Name:- Anuradha Desale

Practical:- 6

Roll no:-14 Sub:-DV

# Program/Notebook to draw Box plot, Pie Chart & Scatter plot, Heat Maps & Histogram

#Import Requied Libraries import matplotlib.pyplot as plt import numpy as np

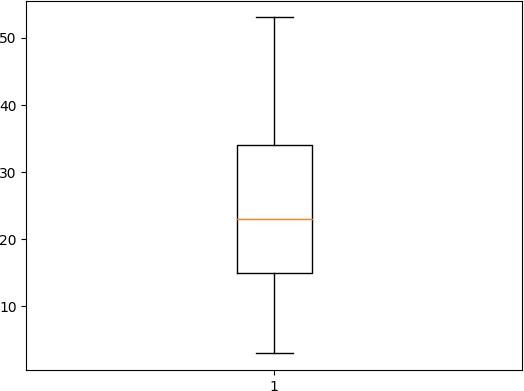
import pandas as pd

1)Box Plot

data = np.array([12,32,53,12,32,18,3,5,34,23,43,23,45,23,34])

plt.boxplot(data)

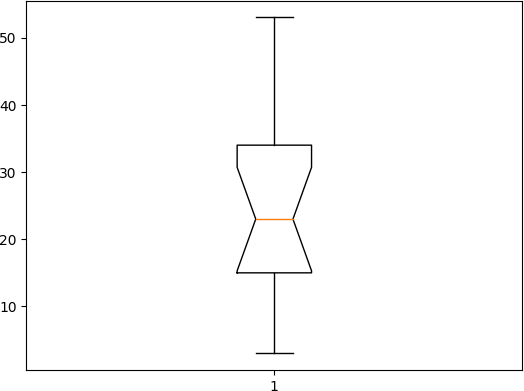
#fig = plt.figure(figsize = (10, 8)) #figsize :- atribute allows us to specify the width and height if figure in unit inches plt.show()



data = np.array([12,32,53,12,32,18,3,5,34,23,43,23,45,23,34])

plt.boxplot(data, notch = "true") #Notch is opetional parameter that accepts boolen values (Ture, Falase). If ture, it creates notched boxplot fig = plt.figure(figsize = (5, 2)) #figsize :- atribute allows us to specify the width and height if figure in unit inches

plt.show()



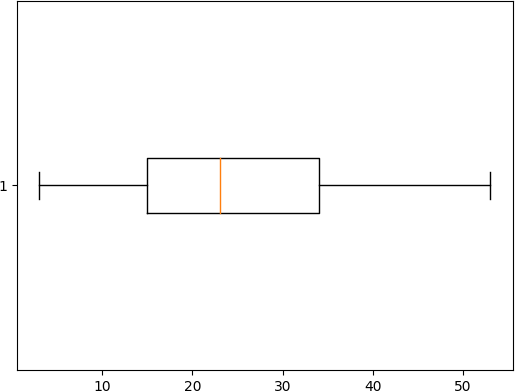
<Figure size 500x200 with 0 Axes>

plt.boxplot(data, vert= 0) #vert accept a boolen value eihter false(0) or True (1) . it specifies whether the boxplot is horizontal or vertical

{'whiskers': [<matplotlib.lines.Line2D at 0x26721cba230>,

<matplotlib.lines.Line2D at 0x26721cba4d0>], 'caps': [<matplotlib.lines.Line2D at 0x26721cba770>,

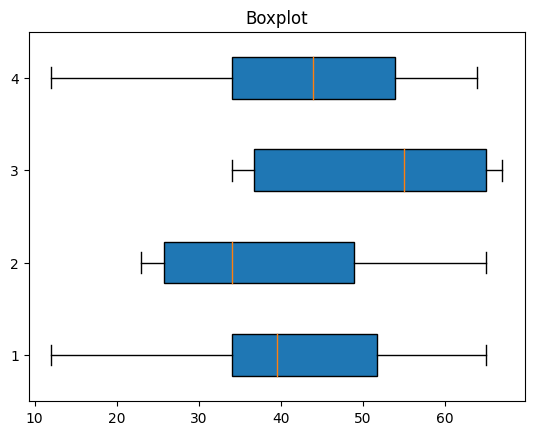
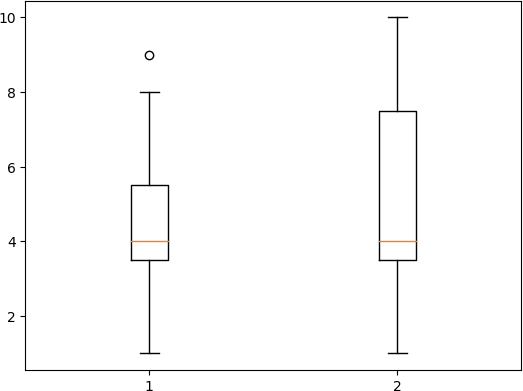
<matplotlib.lines.Line2D at 0x26721cbaa10>],

'boxes': [<matplotlib.lines.Line2D at 0x26721cb9f90>], 'medians': [<matplotlib.lines.Line2D at 0x26721cbacb0>], 'fliers': [<matplotlib.lines.Line2D at 0x26721cbaf50>], 'means': []}

df = pd.read\_csv("bxplt.csv") df

|  |  |
| --- | --- |
| 1 | 0 |
| 0 3 | 2 |
| 1 4 | 1 |
| 2 4 | 7 |
| 3 2 | 9 |
| 4 4 | 10 |
| 5 4 | 8 |
| 6 1 | 6 |
| 7 8 | 4 |
| 8 4 | 4 |
| 9 9 | 3 |
| 10 7 | 4 |

plt.boxplot(df) plt.show()



value1 = [12,34,45,34,54,65]

value2 = [34,34,23,23,54,65]

value3 = [45,34,65,67,34,65]

value4 = [64,34,54,34,54,12]

plt.title("Boxplot")

bxdata = [value1, value2, value3,value4]

box = plt.boxplot(bxdata, vert= 0, patch\_artist=True) plt.show()

value1 = [12,34,45,34,54,65]

value2 = [34,34,23,23,54,65]

value3 = [45,34,65,67,34,65]

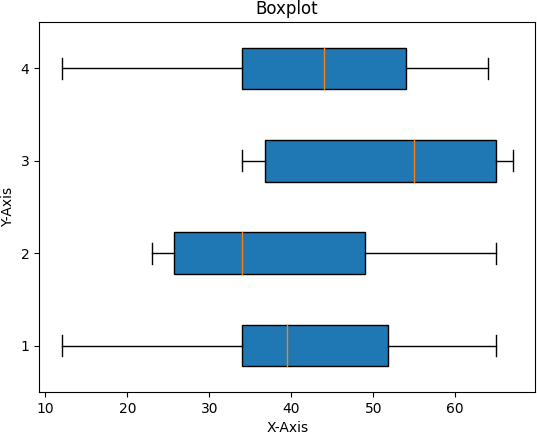
value4 = [64,34,54,34,54,12]

plt.title("Boxplot")

bxdata = [value1, value2, value3,value4]

box = plt.boxplot(bxdata, vert= 0, patch\_artist=True) plt.xlabel("X-Axis")

plt.ylabel('Y-Axis') plt.show()



2) Pie Chart

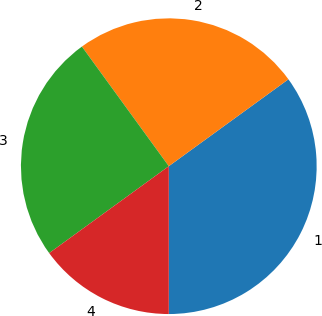
y = np.array([35, 25, 25, 15]) #As We can see the pie chart draws one piece (called a wedge) for each value in the array (in this case [35, 25, 25, 15]). plt.pie(y)

plt.show()

y = np.array([35, 25, 25, 15]) #Lets suppose Total is 100% which is divided into 4 values, which are 35, 25, 25, 15

mylabels = ['1', '2', '3', '4']

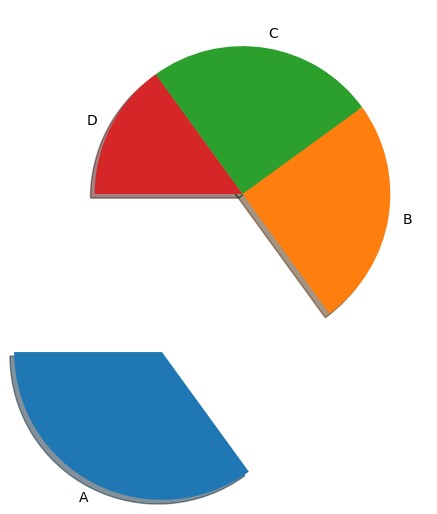
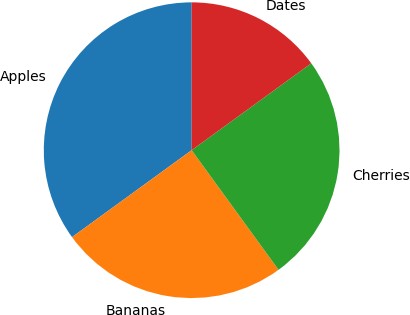
plt.pie(y, labels = mylabels, startangle = 270) #specifying start angle for chart plt.show()



y = np.array([35, 25, 25, 15])

mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels, startangle = 90) plt.show()



y = np.array([35, 25, 25, 15]) mylabels = ['A', 'B', 'C', 'D'] myexplode = [1.2, 0, 0,0]

#Lets suppose Total is 100% which is divided into 4 values, which are 35, 25, 25, 15

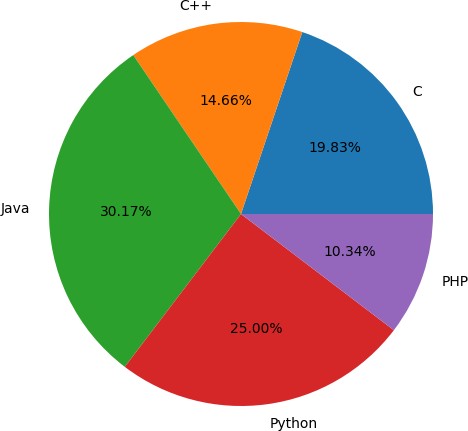
plt.pie(y, labels = mylabels, startangle = 180, explode = myexplode, shadow=True) #we can explode piece of chart and add shadow plt.show()

fig = plt.figure()

ax = fig.add\_axes([0,0,1,1]) ax.axis('equal')

langs = ['C', 'C++', 'Java', 'Python', 'PHP'] students = [23,17,35,29,12]

ax.pie(students, labels = langs,autopct='%1.2f%%') #The proportionate percentage is displayed inside the respective wedge with the help of autopct parameter which is s plt.show()



3) Scatter plot

x =[5, 7, 8, 7, 2, 17, 2, 9,

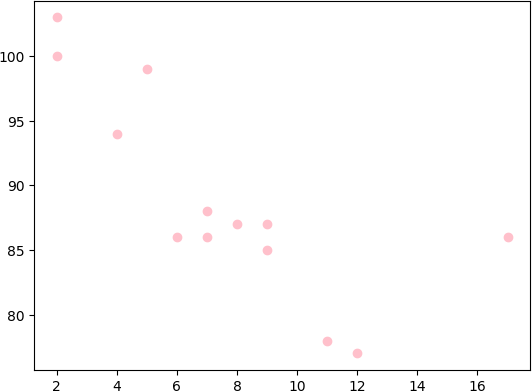
4, 11, 12, 9, 6]

y =[99, 86, 87, 88, 100, 86,

103, 87, 94, 78, 77, 85, 86]

plt.scatter(x, y, c ="pink")

# To show the plot plt.show()



#Draw two plots on the same figure:

#day one, the age and speed of 13 cars:

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])

y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

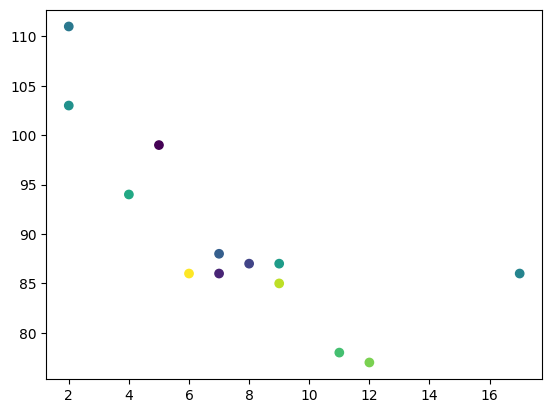
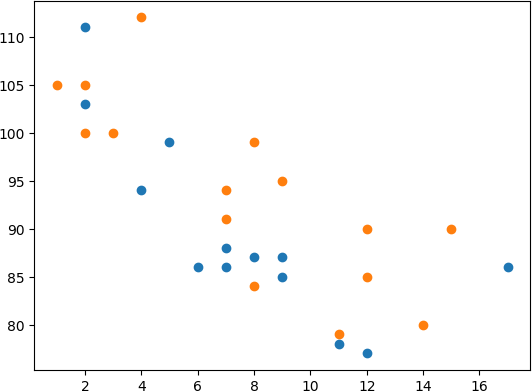
plt.scatter(x, y)

#day two, the age and speed of 15 cars:

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])

y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])

plt.scatter(x, y) plt.show()



x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])

y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='viridis') plt.show()

#Scatter plot with different shape and colour for two datasets. # dataset-1

x1 = [89, 43, 36, 36, 95, 10,

66, 34, 38, 20]

y1 = [21, 46, 3, 35, 67, 95,

53, 72, 58, 10]

# dataset2

x2 = [26, 29, 48, 64, 6, 5,

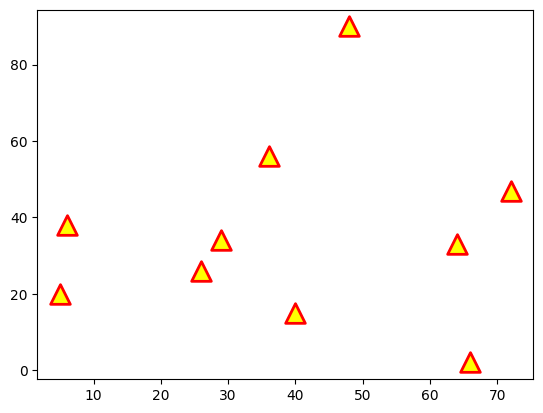
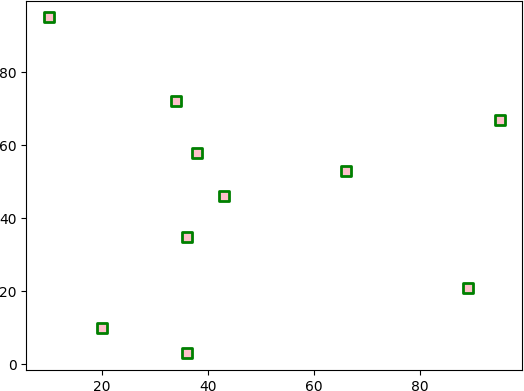
36, 66, 72, 40]

y2 = [26, 34, 90, 33, 38,

20, 56, 2, 47, 15]

plt.scatter(x1, y1, c ="pink", linewidths = 2, marker ="s", edgecolor ="green", s = 50)

<matplotlib.collections.PathCollection at 0x26734fc7dc0>



plt.scatter(x2, y2, c ="yellow", linewidths = 2,

marker ="^", edgecolor ="red", s = 200)

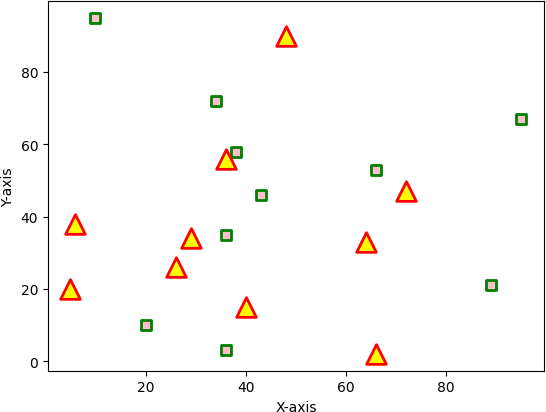
<matplotlib.collections.PathCollection at 0x26735153550>

plt.scatter(x1, y1, c ="pink", linewidths = 2, marker ="s", edgecolor ="green", s = 50)

plt.scatter(x2, y2, c ="yellow", linewidths = 2,

marker ="^", edgecolor ="red", s = 200)

plt.xlabel("X-axis") plt.ylabel("Y-axis") plt.show()



4) Heat Maps

import seaborn as sns

w1 = [20, 24, 25, 26, 30]

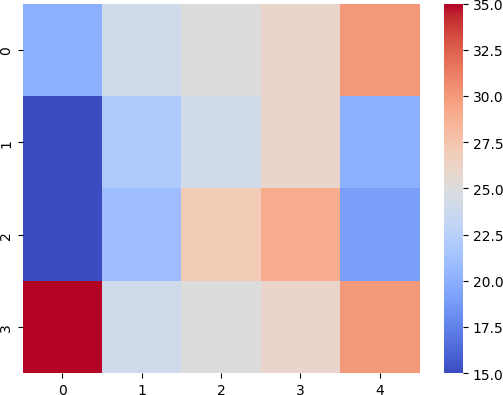
w2 = [15, 22, 24, 26, 20]

w3 = [15, 21, 27, 29, 19]

w4 = [35, 24, 25, 26, 30]

data = np.array([w1, w2, w3, w4])

#sns.heatmap(data) sns.heatmap(data, cmap='coolwarm') plt.show()



df = pd.read\_excel('heat.xlsx') df

|  |  |  |
| --- | --- | --- |
| Rank | Month | Day |
| 0 355 | 1 | 1 |
| 1 375 | 1 | 2 |
| 2 374 | 1 | 3 |
| 3 385 | 1 | 4 |

sns.heatmap(df)

<Axes: >

df\_corr = df.corr()

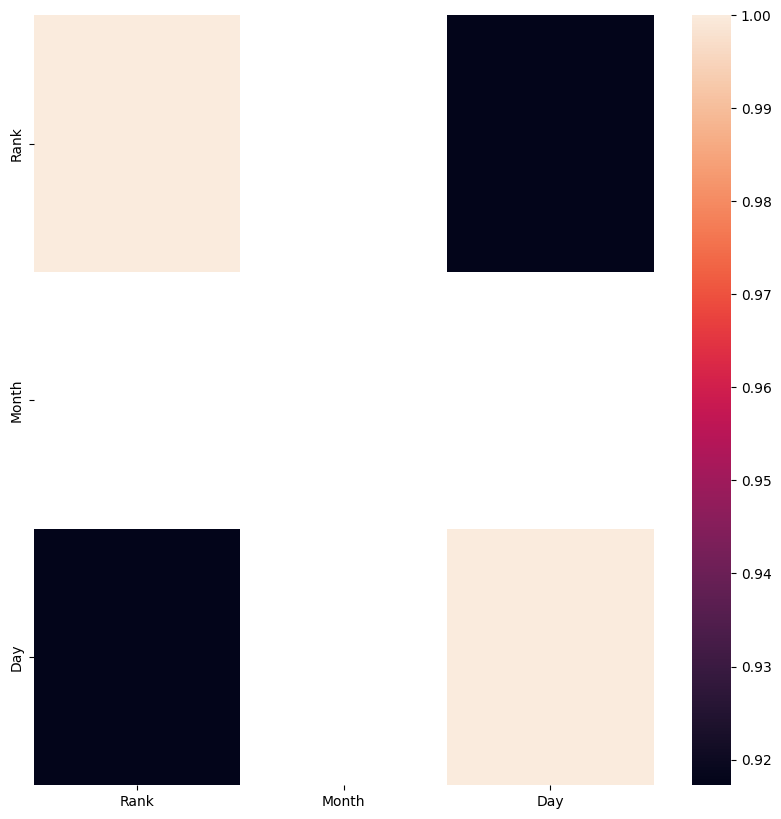
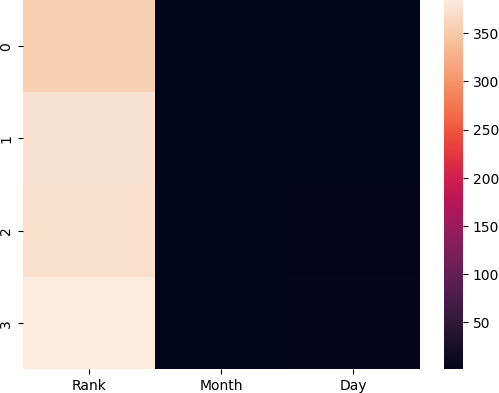


figure =plt.figure(figsize=(10, 10)) sns.heatmap(df\_corr)

#sns.heatmap(df\_corr, annot = True, fmt='.1g') plt.show()

5)Histogram

A histogram is a graph showing frequency distributions.

It is a graph showing the number of observations within each given interval.

## In Matplotlib, we use the hist() function to create histograms.

x = np.random.normal(170, 10, 250)

plt.hist(x) plt.show()

