```
#Storing daily sales data in a NumPy array.
import numpy as np
sales = np.array([150, 200, 250, 300, 400, 350, 500]) # Sales for each day
print("Sales Data:", sales)

#Calculating mean, maximum, and minimum sales.

print("Average Sales:", np.mean(sales))
print("Highest Sale:", np.max(sales))
print("Lowest Sale:", np.min(sales))

#Creating a table-like structure using Pandas.

import pandas as pd
data = {
"Customer": ["Alice", "Bob", "Charlie"],
"Age": [25, 30, 35],
"Amount Spent": [120, 200, 150]
}
df = pd.DataFrame(data)
print(df)
```

```
Sales Data: [150 200 250 300 400 350 500]
Average Sales: 307.14285714285717
Highest Sale: 500
Lowest Sale: 150
  Customer  Age  Amount Spent
0    Alice   25           120
1      Bob   30           200
2  Charlie   35           150
```

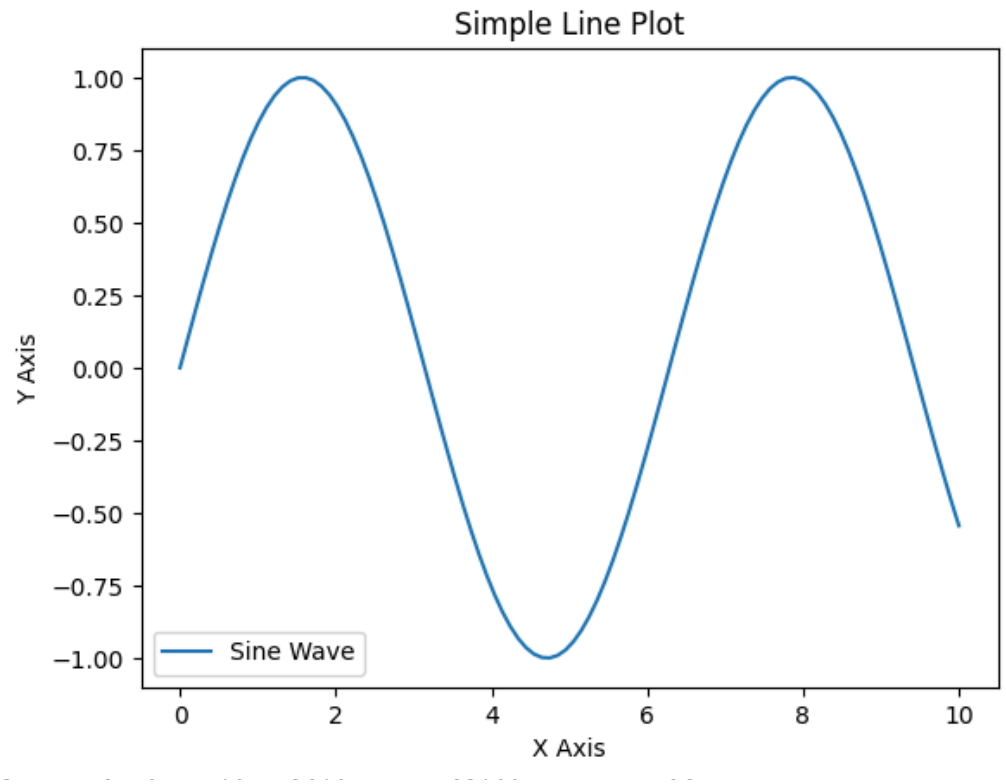
```
#Uploading and reading a CSV file in Pandas.
import pandas as pd
from google.colab import files
uploaded = files.upload() # Upload your CSV file
df = pd.read_csv("students_dataset.csv") # Read and loads the file into a Pandas DataFrame.
df.head()

# Check for missing values

print(df.isnull().sum())

#Remove Missing Values

df_cleaned = df.dropna() # Removes rows with missing values
print(df_cleaned)

#Fill with Mean/Median (For Numerical Data)
df["Age"].fillna(df['Age'].mean(),inplace=True)
df["Marks"].fillna(df["Marks"].median(), inplace=True)
df["Attendance"].fillna(df["Attendance"].mean(),inplace=True)
#Fill with Mode (For Categorical Data)
df["Passed"].fillna(df["Passed"].mode()[0], inplace=True)
print(df)

df.ffill(inplace=True) # Forward fill
df.bfill(inplace=True) # Backward fill
#Remove Duplicates
df.drop_duplicates(inplace=True)
```

```
df.drop_duplicates(inplace=True)

#Standardization (Z-score Normalization)
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
df_scaled=df.copy()
df_scaled[["Marks", "Attendance"]]=scaler.fit_transform(df[["Marks", "Attendance"]])
print(df_scaled)

#Min-Max Scaling
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()

df_scaled[["Marks", "Attendance"]]=scaler.fit_transform(df[["Marks", "Attendance"]])
print(df_scaled)

#One-Hot Encoding
df_encoded = pd.get_dummies(df, columns=["Passed"],drop_first=True)
print(df_encoded)

#Label Encoding
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df["Passed"] = encoder.fit_transform(df["Passed"])
print(df)

#Creating New Features

def performance_category(marks):
  if marks>=85:
    return "High"
  elif marks >= 70:
    return "Medium"
  else:
    return "Low"

df["Performance"] = df["Marks"].apply(performance_category)
print(df)
#Binning (Converting Continuous to Categorical Data)

df["Age_Group"] = pd.cut(df["Age"], bins=[18, 21, 24],labels=["Young", "Adult"])
print(df)
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving students_dataset.csv to students_dataset.csv
Name          0
Age           4
Marks         4
Attendance    4
Passed        0
dtype: int64
     Name   Age  Marks  Attendance Passed
0   Alice  20.0   85.0        90.0    Yes
3   David  22.0   90.0        85.0    Yes
4     Eve  20.0   88.0        95.0     No
8     Ivy  24.0   92.0        92.0    Yes
12   Mona  21.0   83.0        80.0     No
       Name   Age  Marks  Attendance Passed
0     Alice  20.0   85.0       90.00    Yes
1       Bob  21.0   82.0       80.00     No
2   Charlie  22.0   78.0       85.25    Yes
3     David  22.0   90.0       85.00    Yes
4       Eve  20.0   88.0       95.00     No
5     Frank  22.0   76.0       70.00    Yes
6     Grace  23.0   82.0       88.00     No
7      Hank  21.0   80.0       85.25    Yes
8       Ivy  24.0   92.0       92.00    Yes
9      Jack  22.0   79.0       85.00     No
10    Kelly  22.0   82.0       78.00    Yes
11      Leo  23.0   87.0       85.25    Yes
12     Mona  21.0   83.0       80.00     No
13     Nina  22.0   77.0       89.00    Yes
14    Oscar  25.0   82.0       91.00     No
15     Paul  22.0   81.0       85.25    Yes
<ipython-input-3-8ba44832badd>:18: FutureWarning: A value is trying to be set on a copy of a Da
The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu


  df["Age"].fillna(df['Age'].mean(),inplace=True)
<ipython-input-3-8ba44832badd>:19: FutureWarning: A value is trying to be set on a copy of a Da
The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu


  df["Marks"].fillna(df["Marks"].median(), inplace=True)
<ipython-input-3-8ba44832badd>:20: FutureWarning: A value is trying to be set on a copy of a Da
The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu


  df["Attendance"].fillna(df["Attendance"].mean(),inplace=True)
<ipython-input-3-8ba44832badd>:22: FutureWarning: A value is trying to be set on a copy of a Da
The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
```

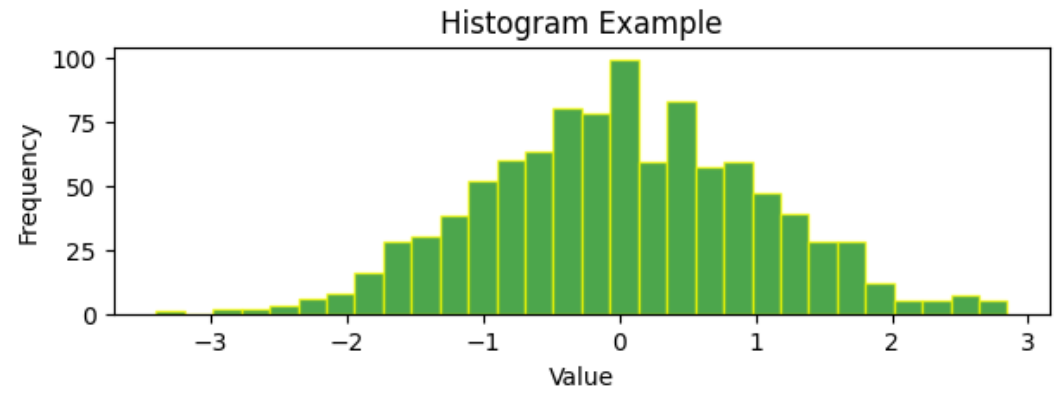
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y, label="Sine Wave")
```

```
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.title("Simple Line Plot")
plt.legend()
plt.show()
```




```
4       Eve  20.0    88.0        95.00        False
```

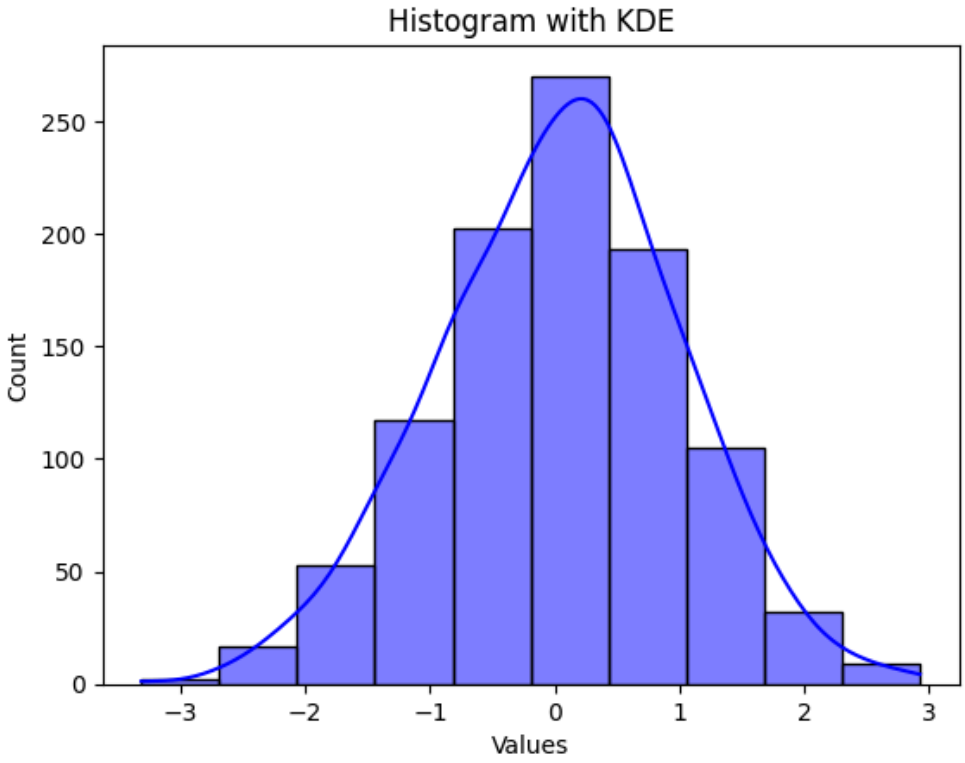
```
#1.3 Histogram (Distribution of Data)

data = np.random.randn(1000)
#print(data)
plt.figure(figsize=(7, 2))
plt.hist(data, bins=30, color='green', edgecolor='yellow', alpha=0.7)
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.title("Histogram Example")
plt.show()
```




```
#2. Seaborn - Statistical Data Visualization ,2.1 Histogram & KDE Plot
```

```
import seaborn as sns
import pandas as pd

# Creating sample data
data = np.random.randn(1000)
df = pd.DataFrame(data, columns=['Values'])
print(df['Values'])
# Plot
sns.histplot(df['Values'], bins=10, kde=True, color='blue')
plt.title("Histogram with KDE")
plt.show()
```

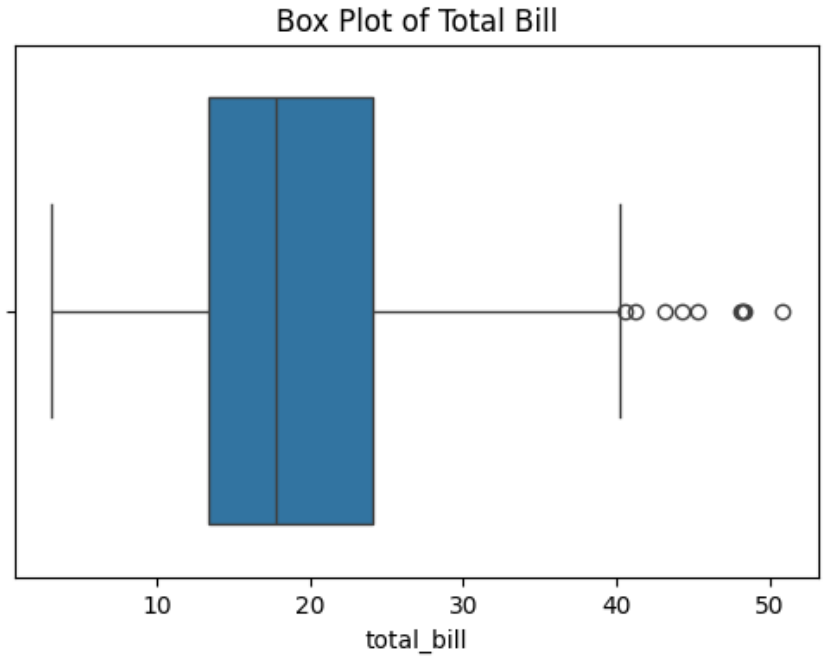
```
0      1.335194
1      0.913911
2      0.245516
3      1.498306
4      0.255534
       ...
995   -0.334334
996   -1.105520
997   -1.353480
998    0.063465
999   -1.431960
Name: Values, Length: 1000, dtype: float64
```



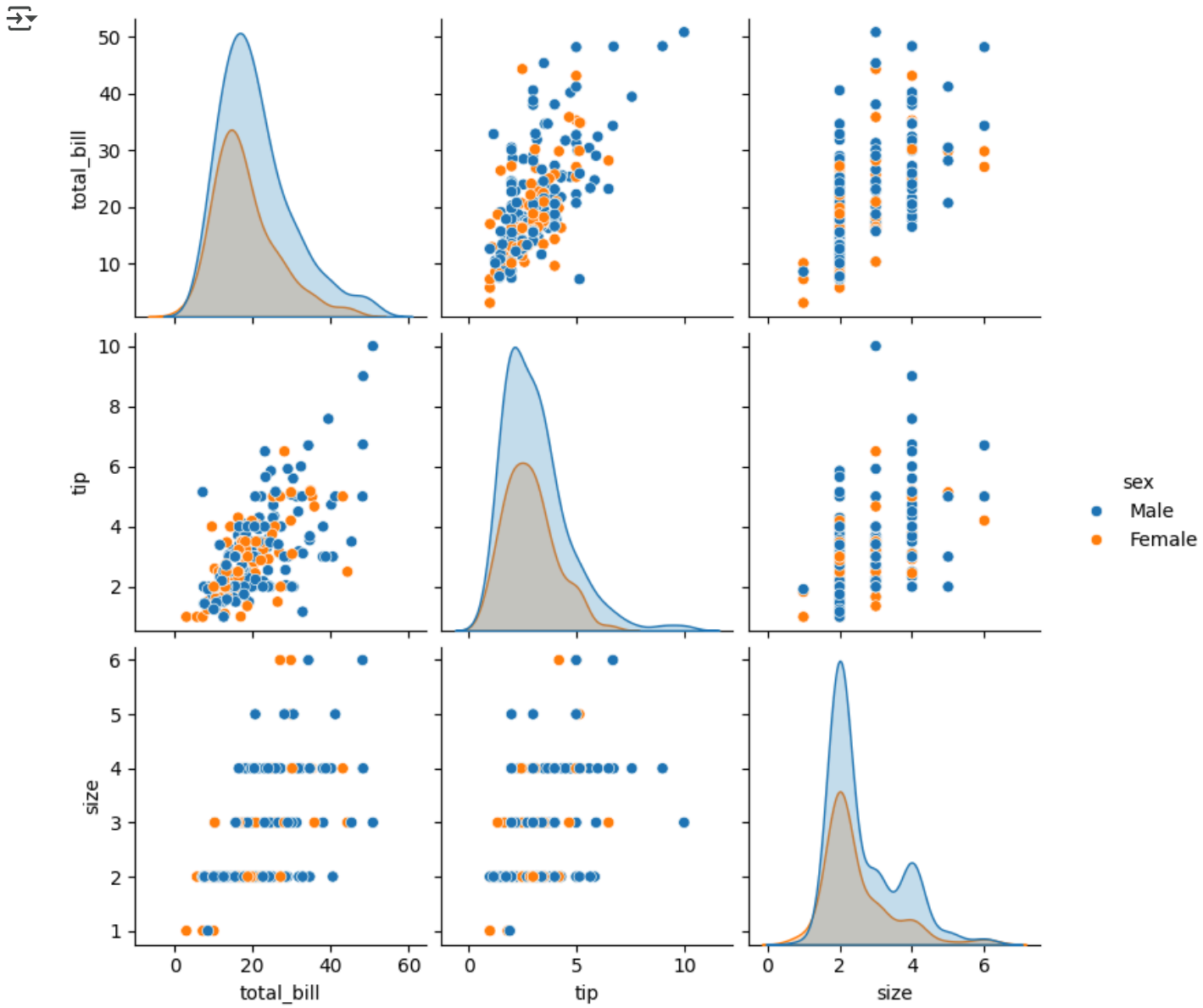

```
#2.2 Box Plot (Detecting Outliers)

tips = sns.load_dataset('tips')
print(tips)
plt.figure(figsize=(6, 4))
sns.boxplot(x=tips['total_bill'])
plt.title("Box Plot of Total Bill")
plt.show()
```

	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

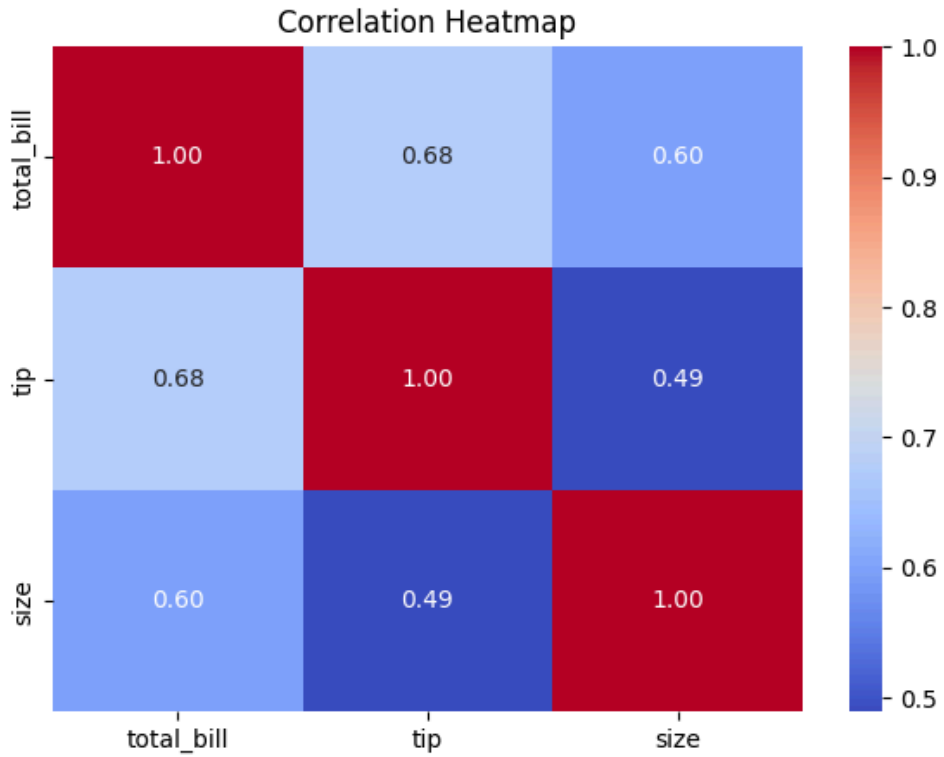
[244 rows x 7 columns]




```
#2.3 Pair Plot (Exploring Relationships)
sns.pairplot(tips, hue='sex')
plt.show()
```

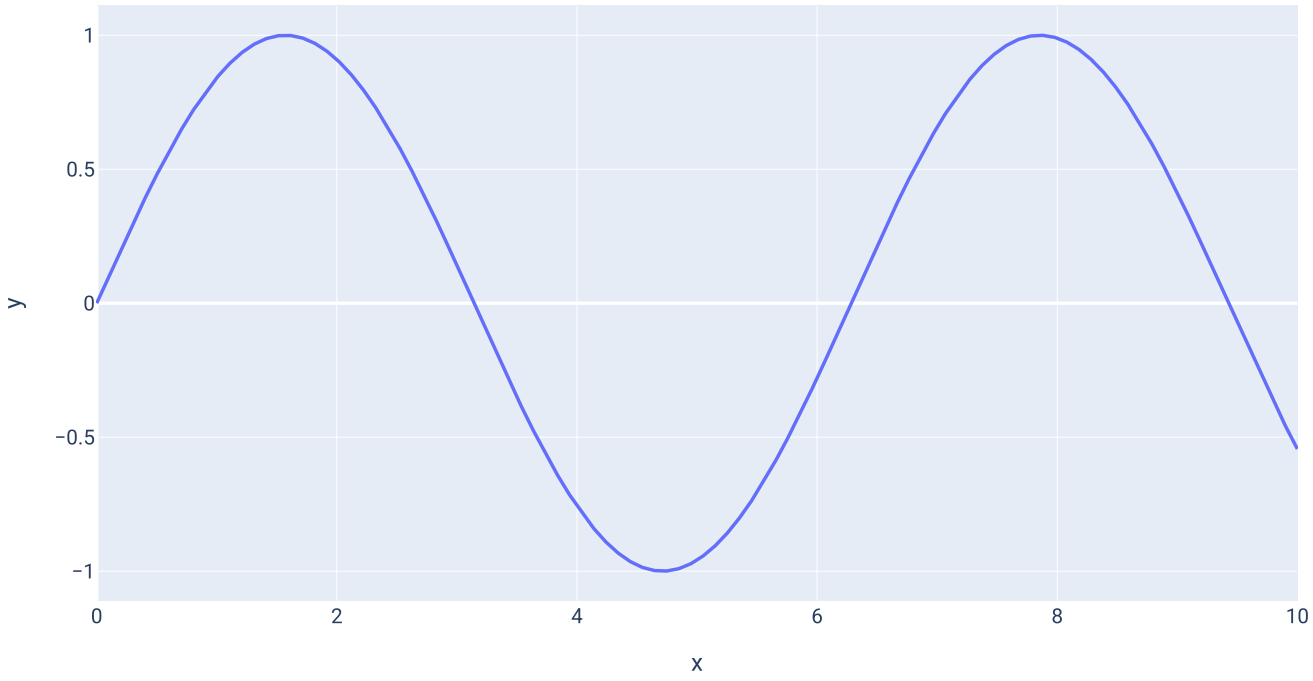



```
#2.4 Heatmap (Correlation Analysis)
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
tips = sns.load_dataset('tips')

# Select only numeric columns for correlation
numeric_tips = tips.select_dtypes(include='number')
corr_matrix = numeric_tips.corr()

# Plot the heatmap
plt.figure(figsize=(7, 5))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



#3. Pyplot,3.1 Interactive Line Plot

```
import pandas as pd
import numpy as np
import plotly.express as px

df = pd.DataFrame({"x": np.linspace(0, 10, 100),"y": np.sin(np.linspace(0, 10, 100))
})

fig = px.line(df, x='x', y='y', title="Interactive Sine Wave")
fig.show()
```




```
#3.3 Interactive 3D Scatter Plot
import plotly.graph_objects as go

fig = go.Figure(data=[go.Scatter3d(
    x=tips['total_bill'],
    y=tips['tip'],
    z=tips['size'],
    mode='markers',
    marker=dict(size=5, color=tips['total_bill'], colorscale='Viridis')
)])

fig.update_layout(title="3D Scatter Plot of Total Bill, Tip & Size")
fig.show()
```