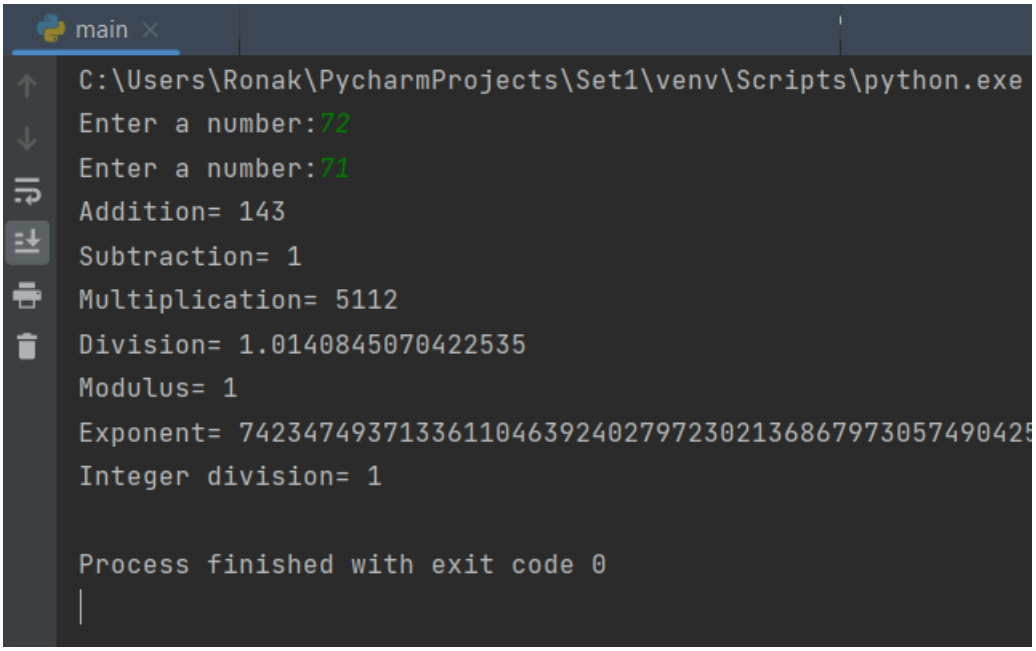
Practical Set-1: Basics of Python

Practical -1

Aim: Write a python program to create a simple arithmetic application including operations(addition, subtraction, multiplication, division, modulus, exponent, integer division).

```
a=int(input("Enter a number:"))
b=int(input("Enter a number:"))
print("Addition=",(a+b))
print("Subtraction=",(a-b))
print("Multiplication=",(a*b))
print("Division=",(a/b))
print("Modulus=",(a%b))
print("Exponent=",(a**b))
print("Integer division=",(a//b))
```

Output:

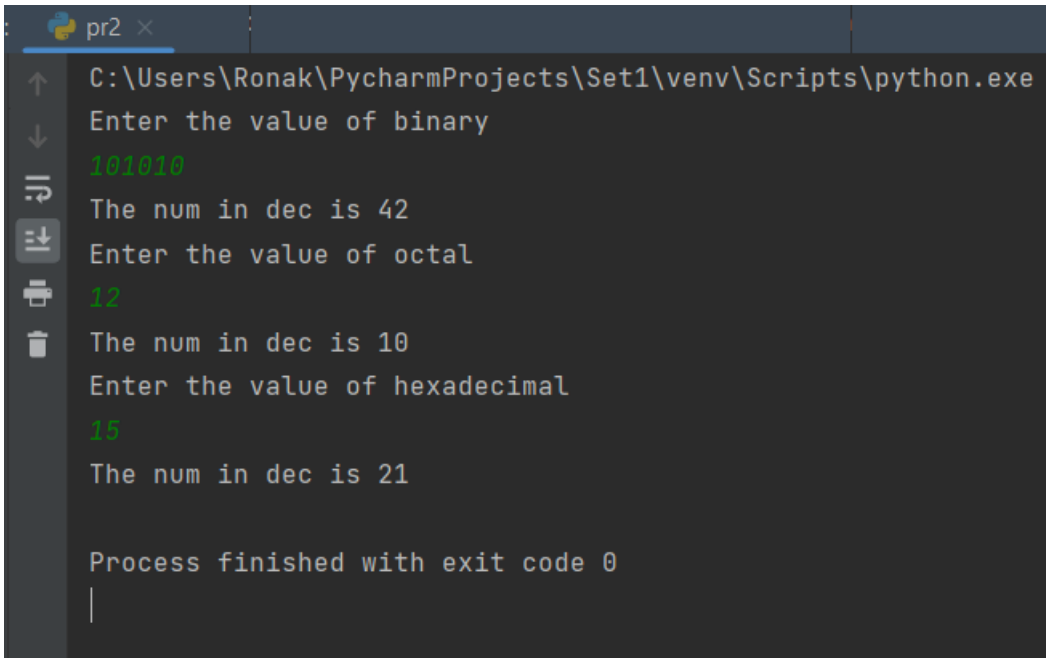


Practical -2

Aim: Write a python program to convert numbers from octal, binary and hexadecimal systems into decimal number system.

```
a=input("Enter the value of binary \n")
print("The num in dec is",int(a,2))
a=input("Enter the value of octal \n")
print("The num in dec is",int(a,8))
a=input("Enter the value of hexadecimal \n")
print("The num in dec is",int(a,16))
```

Output:

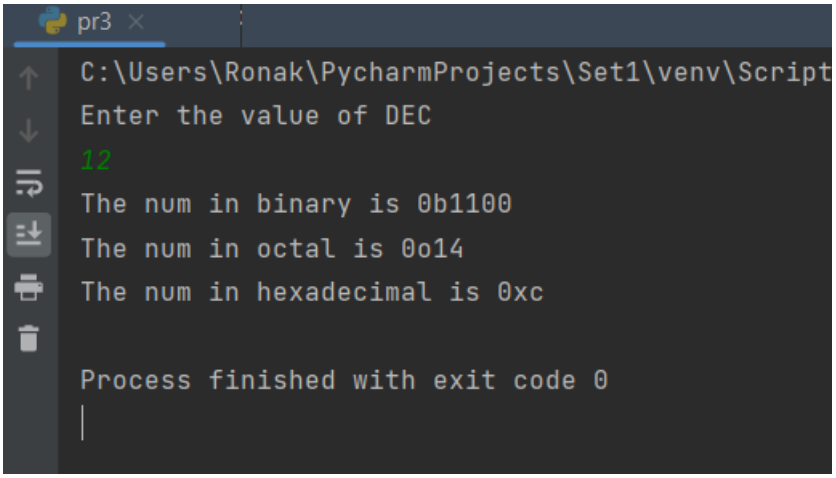


Practical -3

Aim: Write a python program to convert numbers from decimal number system into octal, binary and hexadecimal system.

```
a=int(input("Enter the value of DEC \n"))
print("The num in binary is",bin(a))
print("The num in octal is",oct(a))
print("The num in hexadecimal is",hex(a))
```

Output:

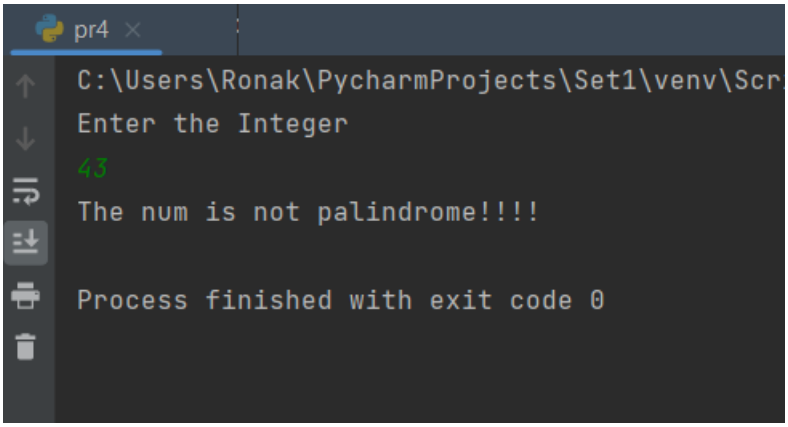


Practical -4

Aim: Write a python program to check whether the given number is palindrome or not.

```
no=int(input("Enter the Integer \n"))
sum=0
temp=no
while(no>0):
    r=no%10
    sum=(sum*10)+r
    no=int(no/10)
if(temp==sum):
    print(f"The num is paalindrome {no}")
else:
    print("The num is not palindrome!!!!")
```

Output:

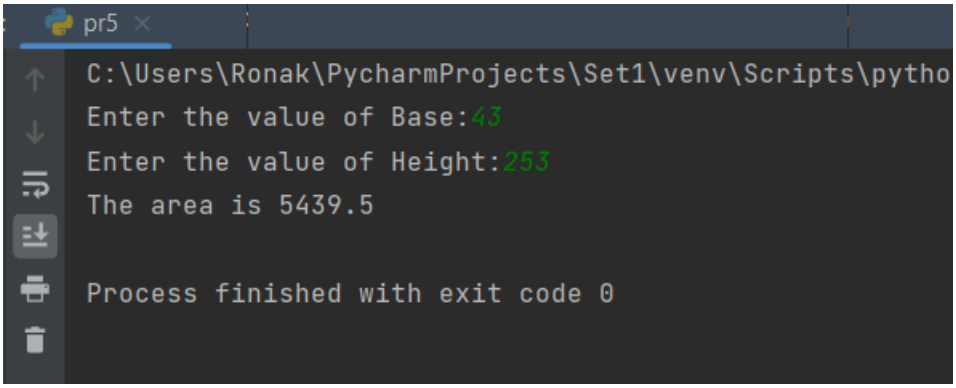


Practical -5

Aim: Write a python program to calculate area of a triangle.

```
x=float(input("Enter the value of Base:"))
y=float(input("Enter the value of Height:"))
Area=(x*y)/2
print("The area is",Area)
```

Output:

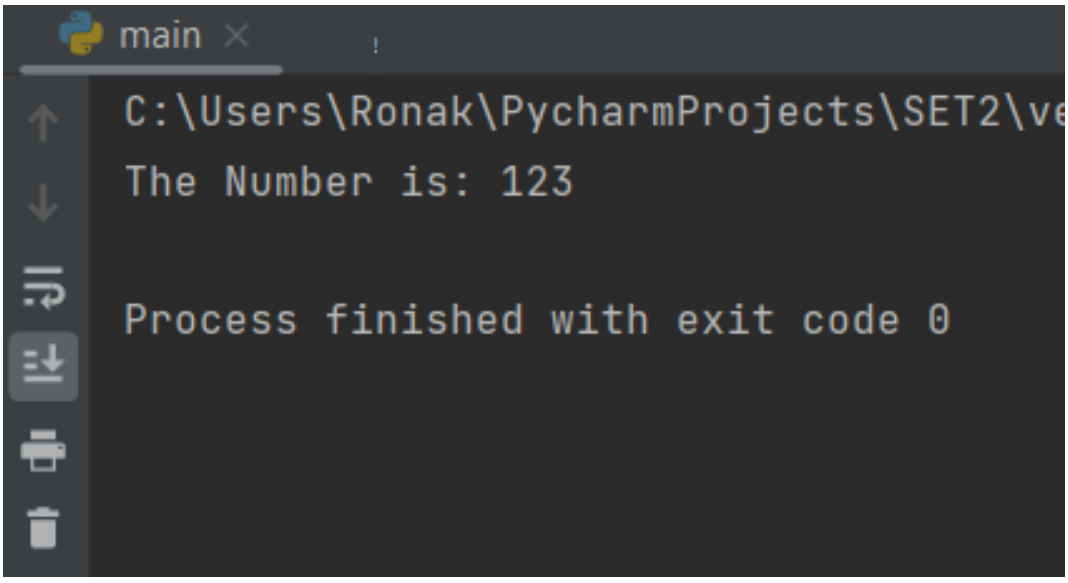


Practical: 6

Aim: Write a python program to display maximum of given 3 numbers.

```
x=5
y=7
z=123
if(x>=y)and(x>=z):
   largest=x
elif(y>=x)and(y>=z):
   largest=y
else:
   largest=z
print("The Number is:",largest)
```

Output:



Practical: 7

Aim: Write a python program to find those numbers which are divisible by 3 and multiple of 5 within 500.

```
for x in range(1,501):
    if(x % 3==0 and x % 5==0):
        print(x, end='')
```

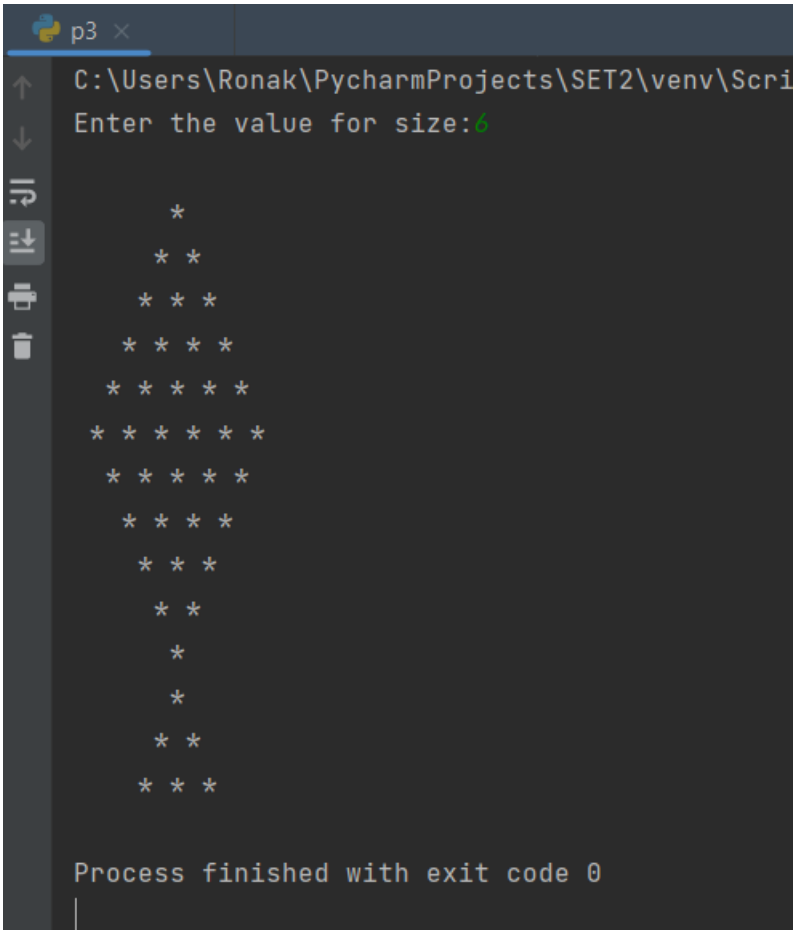
Output:

Practical: 8

Aim: Write a python program to draw kite pattern using for loop.

```
r=int(input("Enter the value for size:"))
for x in range(r,0,-1):
    print(" "*x,"* "*(r-x))
for x in range(0,r,1):
    print(" "*x,"* "*(r-x))
for x in range(r-1,int(r/2)-1,-1):
    print(" "*x,"* "*(r-x))
```

Output:



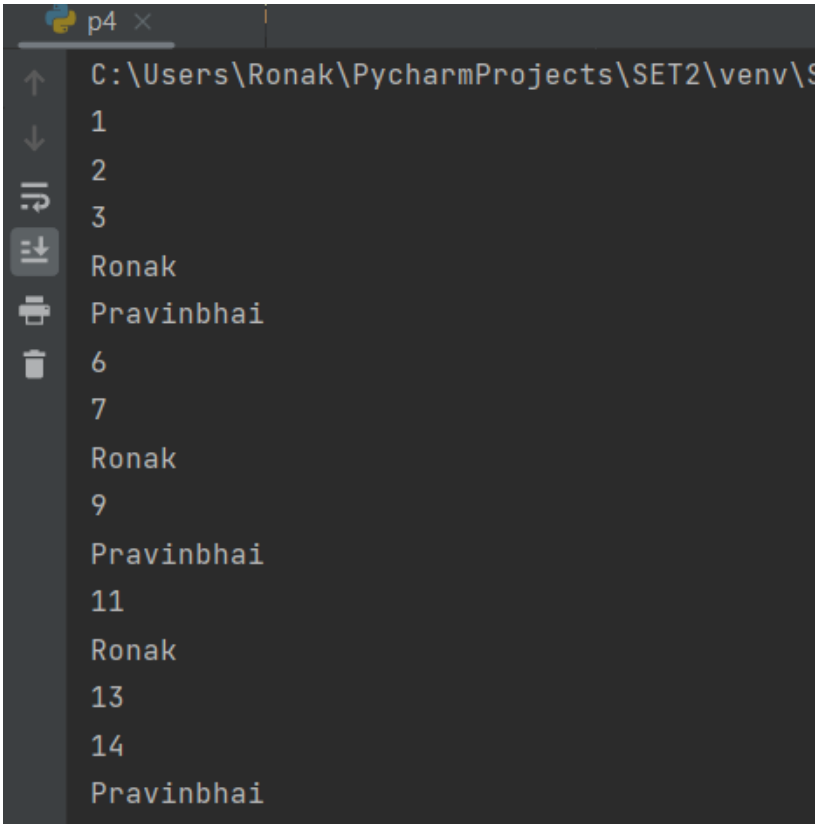
Practical Set-2: Looping and Data Structure with Python

Practical: 1

Aim: Write a python program to print numbers from 1 to 50. For multiple of 4 print name instead of number and for multiple of 5 print father name. For the numbers which are multiple of both 4 and 5 print surname.

```
for i in range(1,51):
    b=i
    if(i%4==0):
        b="Ronak"
    if(i%5==0):
        b="Pravinbhai"
    if(i%4==0 and i%5==0):
        b="Patel"
    print(b)
```

Output:

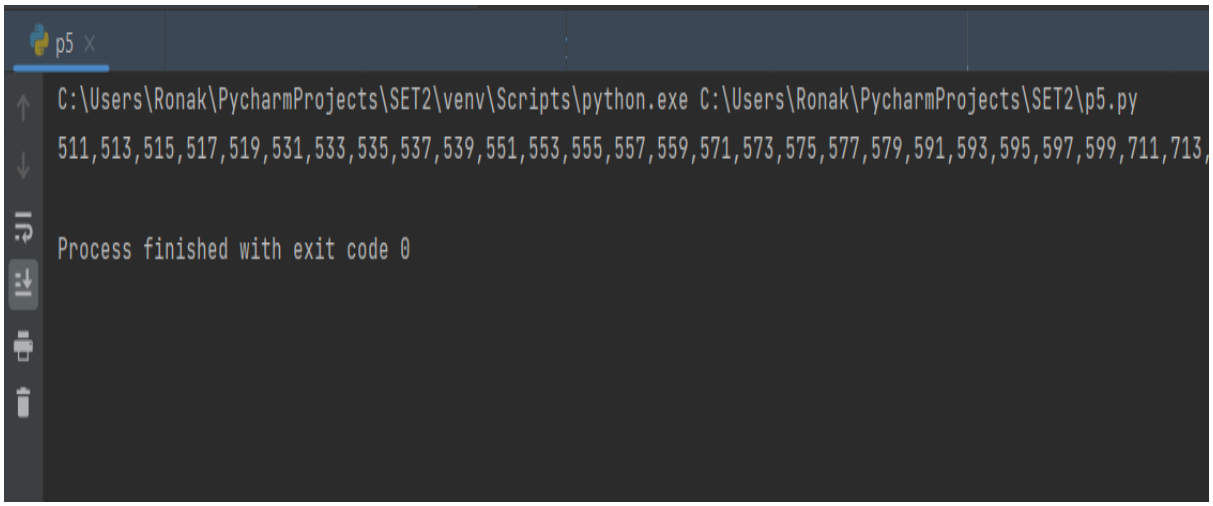


Practical: 2

Aim: Write a python program to find numbers between 500 and 800 when each digit of number is ODD and the number should be printed in sequence separated by comma.

```
item=[]
for i in range(500,801):
    s=str(i)
    if(int(s[0])%2!=0) and (int(s[1])%2!=0) and (int(s[2])%2!=0):
        item.append(s)
print(",".join(item))
```

Output:

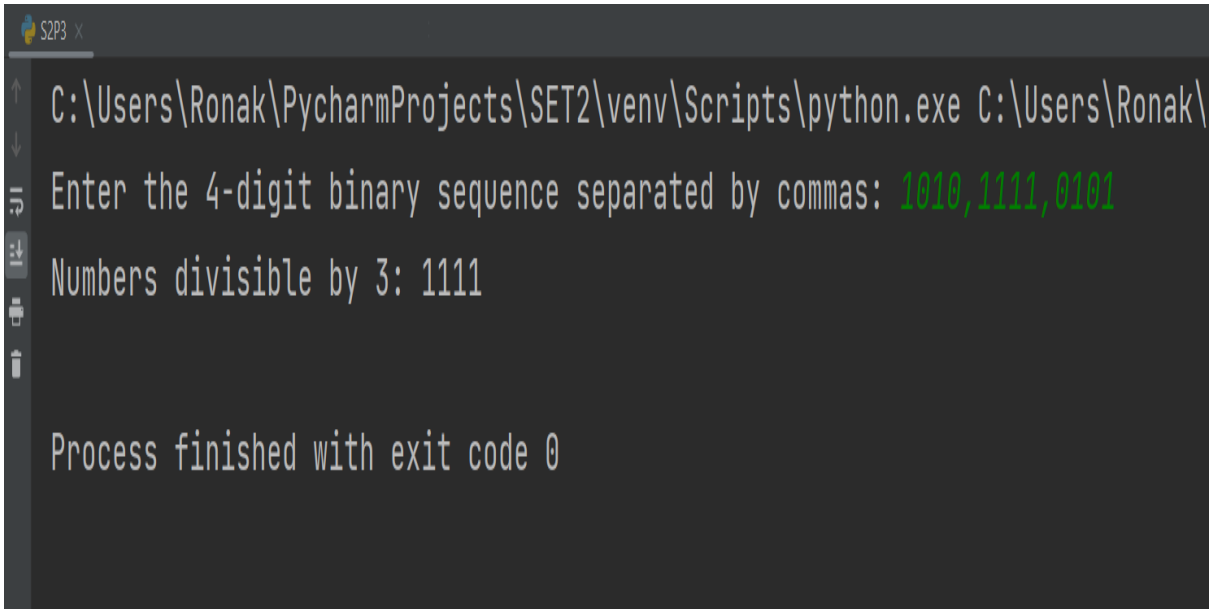


Practical: 3

Aim: Write a python program which accept a sequence of 4-digit binary numbers separated by comma and also print the numbers which are divisible by 3 in sequence separated by comma.

```
def is_divisible_by_3(binary_str):
    decimal_value = int(binary_str, 2)
    return decimal_value % 3 == 0

def main():
    binary_sequence = input("Enter the 4-digit binary sequence separated by commas: ")
    binary_numbers = binary_sequence.split(",")

    divisible_by_3 = [binary for binary in binary_numbers if is_divisible_by_3(binary)]

    if divisible_by_3:
        print("Numbers divisible by 3:", ", ".join(divisible_by_3))
    else:
        print("No numbers in the sequence are divisible by 3.")

if __name__ == "__main__":
    main()
```

Output:

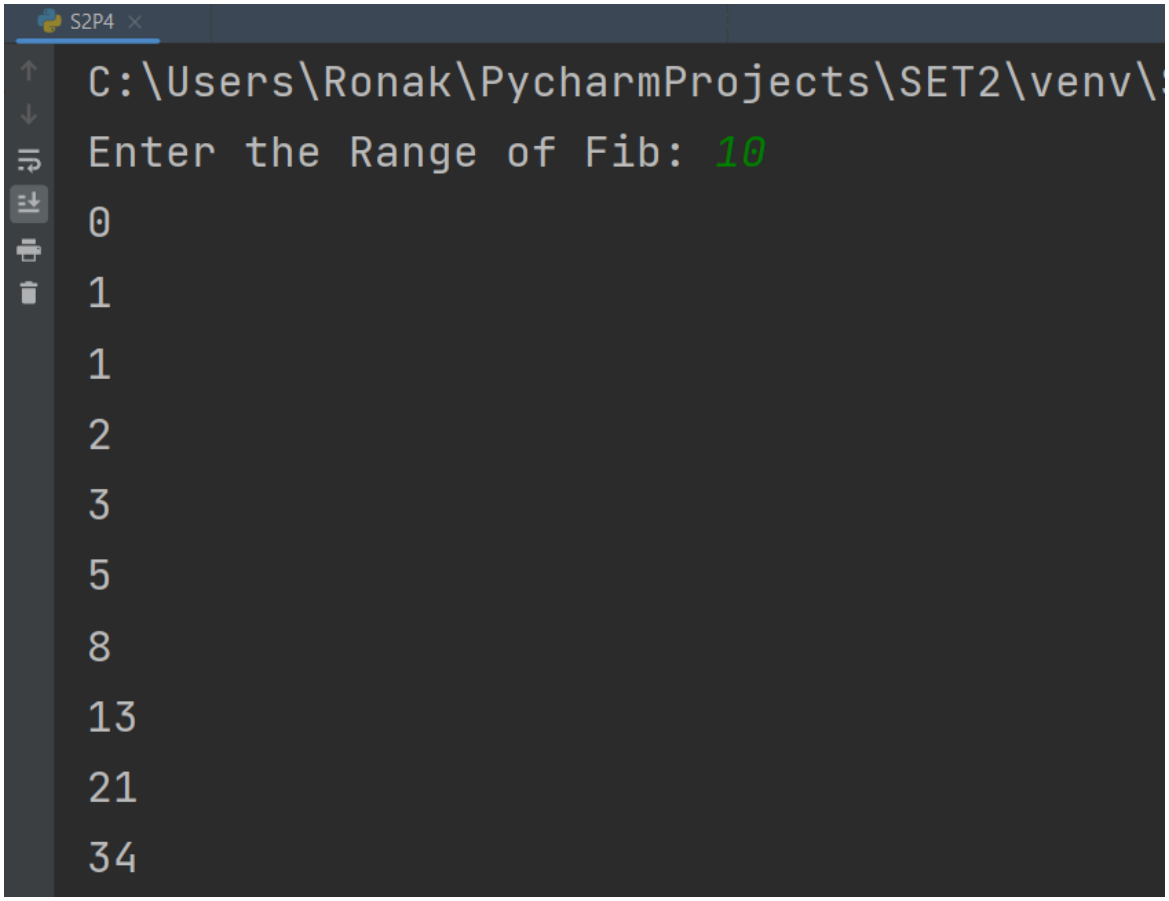


Practical: 4

Aim: Write a python program to display Fibonacci sequence up to nth term using recursive functions.

```
def recurr_fib(n):
    if n <= 1:
        return n
    else:
        return recurr_fib(n - 1) + recurr_fib(n - 2)

num = int(input("Enter the Range of Fib: "))

if num <= 0:
    print("Enter a positive Number")
else:
    for i in range(num):
        print(recurr_fib(i))
```

Output:

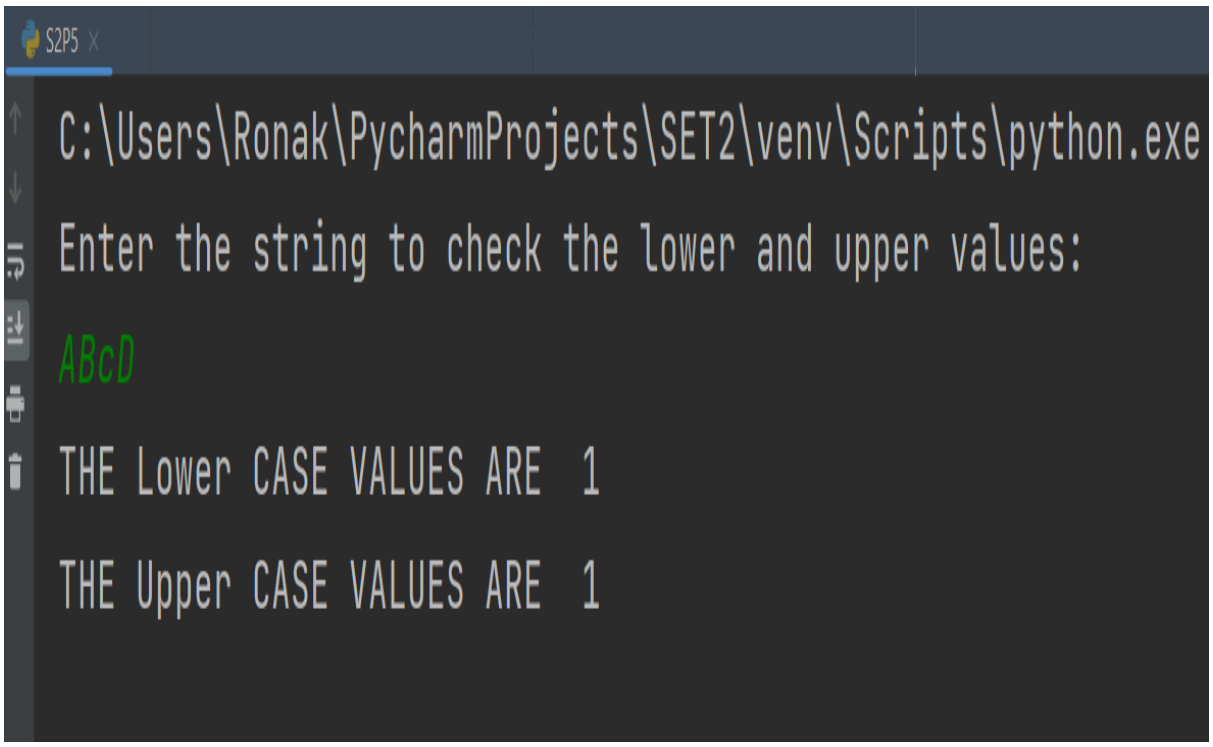


Practical: 5

Aim: Write a python program that accept a string and calculate the number of uppercase and lowercase letter.

```
str=input("Enter the string to check the lower and upper values: \n")
b=0
m=0
for i in str:
    if(i<='z' and i>='a'):
        b=b+1
if( i>='A' and i<='Z'):
    m=m+1
print("THE Lower CASE VALUES ARE ",b)
print("THE Upper CASE VALUES ARE ",m)
```

Output:

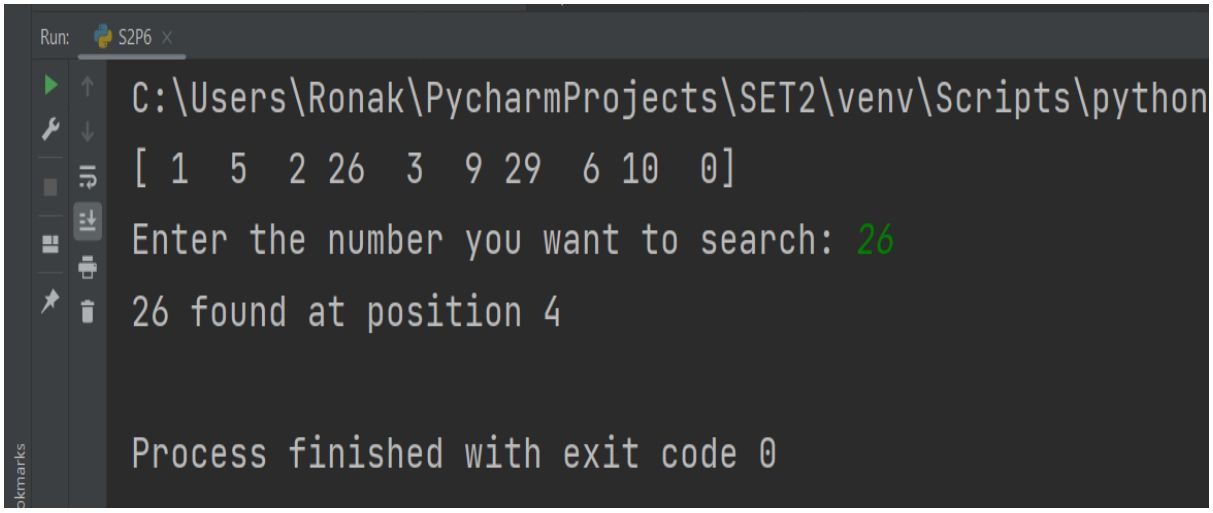


Practical: 6

Aim: Write a python program to search a number in array using sequential search.

```
import numpy as np

def seq_search(array, num):
    pos = 0
    found = False
    while pos < len(array) and not found:
        if array[pos] == num:
            found = True
        else:
            pos = pos + 1
    return found, pos + 1

array1 = np.random.randint(50, size=(10))
print(array1)
number = int(input("Enter the number you want to search: "))
result, position = seq_search(array1, number)

if result:
    print(f"{number} found at position {position}")
else:
    print(f"{number} not found in the array.")
```

Output:

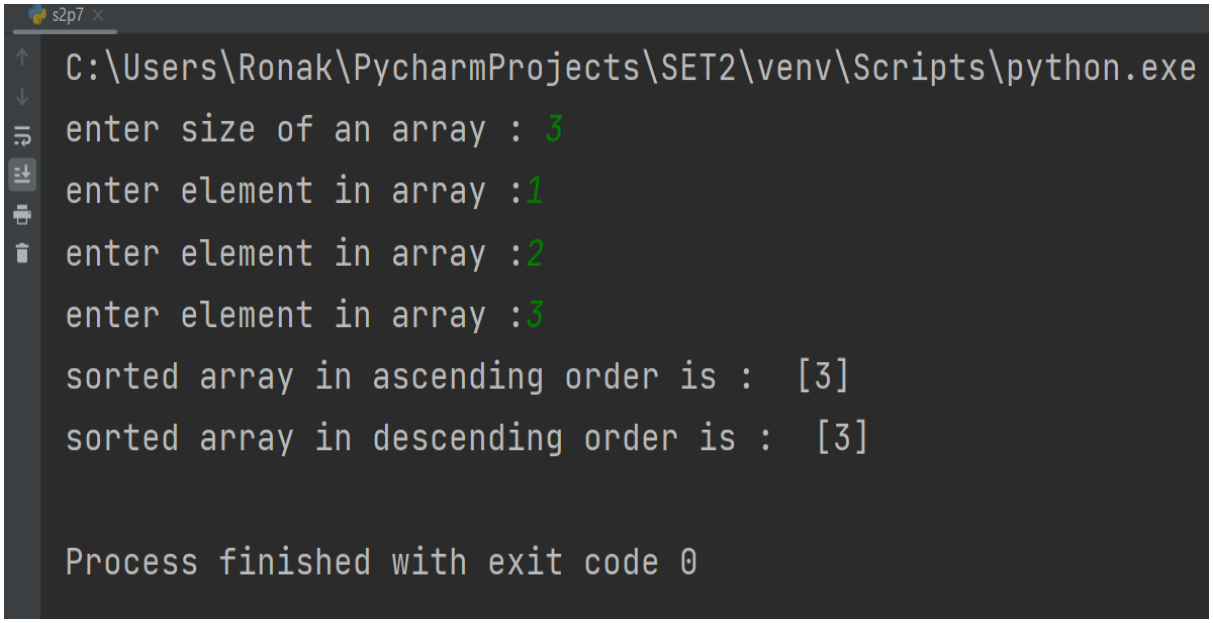


Practical: 7

Aim: Write a python program to sort elements of array.

```
size = int(input("enter size of an array : "))
arr = []
for i in range(size):
    element = int(input("enter element in array :"))
arr.append(element)
arr.sort()
print("sorted array in ascending order is : ",arr)
print("sorted array in descending order is : ",arr[::-1])
```

Output:

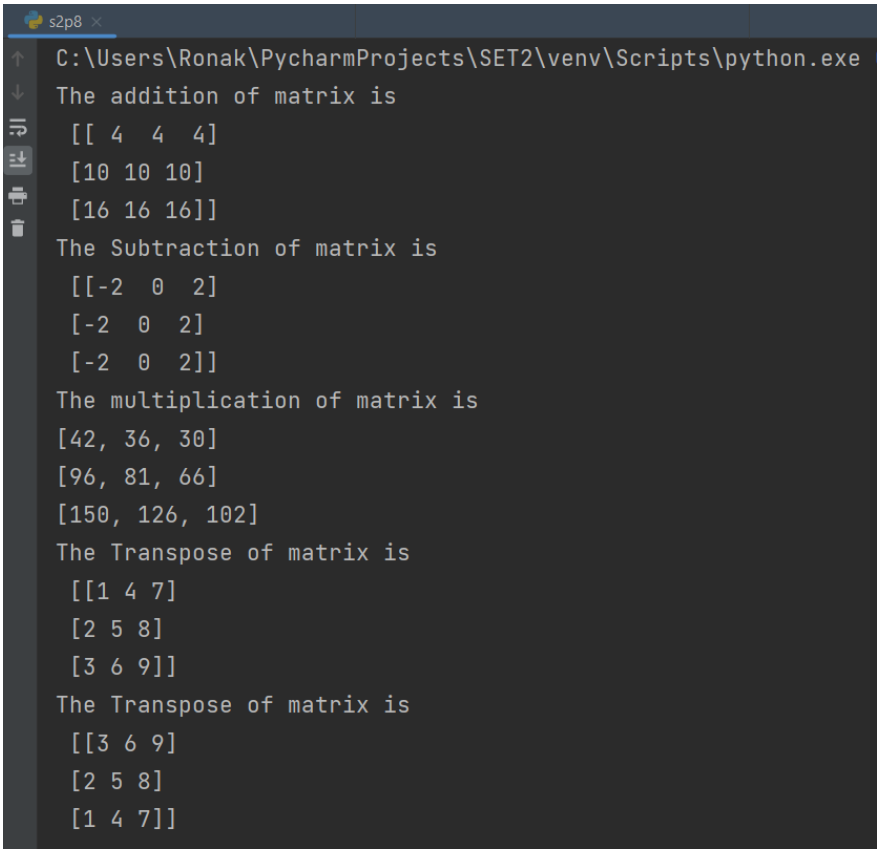


Practical: 8

Aim: Write a python program to input two matrix and perform the following given operation:

```
import numpy
arr1=([1,2,3],[4,5,6],[7,8,9])
arr2=([3,2,1],[6,5,4],[9,8,7])
result=([0,0,0],[0,0,0],[0,0,0])
mat1=numpy.array(arr1)
mat2=numpy.array(arr2)
print("The addition of matrix is \n",mat1+mat2)
print("The Subtraction of matrix is \n",mat1-mat2)
for i in range(0,len(mat1),1):
    for k in range(0,len(mat2[0]),1):
        for j in range(0,len(mat2),1):
            result[i][j] +=mat1[i][k]*mat2[k][j]
print("The multiplication of matrix is ",)
for m in result:
    print(m)
print("The Transpose of matrix is \n",mat1.T)
print("The Transpose of matrix is \n",mat2.T)
```

Output:



Practical Set-3: To study the use of NumPy and Pandas

Practical: 1

Aim: 1) Do as directed:

a) Read student data from given excel file sheet named as "5CSE" into appropriate pandas data structure.

b) Fill missing values in columns "Subject" and "Batch" using forward fill method.

c) Fill value "Jay Patel" in "Mentor" column for students having "Enrolments" column value from "210490131001" to "210490131070" and "Ronak Patel" for remaining students.

d) Add a new column "City" in existing student data and fill that column with residential city of student.

e) Count total number of students subject-wise and batch-wise.

Code:

```
import pandas as pd

df = pd.read_csv('5CSE.csv')

if "Unnamed: 0" in df.columns:
    df.drop("Unnamed: 0", axis=1, inplace=True)

# Use ffill() to fill missing values in "Subject" and "Batch" columns and assign the result
back to the DataFrame
df["Subject"].ffill(inplace=True)
df["Batch"].ffill(inplace=True)

# Define the value function
def value(x):
    if x <= 210490131010:
        return "Viral Prajapati"
    else:
        return "Gaurav Patel"

df['City']=["Vapi","Silvassa","Vapi","Bhilad","Bhilad","Vapi","Vapi","Vapi","Vapi","Vapi",
"Vapi","Vapi","Vapi","Vapi","Silvassa","Vapi","Vapi","Vapi","Vapi","Udvada","Vapi","Va
pi","Vapi","Vapi",
"Vapi","Vapi","Silvassa","Pardi","Vapi","Bhilad","Vapi","Vapi","Silvassa","Vapi","Vapi","
Vapi","Vapi","Vapi","Vapi","Vapi","Bhilad","Vapi"]


df.groupby(["Subject"]).size().reset_index(name='Number of Students')
```

```
df.groupby(["Batch"]).size().reset_index(name='Number of Students')

# Apply the value function to the "Enrollment" column and create a new "Mentor" column
df["Mentor"] = df["Enrollment"].apply(value)

df.groupby(["Subject"]).size().reset_index(name='Number of Students')
df.groupby(["Batch"]).size().reset_index(name='Number of Students')

# Print the modified DataFrame
print(df)

#Save the data in original csv file
df.to_csv('5CSE.csv', index=False)
```

Output: a), b), c), d), e)

Roll No	Enrollment	Name of Student	Subject	Batch	Mentor	City
1	210490131001	Arvindsingh Sarwansingh Deora	PDS	G1	Viral Prajapati	Vapi
2	210490131002	Sanal Saraswat	PDS	G1	Viral Prajapati	Silvassa
3	210490131003	Patel Yashkumar Hitendrabhai	PDS	G1	Viral Prajapati	Vapi
4	210490131004	Harssht Shah	PDS	G1	Viral Prajapati	Bhilad
5	210490131005	Chaudhary Manisha K	PDS	G1	Viral Prajapati	Bhilad
6	210490131006	Nupur Kundan Bhavsar	PDS	G1	Viral Prajapati	Vapi
7	210490131007	Kaustubh Vijay Tambe	PDS	G1	Viral Prajapati	Vapi
8	210490131008	Devendra Ghori	PDS	G1	Viral Prajapati	Vapi
9	210490131009	Harshalkumar Nalinbhai Patel	PDS	G1	Viral Prajapati	Vapi
10	210490131010	Hadal Rinkesh Bhagvanbhai	PDS	G1	Viral Prajapati	Vapi
11	210490131011	Patel Sneh	PDS	G1	Gaurav Patel	Vapi
12	210490131012	Rutik Sumanbhai Patel	PDS	G1	Gaurav Patel	Vapi
13	210490131013	Harshit.B.Dhodi	PDS	G1	Gaurav Patel	Vapi
14	210490131014	Rahul Dora	PDS	G2	Gaurav Patel	Vapi
15	210490131015	Akshit Kantilal Patel	PDS	G2	Gaurav Patel	Silvassa
16	210490131016	Lakhani Brijay	PDS	G2	Gaurav Patel	Vapi
17	210490131017	Krutagna Patel	PDS	G2	Gaurav Patel	Vapi
18	210490131018	Dhruvi Sushilbhai Patel	PDS	G2	Gaurav Patel	Vapi

19	210490131019	Mayur Pareshbhai Parmar	PDS	G2	Gaurav Patel	Vapi
20	210490131020	Bhargav Bhandari	PDS	G2	Gaurav Patel	Udvada
21	210490131021	Gautam Suraj Umaprasad	PDS	G2	Gaurav Patel	Vapi
22	210490131022	Patel Prachi Jitendrabhai	PDS	G2	Gaurav Patel	Vapi
23	210490131023	Ruchi P Mishra	PDS	G2	Gaurav Patel	Vapi
24	210490131024	Hemangi Toke	PDS	G2	Gaurav Patel	Vapi
25	210490131025	Sweta Vinod Jaiswal	PDS	G2	Gaurav Patel	Vapi
26	210490131026	Siddhi Nagesh Bhad	PDS	G2	Gaurav Patel	Vapi
27	210490131027	Amit Kumar Bhagat	CS	G3	Gaurav Patel	Silvassa
28	210490131028	Harshi Patel	CS	G3	Gaurav Patel	Pardi
29	210490131029	Mitaliya Vivek V.	CS	G3	Gaurav Patel	Vapi
30	210490131030	Anuradha Kumari Kharwar	CS	G3	Gaurav Patel	Bhilad
31	210490131031	Yadav Kauntay Mahendra	CS	G3	Gaurav Patel	Vapi
32	210490131032	Saurabh Singh	CS	G3	Gaurav Patel	Vapi
33	210490131033	Hariom Prasad	CS	G3	Gaurav Patel	Silvassa
34	210490131034	Nisarg Ashokbhai Rathod	CS	G3	Gaurav Patel	Vapi
35	210490131035	Rutik	CS	G3	Gaurav Patel	Vapi
36	210490131036	Shivam Jaydeepbhai Desai	CS	G3	Gaurav Patel	Vapi
37	210490131037	Rohan Prasad	CS	G3	Gaurav Patel	Vapi
38	210490131038	Kaushik H Koladiya	CS	G3	Gaurav Patel	Vapi
39	210490131039	Gohil Pratik D.	CS	G3	Gaurav Patel	Vapi
40	210490131040	Tas Shaikh	CS	G3	Gaurav Patel	Vapi
41	210490131041	Saloni Amar Bhatt	CS	G3	Gaurav Patel	Bhilad
42	210490131042	Nehal.M.Tandel	CS	G3	Gaurav Patel	Vapi

e)

```
Batch Count:
  Batch  Number of Students
0    G1                  13
1    G2                  13
2    G3                  16
```

```
Subject Count:
  Subject  Number of Students
0      CS                  16
1     PDS                  26
```

Practical: 2

2) Do as directed:

a) Read data from given csv file into appropriate pandas data structure. Delete rows having missing values.

b) Calculate average price of cars having four- and six-cylinder engines.

c) Find out cheapest and most expensive car details.

d) Find out convertible and sedan car details having maximum engine horsepower.

e) Find average sedan car price

f) Count total number of cars per company.

g) Find each company's highest car price.

Code:

a)

```
import pandas as pd

import numpy as np

df=pd.read_csv("Automobile_data_Miss.csv")

print(df)
```

b)

```
import pandas as pd
import numpy as np
df=pd.read_csv("Automobile_data_Miss.csv")

df.dropna(axis='rows')

a1=df["num-of-doors"]=='four'
a_1=df[a1]["price"].mean()
a2=df["num-of-doors"]=='two'
a_2=df[a2]["price"].mean()
avg=(a_1+a_2)/2
print(avg)
```

c)

```
print("Detail of cheapest car")
df[df.price == df.price.min()]
print(df[df.price == df.price.min()])
print("Detail of expensive car")
```

```
df[df.price == df.price.max()]
print(df[df.price == df.price.min()])
```

d)

```
df1=df.loc[(df['body-style']=='convertible')]

df1=df1.iloc[df1['horsepower'].argmax()]

df2=df.loc[(df['body-style']=='sedan')]

df2=df2.iloc[df2['horsepower'].argmax()]

print('maximum hoursepower of convertible car is\n\n',df1)

print('\n\nmaximum hoursepower of sedan car is\n\n',df2)
```

e)

```
df[df["body-style"]=="sedan"].agg({"price":'mean'})
print(df[df["body-style"] == "sedan"].agg({"price": 'mean'}))
```

f)

```
df['make'].value_counts()
print(df['make'].value_counts())
```

g)

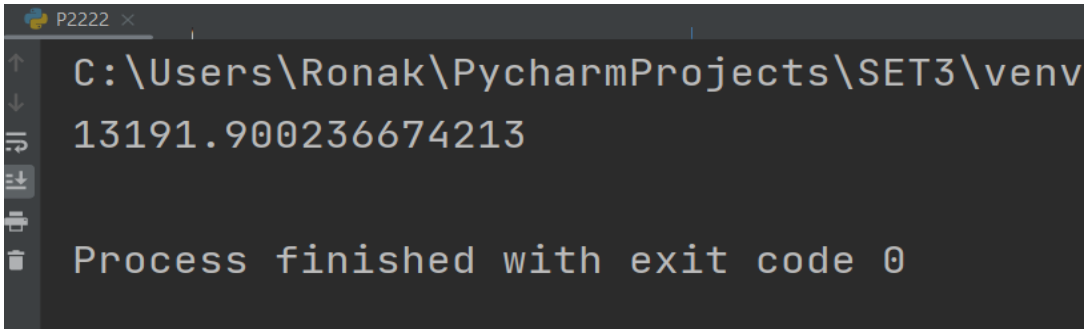
```
car_manufactures = df.groupby('make')

price= car_manufactures["make","price"].max()

print(price)
```

Output:

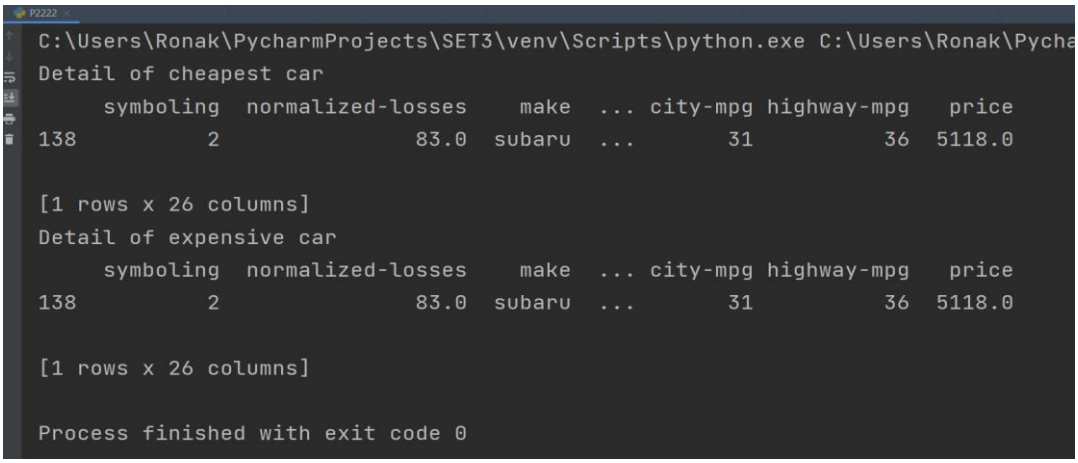
a)

```
C:\Users\Ronak\PycharmProjects\SET3\venv\Scripts\python.exe C:\Users\Ronak\PycharmProjects\SE
     symboling  normalized-losses         make  ... city-mpg highway-mpg    price
0            3                NaN  alfa-romero  ...       21          27  13495.0
1            3                NaN  alfa-romero  ...       21          27  16500.0
2            1                NaN  alfa-romero  ...       19          26  16500.0
3            2              164.0         audi  ...       24          30  13950.0
4            2              164.0         audi  ...       18          22  17450.0
..         ...                ...          ...  ...      ...         ...      ...
200         -1               95.0        volvo  ...       23          28  16845.0
201         -1               95.0        volvo  ...       19          25  19045.0
202         -1               95.0        volvo  ...       18          23  21485.0
203         -1               95.0        volvo  ...       26          27  22470.0
204         -1               95.0        volvo  ...       19          25  22625.0

[205 rows x 26 columns]
```

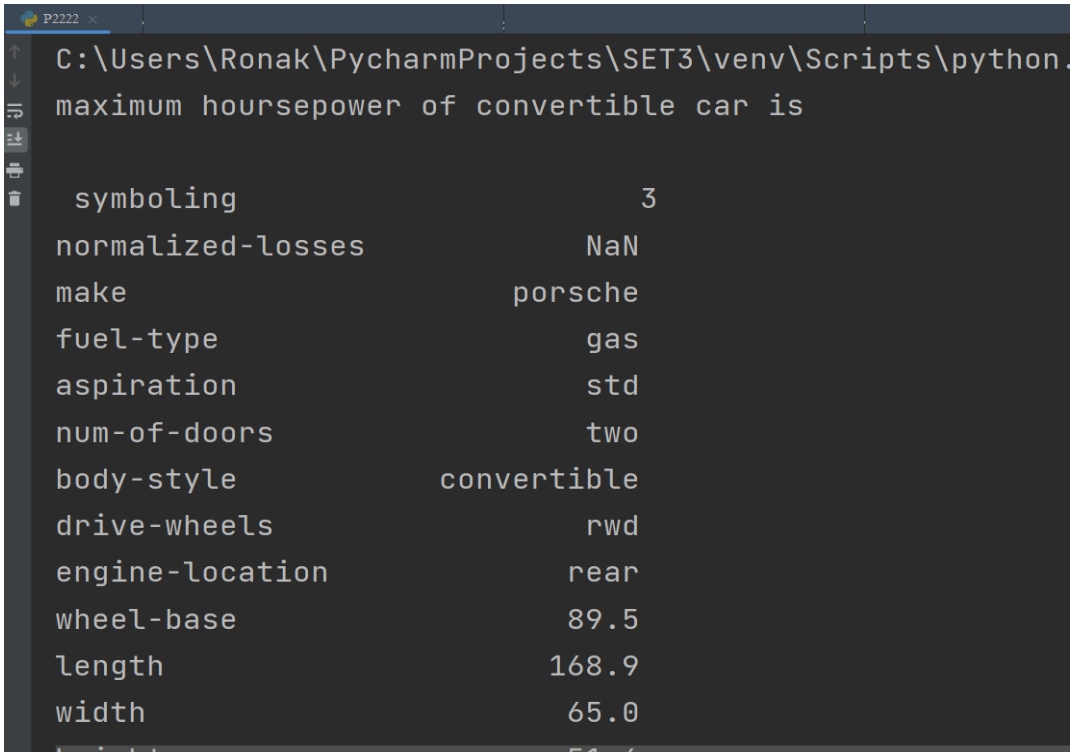
b)


P2222
C:\Users\Ronak\PycharmProjects\SET3\venv
13191.900236674213
Process finished with exit code 0

c)


C:\Users\Ronak\PycharmProjects\SET3\venv\Scripts\python.exe C:\Users\Ronak\Pycha
Detail of cheapest car
symboling normalized-losses make ... city-mpg highway-mpg price
138 2 83.0 subaru ... 31 36 5118.0
[1 rows x 26 columns]
Detail of expensive car
symboling normalized-losses make ... city-mpg highway-mpg price
138 2 83.0 subaru ... 31 36 5118.0
[1 rows x 26 columns]
Process finished with exit code 0

d)


C:\Users\Ronak\PycharmProjects\SET3\venv\Scripts\python.
maximum hoursepower of convertible car is
symboling 3
normalized-losses NaN
make porsche
fuel-type gas
aspiration std
num-of-doors two
body-style convertible
drive-wheels rwd
engine-location rear
wheel-base 89.5
length 168.9
width 65.0

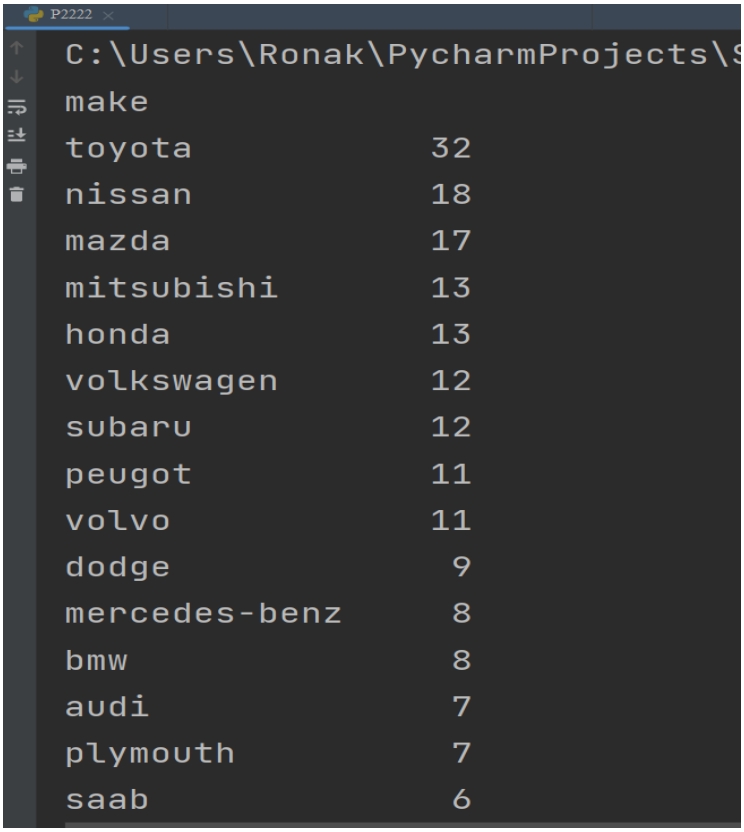
```
maximum hoursepower of sedan car is

 symboling                  0
normalized-losses         NaN
make                    jaguar
fuel-type                  gas
aspiration                 std
num-of-doors               two
body-style               sedan
drive-wheels               rwd
engine-location          front
wheel-base               102.0
length                   191.7
```

Control | Run | Debug | Python Packages | TODO | Python Console | Problems | Terminal

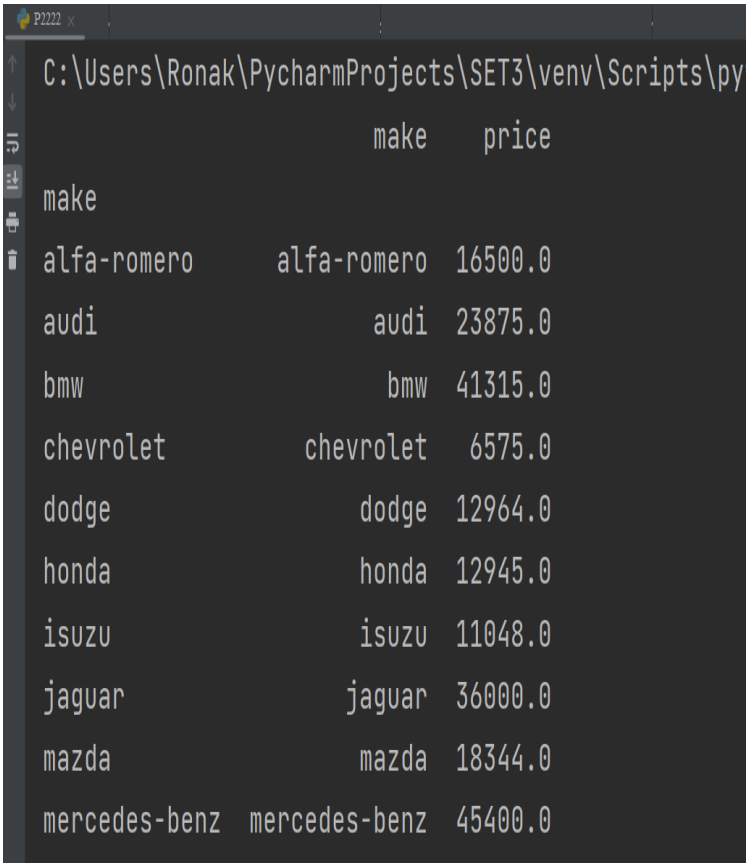
e)

```
P2222
C:\Users\Ronak\PycharmProjects\SET3\venv\
price    14459.755319
dtype: float64

Process finished with exit code 0
```

f)


P2222
C:\Users\Ronak\PycharmProjects\S
make
toyota 32
nissan 18
mazda 17
mitsubishi 13
honda 13
volkswagen 12
subaru 12
peugot 11
volvo 11
dodge 9
mercedes-benz 8
bmw 8
audi 7
plymouth 7
saab 6

g)


P2222
C:\Users\Ronak\PycharmProjects\SET3\venv\Scripts\py
make price
make
alfa-romero alfa-romero 16500.0
audi audi 23875.0
bmw bmw 41315.0
chevrolet chevrolet 6575.0
dodge dodge 12964.0
honda honda 12945.0
isuzu isuzu 11048.0
jaguar jaguar 36000.0
mazda mazda 18344.0
mercedes-benz mercedes-benz 45400.0

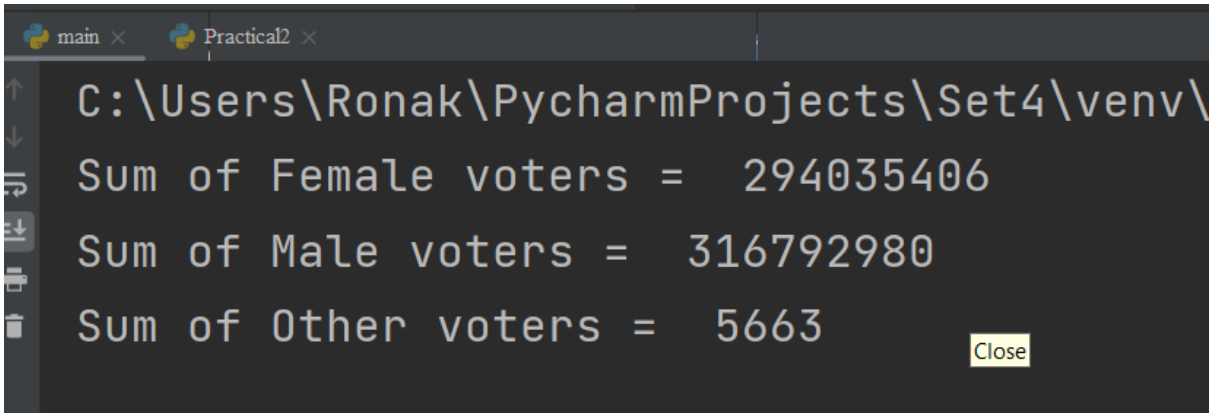
Practical Set-4: Use of matplotlib and pandas Libraries for Data Analysis and Visualization.

Practical: 1

Aim: Plot gender-wise share of overall voters with legend and suitable labels. (Pie chart).

```
from pandas import DataFrame,read_csv

import matplotlib.pyplot as plt

import pandas as pd

result_df=pd.read_csv('Votes 2019.csv')

result_df

x=result_df["Female"].sum()

print("Sum of Female voters=",x)

y=result_df["Male"].sum()

print("Sum of Male voters=",y)

z=result_df["Other"].sum()

print("Sum of Other voters=",z)

sizes=[x,y,z]

labels=["Female","Male","Other"]

colors=['red','palegreen','lightskyblue']

explode=(0,0.1,0)

plt.pie(sizes,explode=explode,labels=labels,colors=colors,startangle=90,autopct="%.1f%%",
shadow=True)

plt.legend(labels,loc=2)

plt.show()
```

Output:


main
Practical2
C:\Users\Ronak\PycharmProjects\Set4\venv\
Sum of Female voters = 294035406
Sum of Male voters = 316792980
Sum of Other voters = 5663
Close

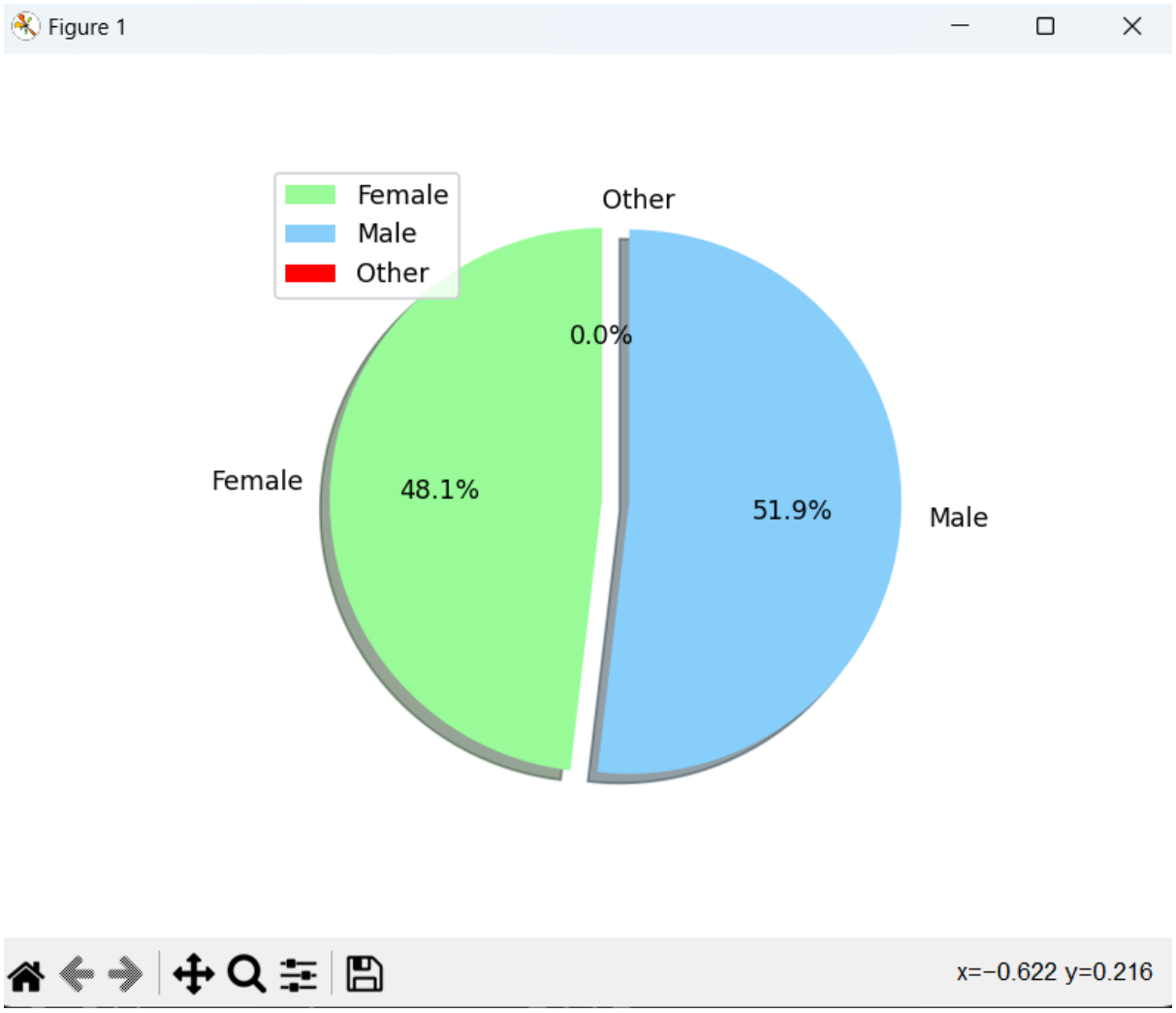


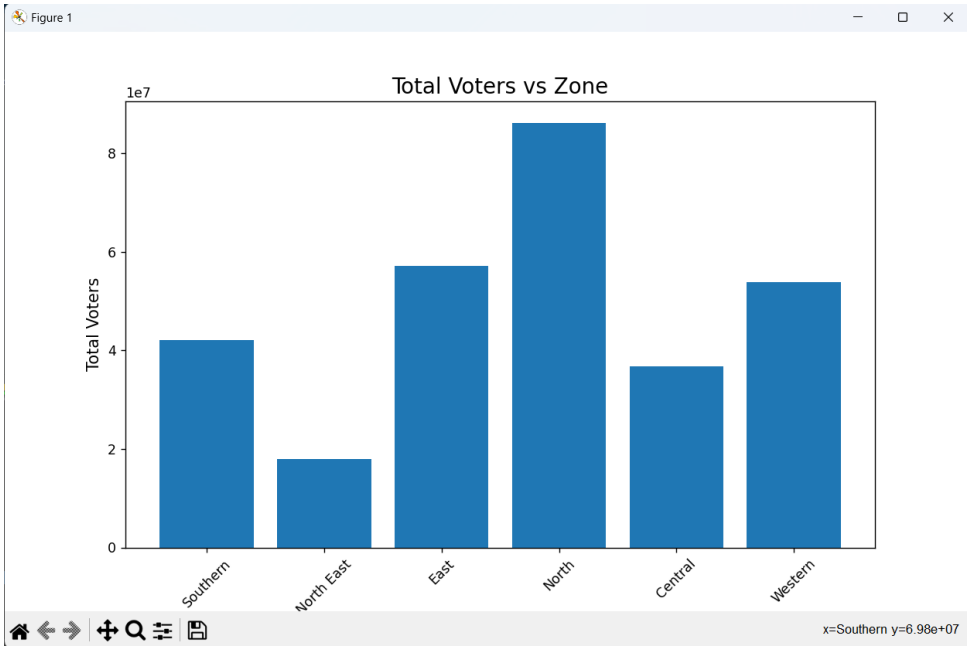
Figure 1
Female
Male
Other
Other
0.0%
Female
48.1%
51.9%
Male
x=−0.622 y=0.216

Practical: 2

Aim: Indian states are divided into six administrative zones: Central, East, North, Northeast, South and Western. Plot six bar chart into single figure to visualize total voters with suitable chart title.

```
from pandas import DataFrame, read_csv
import matplotlib.pyplot as plt
import pandas as pd
result_df = pd.read_csv('Votes 2019.csv')
print(result_df)

zone = ['Southern', 'Southern', 'North East', 'North East', 'East', 'North', 'Central', 'Western',
'Western', 'Western', 'Western', 'North', 'North', 'North', 'East', 'Southern', 'Southern',
'Southern', 'Central', 'Western', 'North', 'North East', 'North East', 'North East', 'North', 'East',
'Southern', 'North', 'North', 'North East', 'Southern', 'Southern', 'North East', 'North', 'North',
'East']
result_df['Zone'] = zone
result_df.to_csv("Votes_2019.csv")
y = result_df['Zone']
fig, ax = plt.subplots(figsize=(10, 6))
l1 = ax.bar(y, result_df['Total Voters'])
ax.set_title("Total Voters vs Zone", size=16)
ax.set_ylabel("Total Voters", size=12)
ax.set_xlabel("Administrative Zones", size=12)
ax.tick_params(axis='x', rotation=45)  # Rotating the x-axis labels for better visibility
plt.show()
```

Output:



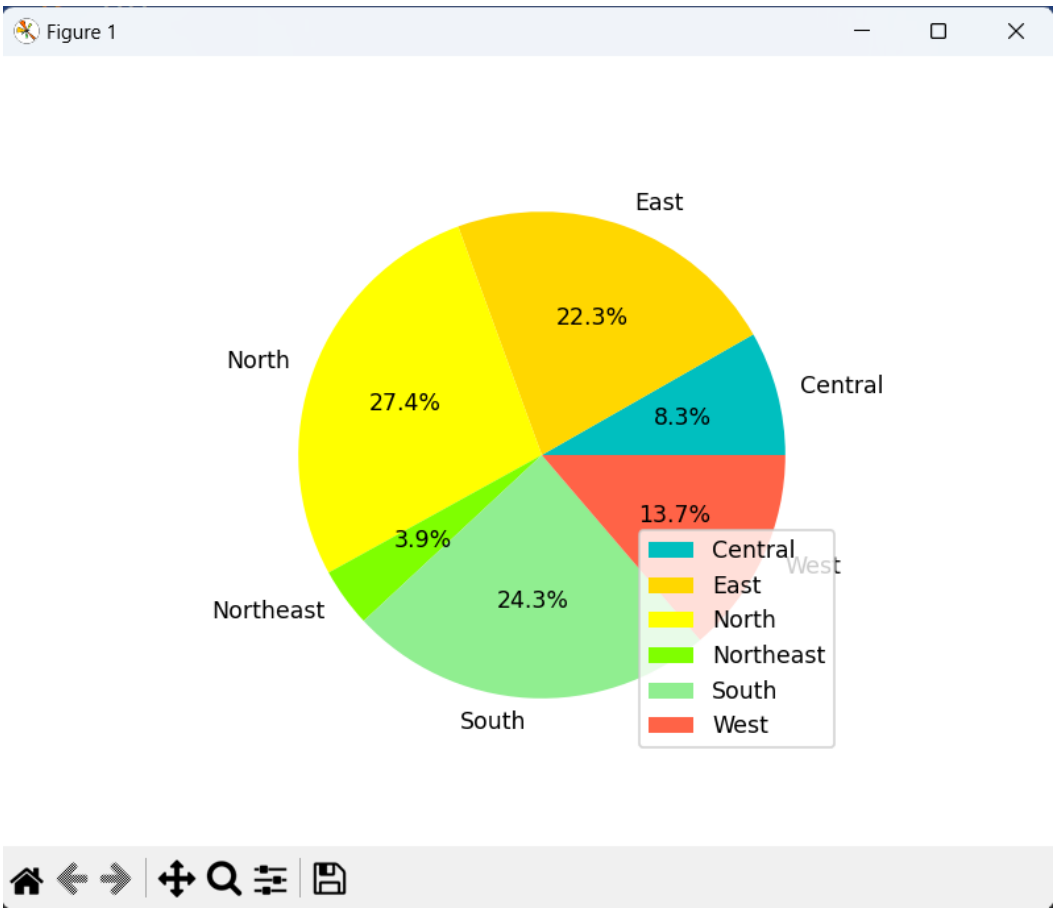
Practical -3

Aim: Plot zone-wise share of total voters with legend and suitable labels. (Pie chart)

```
from pandas import DataFrame, read_csv
import matplotlib.pyplot as plt
import pandas as pd

result_df = pd.read_csv('Votes 2019.csv')

zone = ['Southern', 'Southern', 'North East', 'North East', 'East', 'North', 'Central', 'Western',
'Western', 'Western', 'Western', 'North', 'North', 'North', 'East', 'Southern', 'Southern',
'Southern', 'Central', 'Western', 'North', 'North East', 'North East', 'North East', 'North', 'East',
'Southern', 'North', 'North', 'North East', 'Southern', 'Southern', 'North East', 'North', 'North',
'East']
result_df['Zone'] = zone
result_df.to_csv("Votes_2019.csv")
# Please make sure that the column 'Zone' exists in your DataFrame 'result_df' and adjust the
following code accordingly.
xx = result_df['Total Voters'].groupby(result_df['Zone']).sum()
print(xx)
sizes = [xx["Central"], xx["East"], xx["North"], xx["North East"], xx["Southern"],
xx["Western"]]
labels = ["Central", "East", "North", "Northeast", "South", "West"]
colors = ['c', 'gold', 'yellow', 'chartreuse', 'lightgreen', 'tomato']
plt.pie(sizes, colors=colors, labels=labels, autopct="%.1f%%")
plt.legend(labels, loc=4)
plt.show()
```

Output:

```
main × Practical 3 ×
C:\Users\Ronak\PycharmProjects\Set4\ve
Zone
Central          50419308
East            136404728
North           167613284
North East       23906013
Southern        148583201
Western          83907515
Name: Total Voters, dtype: int64
```

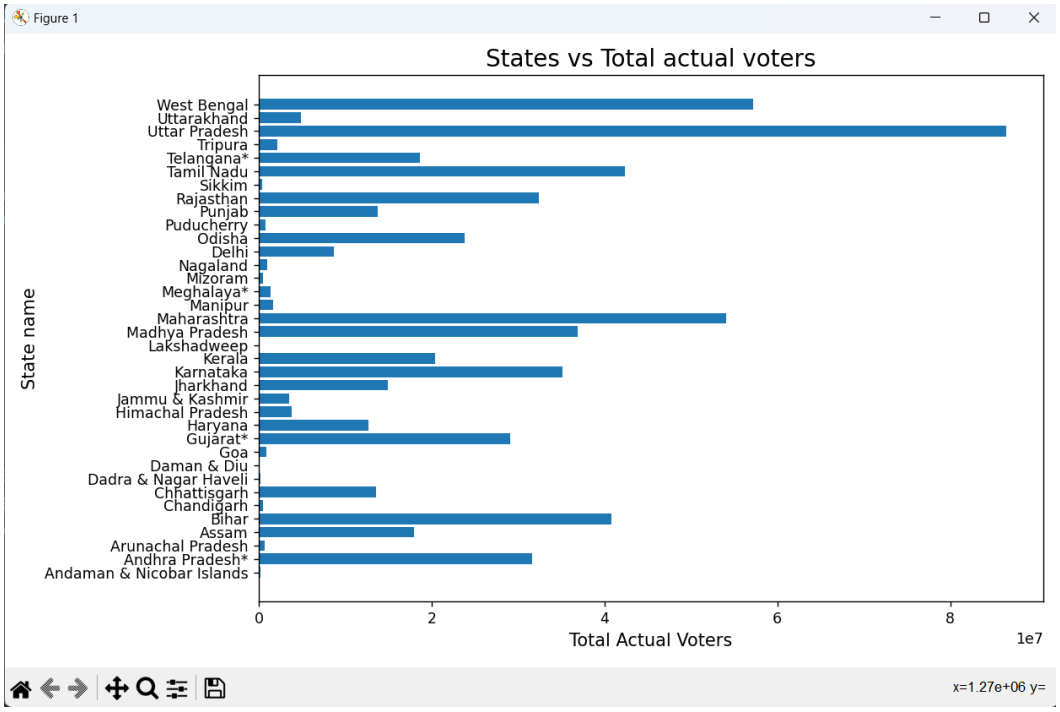


Practical: 4

Aim: Plot horizontal bar chart for states vs total actual votes with suitable labels.

```
from pandas import DataFrame, read_csv
import matplotlib.pyplot as plt
import pandas as pd

result_df = pd.read_csv('Votes 2019.csv')

fig, ax = plt.subplots(figsize=(10, 6))  # Adjusted the figure size

ax.barh(y=result_df['State Name'], width=result_df['Total Actual Votes'])
plt.title("States vs Total actual voters", size=16)
plt.ylabel("State name", size=12)
plt.xlabel("Total Actual Voters", size=12)

# Set the y-axis tick positions and labels
ax.set_yticks(range(len(result_df['State Name'])))
ax.set_yticklabels(result_df['State Name'], fontsize=10)  # Adjusted the font size for y-axis labels

# Adjust the layout to ensure all labels are visible
plt.tight_layout()

plt.show()
```

Output:



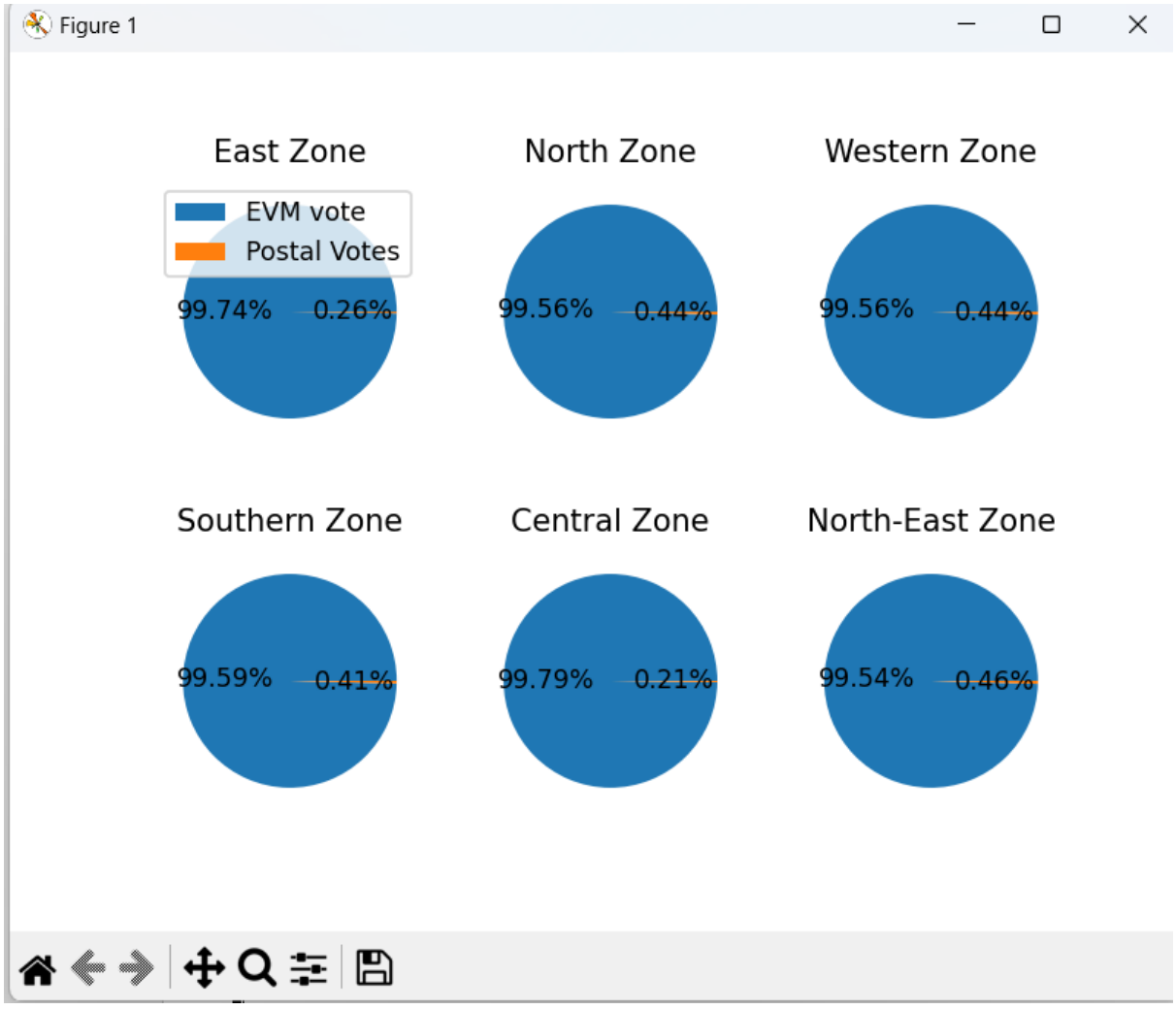
Practical: 5

Aim: Plot type-wise share (EVM and Postal) with legend and suitable labels for each administrative zone into single figure. (Pie Chart).

```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd

result_df = pd.read_csv('Votes 2019.csv')
fig, ax = plt.subplots(2, 3)
zone = ['Southern', 'Southern', 'North East', 'North East', 'East', 'North', 'Central', 'Western',
'Western', 'Western', 'Western', 'North', 'North', 'North', 'East', 'Southern', 'Southern',
'Southern', 'Central', 'Western', 'North', 'North East', 'North East', 'North East', 'North', 'East',
'Southern', 'North', 'North', 'North East', 'Southern', 'Southern', 'North East', 'North', 'North',
'East']
result_df['Zone'] = zone
# Please ensure that the columns 'EVM Vote' and 'Postal Vote' exist in your DataFrame and
adjust the following code accordingly.
xx = result_df['EVM Vote'].groupby(result_df['Zone']).sum()
yy = result_df['Postal Vote'].groupby(result_df['Zone']).sum()

ax[0][0].pie([xx['East'], yy['East']], autopct='%.2f%%')
ax[0][0].set_title('East Zone')
ax[0][1].pie([xx['North'], yy['North']], autopct='%.2f%%')
ax[0][1].set_title('North Zone')
ax[0][2].pie([xx['Western'], yy['Western']], autopct='%.2f%%')
ax[0][2].set_title('Western Zone')
ax[1][0].pie([xx['Southern'], yy['Southern']], autopct='%.2f%%')
ax[1][0].set_title('Southern Zone')
ax[1][1].pie([xx['Central'], yy['Central']], autopct='%.2f%%')
ax[1][1].set_title('Central Zone')
ax[1][2].pie([xx['North East'], yy['North East']], autopct='%.2f%%')
ax[1][2].set_title('North-East Zone')
```

```
# Adjust the legend to show the labels properly
ax[0][0].legend(["EVM vote", "Postal Votes"])

plt.show()
```

Output:

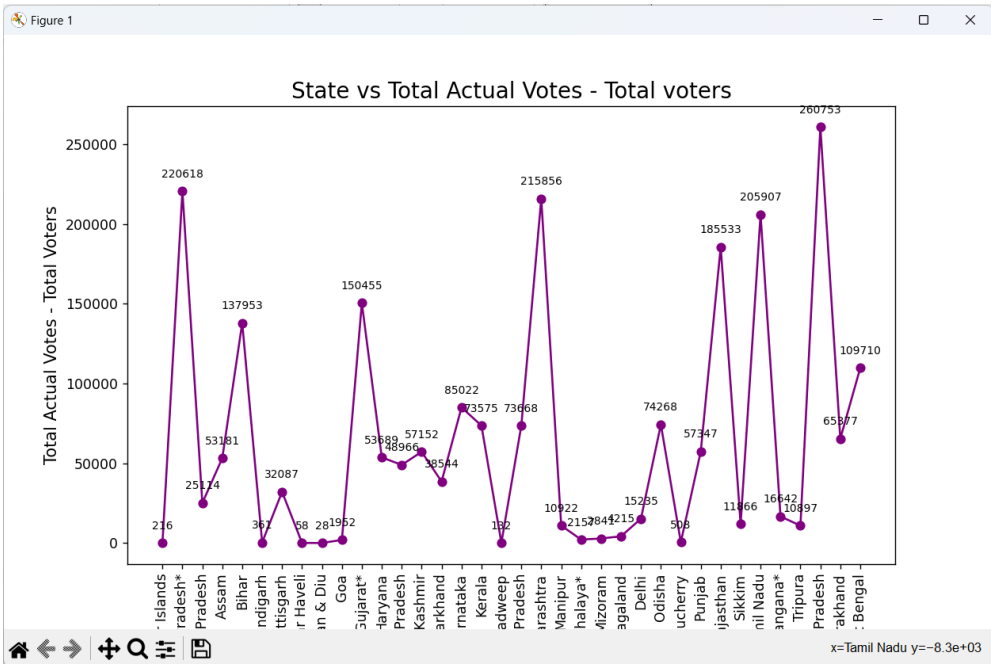


Practical: 6

Aim: Plot vote deficits (Total actual votes – Total voters) for each states using line chart.

```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd

result_df = pd.read_csv('Votes 2019.csv')

result_df.loc[:, "New"] = result_df["Total Actual Votes"] - result_df["Total Voters"]

fig, ax = plt.subplots(figsize=(10, 6))  # Adjusted the figure size

ax.plot(result_df['State Name'], result_df['New'], color='purple', marker='o')
plt.title("State vs Total Actual Votes - Total voters", fontsize=16)
plt.xticks(rotation=90, fontsize=10)  # Adjusted the rotation and font size of xticks
plt.xlabel("State name", fontsize=12)
plt.ylabel("Total Actual Votes - Total Voters", fontsize=12)

# Ensure the labels are displayed properly
for i, txt in enumerate(result_df['New']):
    ax.annotate(txt, (result_df['State Name'][i], result_df['New'][i]), textcoords="offset points",
xytext=(0, 10), ha='center', fontsize=8)

plt.show()
```

Output:

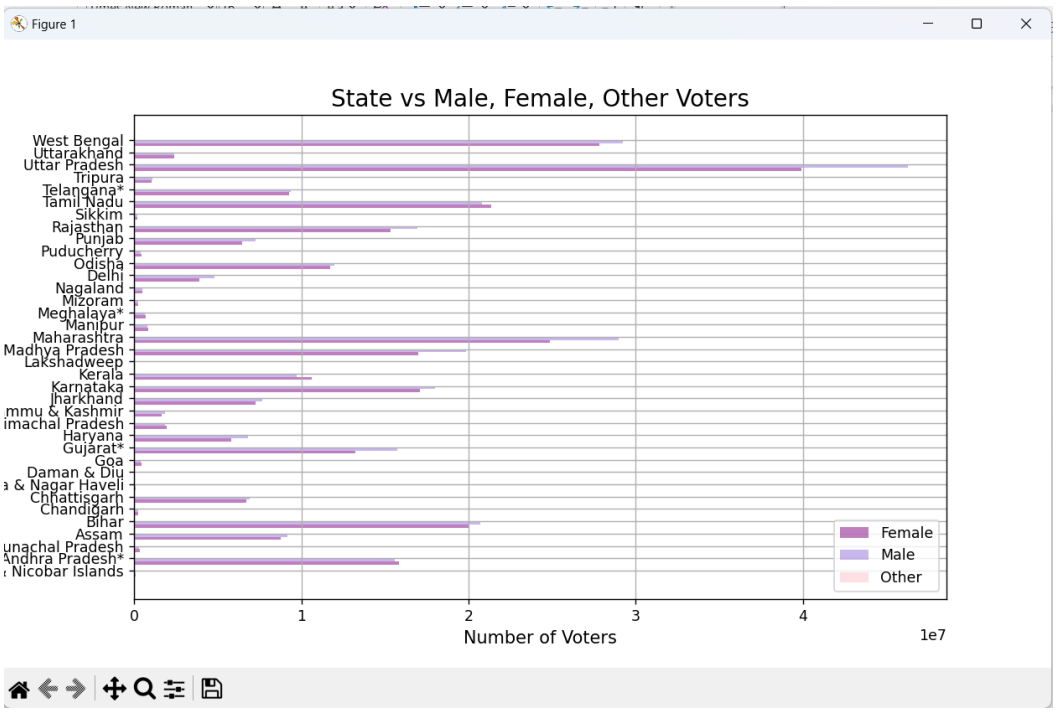


Practical: 7

Aim: Plot horizontal bar chart for states vs male, female and other votes (grouping of bars) with legend and suitable title.

```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd

result_df = pd.read_csv('Votes 2019.csv')

fig, ax = plt.subplots(figsize=(10, 6))  # Adjusted the figure size

pos = np.arange(len(result_df["State Name"]))
width = 0.25

ax.barh(pos, result_df['Female '], width, alpha=0.5, color='purple', label='Female')
ax.barh(pos + width, result_df['Male'], width, alpha=0.5, color='mediumpurple', label='Male')
ax.barh(pos + width * 2, result_df['Other '], width, alpha=0.5, color='pink', label='Other')

ax.set_title('State vs Male, Female, Other Voters', fontsize=16)
ax.set_yticks(pos + width * 1.5)
ax.set_yticklabels(result_df['State Name'], fontsize=10)
ax.set_xlabel('Number of Voters', fontsize=12)
ax.legend(loc='lower right', fontsize=10)

plt.grid()
plt.show()
```

Output:



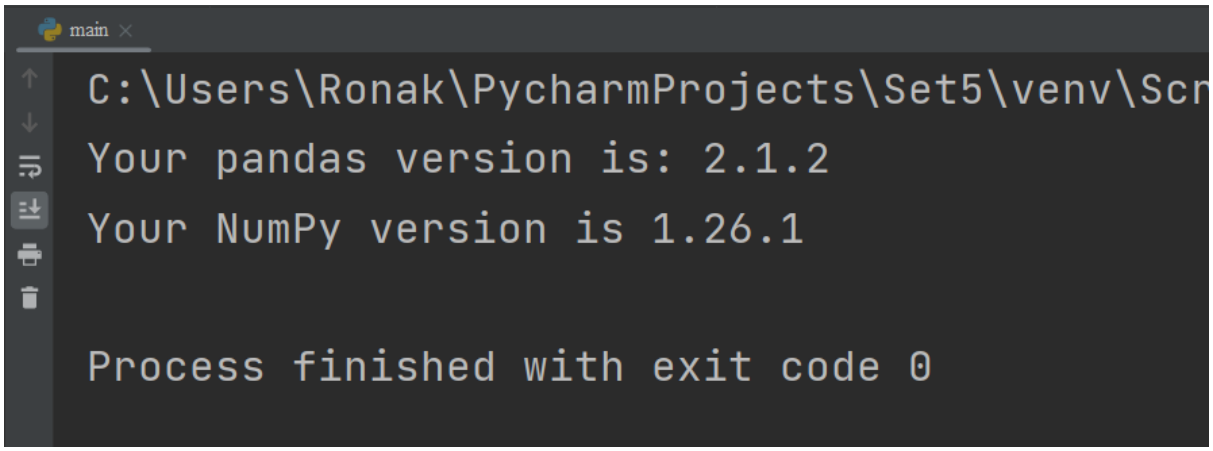
Practical Set-5: To study the sklearn Library and perform various statistics

Practical: 1

Aim: Load iris dataset from sklearn library given iris.csv file into appropriate data structure of pandas.

```
from sklearn.datasets import load_iris
iris = load_iris()
import pandas as pd
import numpy as np
print('Your pandas version is: %s' % pd.__version__)
print('Your NumPy version is %s' % np.__version__)
from sklearn.datasets import load_iris
iris = load_iris()
iris_nparray = iris.data
iris_dataframe = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_dataframe['group'] = pd.Series([iris.target_names[k] for k in iris.target],
dtype="category")
iris_dataframe
```

Output:



Practical: 2

Aim: Perform Descriptive Statistics for Numeric Data, Measuring central tendency, Measuring variance and range.

a) print(iris_dataframe.mean(numeric_only=True))

b) print(iris_dataframe.median(numeric_only=True))

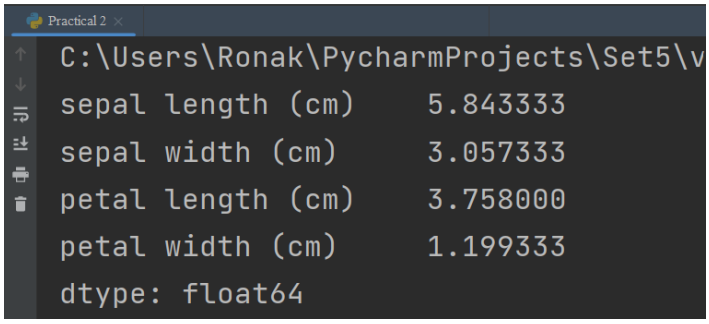
c) print(iris_dataframe.std())

d) print(iris_dataframe.max(numeric_only=True) - iris_dataframe.min(numeric_only=True))

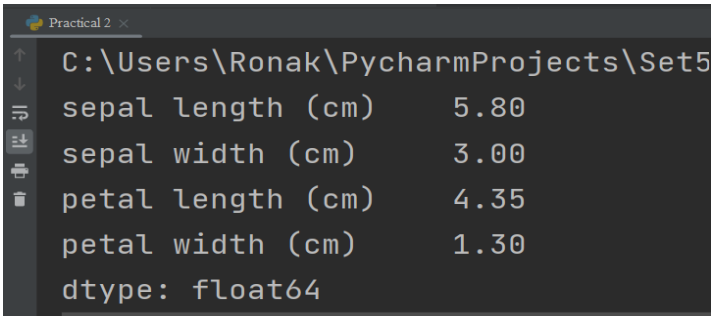
e) print(iris_dataframe.quantile([0,.25,.50,.75,1]))

Output:

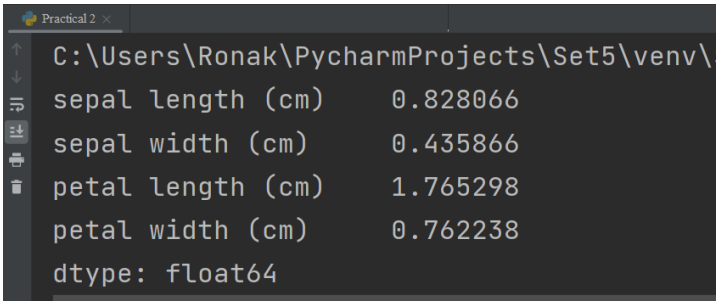
a)



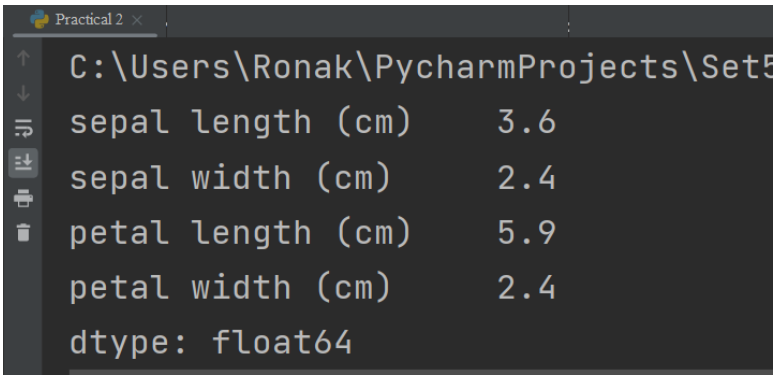
b)



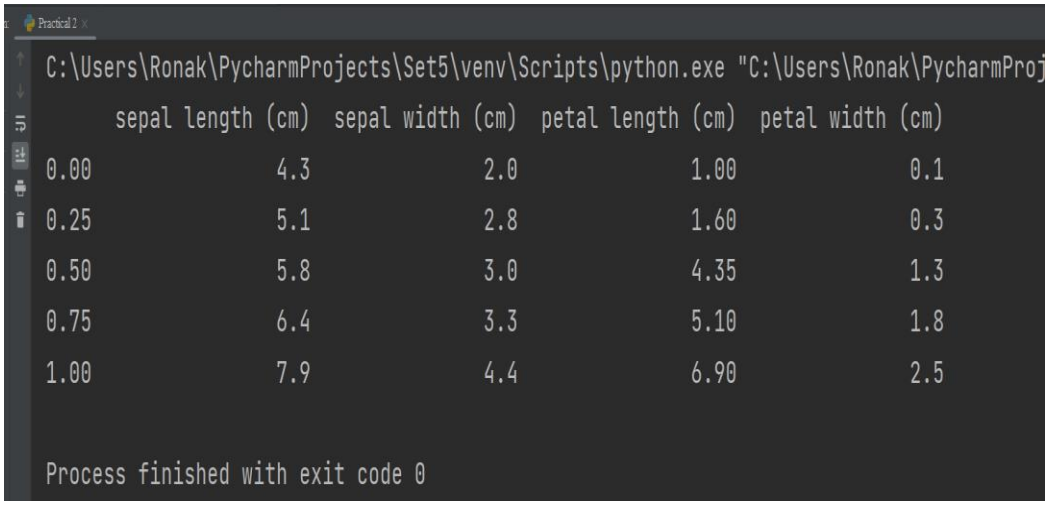
c)



d)


Practical 2
C:\Users\Ronak\PycharmProjects\Set5
sepal length (cm) 3.6
sepal width (cm) 2.4
petal length (cm) 5.9
petal width (cm) 2.4
dtype: float64

e)


Practical 2
C:\Users\Ronak\PycharmProjects\Set5\venv\Scripts\python.exe "C:\Users\Ronak\PycharmProj
sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)
0.00 4.3 2.0 1.00 0.1
0.25 5.1 2.8 1.60 0.3
0.50 5.8 3.0 4.35 1.3
0.75 6.4 3.3 5.10 1.8
1.00 7.9 4.4 6.90 2.5
Process finished with exit code 0

Practical: 3

Aim: To Working with percentiles and defining measures of normality.

a)

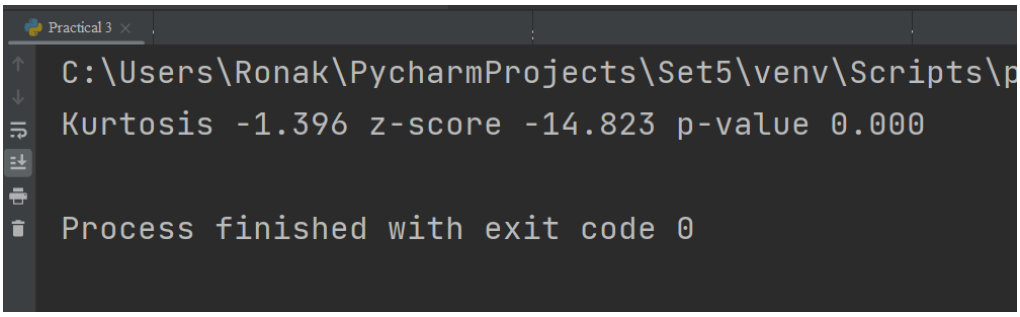
```
from scipy.stats import kurtosis,
kurtosistest variable =
iris_dataframe['petal length (cm)']k
= kurtosis(variable)
zscore, pvalue = kurtosistest(variable)
print('Kurtosis %0.3f z-score %0.3f p-value %0.3f' % (k, zscore, pvalue))
```

b)

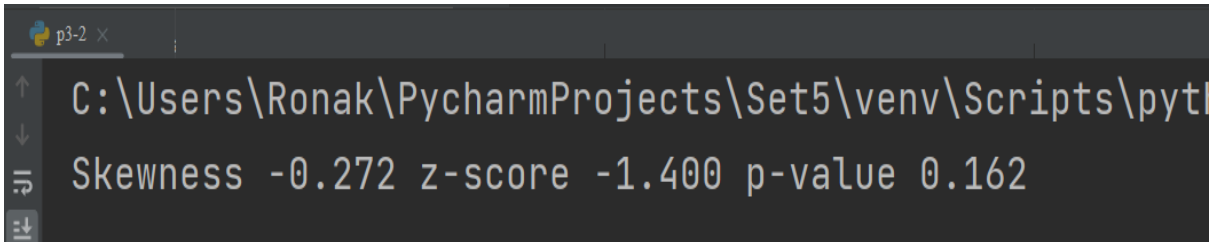
```
from scipy.stats import skew,
skewtest variable =
iris_dataframe['petal length (cm)']s
= skew(variable)
zscore, pvalue = skewtest(variable)
print('Skewness %0.3f z-score %0.3f p-value %0.3f' % (s, zscore, pvalue))
```

Output:

a)



b)



Practical: 4

Aim: To Counting for Categorical Data, understanding frequencies, Creating contingency tables.

a)

```
pcts = [0, .25, .5, .75, 1]
iris_binned = pd.concat(
    [pd.qcut(iris_dataframe.iloc[:,0], pcts,
    precision=1),
    pd.qcut(iris_dataframe.iloc[:,1], pcts,
    precision=1),
    pd.qcut(iris_dataframe.iloc[:,2], pcts,
    precision=1),
    pd.qcut(iris_dataframe.iloc[:,3], pcts,
    precision=1)], join='outer', axis = 1)
print(iris_dataframe['group'].value_counts())
```

b)

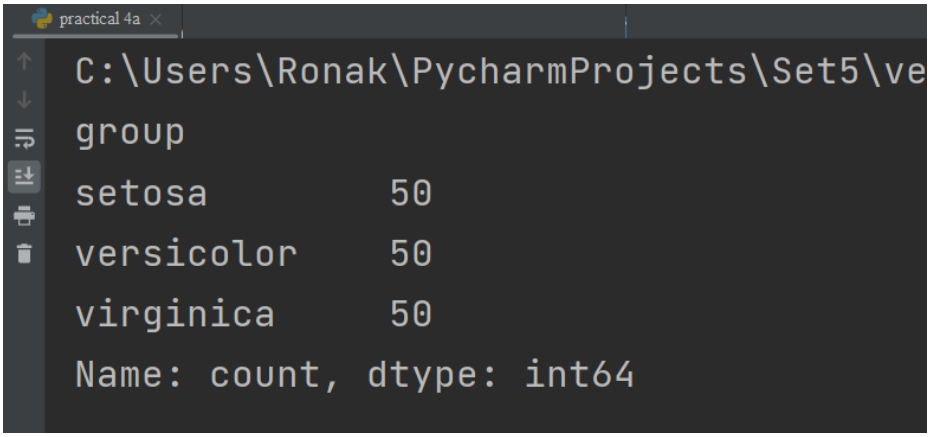
```
print (iris_binned['petal length (cm)'].value_counts())
```

c)

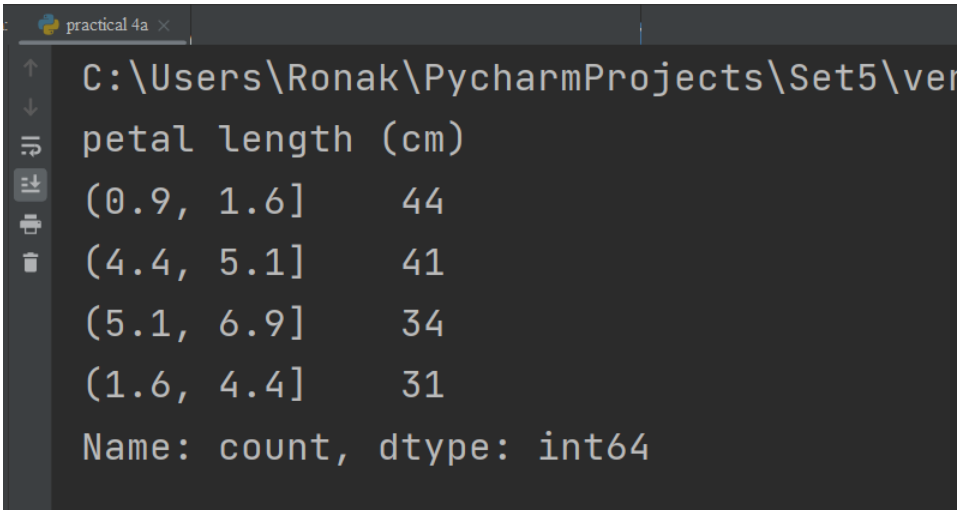
```
print(pd.crosstab(iris_dataframe['group'], iris_binned['petal length (cm)']))
```

Output:

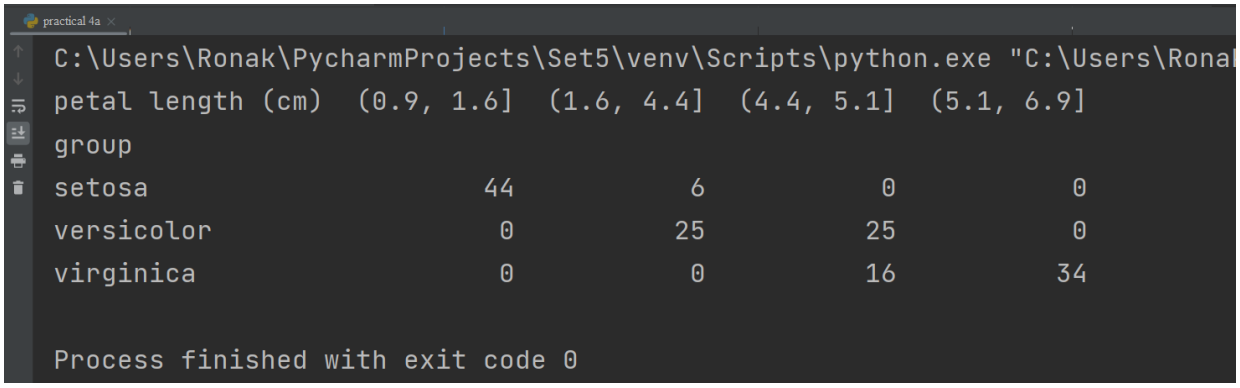
a)


practical 4a
C:\Users\Ronak\PycharmProjects\Set5\ve
group
setosa 50
versicolor 50
virginica 50
Name: count, dtype: int64

b)


practical 4a
C:\Users\Ronak\PycharmProjects\Set5\ver
petal length (cm)
(0.9, 1.6] 44
(4.4, 5.1] 41
(5.1, 6.9] 34
(1.6, 4.4] 31
Name: count, dtype: int64

c)


practical 4a
C:\Users\Ronak\PycharmProjects\Set5\venv\Scripts\python.exe "C:\Users\Ronal
petal length (cm) (0.9, 1.6] (1.6, 4.4] (4.4, 5.1] (5.1, 6.9]
group
setosa 44 6 0 0
versicolor 0 25 25 0
virginica 0 0 16 34
Process finished with exit code 0

Practical: 5

Aim: To Creating Applied Visualization for EDA like boxplots.

```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

iris = load_iris()
iris_nparray = iris.data
iris_dataframe = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_dataframe['group'] = pd.Series([iris.target_names[k] for k in iris.target],
dtype="category")

# Creating the boxplot
ax = iris_dataframe.boxplot(fontsize=9)
ax.set_xticklabels(iris.feature_names, rotation=45, ha='right')  # Setting x-axis labels
ax.set_ylabel('Measurement (cm)')  # Setting y-axis label
plt.title('Boxplot of Iris Dataset Features')  # Setting the title of the plot
plt.show()
```

a)

```
boxplots = iris_dataframe.boxplot(fontsize=9)
```

b)

```
import matplotlib.pyplot as plt
boxplots = iris_dataframe.boxplot(column='petal length (cm)', by='group',
fontsize=10)plt.suptitle("")
plt.show()
```

Output:

a)

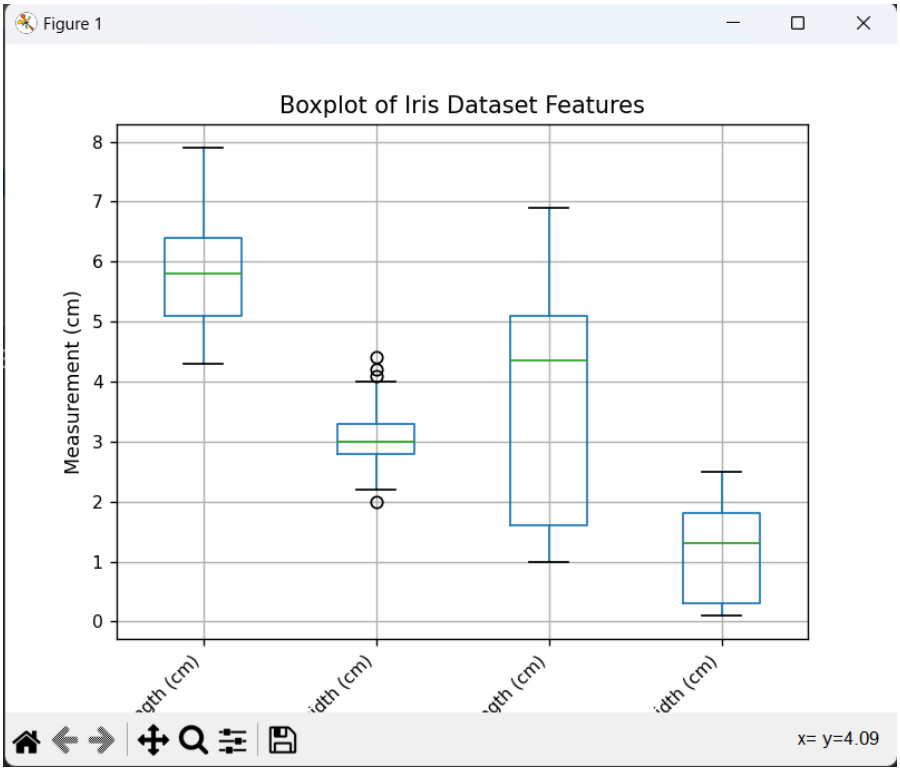


Figure 1
Boxplot of Iris Dataset Features
Measurement (cm)
0
1
2
3
4
5
6
7
8
x= y=4.09

b)

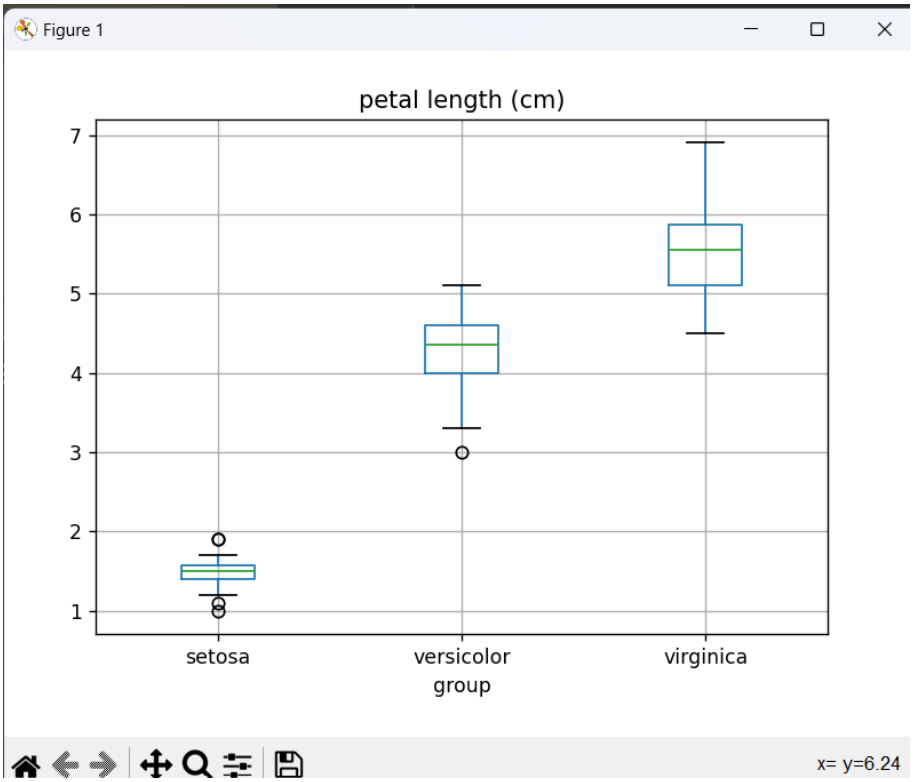


Figure 1
petal length (cm)
1
2
3
4
5
6
7
setosa
versicolor
virginica
group
x= y=6.24

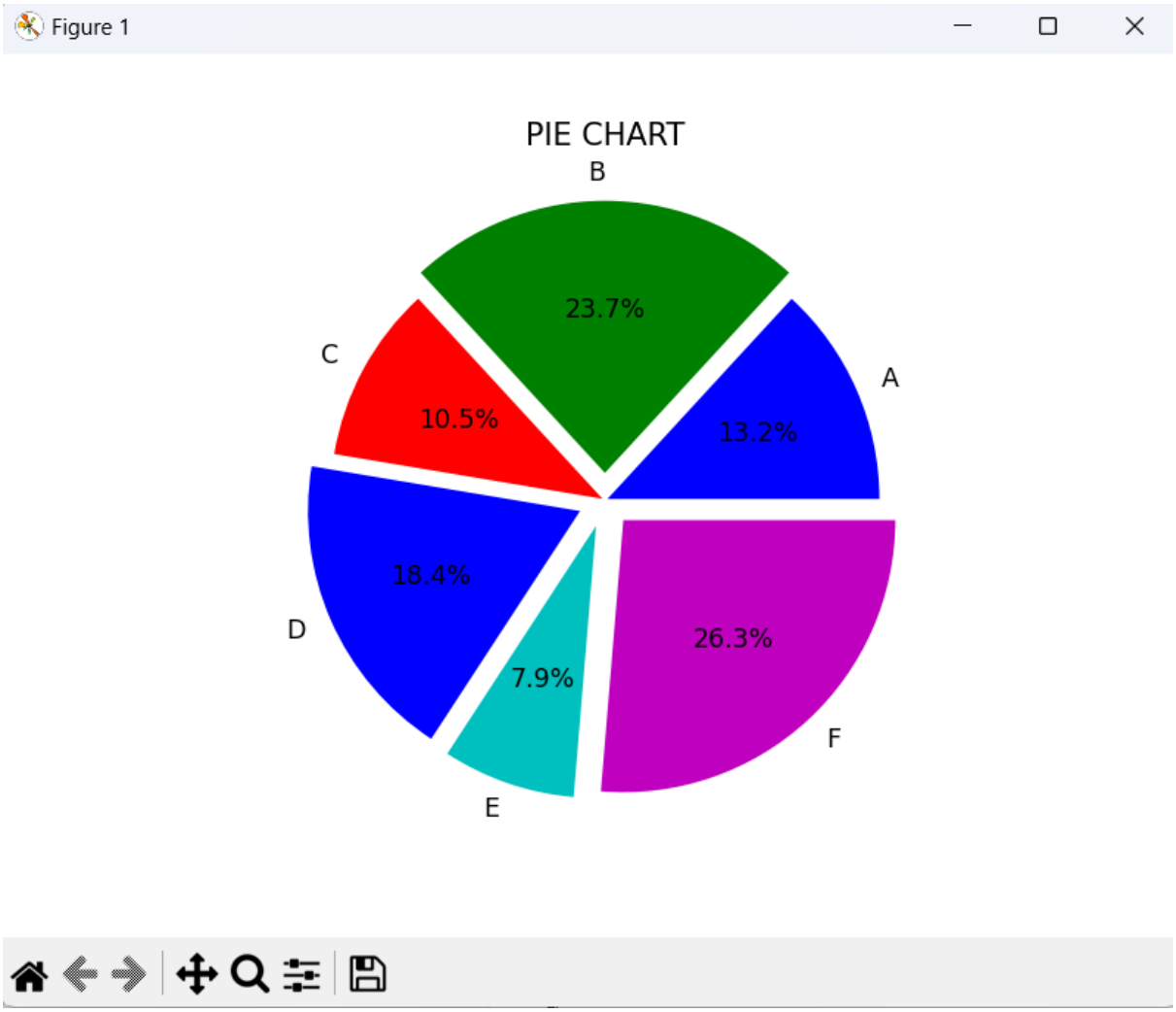
Practical Set-6: Create various plots using matplotlib library.

Practical: 1

Aim: Prepare a Pie charts by taking suitable data as reference.

```
import matplotlib.pyplot as plt
#%matplotlib inline
values = [5,9,4,7,3,10]
c = ['b','g','r','b','c','m']
l = ['A','B','C','D','E','F']
e = (0.01,0.1,0.01,0.1,0.1,0.1)
plt.pie(values,colors=c,labels=l,explode=e,autopct='%1.1f%%',shadow=False)
plt.title('PIE CHART');
plt.show()
```

Output:

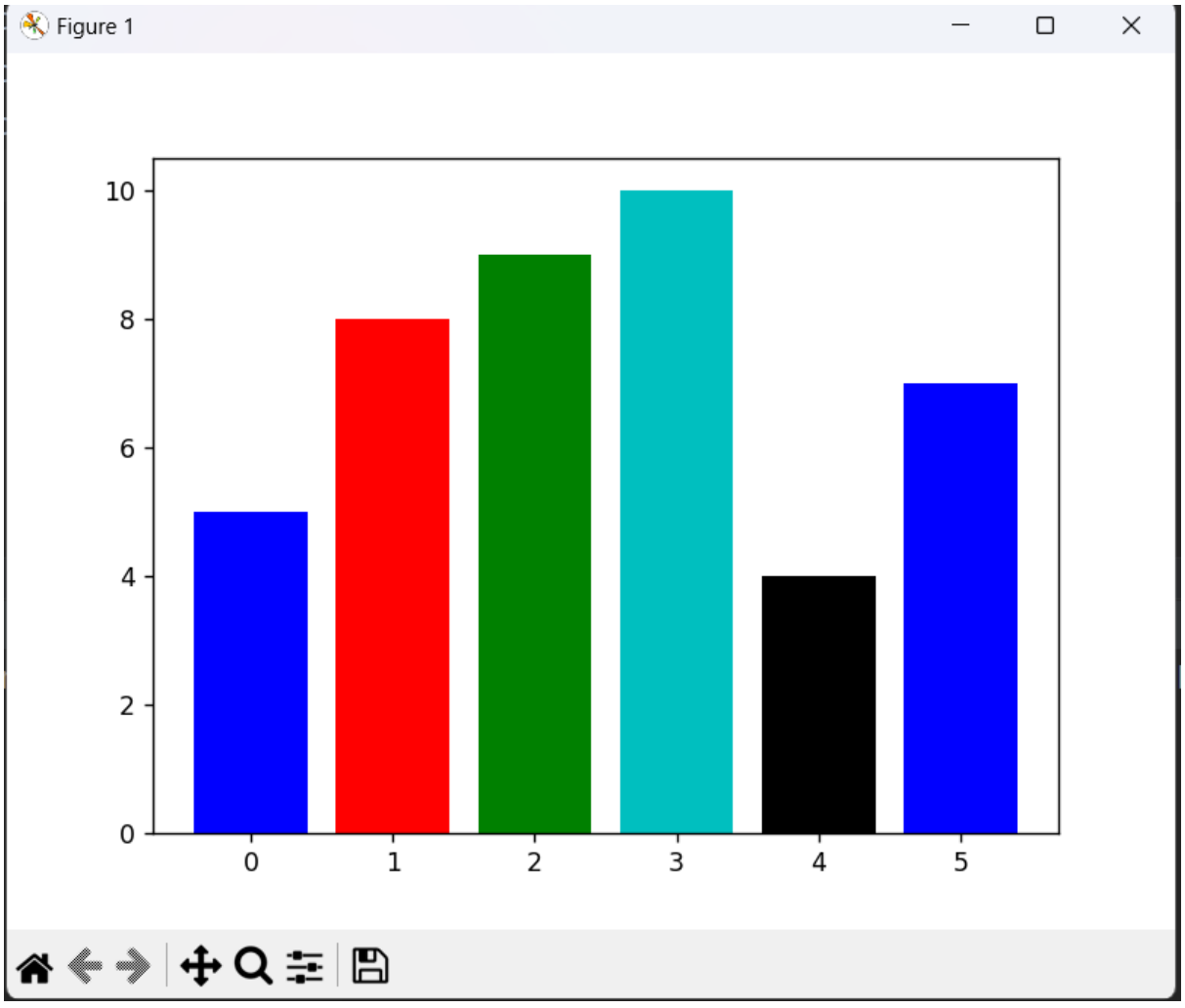


Practical: 2

Aim: Prepare a Bar charts by taking suitable data as reference.

```
import matplotlib.pyplot as plt
#%matplotlib inline
x=[5,8,9,10,4,7]
y=[0.7,0.8,0.7,0.7,0.8,0.7]
colors = ['b','r','g','c','k','b']
plt.bar(range(0,6),x,color=colors,align='center')
plt.show()
```

Output:

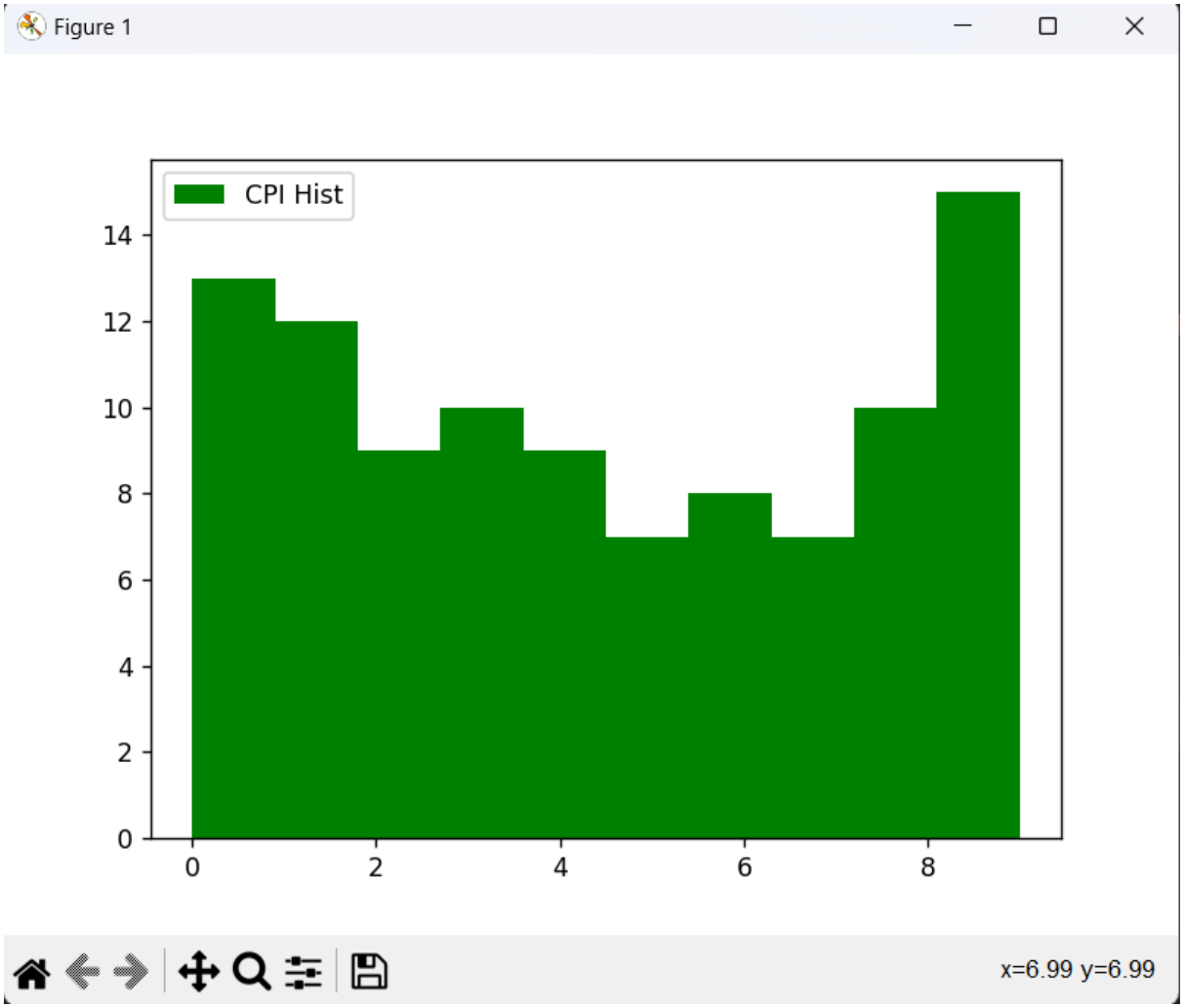


Practical: 3

Aim: Prepare a Histograms by taking suitable data as reference.

```
import matplotlib.pyplot as plt

import numpy as np

#%matplotlib notebook

cpis = np.random.randint(0,10,100)

plt.hist(cpis,bins=10,histtype='stepfilled',align='mid',label='CPI Hist',color='g')

plt.legend()

plt.show()
```

Output:

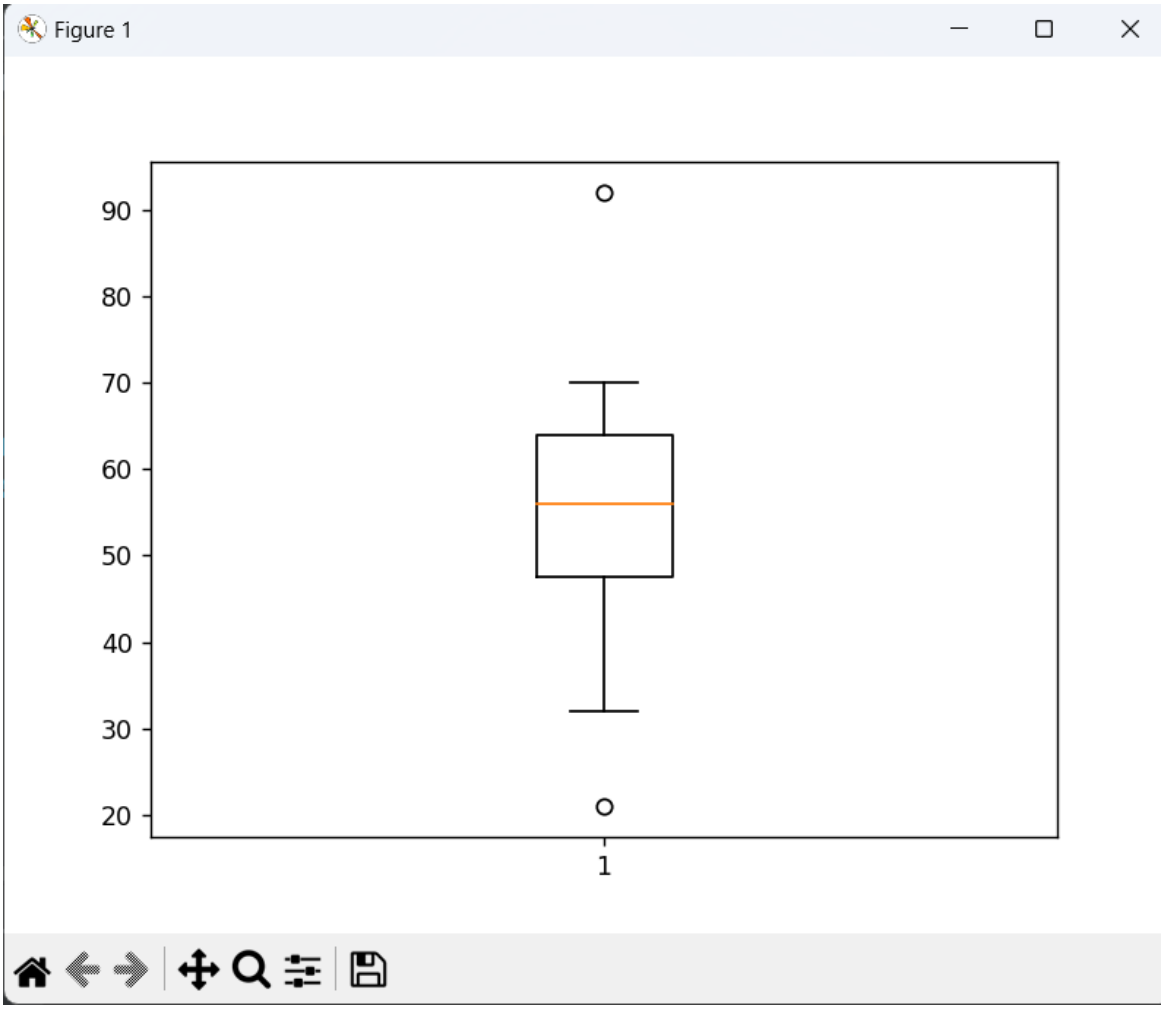


Practical: 4

Aim: Prepare a Box plots by taking suitable data as reference.

```
import pandas as pd

import matplotlib.pyplot as plt

timetaken =pd.Series([50,45,52,63,70,21,56,68,54,57,35,62,65,92,32])

plt.boxplot(timetaken)
plt.show()
```

Output:



Practical: 5

Aim: Prepare a Scatterplots by taking suitable data as reference.

```
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('insurance.csv')
grouped = df.groupby(['Smoker'])
for key, group in grouped:
    plt.scatter(group['bmi'],
    group['Charges'],
    label='Smoke ='+ str(key))

plt.legend()
plt.show()
```

Insurance.csv

bmi	Smoker	Charges
27.9	Yes	16884.9
33.77	No	1725.55
33	No	4449.46
22.705	No	21984.5
28.88	No	3866.86
25.74	No	3756.62
33.44	No	8240.59
27.74	Yes	7281.51
29.83	Yes	6406.41
25.84	No	28923.1
22.4	Yes	36120.9
30.2	Yes	40273.7
28.5	No	1137.01
34.1	Yes	43921.2
31.9	Yes	38282.8
28.6	No	3238.44
24.7	Yes	39611.8
22.3	No	2322.62
36.9	Yes	38511.6
32.2	Yes	43993.8
28.1	No	2234.38
30.6	Yes	40932.4
23.9	No	2128.43
36.7	Yes	38571.9
31.5	Yes	39722.8
27.3	No	3258.96
26.2	No	2897.32
25.1	No	2464.62
23.2	No	2349.43
30.1	Yes	36837.5

Output:

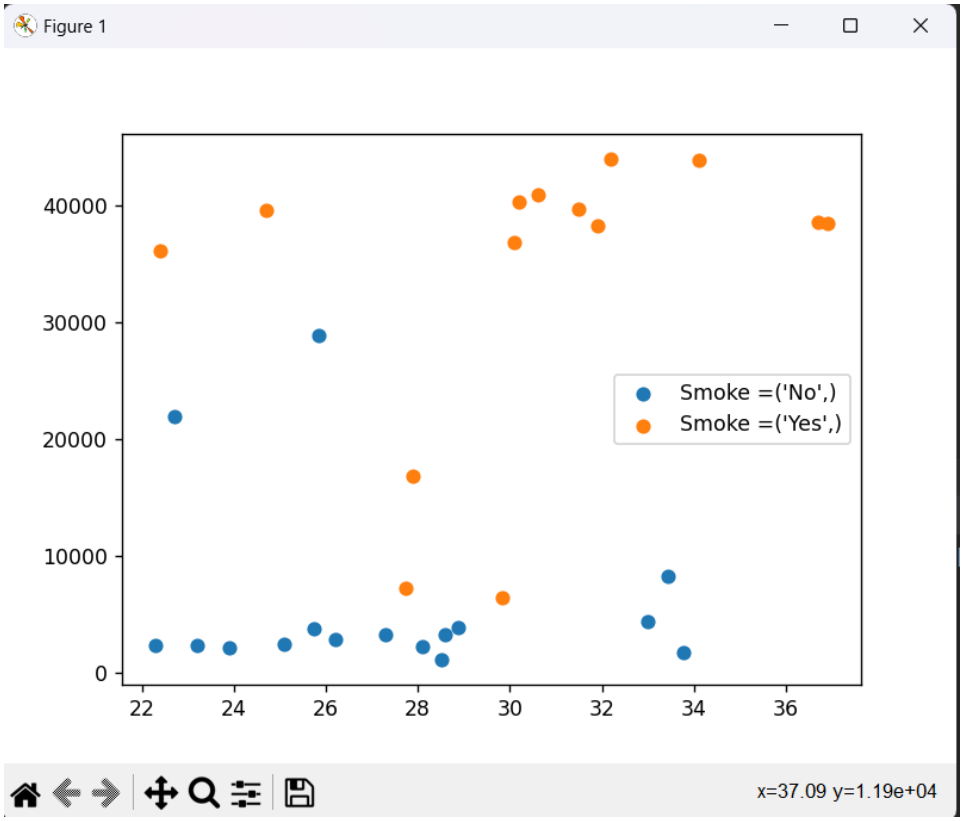


Figure 1
40000
30000
20000
10000
0
22
24
26
28
30
32
34
36
Smoke =('No',)
Smoke =('Yes',)
x=37.09 y=1.19e+04

Practical: 6

Aim: Prepare a Time Series by taking suitable data as reference.

```
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
import numpy as np

start_date = dt.datetime(2022, 10, 20)
end_date = dt.datetime(2022, 11, 5)
daterange = pd.date_range(start_date, end_date)
sales = (np.random.rand(len(daterange)) * 50).astype(int)
df = pd.DataFrame(sales, index=daterange, columns=['Sales'])
df.loc['Oct 4 2022':'Nov 04 2022'].plot()
plt.ylim(0,50)
plt.xlabel('Sales Date')
plt.ylabel('Sale Value')
plt.title('Plotting Time')
plt.show()
```

Output: