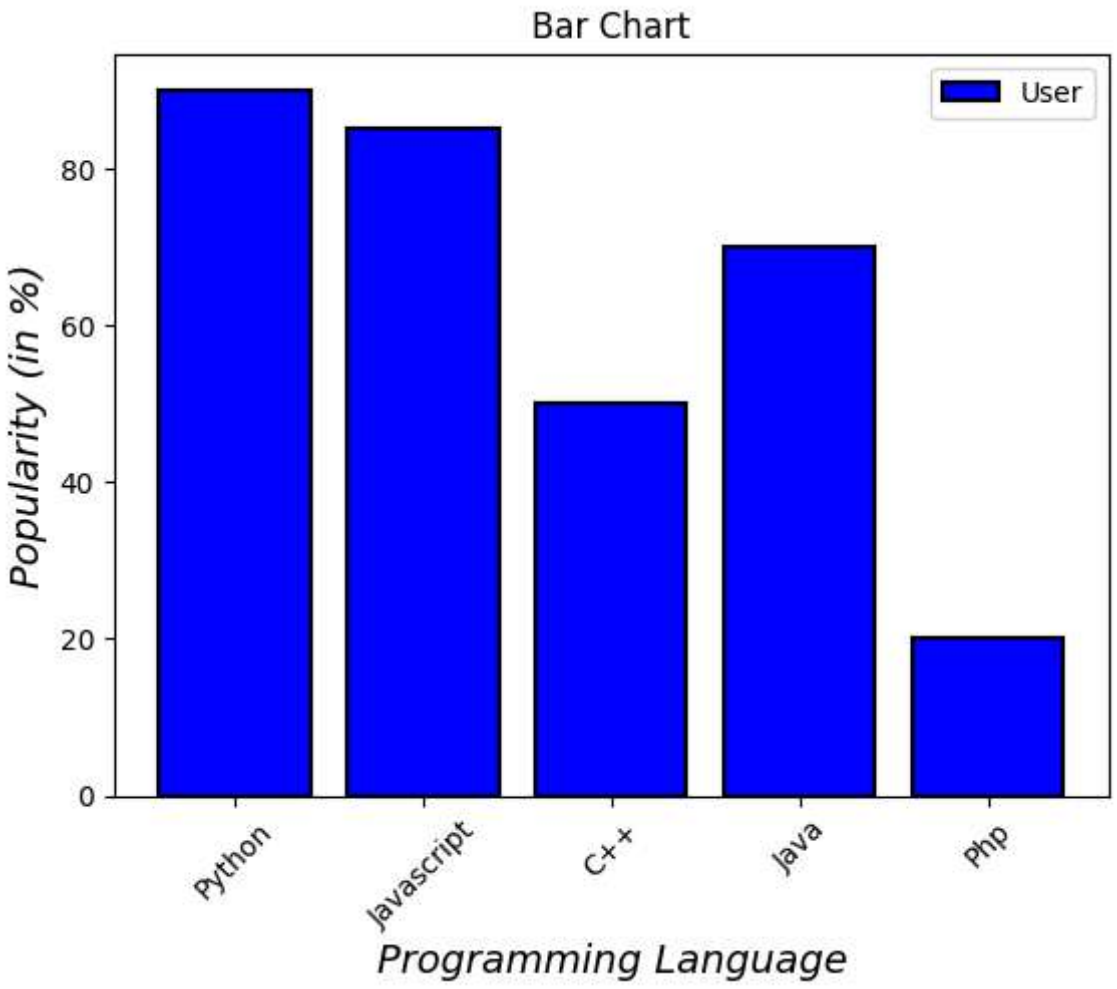
In [3]:
```
# Making Arrays
points = 10
x = np.linspace(1,8,points)
y = np.linspace(3,10,points)



# plotting function
plt.plot(x,y,color = "red",ls = '-',lw = 2, marker ='o',ms = 5) #ls = linestyle,lw = line weight ms = marker size

#Styling functions
plt.title("Line Plot")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.legend(["Trend"])

#output function
plt.show()
```



2. Plotting *Bar* chart:

Example:

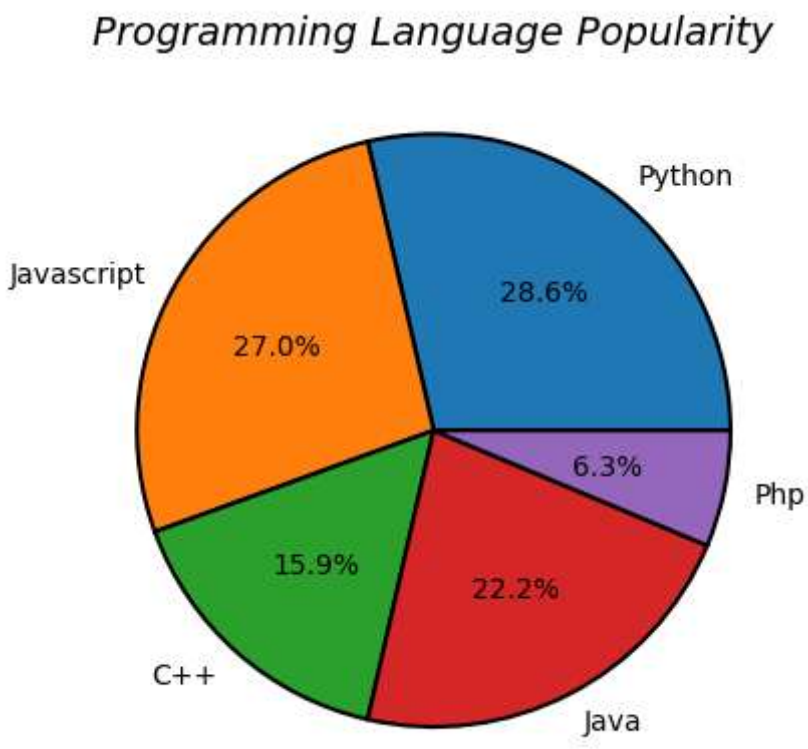
In [3]:
```
#Creating Arrays
languages = ['Python','Javascript','C++', 'Java','Php']
popularities = [90, 85, 50, 70, 20]

# bar function
plt.bar(languages, popularities, color = 'blue',align='center',edgecolor = 'black',lw=1.5)

#Styling functions
plt.xlabel('Programming Language',fontsize=14, fontstyle='italic')
plt.ylabel('Popularity (in %)',fontsize=14, fontstyle='italic')
plt.xticks(rotation = 45)
plt.title("Bar Chart")
plt.legend(['User'])

#Output function
plt.show()
```



3. Plotting *Pie Chart*:

Example:

In [3]:
```
# Creating Arrays
languages = ['Python', 'Javascript', 'C++', 'Java', 'Php']
popularities = [90, 85, 50, 70, 20]

# Create a pie chart
plt.pie(popularities, labels=languages, autopct='%1.1f%%',wedgeprops={'edgecolor': 'black', 'linewidth': 1.5})

# Styling functions
plt.title("Programming Language Popularity", fontsize=14, fontstyle='italic')
# plt.legend(languages, loc="upper right")

# Output function
plt.show()
```



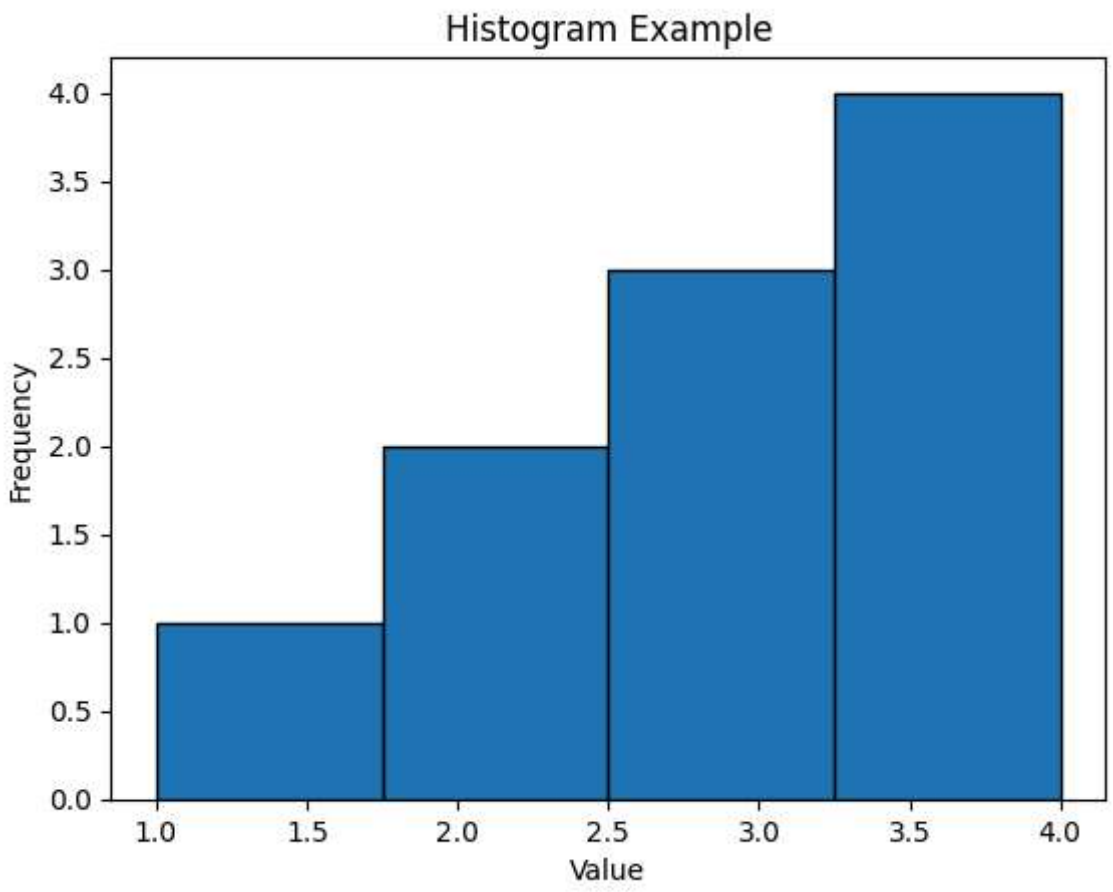
4. Plotting *Histogram*:

- A histogram in matplotlib is a type of bar chart that shows how often different values (or ranges of values) appear in your data. Think of it as a way to "group" your numbers into bins and then count how many numbers fall into each bin.

Example-1:

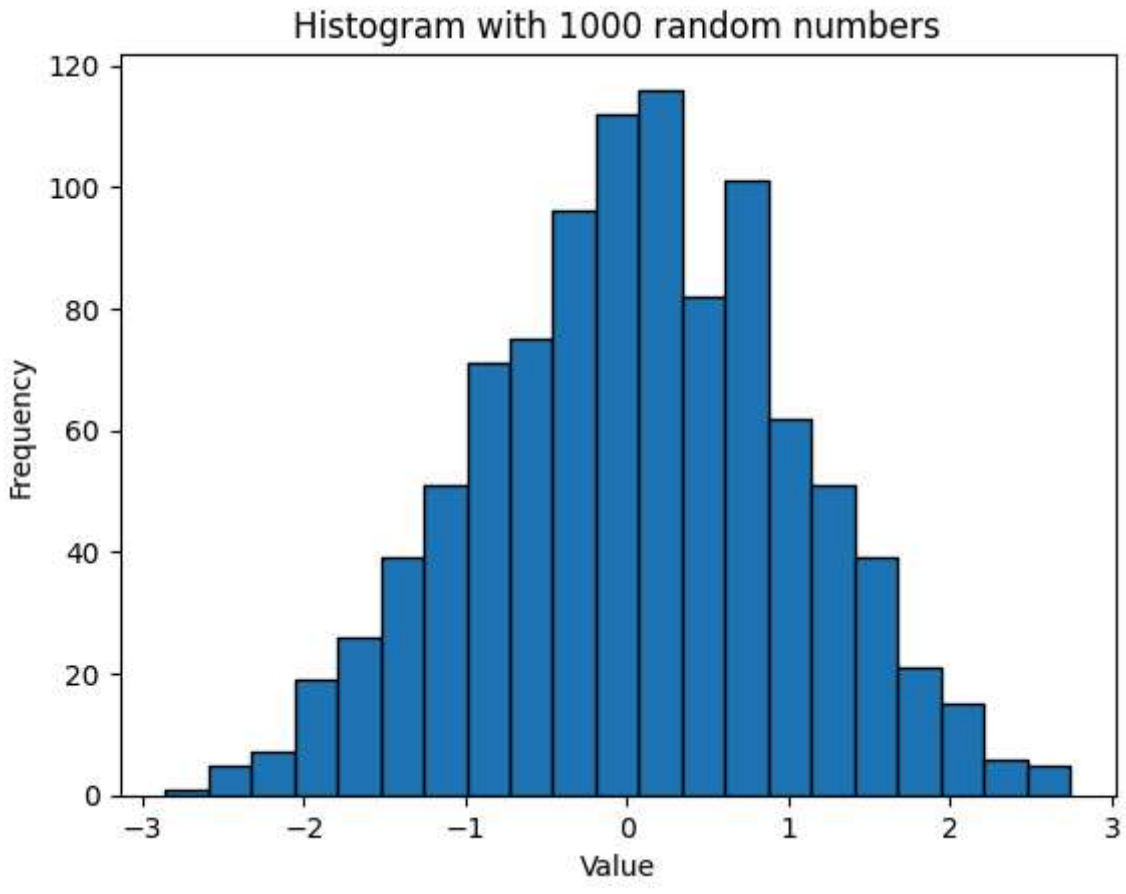
In [32]:
```
import matplotlib.pyplot as plt

# Sample data
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
```

```
# Create a histogram
plt.hist(data, bins=4, edgecolor='black')  # 'bins' defines how many buckets you want
plt.title('Histogram Example')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



Example-2:

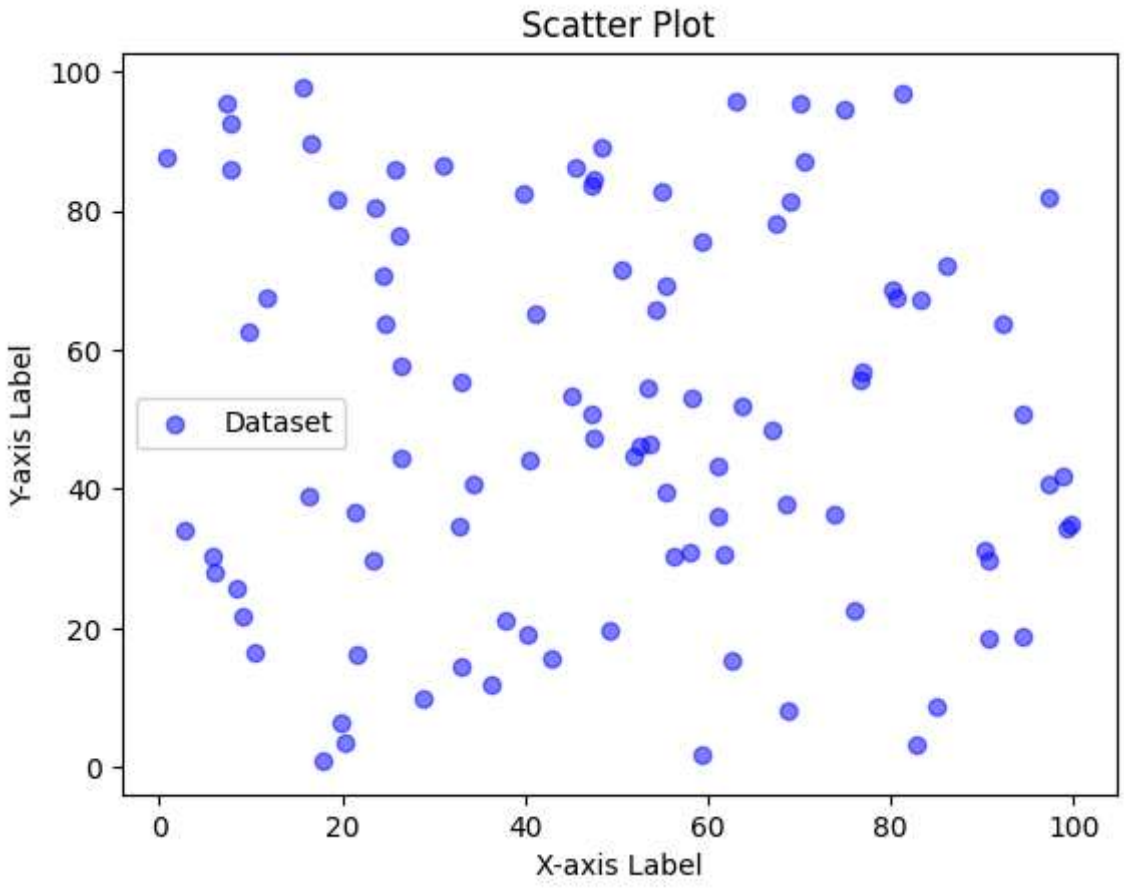
In [23]:
```
points = 1000

# Generate random data (e.g., from a normal distribution)
data = np.random.randn(points)

# Create a histogram with dynamic binning
plt.hist(data, bins='auto', edgecolor='black')

# Add title and labels
plt.title(f'Histogram with {points} random numbers')
plt.xlabel('Value')
plt.ylabel('Frequency')

# Display the plot
plt.show()
```



5. Plotting *Scatter* Plot:

Example:

In [ ]:
```
points = 100

x = np.random.random(points) * 100
y = np.random.random(points) * 100

# scatter function
plt.scatter(x,y,color="#00f",alpha=0.5)  # alpha = transparency


plt.title("Scatter Plot")
plt.xlabel("X-axis Label")
plt.ylabel("Y-axis Label")
plt.legend(["Dataset"])
plt.show()
```

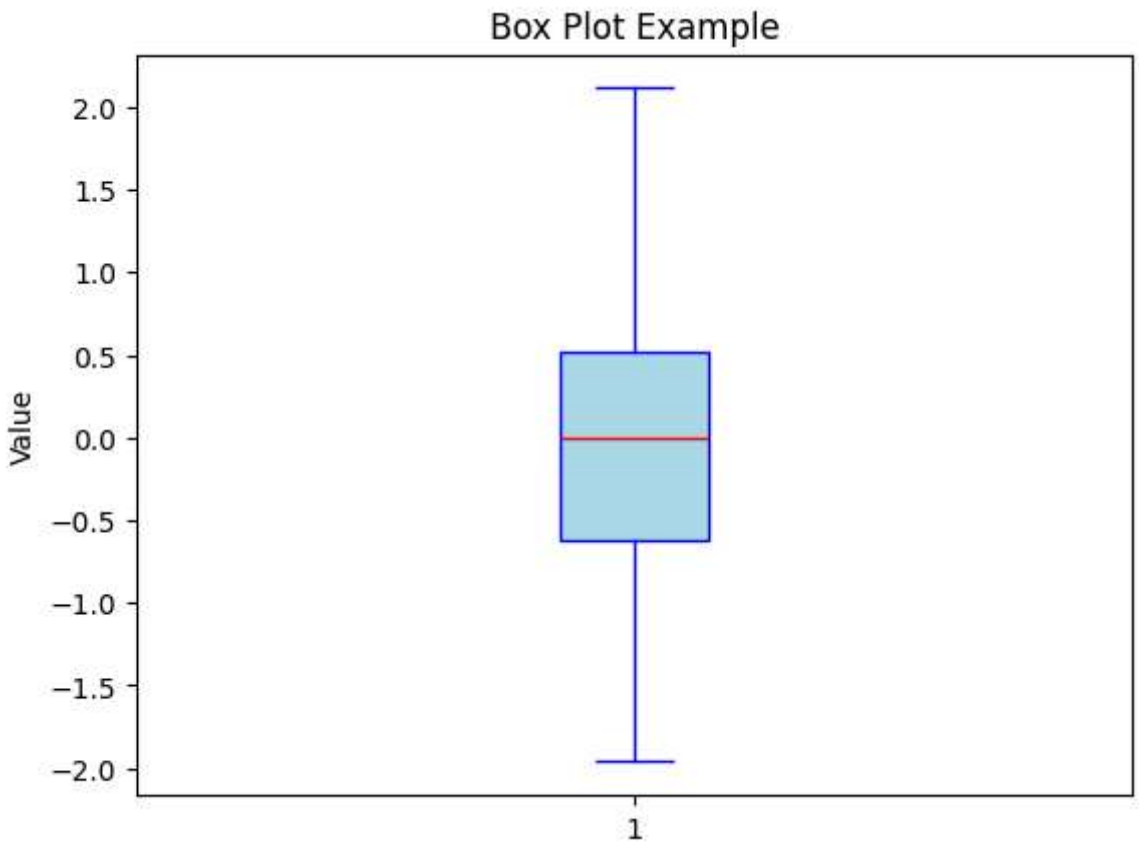


6. Plotting *Box* Plot:

- A box plot (or box-and-whisker plot) is a convenient way to visualize the distribution of a dataset. It shows the median, quartiles, and potential outliers in your data.

Example:

In [33]:
```
import matplotlib.pyplot as plt
import numpy as np

# Generate a sample dataset: 100 random values from a normal distribution
data = np.random.randn(100)

# Create a box plot
plt.boxplot(data, patch_artist=True, boxprops=dict(facecolor='lightblue', color='blue'),
            medianprops=dict(color='red'), whiskerprops=dict(color='blue'),
            capprops=dict(color='blue'), flierprops=dict(markerfacecolor='green', marker='o', markersize=6, linestyle='none'))

# Add title and labels
plt.title("Box Plot Example")
plt.ylabel("Value")

# Display the plot
plt.show()
```

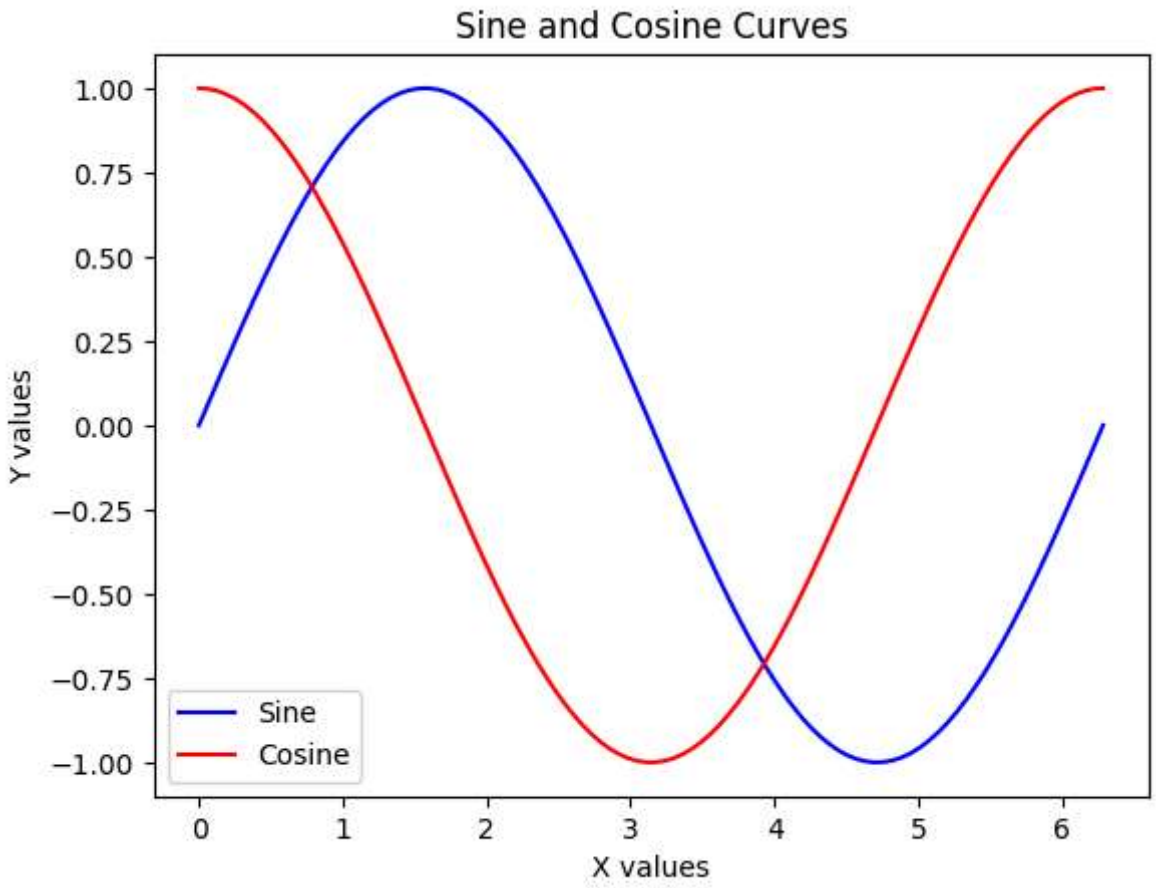


Some Special Tools:

- Multiple Plots
- Multiple Figures
- Matrix Figures
- Subplots

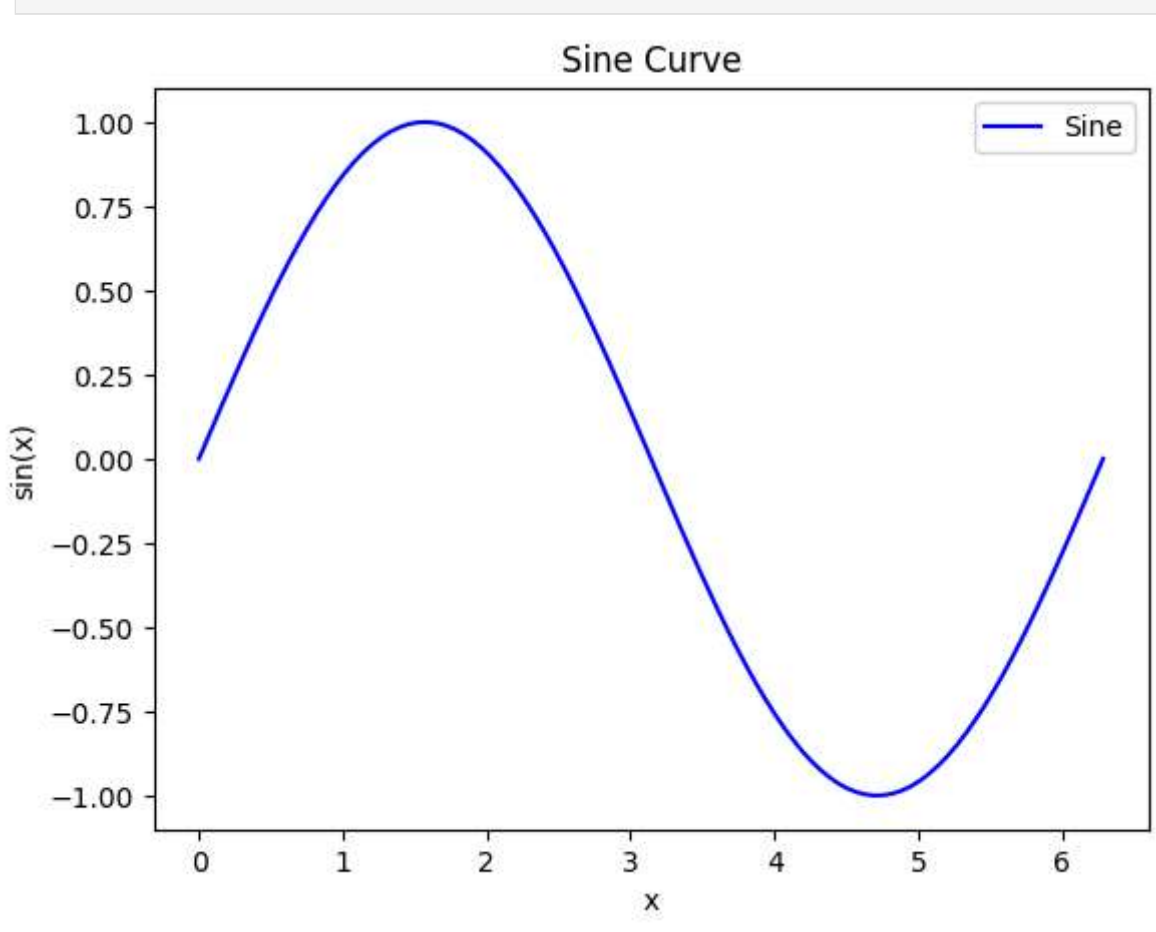
1. Multiple Plots:

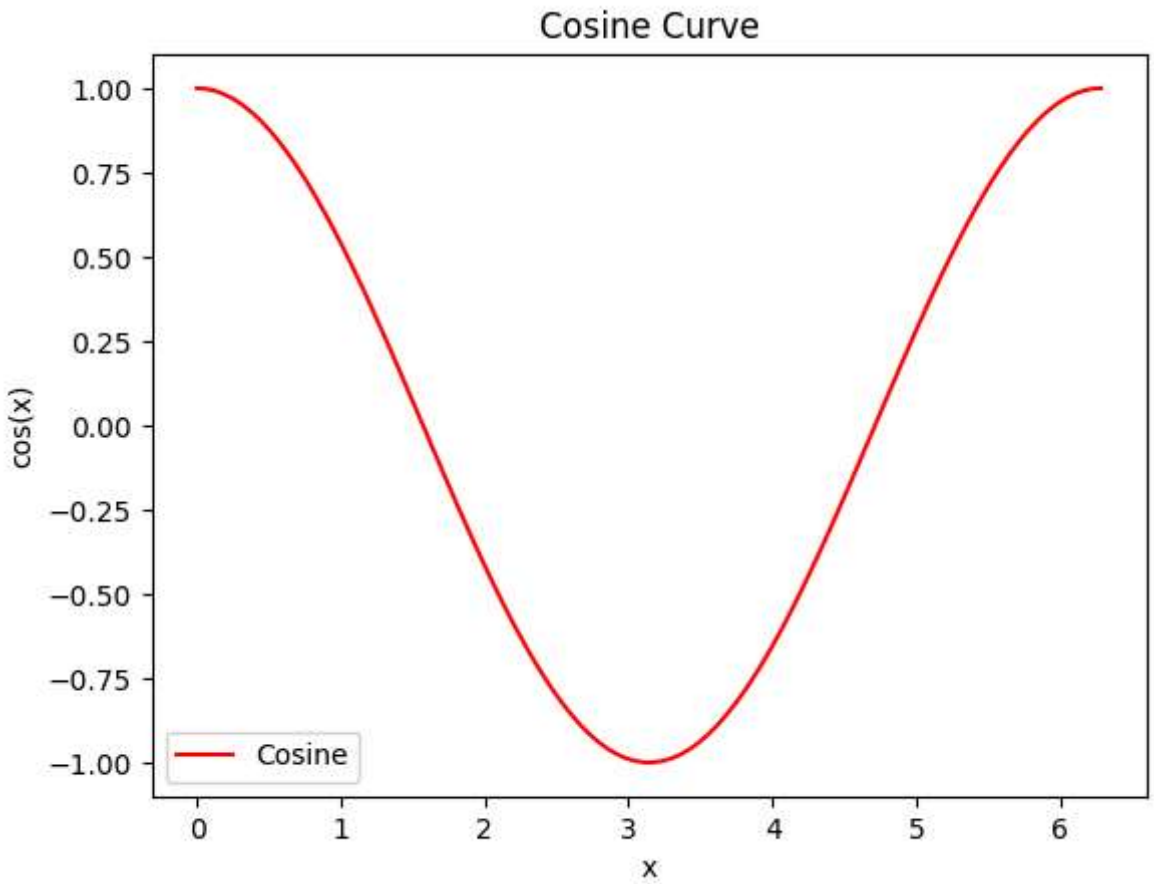
In [5]:
```
# Generate x values from 0 to 2π
x = np.linspace(0, 2 * np.pi, 100)

# Calculate y values for sine and cosine
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the sine curve with a blue line
plt.plot(x, y_sin, label='Sine', color='blue')

# Plot the cosine curve with a red line
plt.plot(x, y_cos, label='Cosine', color='red')

# Add title and labels
plt.title("Sine and Cosine Curves")
plt.xlabel("X values")
plt.ylabel("Y values")

# Add legend to distinguish the curves
plt.legend()

# Display the plot
plt.show()
```



2. Multiple Figures:

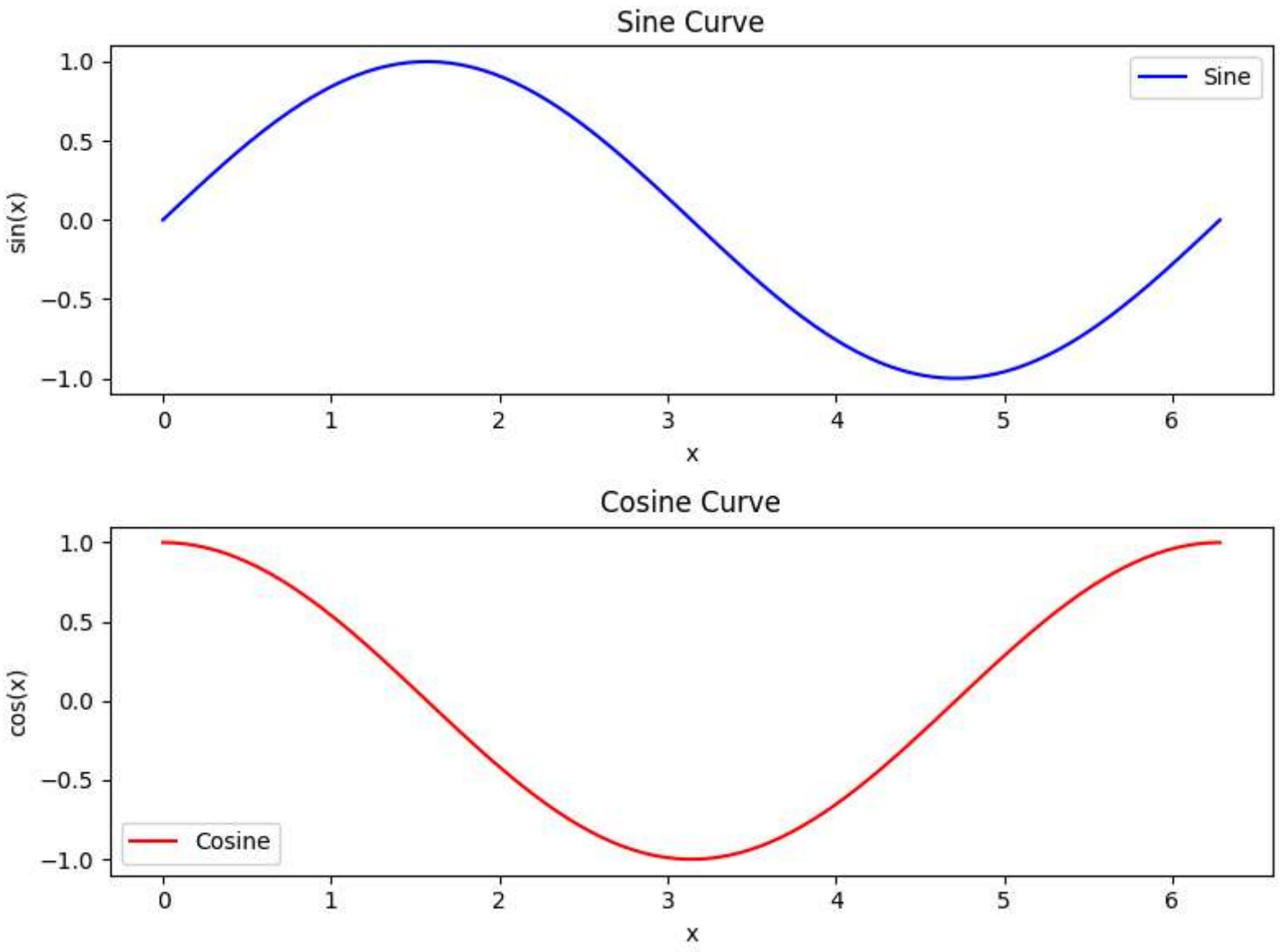
In [6]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Generate x values from 0 to 2π
x = np.linspace(0, 2 * np.pi, 100)

# Calculate y values for sine and cosine
y_sin = np.sin(x)
y_cos = np.cos(x)

# Create the first figure for the sine curve
plt.figure()  # Start a new figure
plt.plot(x, y_sin, label='Sine', color='blue')
plt.title("Sine Curve")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.legend()

# Create the second figure for the cosine curve
plt.figure()  # Start another new figure
plt.plot(x, y_cos, label='Cosine', color='red')
plt.title("Cosine Curve")
plt.xlabel("x")
plt.ylabel("cos(x)")
plt.legend()

# Display both figures
plt.show()
```





3. Subplots:

In [ ]:
```python
# Generate x values from 0 to 2π
x = np.linspace(0, 2 * np.pi, 100)

# Calculate y values for sine and cosine
y_sin = np.sin(x)
y_cos = np.cos(x)

# Create a figure with a 2x1 grid of subplots
plt.figure(figsize=(8, 6))

# First subplot for the sine curve
plt.subplot(2, 1, 1)
plt.plot(x, y_sin, label='Sine', color='blue')
plt.title("Sine Curve")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.legend()

# Second subplot for the cosine curve
plt.subplot(2, 1, 2)
plt.plot(x, y_cos, label='Cosine', color='red')
plt.title("Cosine Curve")
plt.xlabel("x")
plt.ylabel("cos(x)")
plt.legend()

# Adjust layout for better spacing between subplots
plt.tight_layout()

# Display the figure with both subplots
plt.show()
```



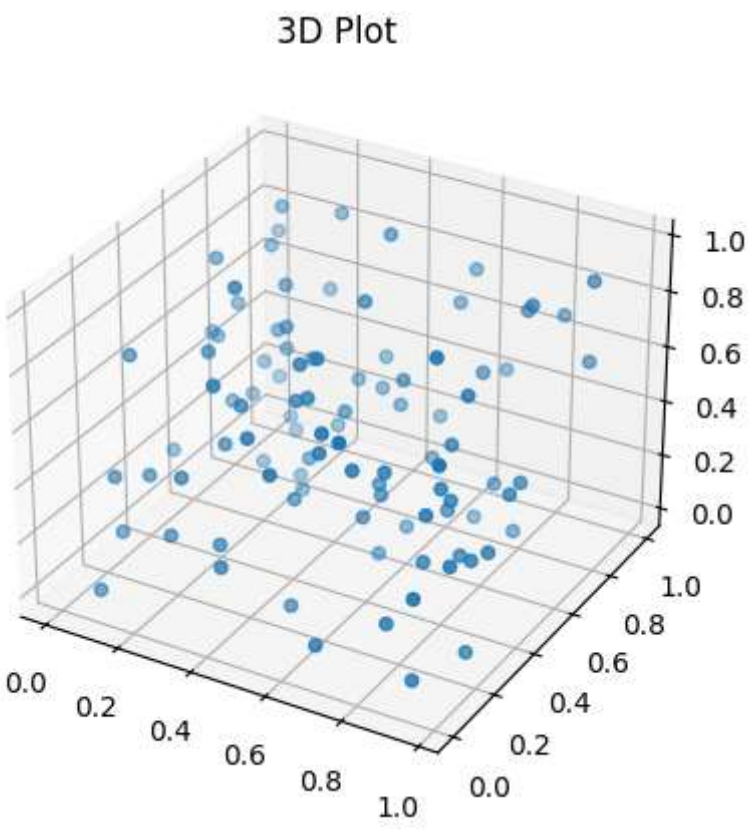
Advanced Applications of Matplotlib:

- 3D Plotting
- Animations

1.Plotting *3D*:

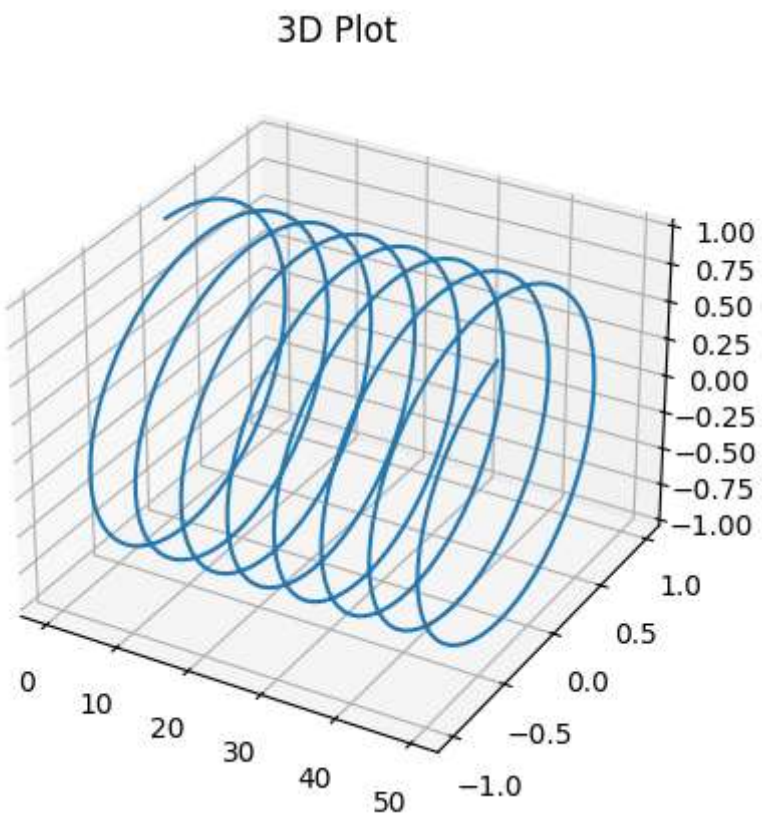
Example-1:

In [ ]:
```
# Create a 3D axes object
ax = plt.axes(projection='3d')

# Generate random data for x, y, and z axes
x = np.random.random(100)
y = np.random.random(100)
z = np.random.random(100)

# Create a scatter plot in 3D
ax.scatter(x, y, z)

# Set the title for the 3D plot
ax.set_title("3D Plot")

# Display the plot
plt.show()
```



Example-2:

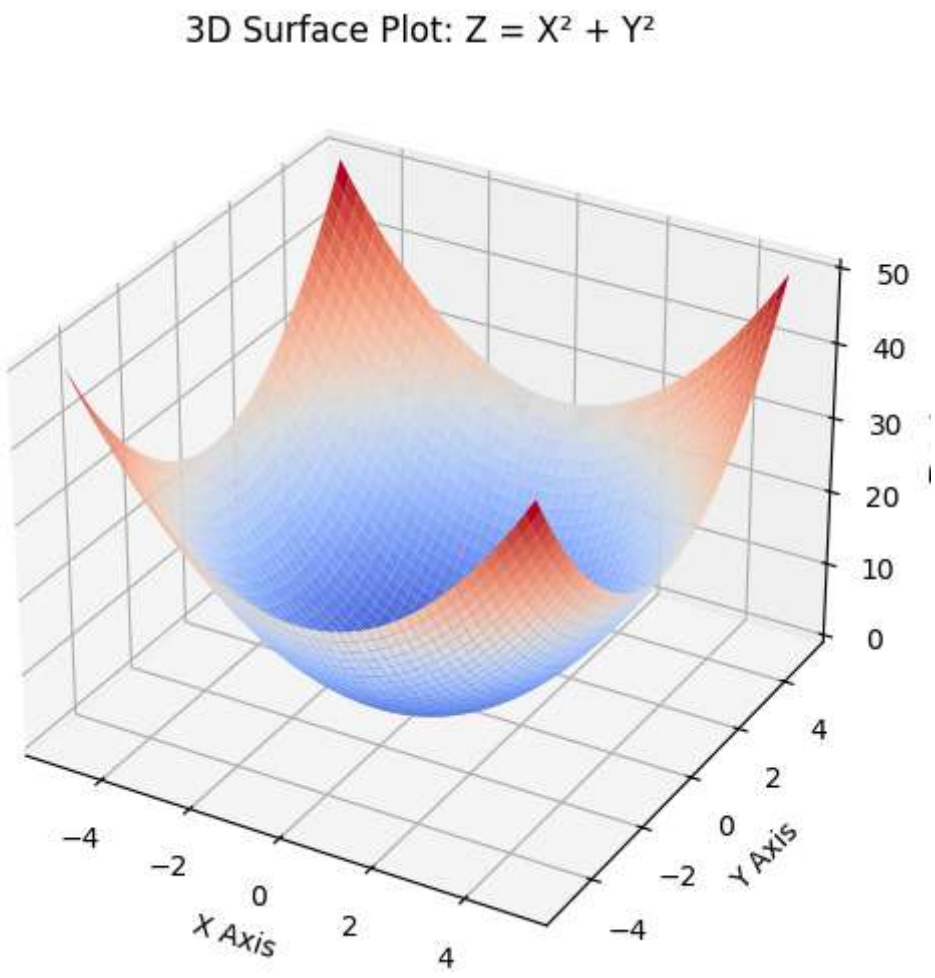
```
In [11]: # Create a 3D axes object
ax = plt.axes(projection='3d')

# Generate random data for x, y, and z axes
x = np.arange(0, 50, 0.1)
y = np.sin(x)
z = np.cos(x)

# Create a line plot in 3D
ax.plot(x, y, z)

# Set the title for the 3D plot
ax.set_title("3D Plot")

# Display the plot
plt.show()
```



Example-3:

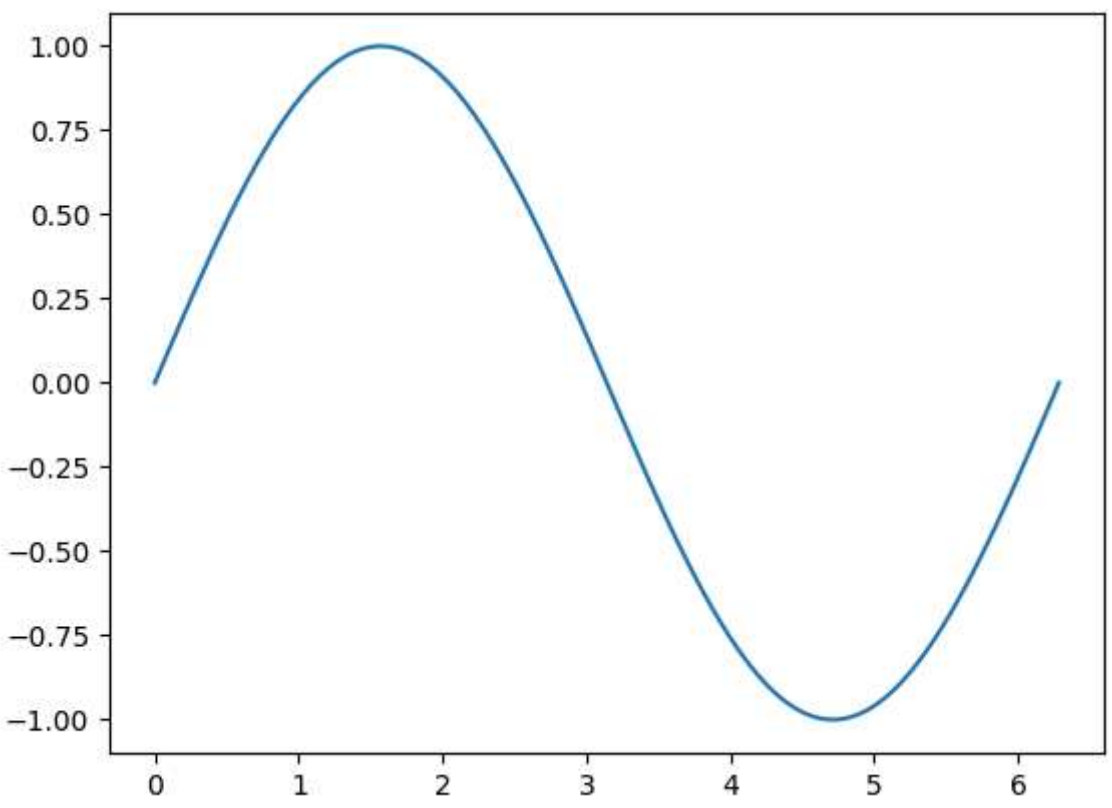
```
In [13]: # Create x and y values
x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)

# Create a meshgrid for x and y
X, Y = np.meshgrid(x, y)

# Compute Z as a simple function of X and Y (paraboloid)
Z = X**2 + Y**2

# Create a figure and add a 3D subplot
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
```

```
# Plot the surface with a colormap
surf = ax.plot_surface(X, Y, Z, cmap='coolwarm')

# Set title and labels
ax.set_title("3D Surface Plot: Z = X² + Y²")
ax.set_xlabel("X Axis")
ax.set_ylabel("Y Axis")
ax.set_zlabel("Z Axis")

# Display the plot
plt.show()
```



2.Plotting *Animation*:

In [4]:
```
import matplotlib.animation as animation

# Create a figure and axis
fig, ax = plt.subplots()
x = np.linspace(0, 2 * np.pi, 100)
line, = ax.plot(x, np.sin(x))  # initial sine wave

# Function to update the sine wave for each frame
def update(frame):
    # Update the y-data with a shifting sine wave
    line.set_ydata(np.sin(x + frame / 10.0))
    return line,

# Create the animation: update every 50 ms for 100 frames
ani = animation.FuncAnimation(fig, update, frames=100, interval=50, blit=True)

plt.show()
```



Introduction to Seaborn for Advanced Visualization

What is Seaborn?

- Built on the top of Matplotlib, Seaborn was born to carry out more customization to matplotlib. It provides high level interfaces for statistical graphs.

0. Installing *Seaborn*:

In [ ]:
```
%pip install seaborn
```

1. Import *Seaborn*:

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Importing Seaborn Builtin dataset:

In [2]:
```
sns.get_dataset_names()
```

Out[2]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

3. Loading a dataset:

In [3]:
```
tips = sns.load_dataset('tips')
titanic = sns.load_dataset('titanic')
iris = sns.load_dataset('iris')
```

4. Visualizing a dataset:

In [30]:
```
tips
```

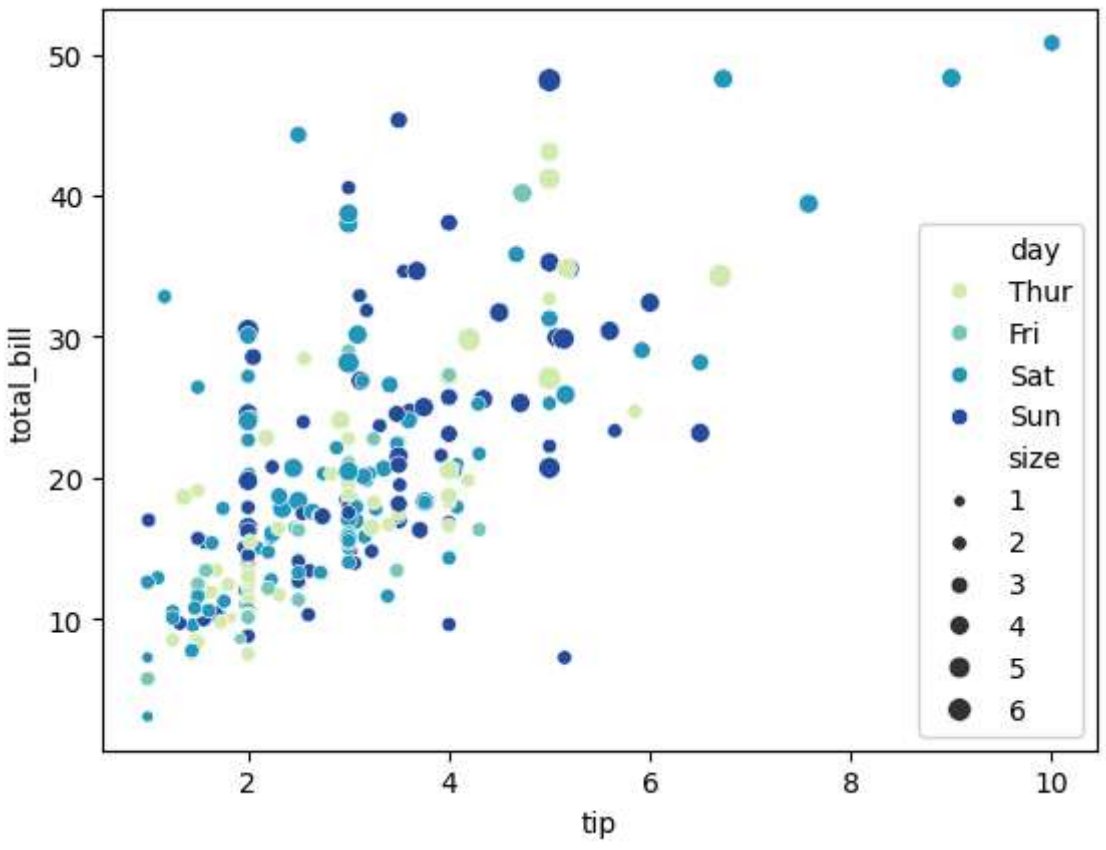
Out[30]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [15]:
```
sns.scatterplot(x = 'tip', y = 'total_bill', data = tips, hue = 'day', size='size',palette="YlGnBu")
```

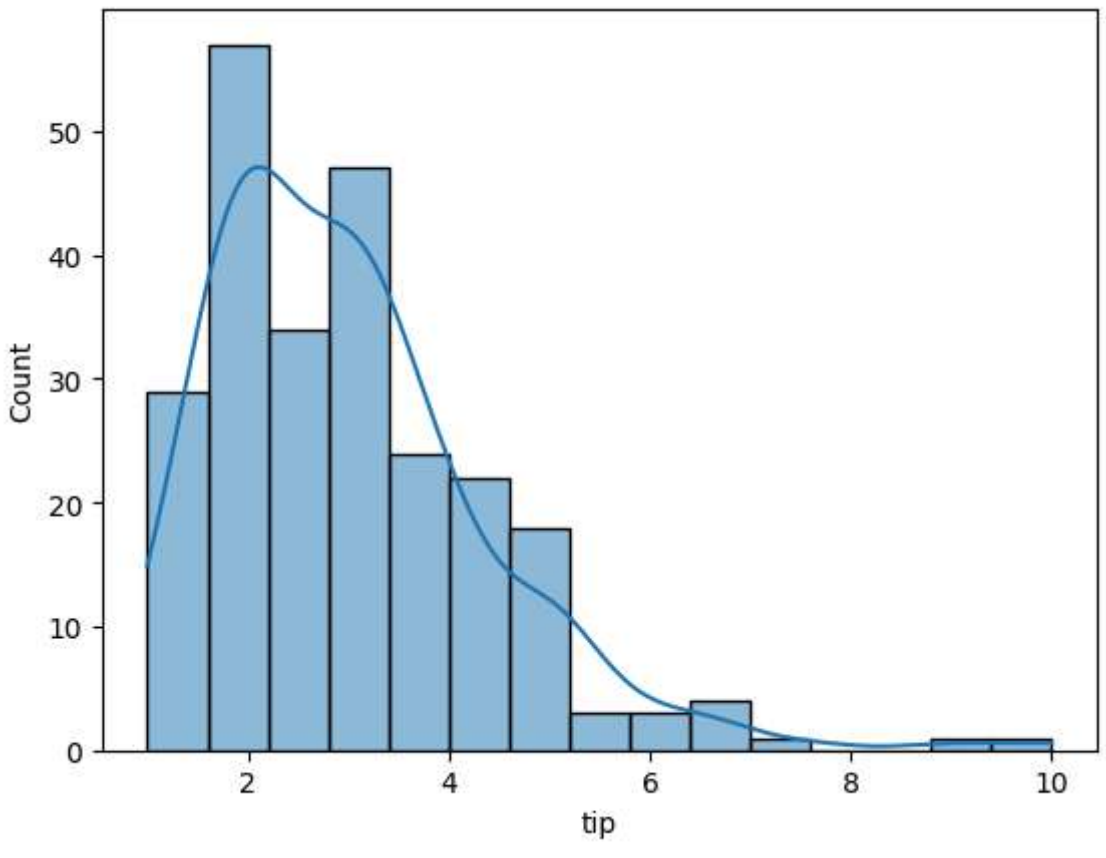
Out[15]: <Axes: xlabel='tip', ylabel='total_bill'>



5. Seabom *Histogram*:

In [18]:
```
sns.histplot(tips['tip'], kde = True, bins = 15)
```

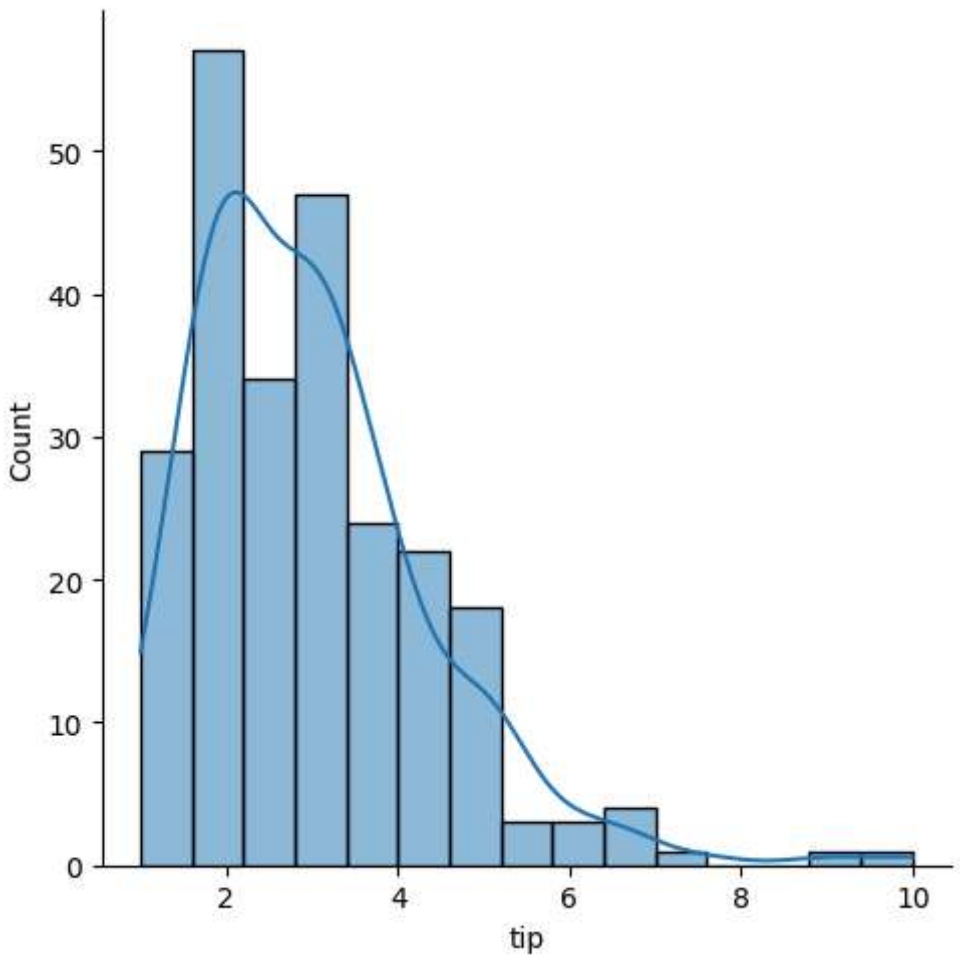
Out[18]: <Axes: xlabel='tip', ylabel='Count'>



6. Seabom *Distribution* Plot:

In [21]:
```
sns.displot(tips['tip'], kde = True, bins = 15)
```

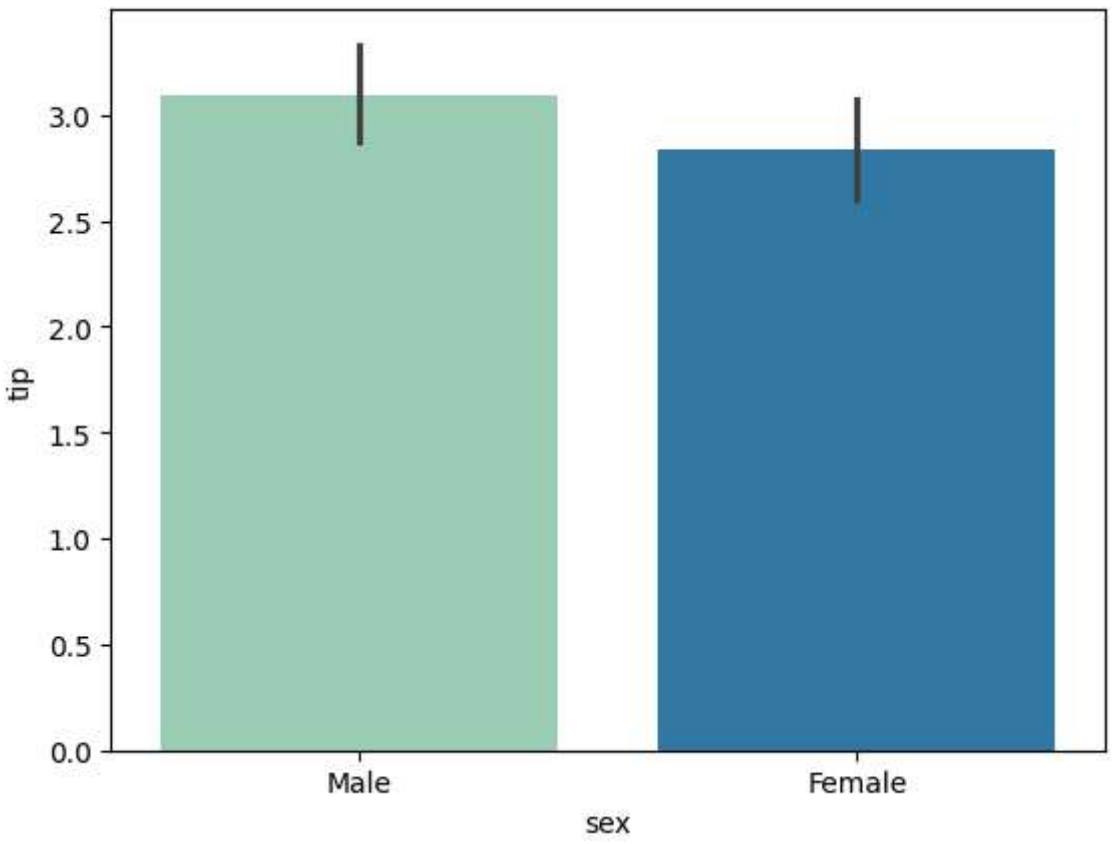
Out[21]: <seaborn.axisgrid.FacetGrid at 0x21f177a7890>



7. Seabom *Bar* Plot:

```
In [25]: sns.barplot(x = 'sex', y = 'tip',hue = 'sex', data = tips, palette = 'YlGnBu')
```

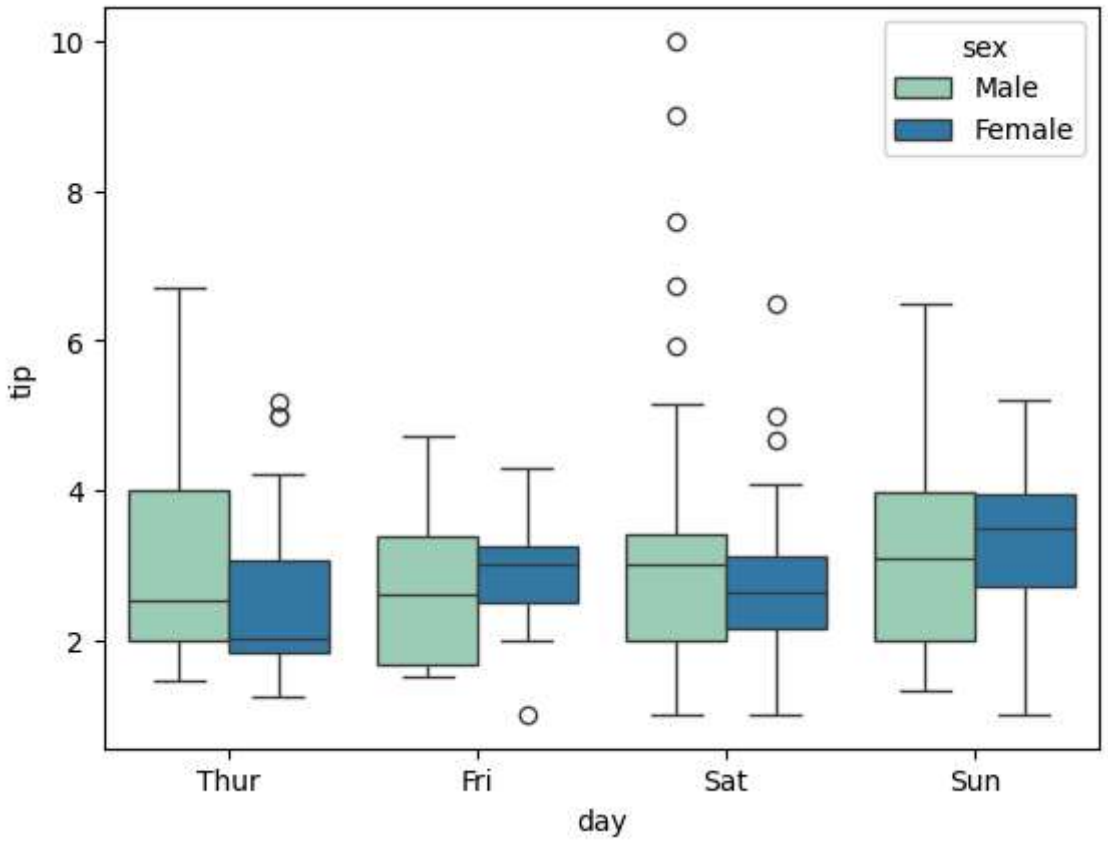
```
Out[25]: <Axes: xlabel='sex', ylabel='tip'>
```



8. Seabom *Box* Plot:

```
In [27]: sns.boxplot(x = 'day', y = 'tip', data = tips, hue = 'sex', palette = 'YlGnBu' )
```

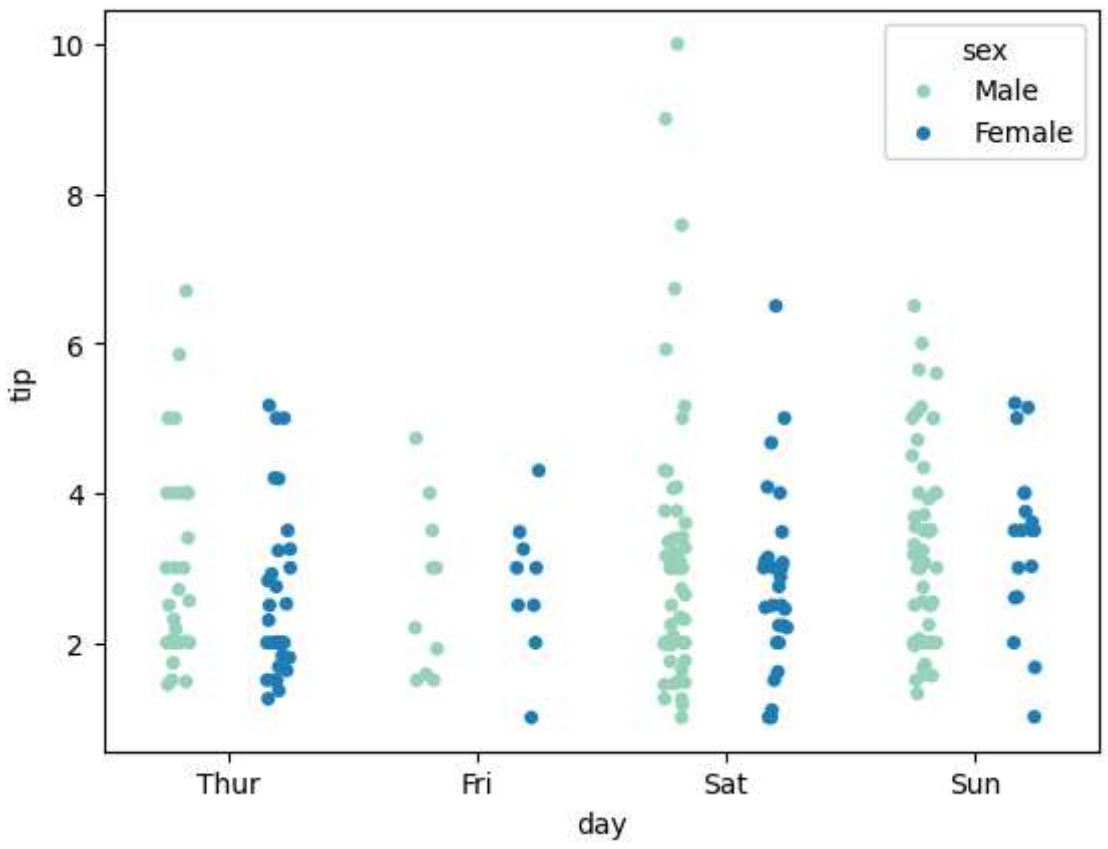
```
Out[27]: <Axes: xlabel='day', ylabel='tip'>
```



9. Seabom *Strip* Plot:

```
In [9]: sns.stripplot(x = 'day', y = 'tip', data = tips, hue = 'sex', palette = 'YlGnBu' , dodge=True)
```

```
Out[9]: <Axes: xlabel='day', ylabel='tip'>
```

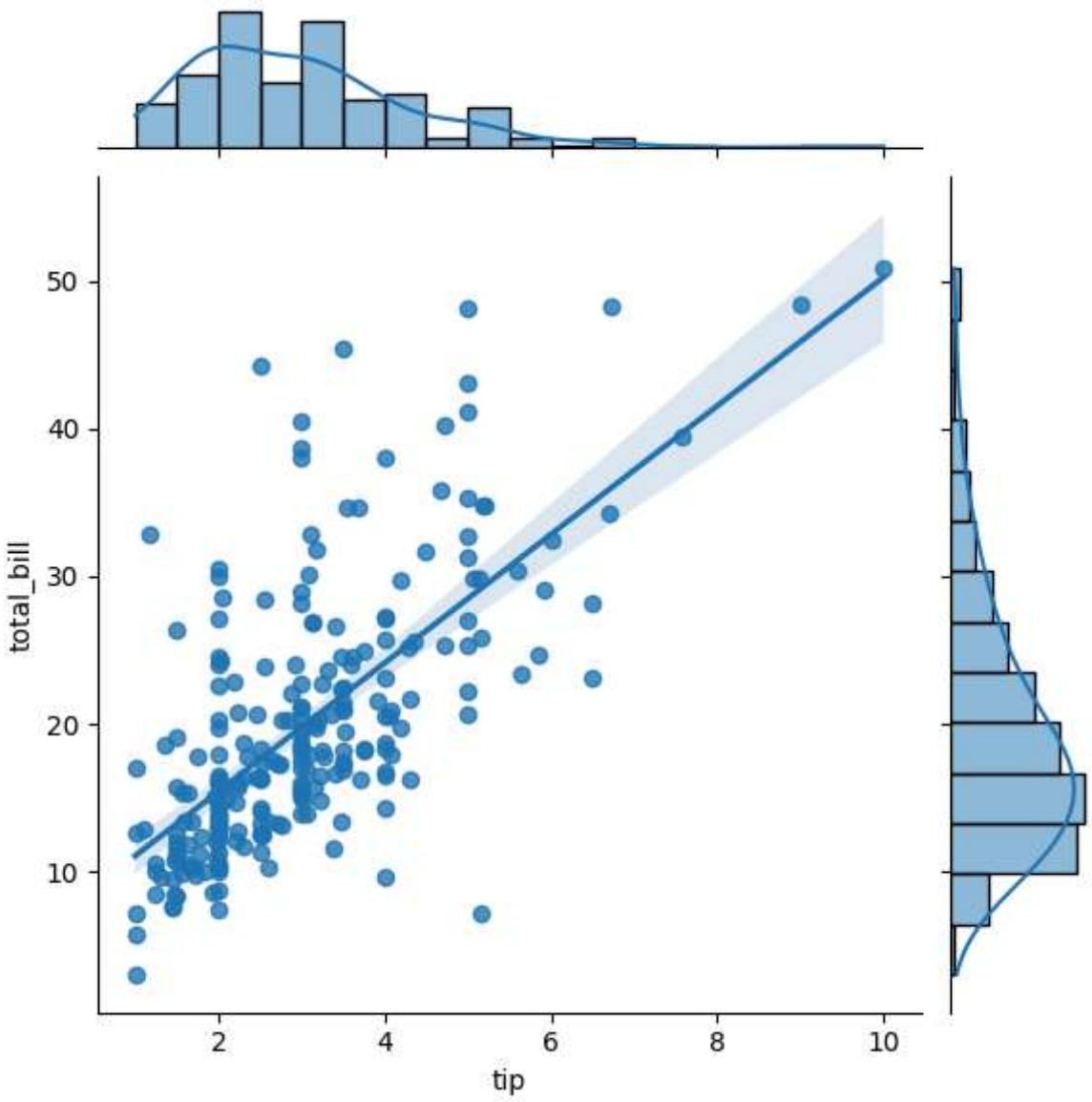


10. Seabom *Join* Plot:

Example-1:

```
In [11]: sns.jointplot(x = 'tip', y = 'total_bill', data=tips, kind = 'reg')
```

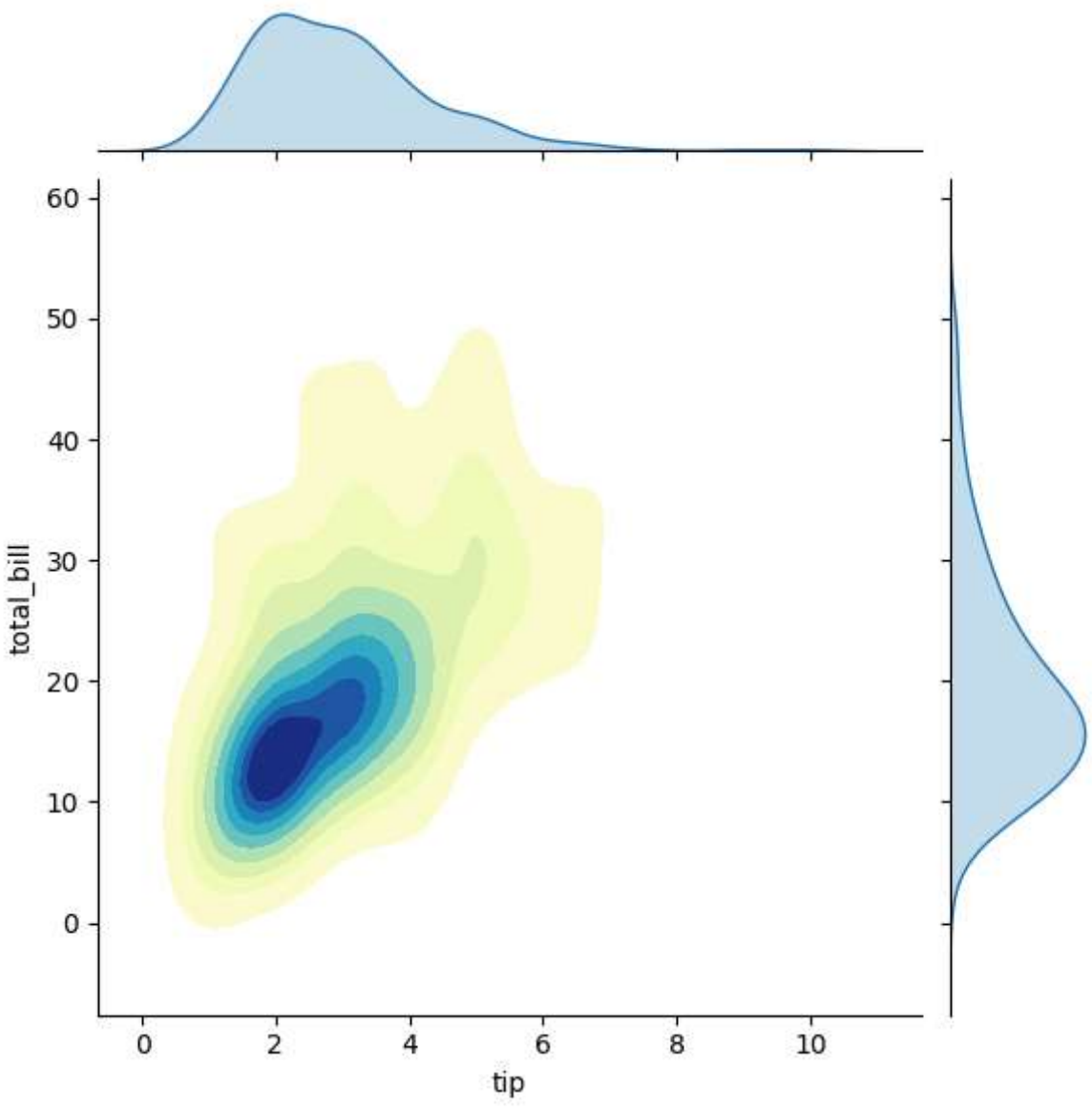
```
Out[11]: <seaborn.axisgrid.JointGrid at 0x1fbfe468b90>
```



Example-2:

In [15]:
```
sns.jointplot(x = 'tip', y = 'total_bill', data=tips, kind = 'kde', fill = True, cmap = 'YlGnBu')
```

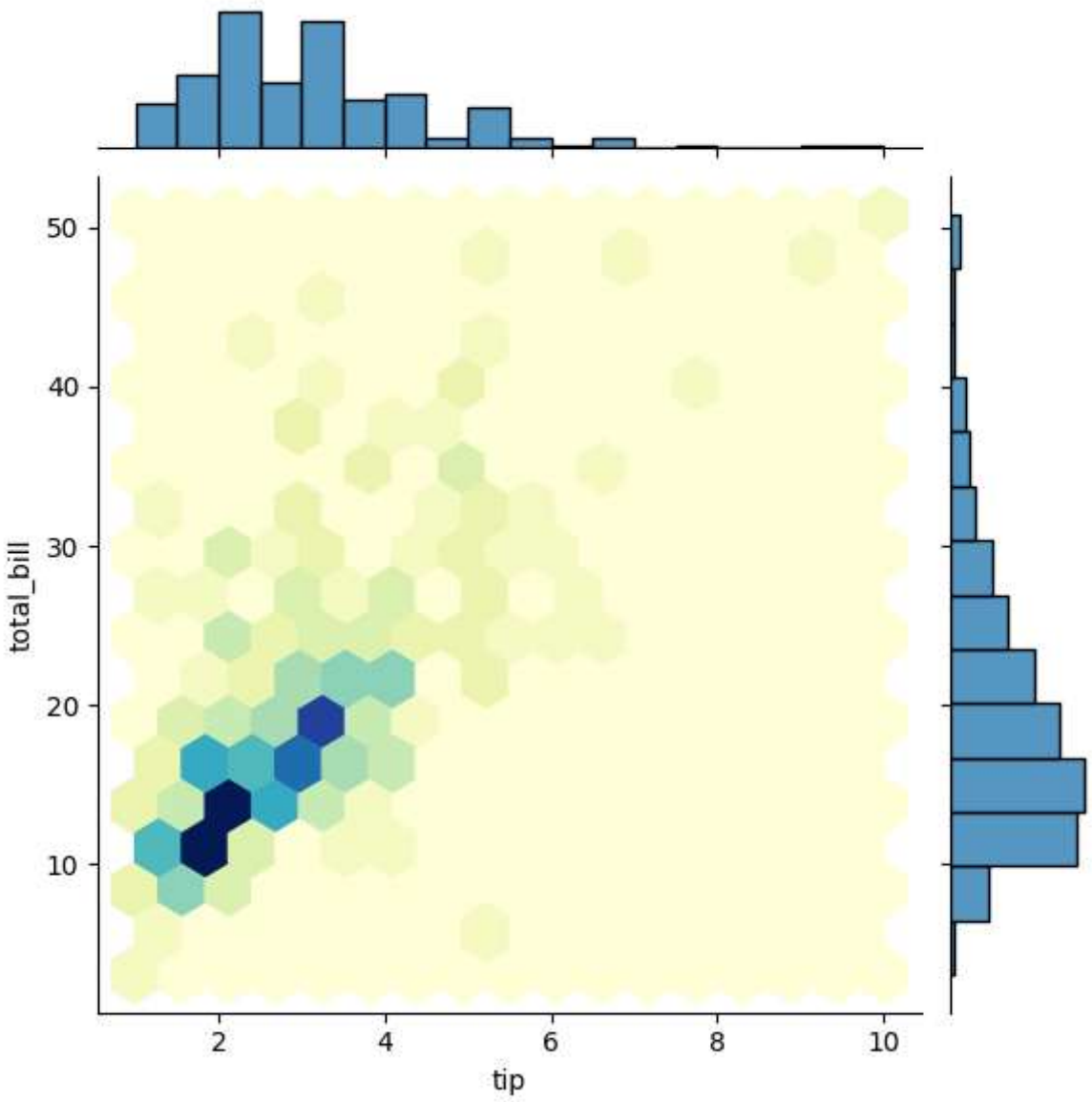
Out[15]: <seaborn.axisgrid.JointGrid at 0x1fbfcff7d10>



Example-3:

In [18]:
```
sns.jointplot(x = 'tip', y = 'total_bill', data=tips, kind = 'hex', cmap = 'YlGnBu')
```

Out[18]: <seaborn.axisgrid.JointGrid at 0x1fb82c55150>



10. Seabom *Pair* Plot:

In [33]:
```
titanic
```

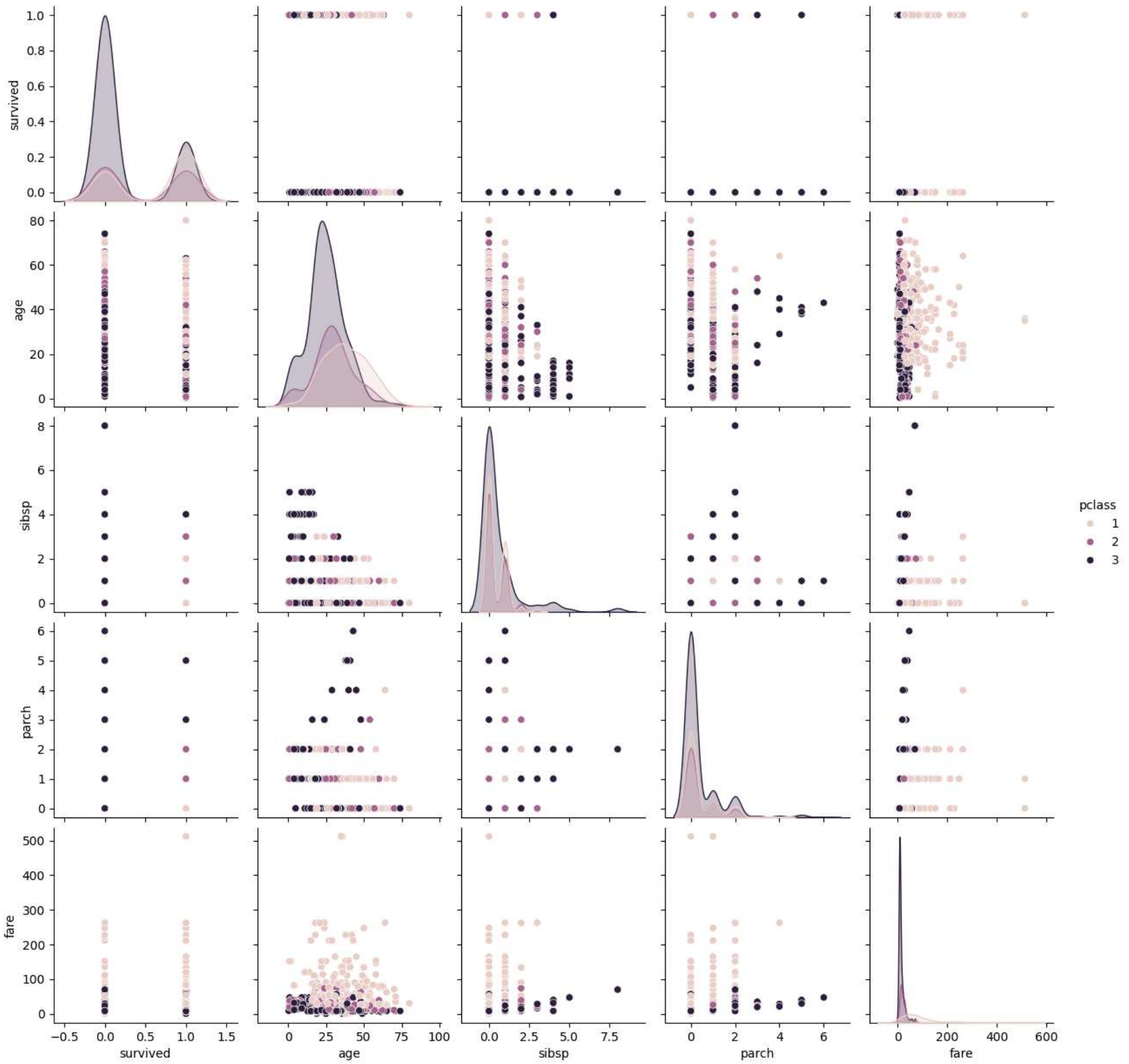
Out[33]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

In [35]:
```
sns.pairplot(titanic.select_dtypes(['number']),hue='pclass')
```

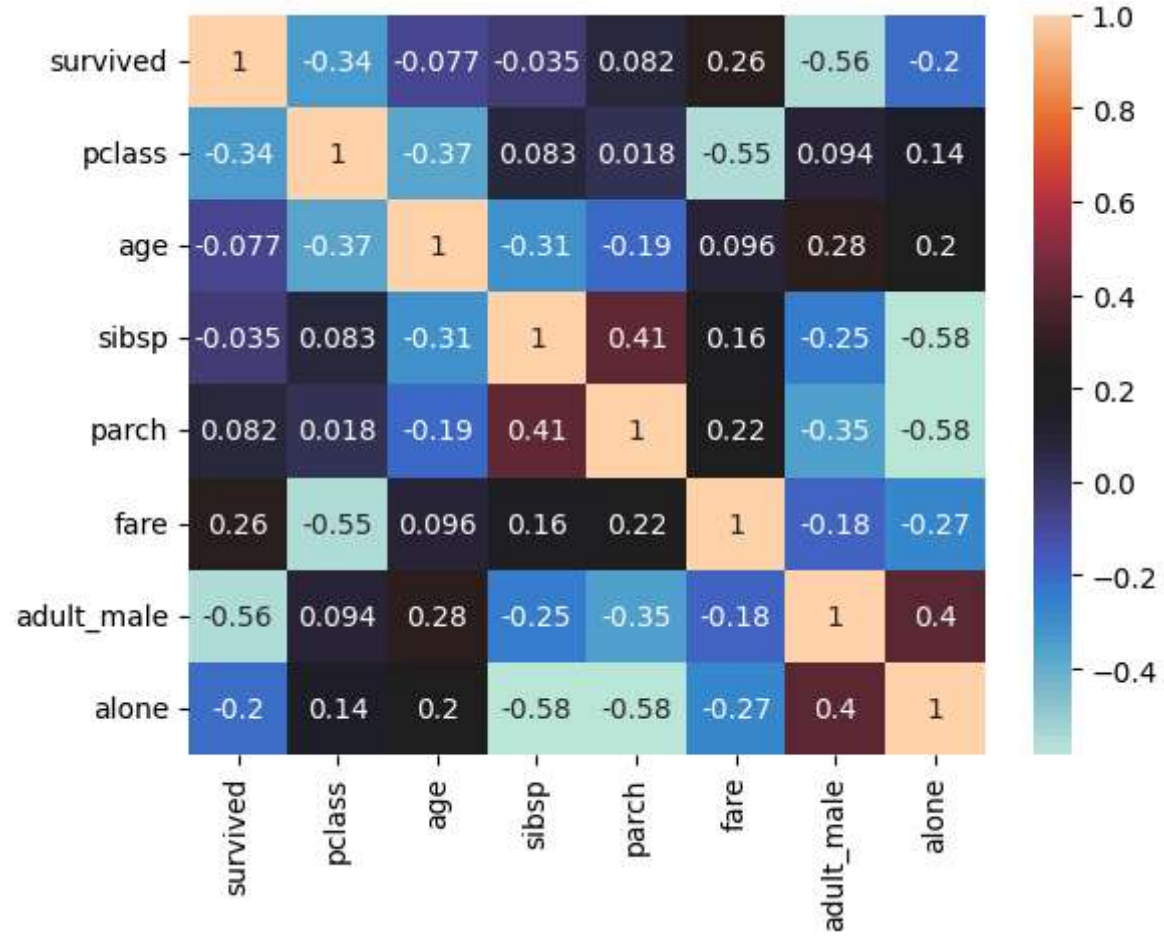
Out[35]:
```
<seaborn.axisgrid.PairGrid at 0x1fbfd116ea0>
```



11. Seabom *Heatmap* Plot:

```
In [37]: sns.heatmap(titanic.corr(numeric_only=True),annot=True,cmap='icefire')
```

```
Out[37]: <Axes: >
```



12. Seabom *Clustermap* Plot:

In [40]:
```
iris
```

Out[40]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [4]:
```
sns.clustermap(iris.drop("species", axis=1))
```

Out[4]:
```
<seaborn.matrix.ClusterGrid at 0x28a2de1ba10>
```