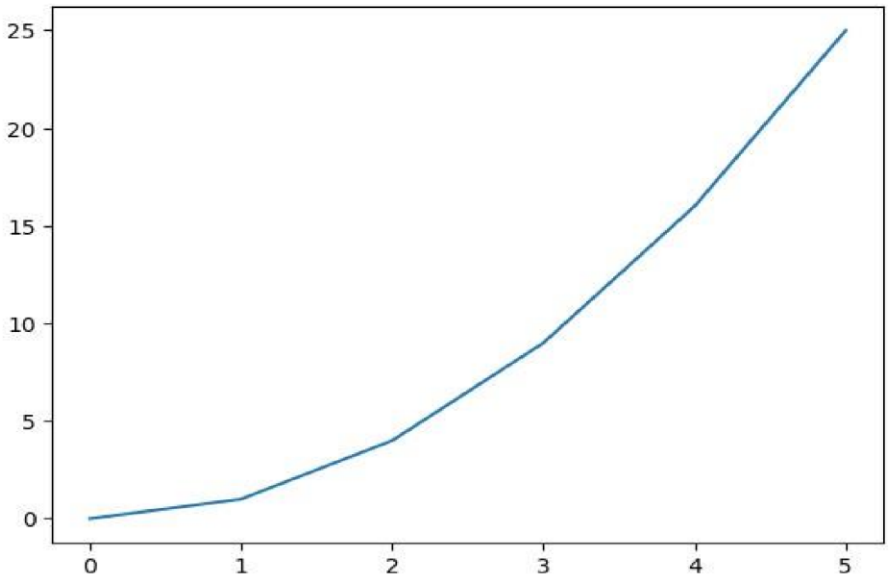
YearsExpe	Salary
1.1	39343
1.3	46205
1.5	37731
2	43525
2.2	39891
2.9	56642
3	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218
4	55794
4	56957
4.1	57081
4.5	61111
4.9	67938
5.1	66029
5.3	83088
5.9	81363
6	93940
6.8	91738
7.1	98273
7.9	101302
8.2	113812
8.7	109431
9	105582
9.5	116969
9.6	112635
10.3	122391
10.5	121872

Name: Omkar Karlekar
Roll no. : 644
Batch : F3

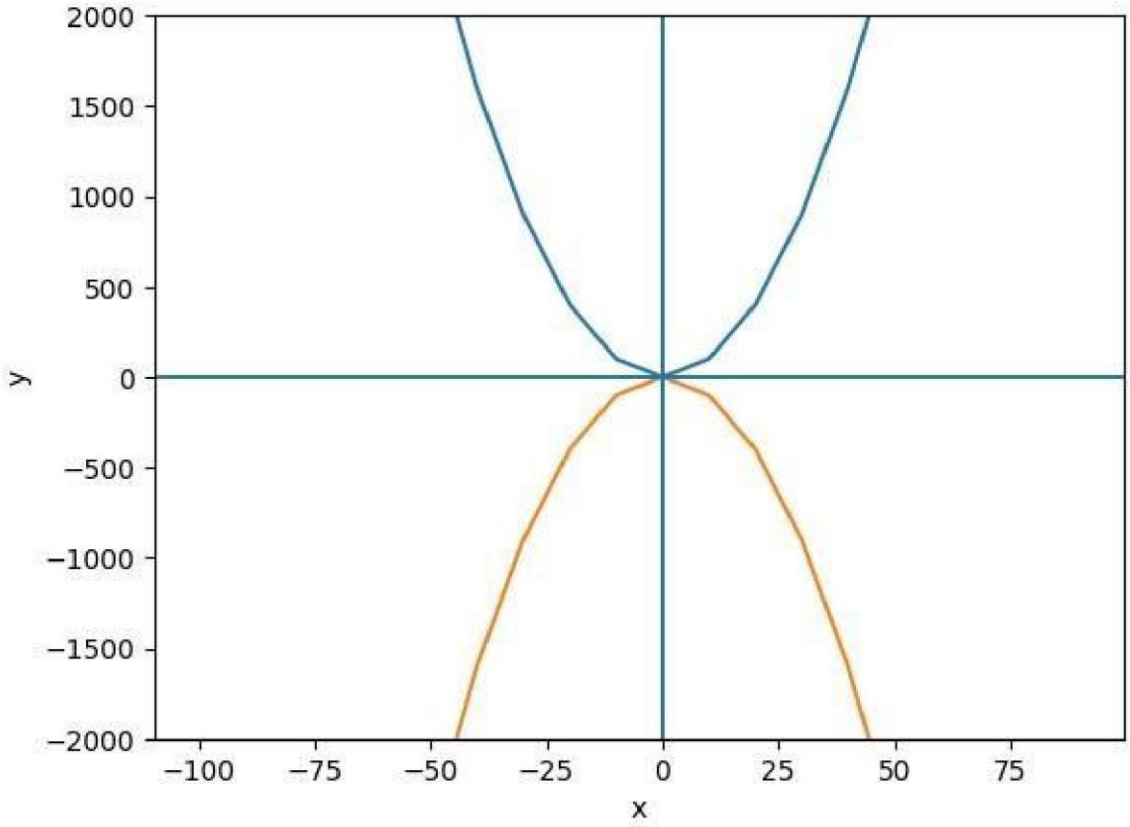
In [4]:

```
import matplotlib.pyplot as plt
x_values=[0,1,2,3,4,5]
y_values=[0,1,4,9,16,25]

plt.plot(x_values, y_values)
plt.show()
```



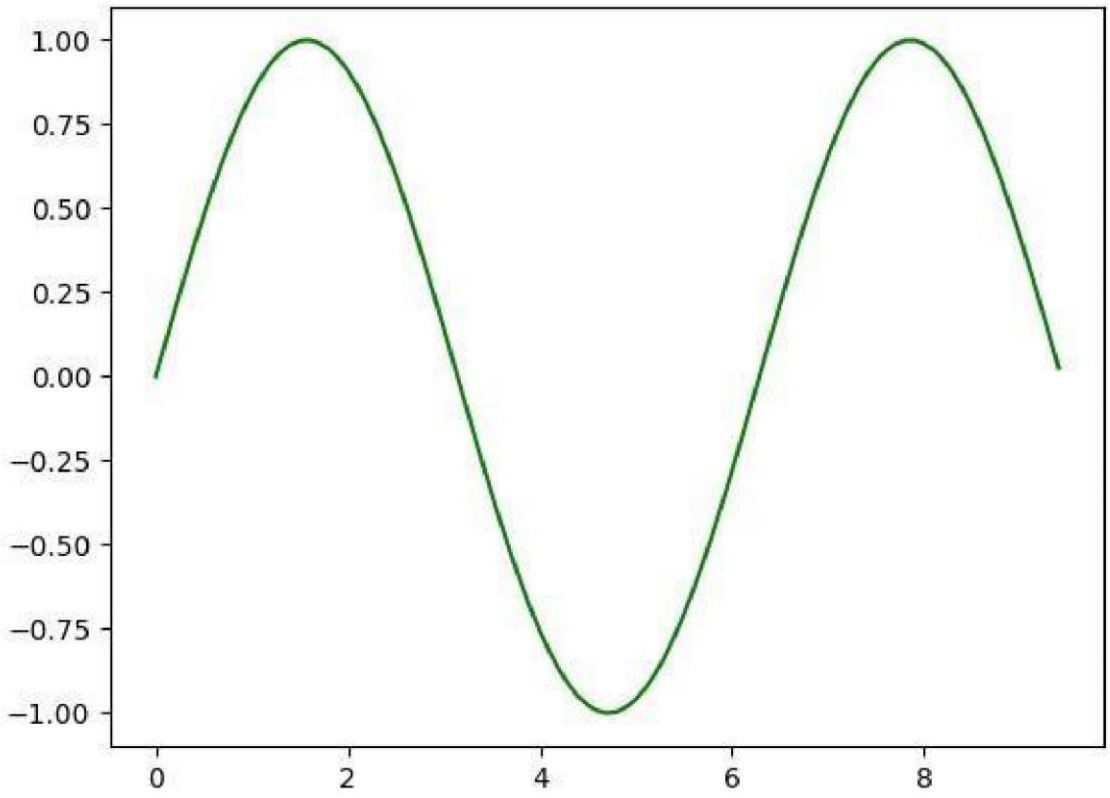
In [5]:

```
y1=[]
y2=[]
x=range(-100,100,10)
for i in x:y1.append(i**2)
for i in x:y2.append(-i**2)

plt.plot(x,y1)
plt.plot(x,y2)
plt.xlabel("x")
plt.ylabel("y")
plt.ylim(-2000,2000)
plt.axhline(0)
plt.axvline(0)

plt.show()
```



In [6]:

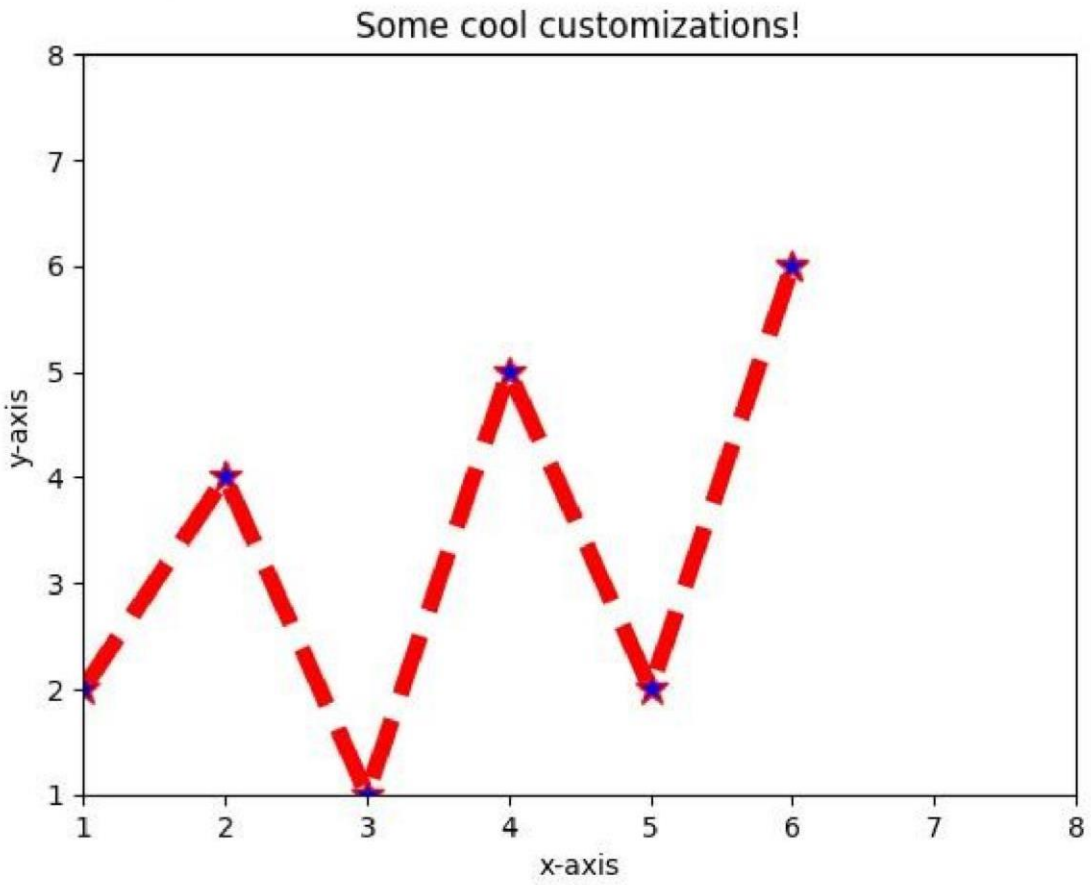
```
import numpy as np
x=np.arange(0,3*np.pi,0.1)
y1=np.sin(x)
plt.plot(x,y1,'green')
plt.show()
```



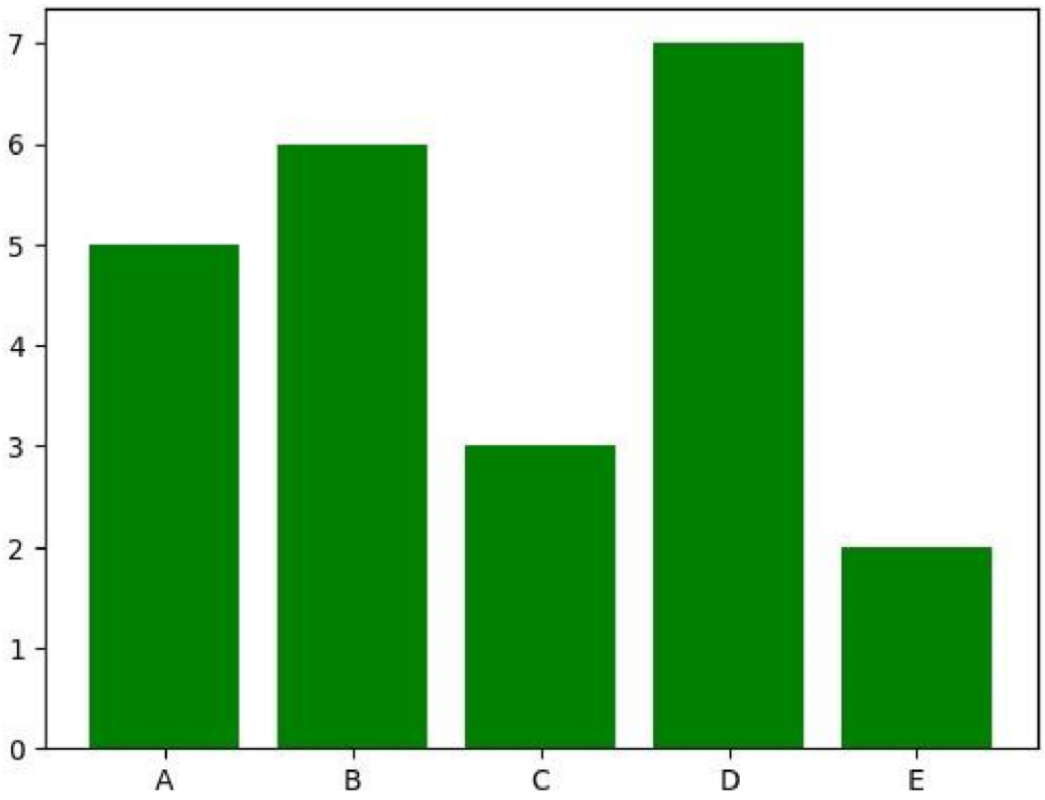
In [8]:

```
import matplotlib.pyplot as plt

#x axis values
x=[1,2,3,4,5,6]
#corresponding y axis values
y=[2,4,1,5,2,6]

#plotting the points
plt.plot(x, y, color='r',linestyle='dashed',linewidth=6,marker='*'

#setting x and y axis range
plt.ylim(1,8)
plt.xlim(1,8)

#namingthe x axis
plt.xlabel('x-axis')
#naming the y asix
plt.ylabel('y-axis')

#giving a title to my graph
plt.title('Some cool customizations!')

#function to show the plot
plt.show
```

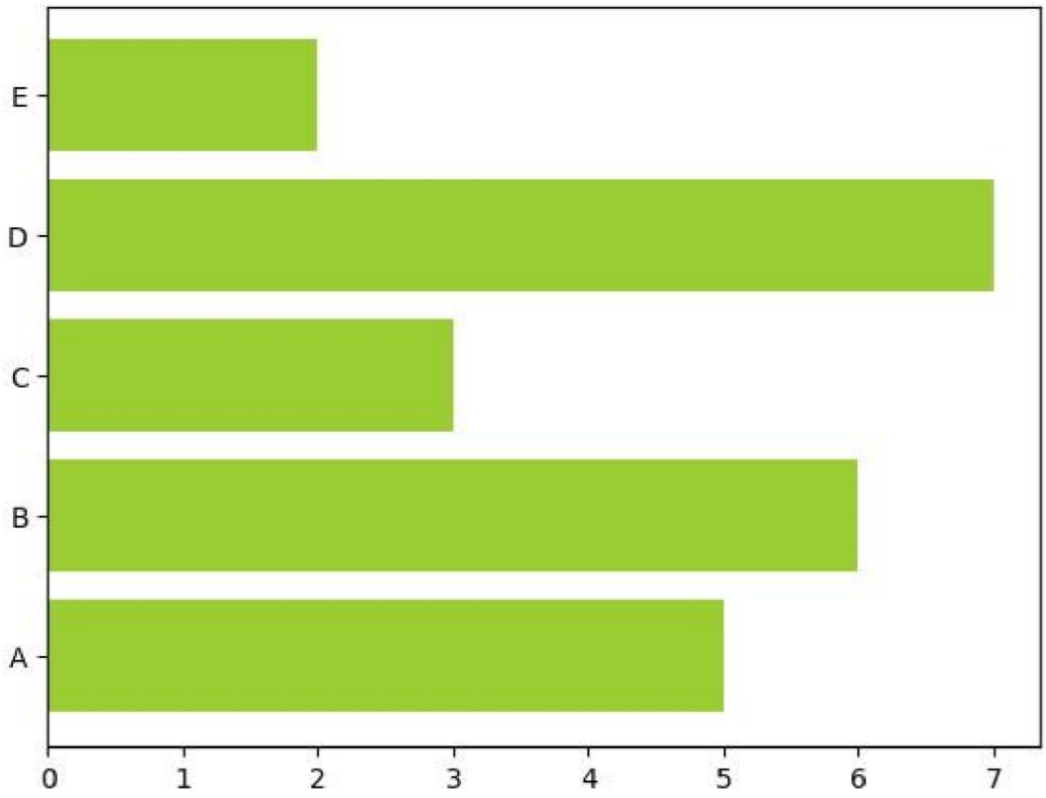
Out [8]: <function matplotlib.pyplot.show(close=None, block=None)>



In [9]:
```
import matplotlib.pyplot as plt

#create data for plotting
values=[5,6,3,7,2]
names=["A","B","C","D","E"]
plt.bar(names,values,color='g')
plt.show()
```

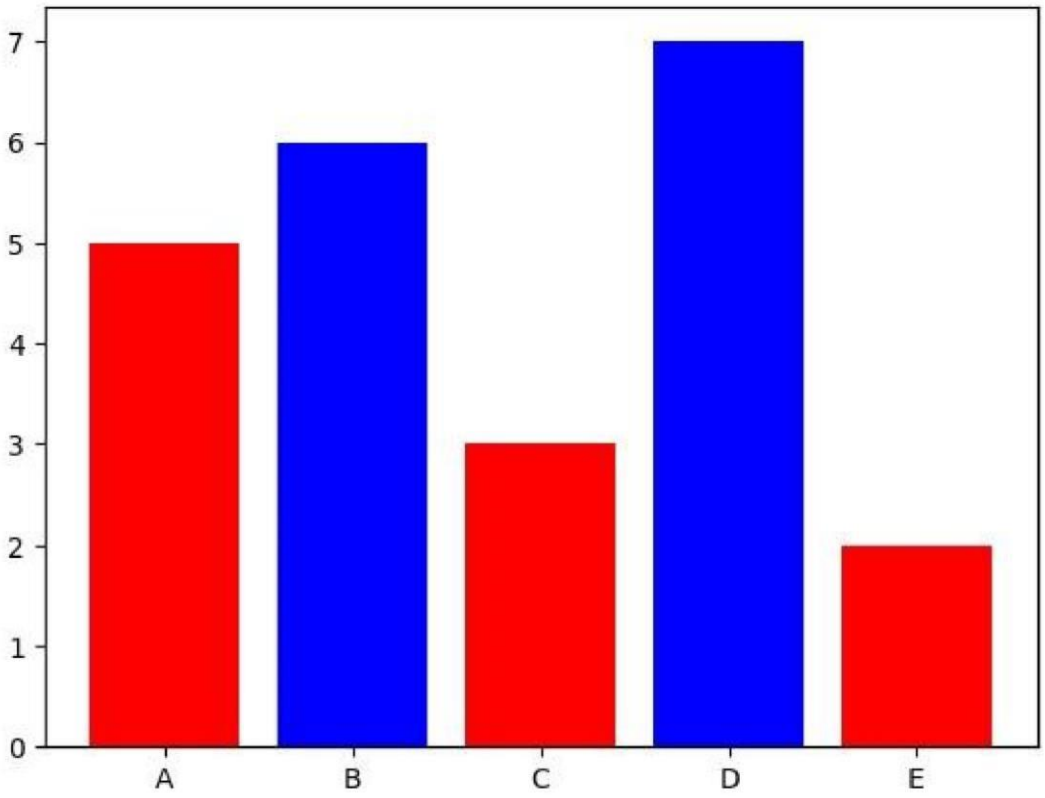


In [10]:
```
plt.barh(names,values,color="yellowgreen")
plt.show()
```



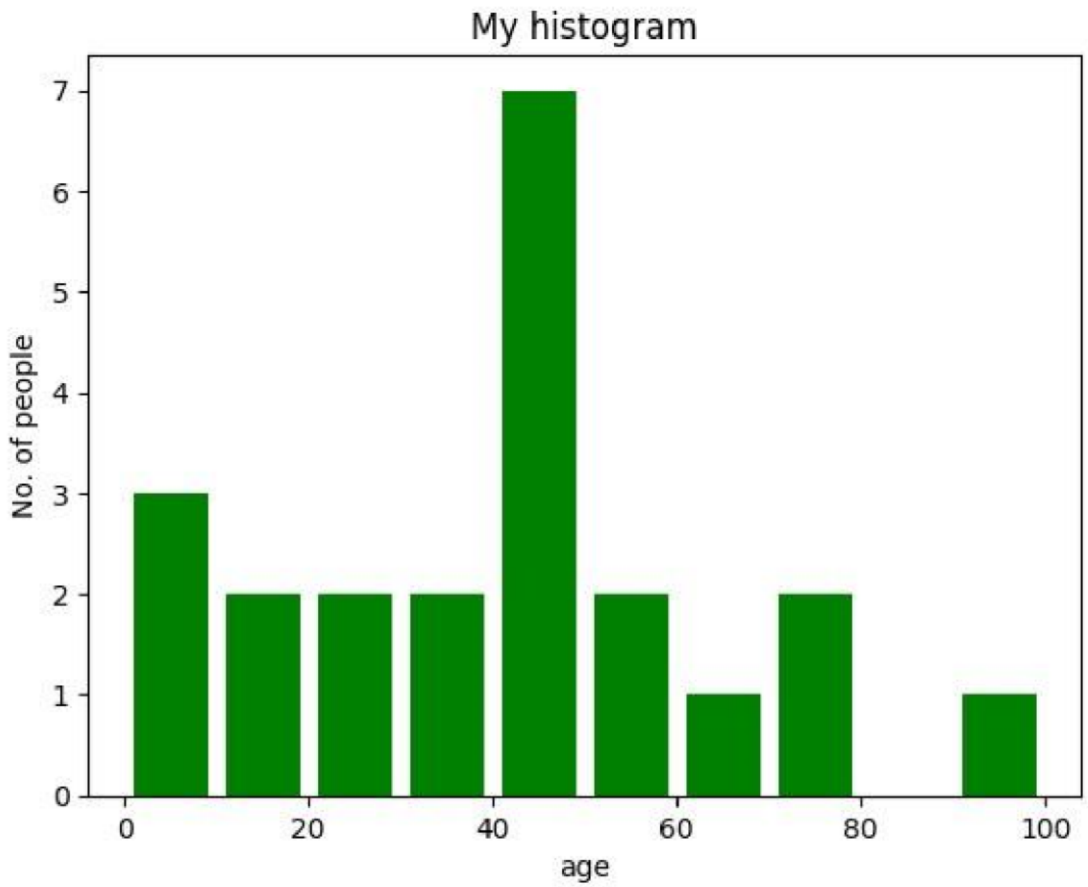
In [14]:

```
import matplotlib.pyplot as plt

#create data for plotting
values=[5,6,3,7,2,]
names=["A","B","C","D","E"]
c1=['r','b']
plt.bar(names,values,color=c1)
plt.show()
```



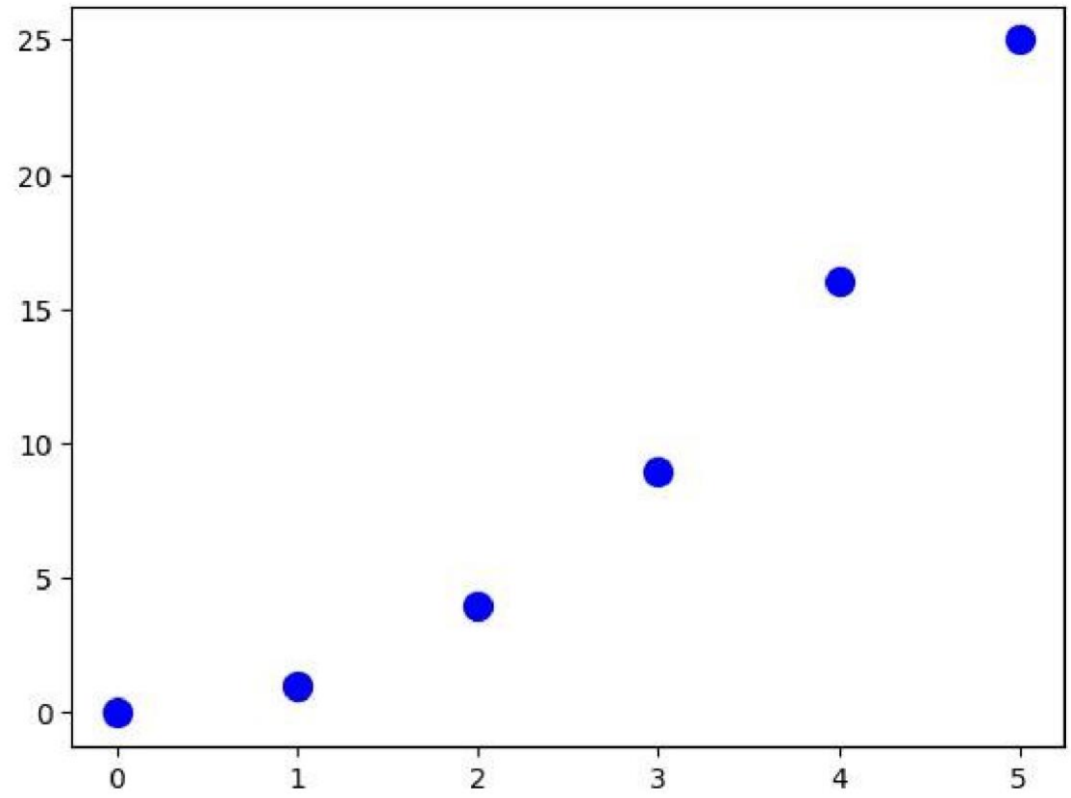
In [16]:

```
import matplotlib.pyplot as plt
#frequencies
ages=[2,5,70,40,30,45,50,45,43,40,44,60,7,13,57,18,90,77,32,21,20,4

#setting the ranges and no. of intervals
range=(0,100)
bins=10

#plotting a histogram
plt.hist(ages,bins,range,color='g',histtype='bar',rwidth=0.8)

#x axis label
plt.xlabel('age')
#frequencylabel
plt.ylabel("No. of people")
#plot title
plt.title('My histogram')

#function to show the plot
plt.show()
```



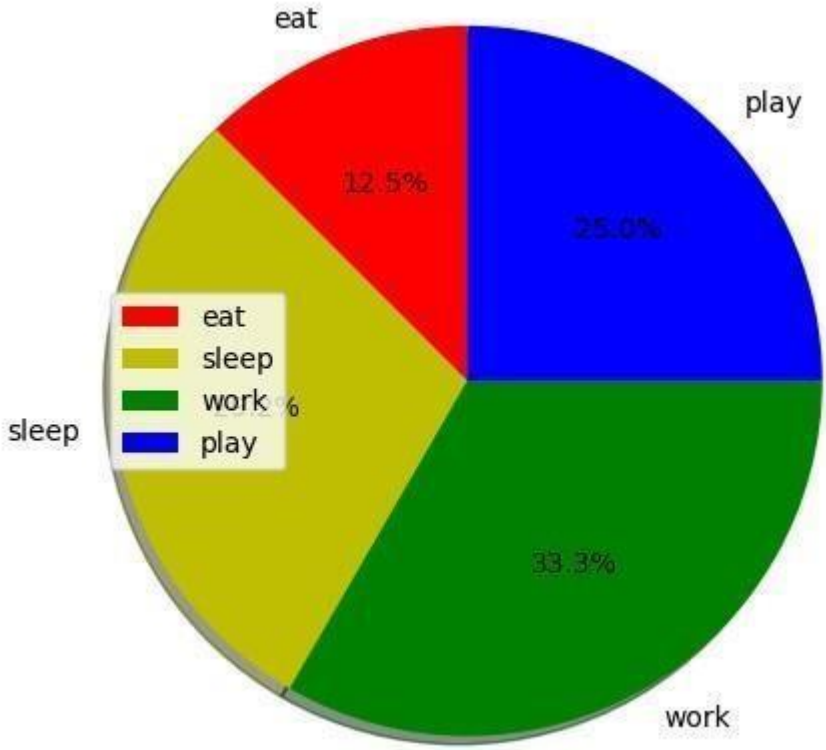
In [17]:

```
#scatter plot

x_values=[0,1,2,3,4,5]
y_values=[0,1,4,9,16,25]
plt.scatter(x_values,y_values,s=100,color='b')
plt.show()
```



In [18]:

```
#pie chart
import matplotlib.pyplot as plt

#defining labels
activities=['eat','sleep','work','play']

#portion covered by each label
slices=[3,7,8,6]

#color for each label
colors=['r','y','g','b']

#plotting the pie chart
plt.pie(slices,labels = activities, colors=colors, startangle=90,

#plotting legend
plt.legend()

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


eat
play
12.5%
25.0%
eat
sleep
work
play
sleep
work
33.3%

In [3]:

```
import numpy as nm
import matplotlib.pyplot as mplt
import pandas as pd

data_set=pd.read_csv('/content/sample_data/salary_data.csv')
data_set
```

Out [3]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582

26	9.5	116969	
27	9.6	112635	

	YearsExperience	Salary
28	10.3	122391
29	10.5	121872

In [4]:
```
x=data_set.iloc[:,:-1].values
y=data_set.iloc[:,1].values
print(x)
print(y)
```

```
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]
 [ 2.9]
 [ 3. ]
 [ 3.2]
 [ 3.2]
 [ 3.7]
 [ 3.9]
 [ 4. ]
 [ 4. ]
 [ 4.1]
 [ 4.5]
 [ 4.9]
 [ 5.1]
 [ 5.3]
 [ 5.9]
 [ 6. ]
 [ 6.8]
 [ 7.1]
 [ 7.9]
 [ 8.2]
 [ 8.7]
 [ 9. ]
 [ 9.5]
 [ 9.6]
 [10.3]
 [10.5]]
[ 39343  46205  37731  43525  39891  56642  60150  54445  64445 57189
  63218  55794  56957  57081  61111  67938  66029  83088  81363  93940
  91738  98273 101302 113812 109431 105582 116969 112635 122391 121872]
```

Step 1: Splitting the dataset into training and test set

In[5]:
```
#splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size
print(x_train)
```

```
[[ 2.9]
 [ 5.1]
 [ 3.2]
 [ 4.5]
 [ 8.2]
 [ 6.8]
 [ 1.3]
 [10.5]
 [ 3. ]
 [ 2.2]
 [ 5.9]
 [ 6. ]
 [ 3.7]
```

```
[ 3.2]
[ 9. ]
[ 2. ]
[ 1.1]
[ 7.1]
[ 4.9]
[ 4. ]]
```

Step 2: Fitting the simple linear regression model to the training dataset

In [6]:
```
#fitting the simple linear regression model to the training datase
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)
```

Out [6]:
```
▾ LinearRegression
LinearRegression()
```

Step 3: Prediction of test and training set result

In [7]:
```
#Prediction of test and Training set result
y_pred= regressor.predict(x_test)
x_pred= regressor.predict(x_train)
print(x_pred)
print(y_pred)
```

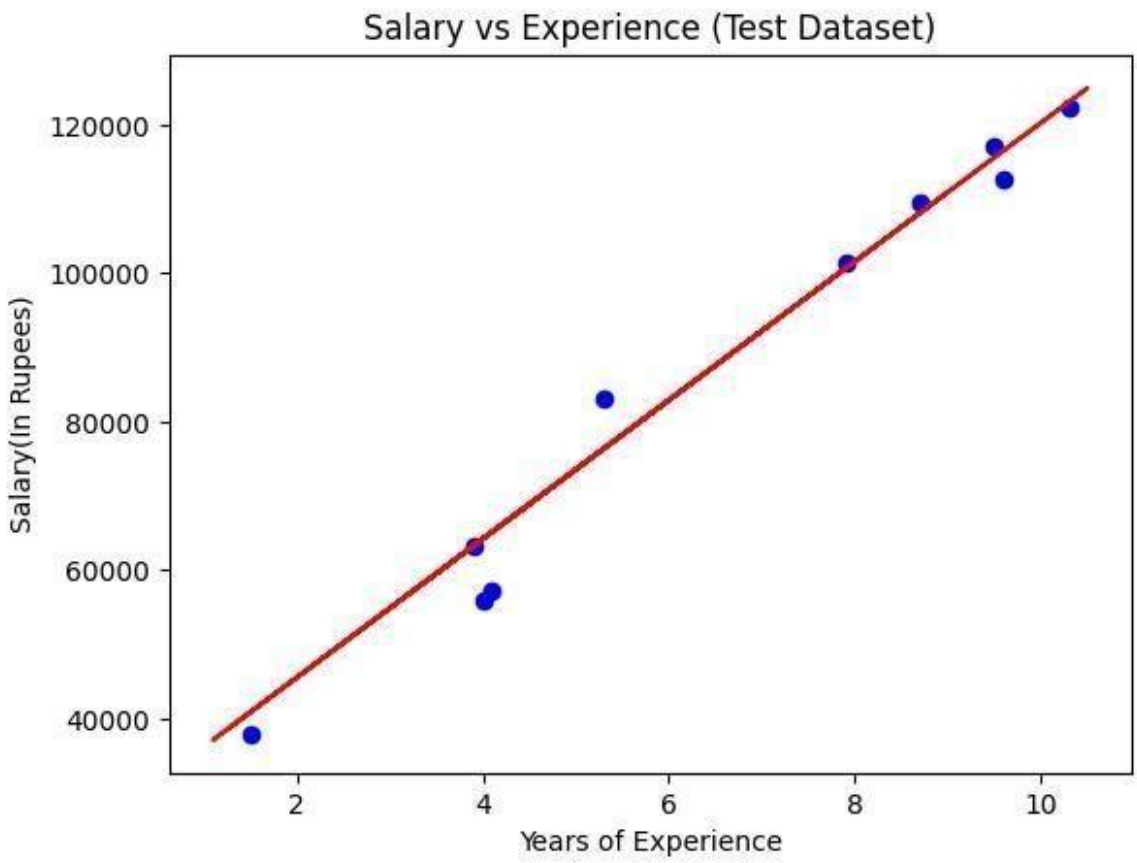
```
[ 53919.42532909  74480.49870396  56723.20806202  68872.93323808
 103452.92027763  90368.60085726  38965.91742009 124948.58789682
  54854.0195734   47377.2656189   81957.25265845  82891.84690277
  61396.17928358  56723.20806202 110929.67423213  45508.07713028
  37096.72893147  93172.3835902   72611.31021533  64199.96201652]
[ 40835.10590871 123079.39940819  65134.55626083  63265.36777221
 115602.64545369 108125.8914992  116537.23969801  64199.96201652
  76349.68719258 100649.1375447 ]
```

Step 4: Visualizing

In [9]:
```
import matplotlib.pyplot as mtp
mtp.scatter(x_train, y_train,color="green")
mtp.plot(x_train, x_pred, color="red")
mtp.title("Salary vs Experience (Training Dataset)")
mtp.xlabel("Years of Experience")
mtp.ylabel("Salary(In Rupees)")
mtp.show()
```

In [10]:

```
#visualizing the test set results
mtp.scatter(x_test, y_test, color="blue")
mtp.plot(x_train, x_pred, color="red")
mtp.title("Salary vs Experience (Test Dataset)")
mtp.xlabel("Years of Experience")
mtp.ylabel("Salary(In Rupees)")
mtp.show()
```


Salary vs Experience (Test Dataset)
Salary(In Rupees)
120000
100000
80000
60000
40000
2
4
6
8
10
Years of Experience

Name-Omkar Karlekar
Roll-644
PRN-202201090088
Batch-F3

```
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/drive/MyDrive/Colab Notebooks/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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
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()
```

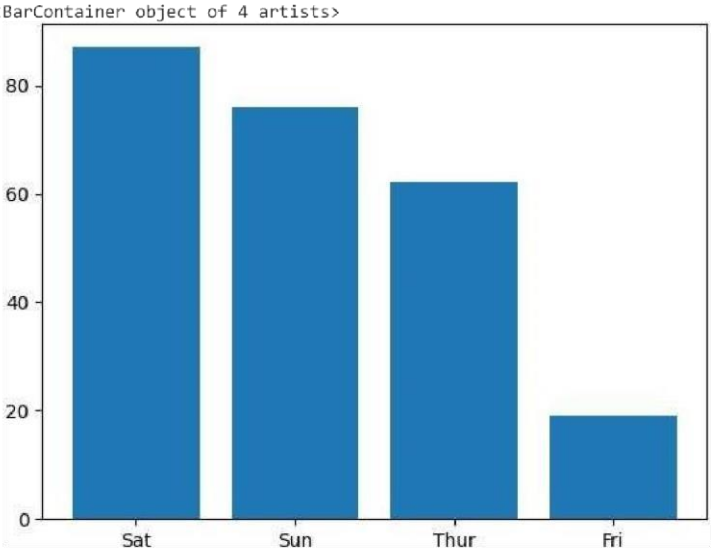
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    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()
```

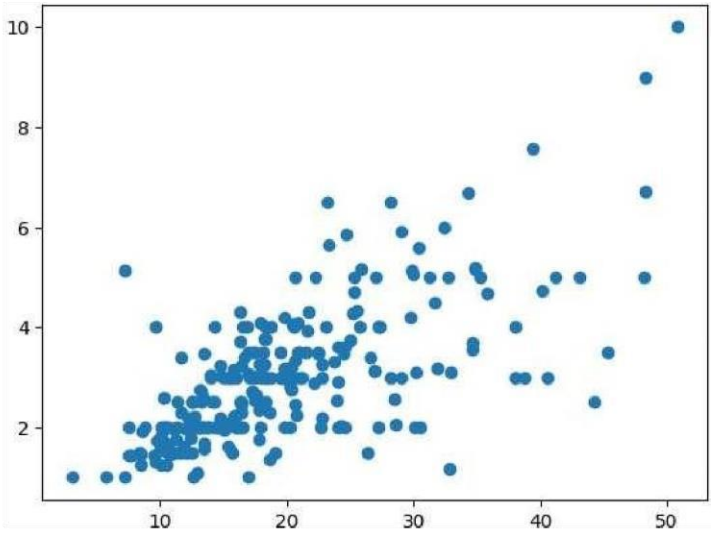
	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000

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

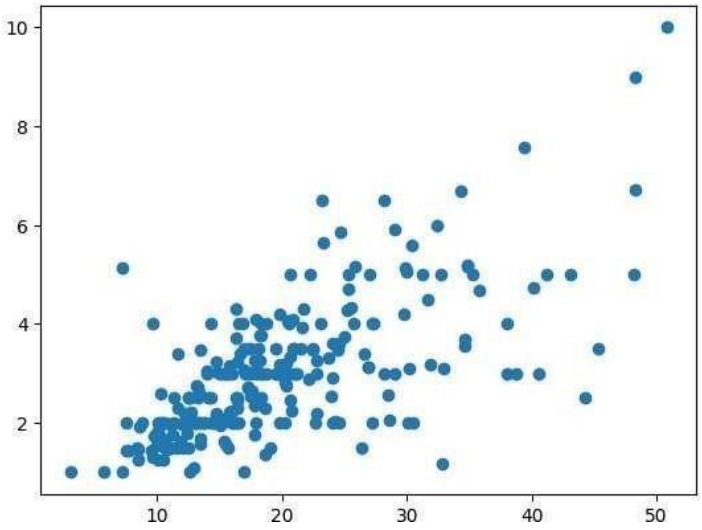
```
<BarContainer object of 4 artists>
```

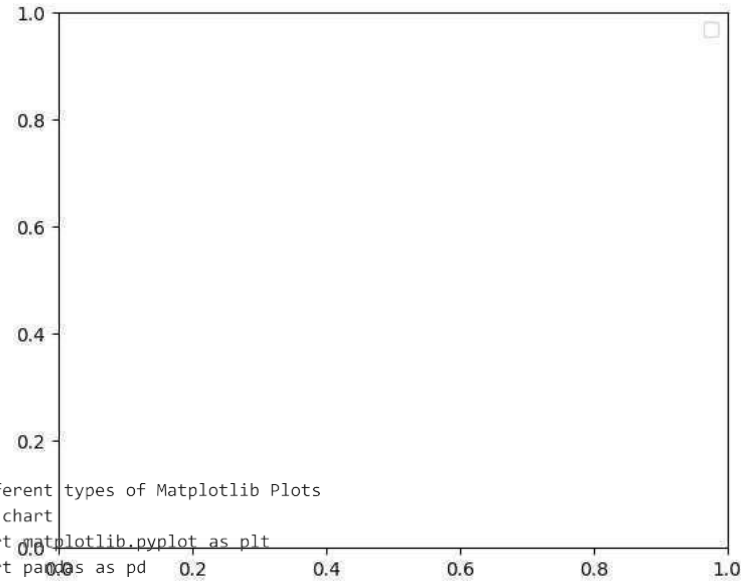



```
plt.scatter(df1['total_bill'],df1['tip'])
plt.show()
```

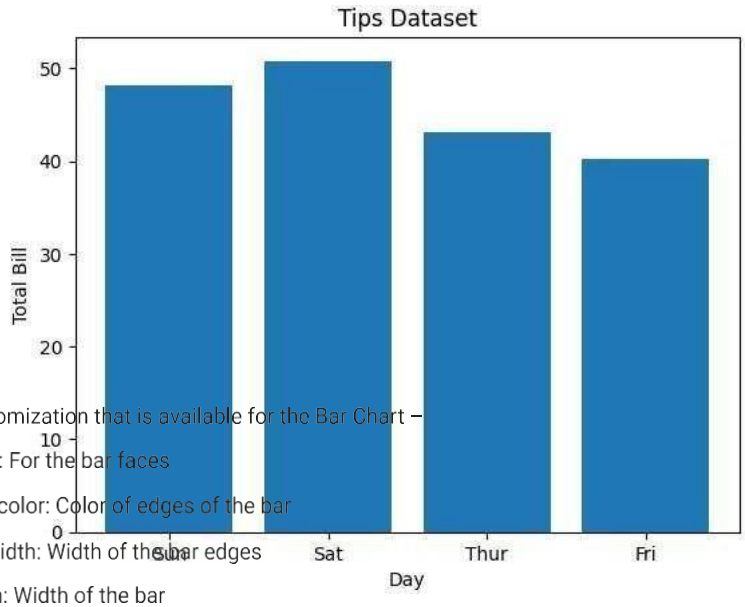



```
plt.scatter(x='total_bill',y='tip',data=df1)
fig=plt.figure(figsize=(5,4))
ax=fig.add_axes([1,1,1,1])
ax.legend(labels=('sun','mon','tue'))
plt.show()
```






```
#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/sample_data/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

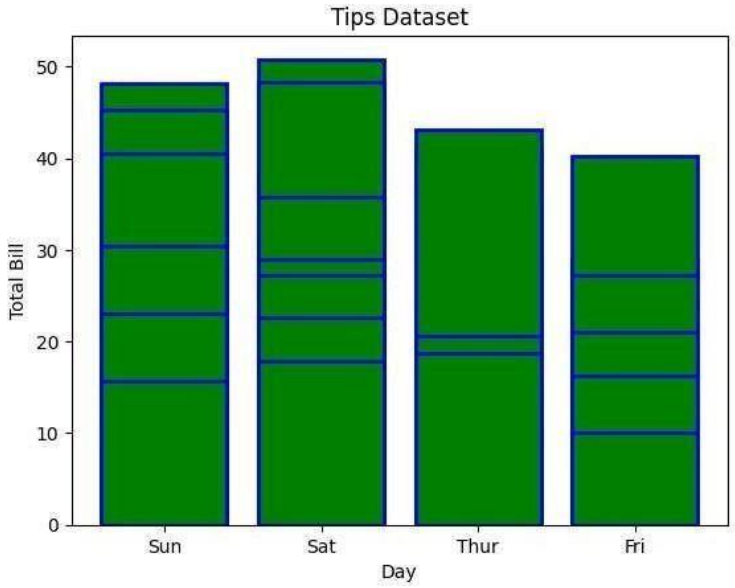
edgecolor: Color of edges of the bar

linewidth: Width of the bar edges

width: Width of the bar

```
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()
```



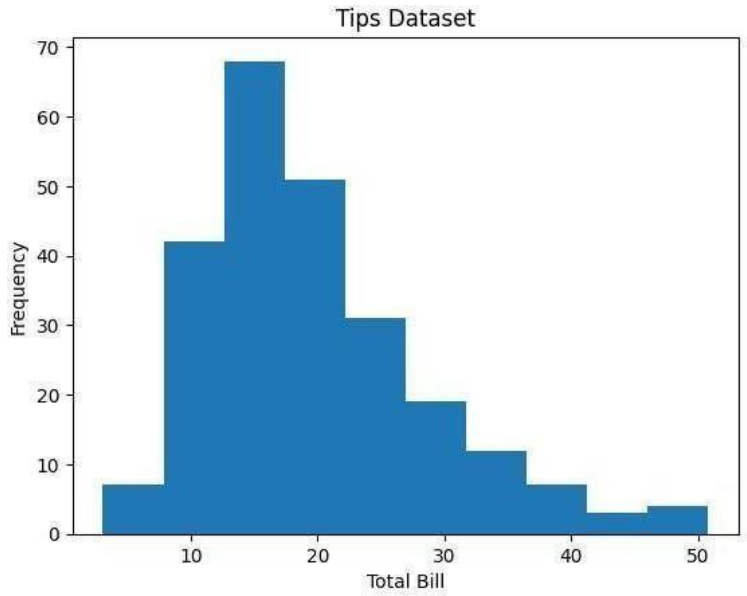
Histogram A histogram is basically used to represent data provided in a form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The hist() function is used to compute and create histogram of x.

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

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

# plotting the data
plt.hist(x, 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')
```

https://colab.research.google.com/drive/13m_L3-SRAFdfFelOLc0h5MQIA2RhHMEw#scrollTo=bqsKn8hATNLe&printMode=true

```
# 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()
```

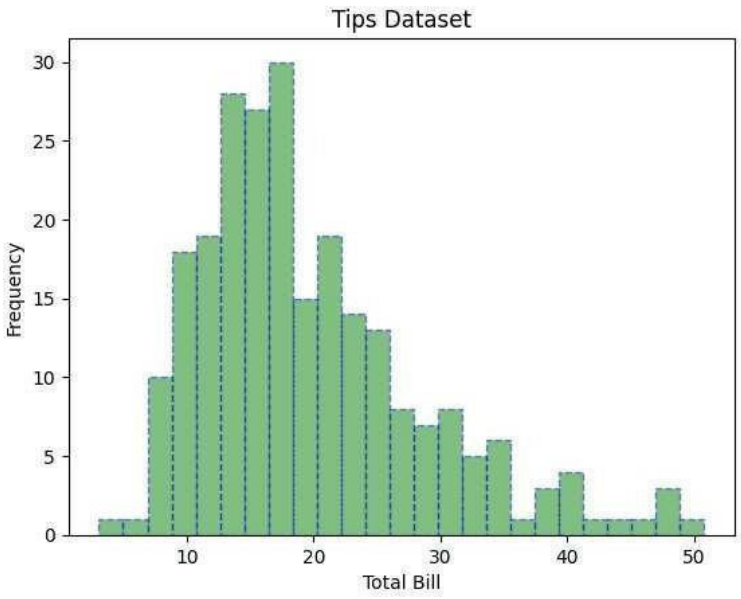


Customization that is available for the Histogram –

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)

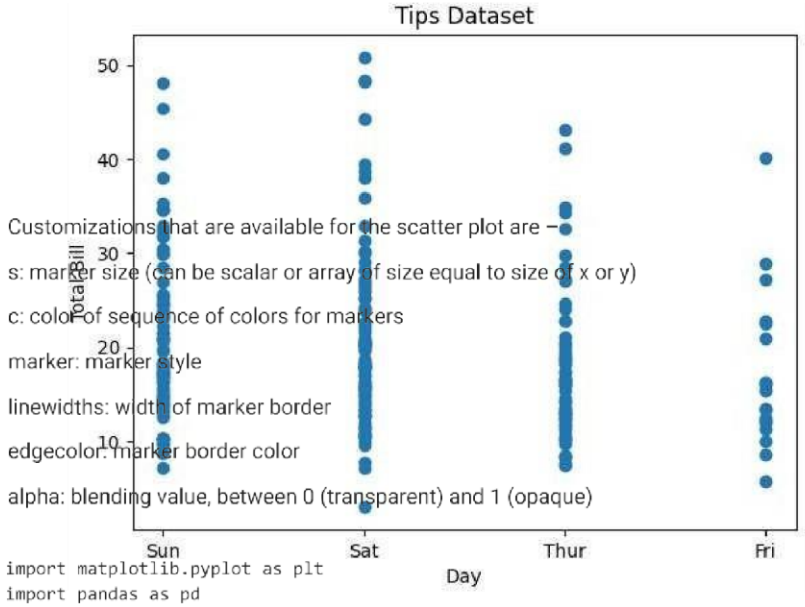
```
import matplotlib.pyplot as plt
import pandas as pd
```

```
# 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

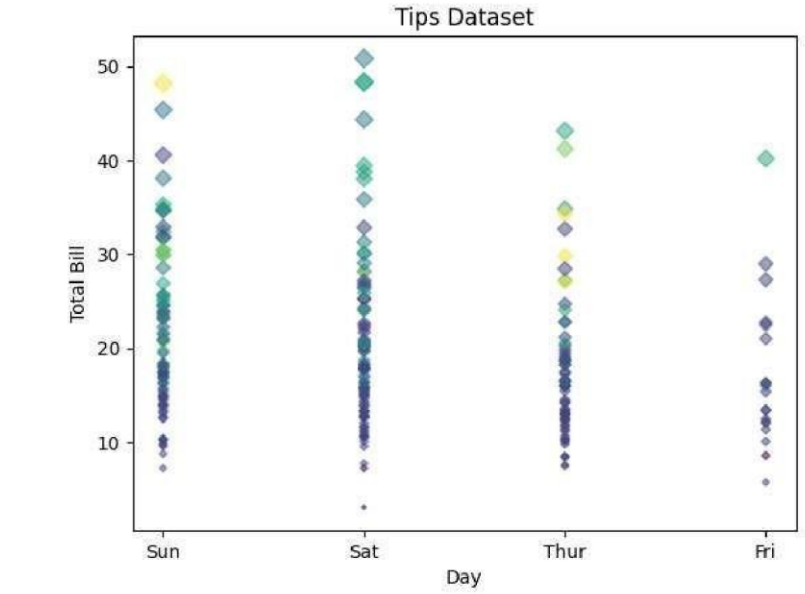
marker: marker style

linewidths: width of marker border

edgecolor: marker border color

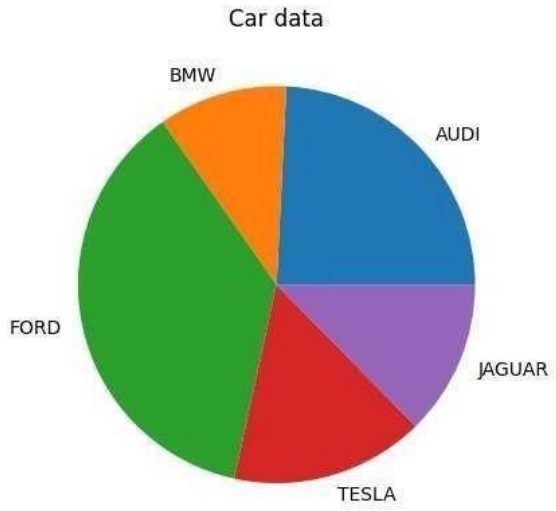
alpha: blending value, between 0 (transparent) and 1 (opaque)

```
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.

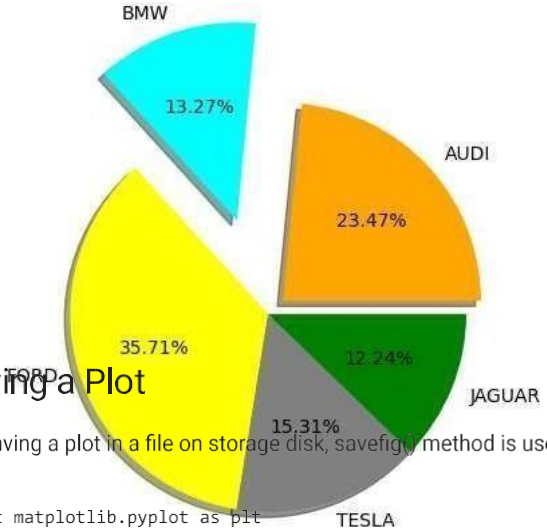
```
import matplotlib.pyplot as plt
import pandas as pd


# initializing the data
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR',]
data = [23, 10, 35, 15, 12]

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

# Adding title to the plot
plt.title("Car 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.

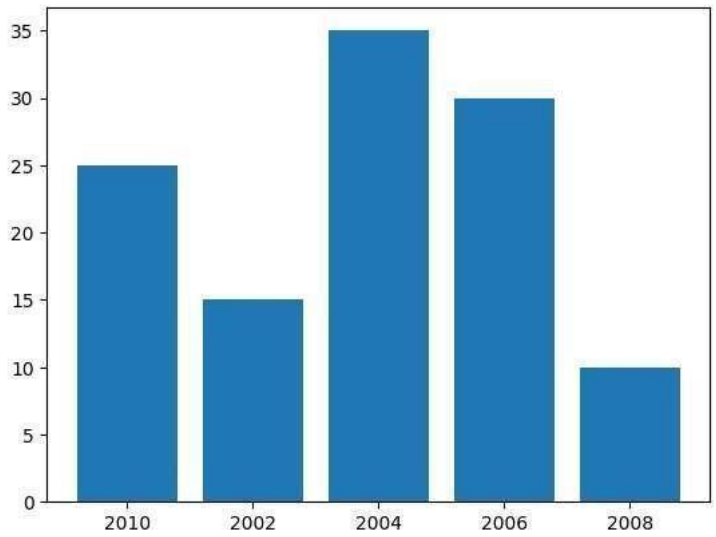
```
import matplotlib.pyplot as plt
import pandas as pd

# initializing the data
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR',]
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=cars, 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 = ['2010', '2002', '2004', '2006', '2008']
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)
```



✓ 0s completed at 2:48 PM