NAME: Pranav Sanjay Kakade

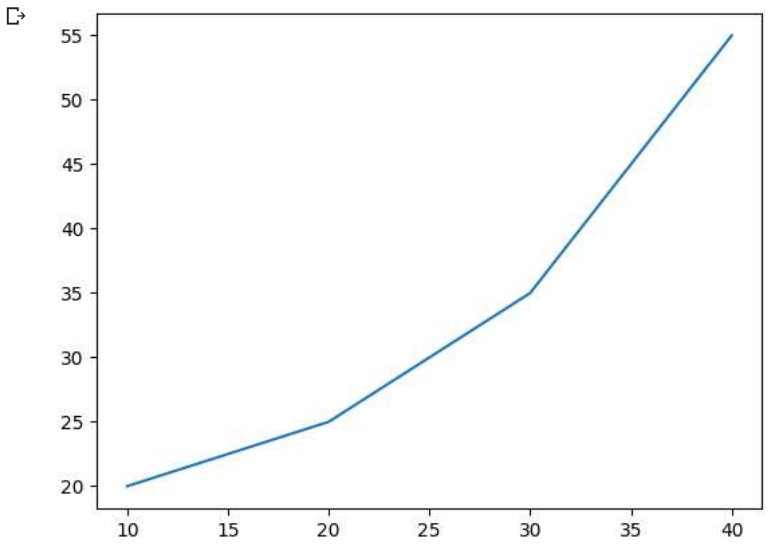
Roll No:623

DIV:F(F2)

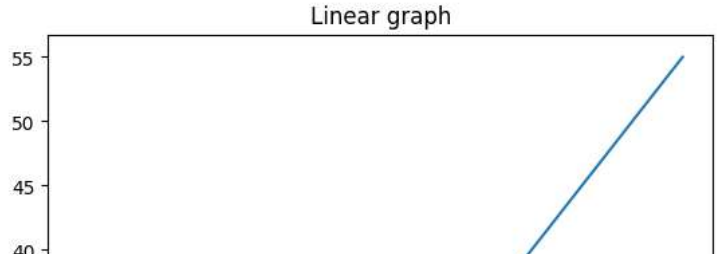
PRN:202201090089

ASSIGNMENT 5-A

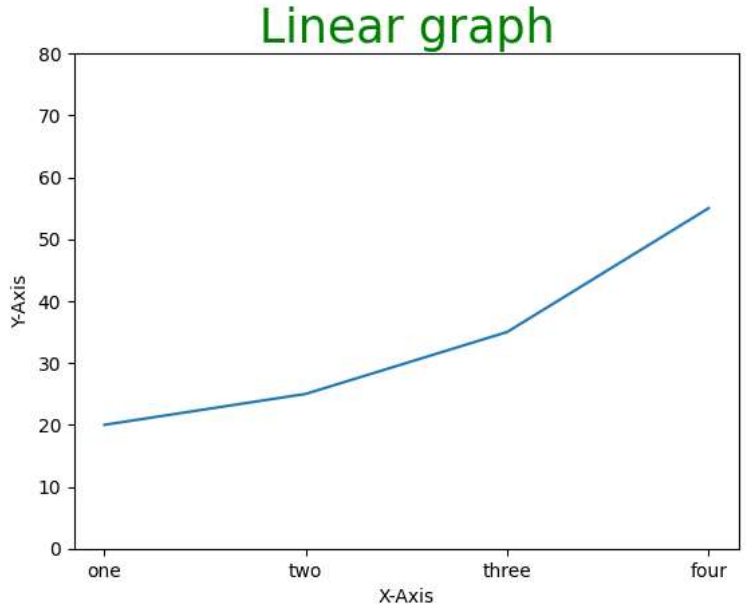
```
# https://www.geeksforgeeks.org/data-visualization-using-matplotlib/
#https://www.w3resource.com/graphics/matplotlib/basic/index.php
# https://gist.github.com/pb111/a6bdc9fd610344973e06da478604716f
import matplotlib.pyplot as plt

# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# plotting the data
plt.plot(x, y)

plt.show()
```




```
# Adding Title
# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# plotting the data
plt.plot(x, y)

# Adding title to the plot
plt.title("Linear graph")

plt.show()
```

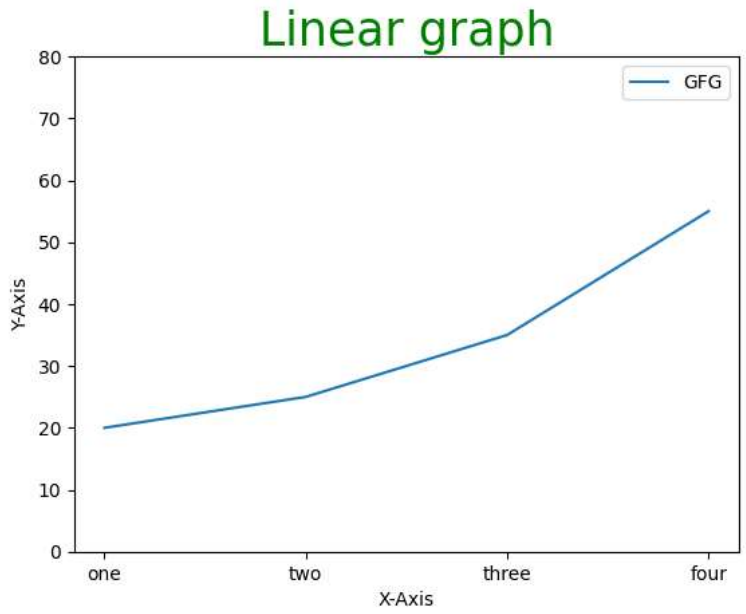



```
#Setting Limits and Tick labels
import matplotlib.pyplot as plt


# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# plotting the data
plt.plot(x, y)

# Adding title to the plot
plt.title("Linear graph", fontsize=25, color="green")

# Adding label on the y-axis
plt.ylabel('Y-Axis')

# Adding label on the x-axis
plt.xlabel('X-Axis')

# Setting the limit of y-axis
plt.ylim(0, 80)

# setting the labels of x-axis
plt.xticks(x, labels=["one", "two", "three", "four"])

plt.show()
```

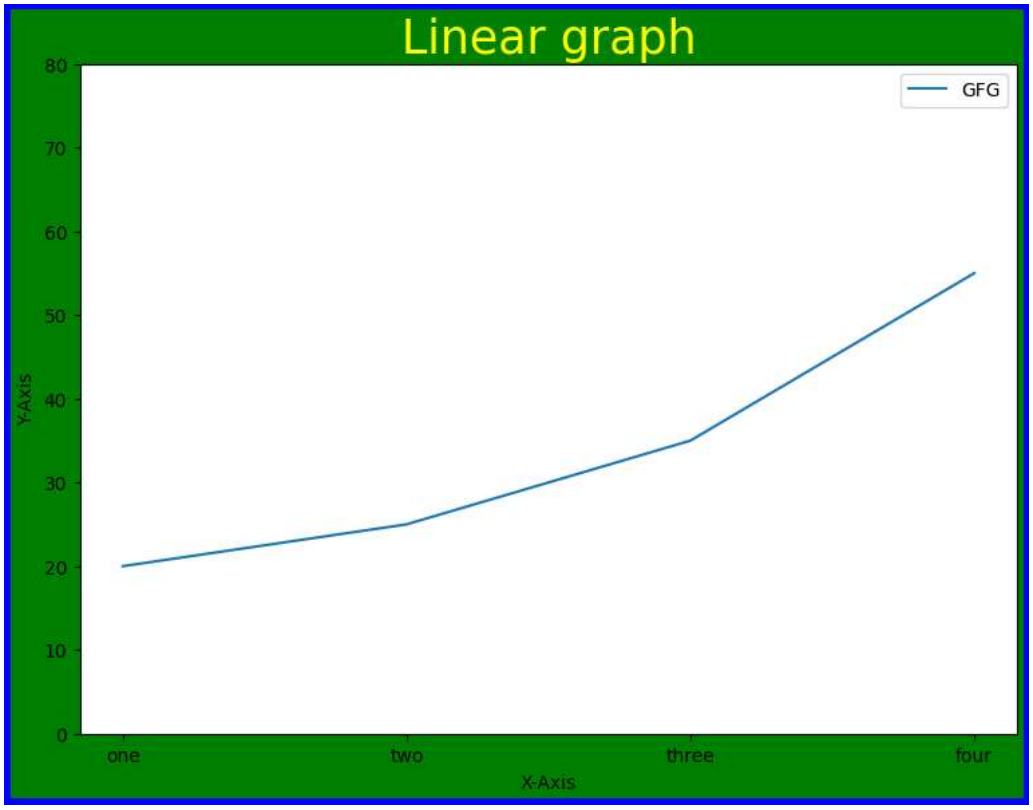



```
#Adding Legends
import matplotlib.pyplot as plt


# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# plotting the data
plt.plot(x, y)

# Adding title to the plot
plt.title("Linear graph", fontsize=25, color="green")

# Adding label on the y-axis
plt.ylabel('Y-Axis')
```

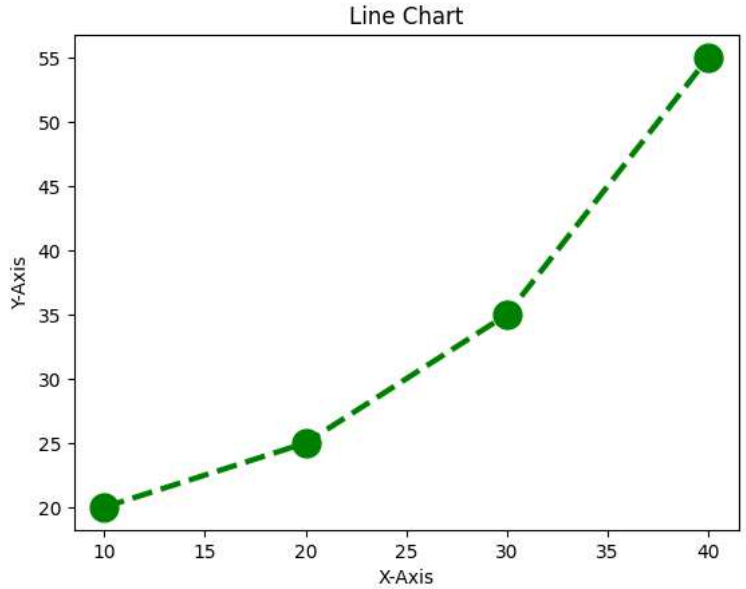
```
# Adding label on the x-axis
plt.xlabel('X-Axis')

# Setting the limit of y-axis
plt.ylim(0, 80)

# setting the labels of x-axis
plt.xticks(x, labels=["one", "two", "three", "four"])

# Adding legends
plt.legend(["GFG"])

plt.show()
```



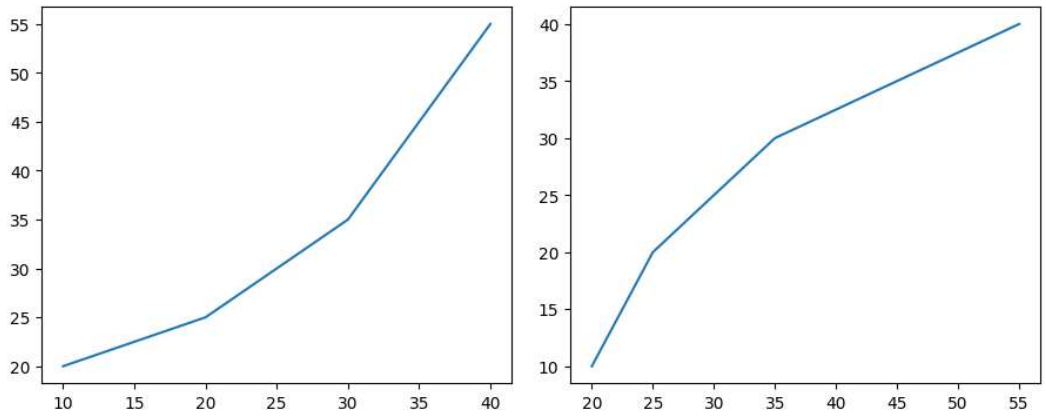

```
#Figure class
# Python program to show pyplot module
import matplotlib.pyplot as plt
from matplotlib.figure import Figure

# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# Creating a new figure with width = 7 inches
# and height = 5 inches with face color as
# green, edgecolor as red and the line width
# of the edge as 7
fig = plt.figure(figsize =(7, 5), facecolor='g',
                 edgecolor='b', linewidth=7)

# Creating a new axes for the figure
ax = fig.add_axes([1, 1, 1, 1])

# Adding the data to be plotted
ax.plot(x, y)

# Adding title to the plot
plt.title("Linear graph", fontsize=25, color="yellow")

# Adding label on the y-axis
plt.ylabel('Y-Axis')

# Adding label on the x-axis
plt.xlabel('X-Axis')

# Setting the limit of y-axis
plt.ylim(0, 80)

# setting the labels of x-axis
plt.xticks(x, labels=["one", "two", "three", "four"])

# Adding legends
plt.legend(["GFG"])

plt.show()
```




```
#Different line styles
import matplotlib.pyplot as plt


# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# plotting the data
plt.plot(x, y, color='green', linewidth=3, marker='o',
        markersize=15, linestyle='--')

# Adding title to the plot
plt.title("Line Chart")

# Adding label on the y-axis
plt.ylabel('Y-Axis')

# Adding label on the x-axis
plt.xlabel('X-Axis')

plt.show()
```

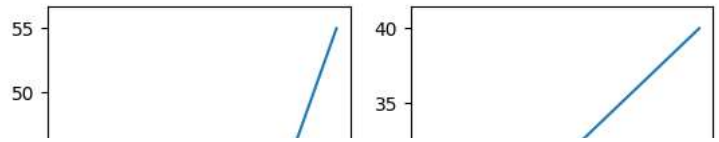


```
#Multiple Plots
# Python program to show pyplot module
import matplotlib.pyplot as plt
from matplotlib.figure import Figure

# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# Creating a new figure with width = 5 inches
# and height = 4 inches
fig = plt.figure(figsize =(5, 4))

# Creating first axes for the figure
ax1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])

# Creating second axes for the figure
ax2 = fig.add_axes([1, 0.1, 0.8, 0.8])

# Adding the data to be plotted
ax1.plot(x, y)
ax2.plot(y, x)

plt.show()
```




```
#Using subplot() method.
import matplotlib.pyplot as plt


# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]


# Creating figure object
plt.figure()

# adding first subplot
plt.subplot(121)
plt.plot(x, y)

# adding second subplot
plt.subplot(122)
plt.plot(y, x)
```

```
[<matplotlib.lines.Line2D at 0x7f1100217ca0>]
```



ASSINGNMENT 5-B

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas import Series, DataFrame


# Reading the tips.csv file
df1=pd.read_csv('/content/tips.csv')

df1.head()
```

	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

```
df1.tail()
```

	total_bill	tip	sex	smoker	day	time	size
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

```
df1.columns
```

```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')
```

```
df1.info()
```

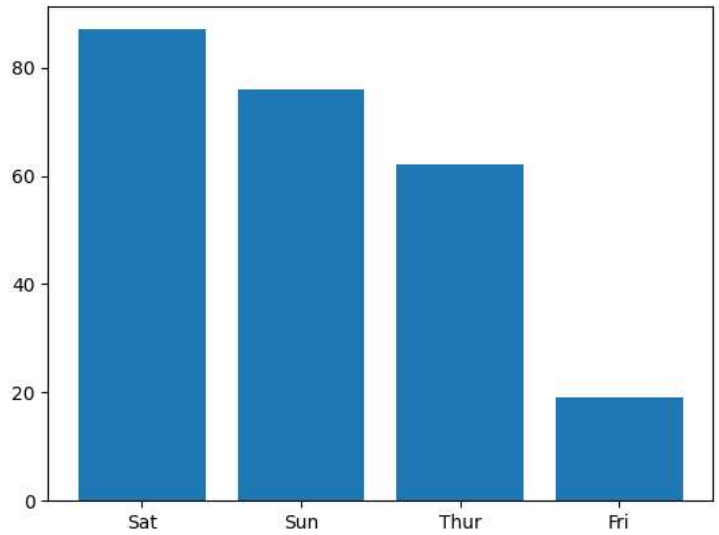
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object 
 3   smoker      244 non-null    object 
 4   day         244 non-null    object 
 5   time        244 non-null    object 
 6   size        244 non-null    int64  
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

```
df1.describe()
```

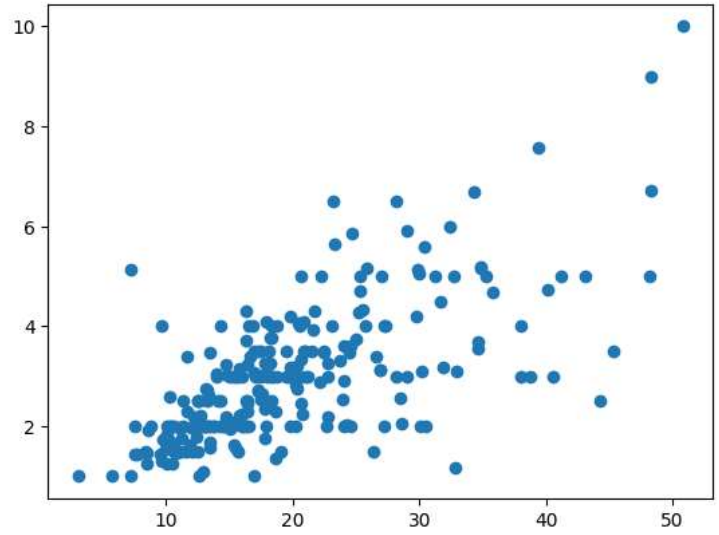
	total_bill	tip	size
count	244.000000	244.000000	244.000000

```
a=pd.DataFrame(df1['day'].value_counts())
a.reset_index(inplace=True)
plt.bar(a['index'],a['day'])
```

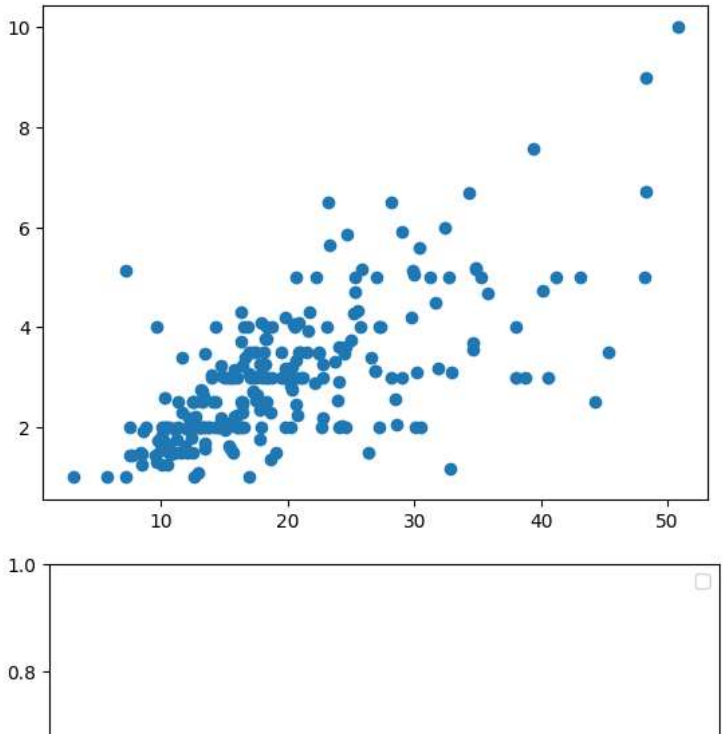
```
<BarContainer object of 4 artists>
```



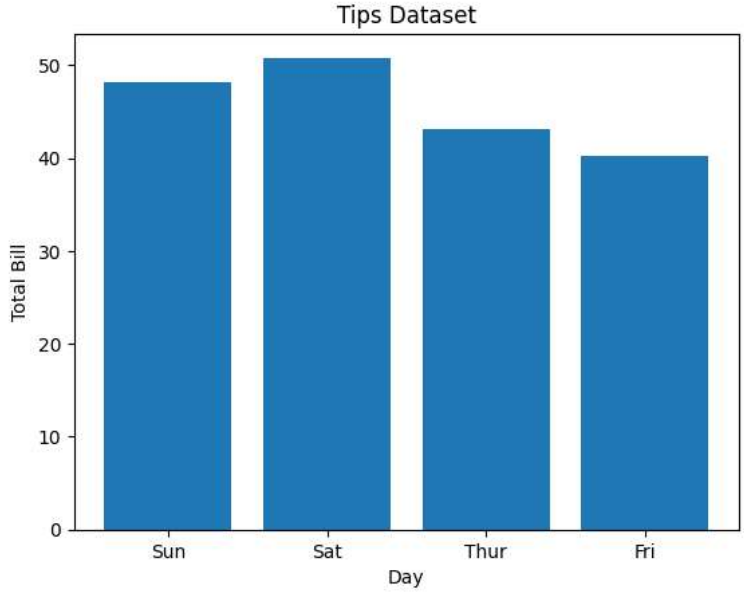

```
plt.scatter(df1['total_bill'],df1['tip'])
plt.show()
```




```
plt.scatter(x='total_bill',y='tip',data=df1)
fig=plt.figure(figsize=(5,4))
ax=fig.add_axes([1,1,1,1])
ax.legend(labels=('sun','mon','tue'))
plt.show()
```




```
#Different types of Matplotlib Plots
#bar chart
import matplotlib.pyplot as plt
import pandas as pd

# Reading the tips.csv file
data = pd.read_csv('/content/tips.csv')

# initializing the data
x = data['day']
y = data['total_bill']

# plotting the data
plt.bar(x, y)

# Adding title to the plot
plt.title("Tips Dataset")

# Adding label on the y-axis
plt.ylabel('Total Bill')

# Adding label on the x-axis
plt.xlabel('Day')

plt.show()
```



Customization that is available for the Bar Chart –

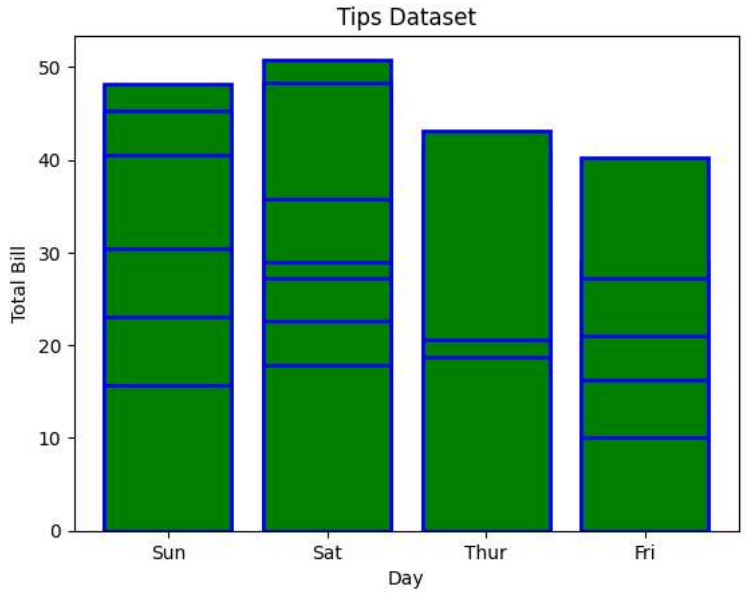
color: For the bar faces

edgecolor: Color of edges of the bar

linewidth: Width of the bar edges

width: Width of the ba

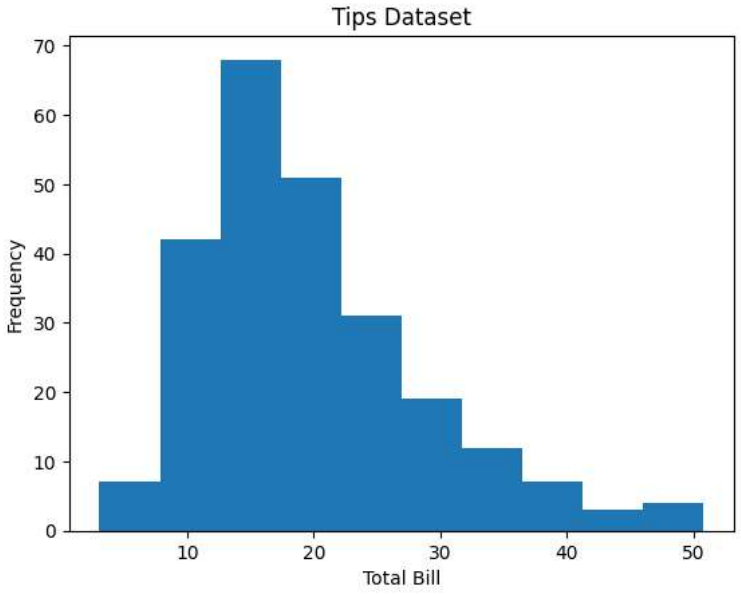
```
import matplotlib.pyplot as plt
import pandas as pd



# initializing the data
x = data['day']
y = data['total_bill']

# plotting the data
plt.bar(x, y, color='green', edgecolor='blue',
        linewidth=2)

# Adding title to the plot
plt.title("Tips Dataset")

# Adding label on the y-axis
plt.ylabel('Total Bill')

# Adding label on the x-axis
plt.xlabel('Day')

plt.show()
```



Histogram A histogram is basically used to represent data provided in a form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The hist() function is used to compute and create histogram of x.

```
import matplotlib.pyplot as plt
import pandas as pd



# initializing the data
x = data['total_bill']

# plotting the data
plt.hist(x)

# Adding title to the plot
plt.title("Tips Dataset")

# Adding label on the y-axis
plt.ylabel('Frequency')
```

```
# Adding label on the x-axis
plt.xlabel('Total Bill')

plt.show()
```



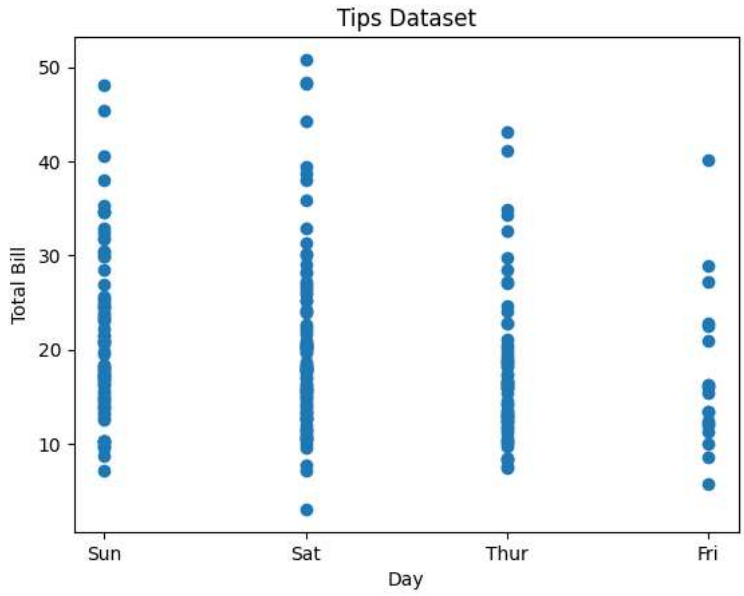
Customization that is available for the Histogram –

bins: Number of equal-width bins color: For changing the face color edgecolor: Color of the edges linestyle: For the edgelines alpha: blending value, between 0 (transparent) and 1 (opaque)

```
import matplotlib.pyplot as plt
import pandas as pd



# initializing the data
x = data['total_bill']

# plotting the data
plt.hist(x, bins=25, color='green', edgecolor='blue',
        linestyle='--', alpha=0.5)

# Adding title to the plot
plt.title("Tips Dataset")

# Adding label on the y-axis
plt.ylabel('Frequency')

# Adding label on the x-axis
plt.xlabel('Total Bill')

plt.show()
```

Scatter Plot Scatter plots are used to observe relationships between variables. The scatter() method in the matplotlib library is used to draw a scatter plot.

```
import matplotlib.pyplot as plt
import pandas as pd



# initializing the data
x = data['day']
y = data['total_bill']

# plotting the data
plt.scatter(x, y)

# Adding title to the plot
plt.title("Tips Dataset")

# Adding label on the y-axis
plt.ylabel('Total Bill')

# Adding label on the x-axis
plt.xlabel('Day')

plt.show()
```



Customizations that are available for the scatter plot are –

s: marker size (can be scalar or array of size equal to size of x or y)

c: color of sequence of colors for markers

marker: marker style

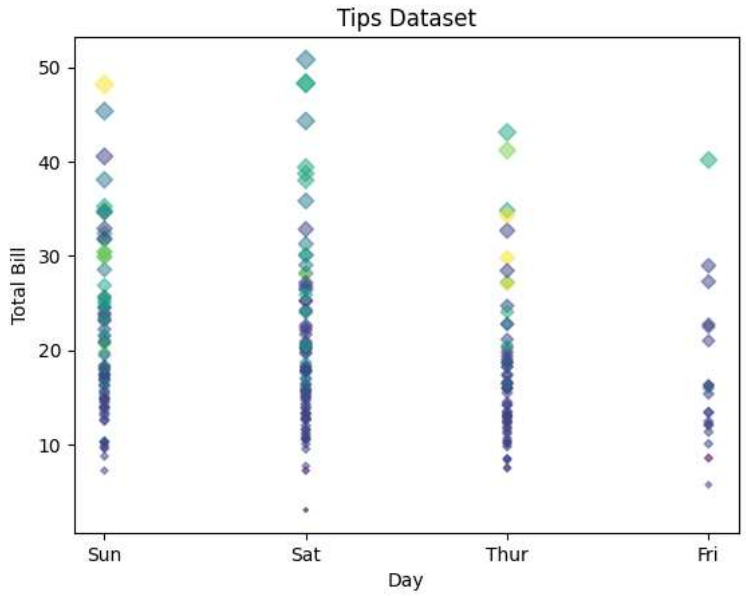
linewidths: width of marker border

edgecolor: marker border color

alpha: blending value, between 0 (transparent) and 1 (opaque)

```
import matplotlib.pyplot as plt
import pandas as pd



# initializing the data
x = data['day']
y = data['total_bill']

# plotting the data
plt.scatter(x, y, c=data['size'], s=data['total_bill'],
            marker='D', alpha=0.5)
```

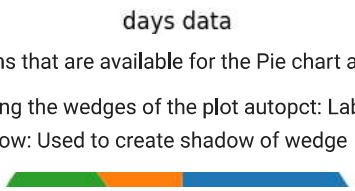
```
# Adding title to the plot
plt.title("Tips Dataset")

# Adding label on the y-axis
plt.ylabel('Total Bill')

# Adding label on the x-axis
plt.xlabel('Day')

plt.show()
```



Pie Chart

Pie chart is a circular chart used to display only one series of data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges. It can be created using the pie() method.

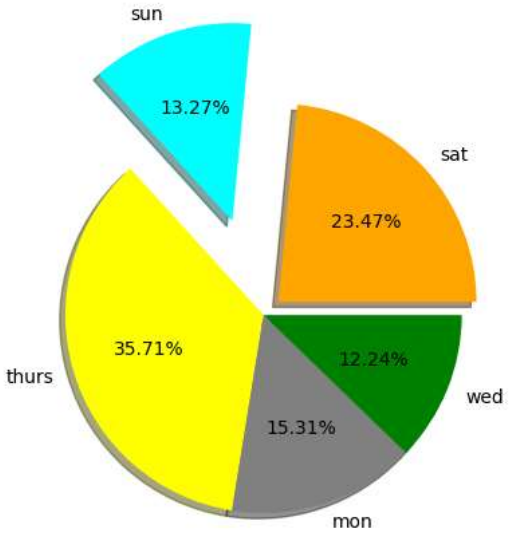
```
import matplotlib.pyplot as plt
import pandas as pd


# initializing the data
day = ['sat', 'sun', 'fri',
        'wed', 'thurs',]
data = [23, 10, 35, 15, 12]

# plotting the data
plt.pie(data, labels=day)

# Adding title to the plot
plt.title("days data")

plt.show()
```



Customizations that are available for the Pie chart are –

explode: Moving the wedges of the plot autopct: Label the wedge with their numerical value. color: Attribute is used to provide color to the wedges. shadow: Used to create shadow of wedge

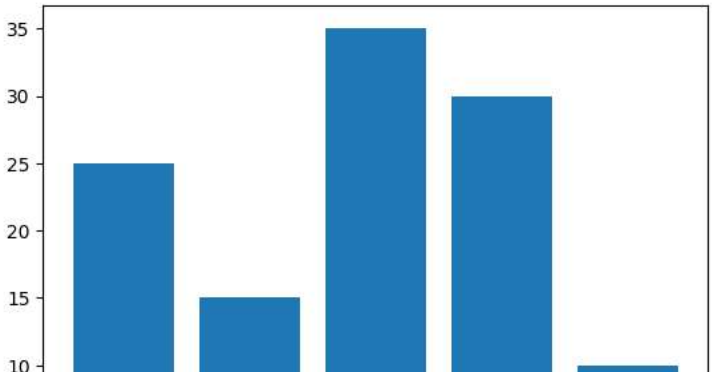

```
import matplotlib.pyplot as plt
import pandas as pd

# initializing the data
days = ['sat', 'sun', 'thurs',
        'mon', 'wed',]
data = [23, 13, 35, 15, 12]

explode = [0.1, 0.5, 0, 0, 0]

colors = ( "orange", "cyan", "yellow",
        "grey", "green",)

# plotting the data
plt.pie(data, labels=days, explode=explode, autopct='%1.2f%%',
        colors=colors, shadow=True)

plt.show()
```



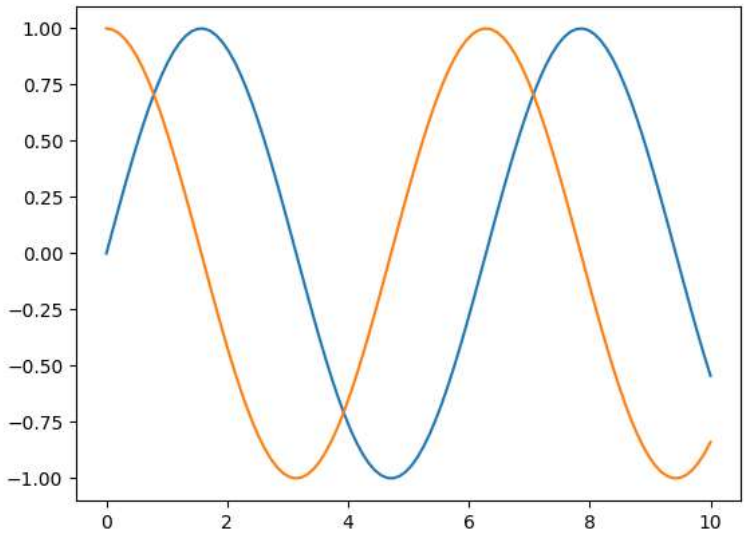
Saving a Plot

For saving a plot in a file on storage disk, savefig() method is used. A file can be saved in many formats like .png, .jpg, .pdf, etc

```
import matplotlib.pyplot as plt

# Creating data
year = ['sat', 'sun', 'thurs', 'mon', 'tue']
production = [25, 15, 35, 30, 10]

# Plotting barchart
plt.bar(year, production)

# Saving the figure.
plt.savefig("output.jpg")

# Saving figure by changing parameter values
plt.savefig("output1", facecolor='y', bbox_inches="tight",
            pad_inches=0.3, transparent=True)
```




```
x = np.linspace(0,10,100)
fig = plt.figure()
plt.plot(x,np.sin(x))
plt.plot(x,np.cos(x))
fig.savefig('Graph1.png')
```



✓ 0s completed at 8:02 PM