1. a) Write a python program to find the best of two test average marks out of three test's marks accepted from the user.

```
m1 = int(input("Enter marks for test1 : "))
m2 = int(input("Enter marks for test2 : "))
m3 = int(input("Enter marks for test3 : "))

if m1 <= m2 and m1 <= m3:
    avgMarks = (m2+m3)/2
 elif m2 <= m1 and m2 <= m3:
             avgMarks = (m1+m3)/2
 elif m3 <= m1 and m2 <= m2:
     avgMarks = (m1+m2)/2

best_of_two = sorted([m1, m2, m3], reverse=True)[:2]
average_best_of_two = sum(best_of_two)/2

print("Average of best two test marks out of three test's marks is", average_best_of_two);
```

OUTPUT

```
Enter marks for test1 : 45
Enter marks for test2 : 39
Enter marks for test3 : 48
Average of best two test marks  out of three test's marks is 46.5
```

1.b) Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.

```
from collections import Counter

value = input("Enter a value : ")
if value == value[::-1]:
    print("Palindrome")
else:
    print("Not Palindrome")

counted_dict = Counter(value)
for key in sorted(counted_dict.keys()):
    print(f'{key} appears {counted_dict[key]} times');

#Alternate way to count appearances
for i in range(10):
    if value.count(str(i)) > 0:
        print(f'{str(i)} appears {value.count(str(i))} times')
```

OUTPUT 1:

```
Enter a value : 1234234
Not Palindrome
1 appears 1 times
2 appears 2 times
3 appears 2 times
4 appears 2 times
```

OUTPUT 2:

```
Enter a value : 12321
Palindrome
1 appears 2 times
2 appears 2 times
3 appears 1 times
```

2. a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.

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

try:
    num = int(input("Enter a number : "))
    if num > 0:
        print(f' fn({num}) = {fn(num)}')
    else:
        print("Input should be greater than 0")
except ValueError:
    print("Try with numeric value")
```

OUTPUT 1

```
Enter a number : 5
fn(5) = 3
```

OUTPUT 2

```
Enter a number: -3
Input should be greater than 0
```

OUTPUT 3

```
Enter a number: abc
Try with numeric value
```

2. b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions.

```
def bin2Dec(val):
    rev=val[::-1]
    dec = 0
    i = 0
    for dig in rev:
        dec += int(dig) * 2**i
        i += 1

return dec


def oct2Hex(val):
    rev=val[::-1]
    dec = 0
    i = 0
    for dig in rev:
        dec += int(dig) * 8**i
        i += 1
    list=[]
    while dec != 0:
        list.append(dec%16)
        dec = dec // 16

    nl=[]
    for elem in list[::-1]:
        if elem <= 9:
            nl.append(str(elem))
        else:
            nl.append(chr(ord('A') + (elem -10)))
    hex = "".join(nl)

    return hex


base = 2
num1 = input("Enter a binary number : ")
# print(bin2Dec(num1))
print(int(num1, base))
"""
#A better implementation

def bin2Dec(val):
    return int(val, 2)

def oct2Hex(val):
    return int(val, 8)

try:
    num1 = input("Enter a binary number : ")
    print(bin2Dec(num1))
except ValueError:
    print("Invalid literal in input with base 2")

try:
    num2 = input("Enter a octal number : ")
    print(oct2Hex(num2))
except ValueError:
    print("Invalid literal in input with base 8")
```

OUTPUT 1

```
Enter a binary number: 101010
42
```

OUTPUT 2

```
Enter an octal number: 755
0x1FD
```

OUTPUT 3

```
Enter a binary number: 11011a
Invalid literal in input with base 2
```

OUTPUT 4

```
Enter an octal number: 1298
Invalid literal in input with base 8
```

3. a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters.

```
import  string

sentence = input("Enter a sentence : ")

wordList = sentence.strip().split(" ")
print(f'This sentence has {len(wordList)} words', end='\n\n')

digit_count = uppercase_count = lowercase_count = 0

for ch in sentence:
    if '0' <= ch <= '9':
        digit_count += 1
    elif 'A' <= ch <= 'Z':
        uppercase_count += 1
    elif 'a' <= ch <= 'z':
        lowercase_count += 1

for character in sentence:
    if character in string.digits:
        digit_count += 1
    elif character in string.ascii_uppercase:
        uppercase_count += 1
    elif character in string.ascii_lowercase:
        lowercase_count += 1

print(f'This sentence has {digit_count} digits',
      f' {uppercase_count} upper case letters',
      f' {lowercase_count} lower case letters', sep='\n')
```

OUTPUT 1

```
Enter a sentence : Rama went to Devaraja market to pick 2 kgs of vegetable
This sentence has 11 words

This sentence has 1 digits
2 upper case letters
42 lower case letters
```

OUTPUT 2

```
Enter a sentence: Python is Fun!
This sentence has 3 words

This sentence has 0 digits
3 uppercase letters
```

9 lowercase letters

b) Write a Python program to find the string similarity between two given strings

Sample Output: Sample Output:

```
Original string:                        Original string:
Python Exercises                        Python Exercises
Python Exercises                        Python Exercise
Similarity between two said strings: Similarity between two said strings:1.0
                                                   0.967741935483871
```

```
str1 = input("Enter String 1 \n").lower()
str2 = input("Enter String 2 \n").lower()

# if len(str2) < len(str1):
#     short = len(str2)
#     long = len(str1)
# else:
#     short = len(str1)
#     long = len(str2)

string_1_length = len(str1)
string_2_length = len(str2)

short_string_length, long_string_length = min(string_1_length, string_2_length),
max(string_1_length, string_2_length)


match_count = 0
for i in range(short_string_length):
    if str1[i] == str2[i]:
        match_count += 1

print("Similarity between two said strings:")
print(match_count/long_string_length)
```

OUTPUT 1

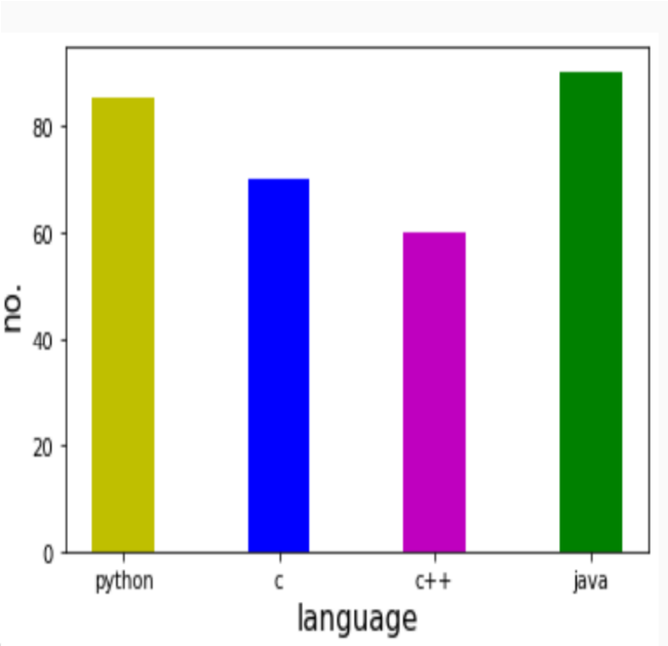
```
Enter String 1 : Python Exercises
Enter String 2 : Python Exercise
Similarity between strings "Python Exercises" and "Python Exercise" is :  0.967741935483871
```

OUTPUT 2

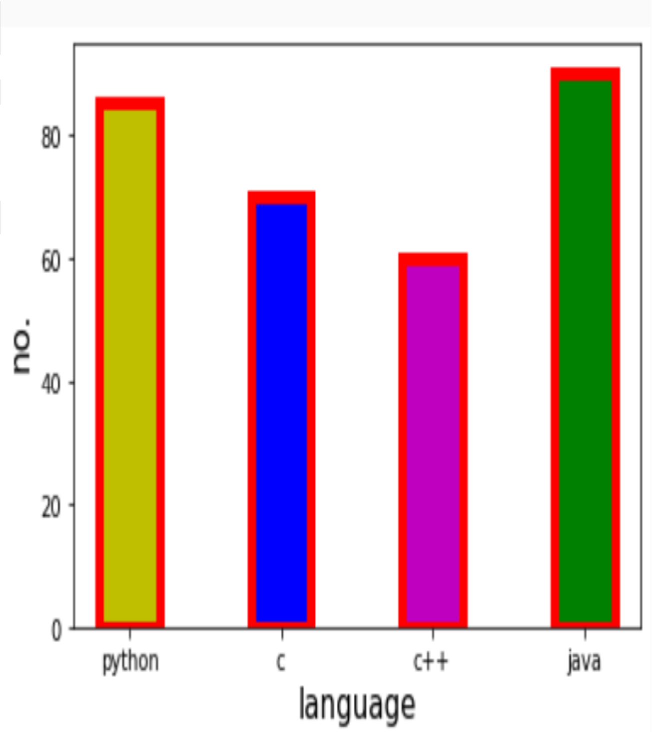
```
Enter String 1 : Python Exercises
Enter String 2 : Python Exercises
Similarity between strings "Python Exercises" and "Python Exercises" is :  1.0
```

4. a) Write a Python program to Demonstrate how to Draw a Bar Plot using Matplotlib.

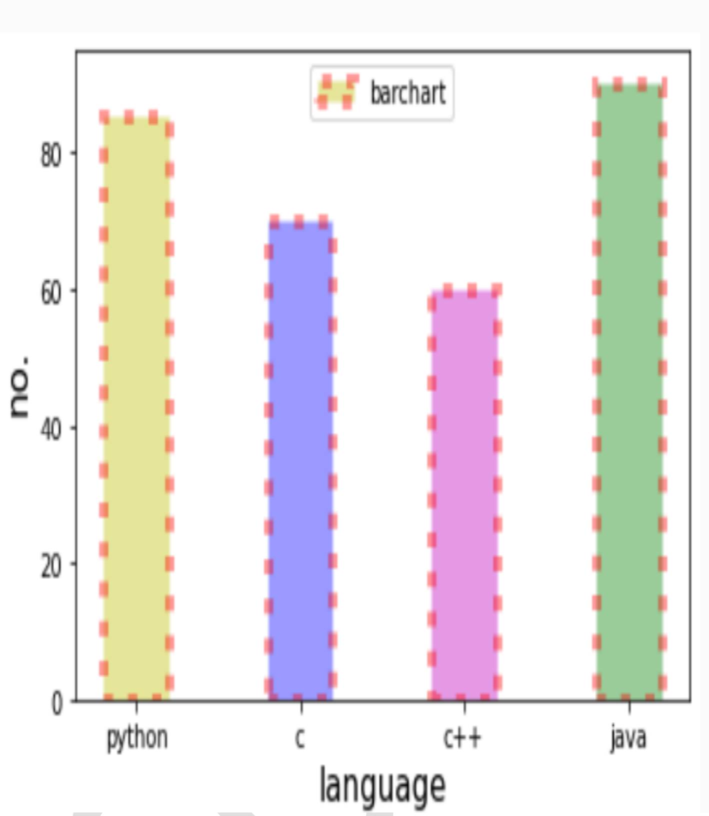
```
import  matplotlib.pyplot  as plt
x=["python","c","c++","java"]
y=[85,70,60,90]
plt.xlabel("language",fontsize=15)
plt.ylabel("no.",fontsize=15)
c=["y","b","m","g"]
plt.bar(x,y,width=0.4,color=c,align="center")
plt.show()
```



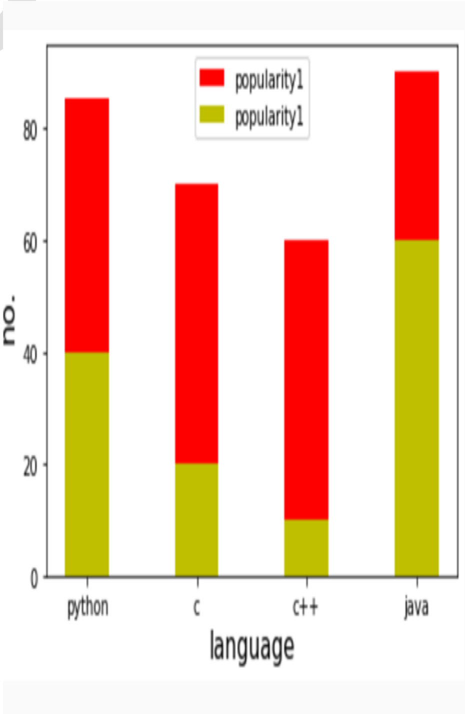
```
import matplotlib.pyplot as plt
x=["python","c","c++","java"]
y=[85,70,60,90]
plt.xlabel("language",fontsize=15)
plt.ylabel("no.",fontsize=15)
c=["y","b","m","g"]
plt.bar(x,y,width=0.4,color=c,align="cente",
edgecolor="r",linewidth=5)
plt.show()
```



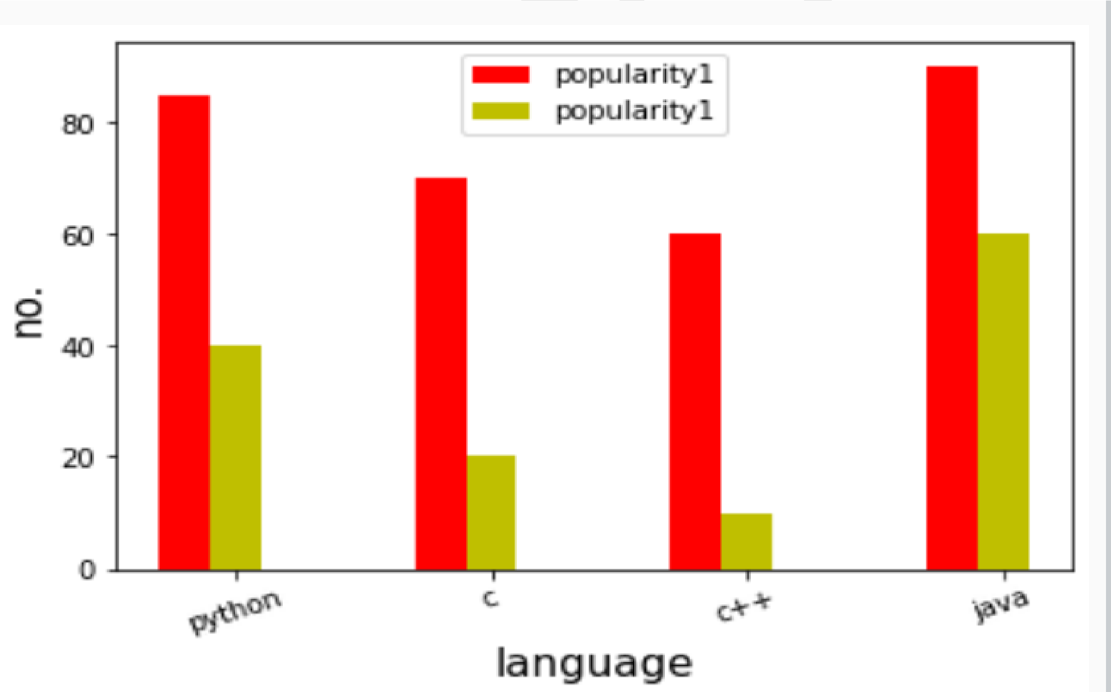
```
import  matplotlib.pyplot as plt
x=["python","c","c++","java"]
y=[85,70,60,90]
plt.xlabel("language",fontsize=15)
plt.ylabel("no.",fontsize=15)
c=["y","b","m","g"]
plt.bar(x,y,width=0.4,color=c,align="center",edgecolor="r",linewidth=5,linestyle=":",
alpha=0.4,label="barchart”)
plt.legend()
plt.show()
```



```
import matplotlib.pyplot as plt
x=["python","c","c++","java"]
y=[85,70,60,90]
z=[40,20,10,60]
plt.xlabel("language",fontsize=15)
plt.ylabel("no.",fontsize=15)
plt.bar(x,y,width=0.4,color="r",align="center",label="popularity1")
plt.bar(x,z,width=0.4,color="y",align="center",label="popularity1")
plt.legend()
plt.show()
```

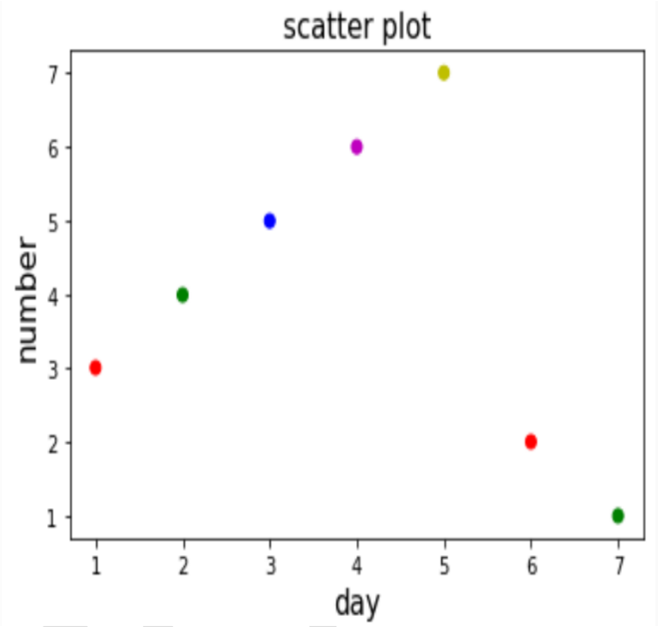


```
Importmatplotlib.pyplot as plt
import numpy as np


x=["python","c","c++","java"]
y=[85,70,60,90]
z=[40,20,10,60]

width=0.2
p=np.arange(len(x))
p1=[j+width for j in p]

plt.xlabel("language",fontsize=15)
plt.ylabel("no.",fontsize=15)

plt.bar(p,y,width,color="r",label="popularity1")
plt.bar(p1,z,width,color="y",label="popularity1")

plt.xticks(p+width,x,rotation=20)
plt.legend()
plt.show()
```

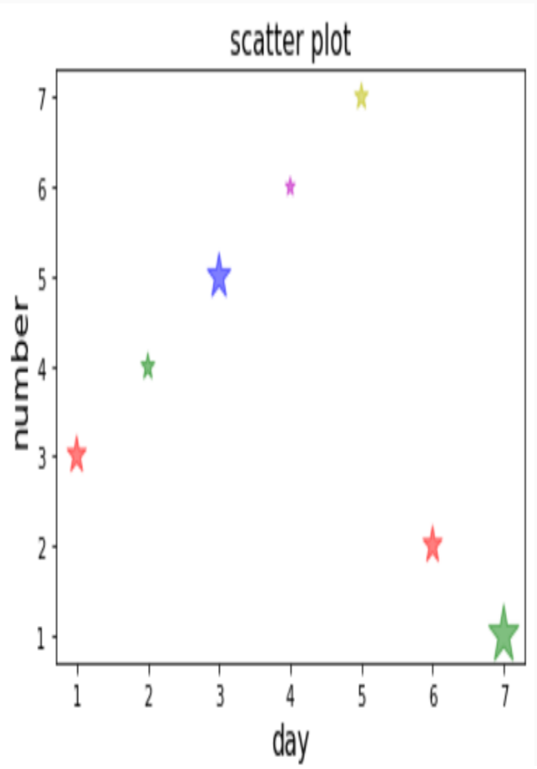


b) Write a Python program to Demonstrate how to Draw a Scatter Plot using Matplotlib.

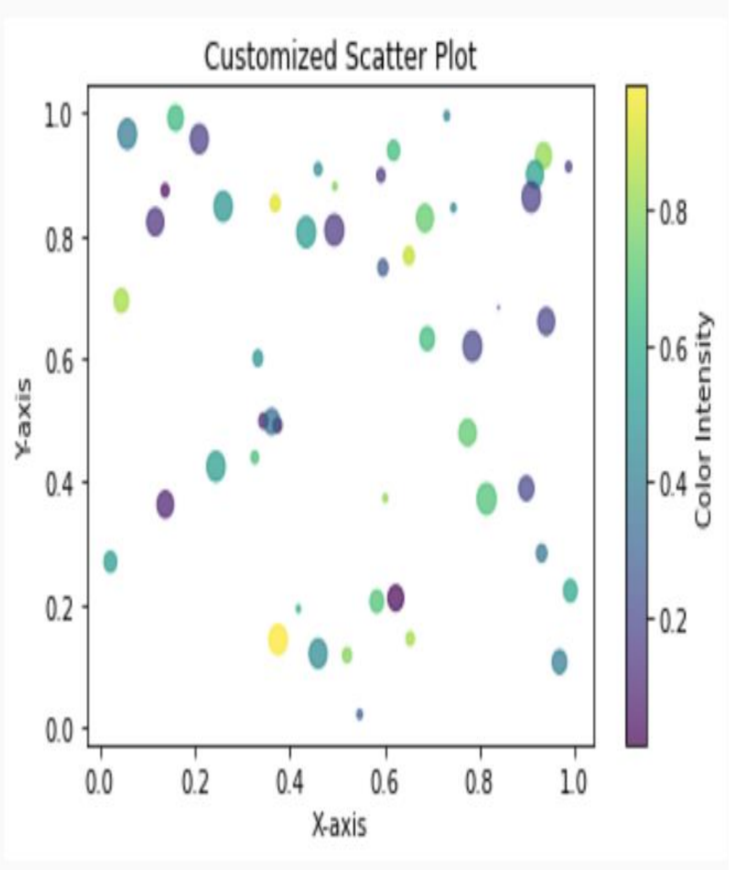
```
import matplotlib.pyplot as plt
day=[1,2,3,4,5,6,7]
number=[3,4,5,6,7,2,1]
colors=("r","g","b","m","y","r","g")
plt.scatter(day,number,c=colors)
plt.title("scatter plot",fontsize=15)
plt.xlabel("day",fontsize=15)
plt.ylabel("number",fontsize=15)
plt.show()
```



```
import matplotlib.pyplot as plt
day=[1,2,3,4,5,6,7]
number=[3,4,5,6,7,2,1]
colors=("r","g","b","m","y","r","g")
sizes=[200,100,300,50,100,200,500]
plt.scatter(day,number,c=colors,s=sizes,alpha=0.5,marker=
"*")
plt.title("scatter plot",fontsize=15)
plt.xlabel("day",fontsize=15)
plt.ylabel("number",fontsize=15)
plt.show()
```

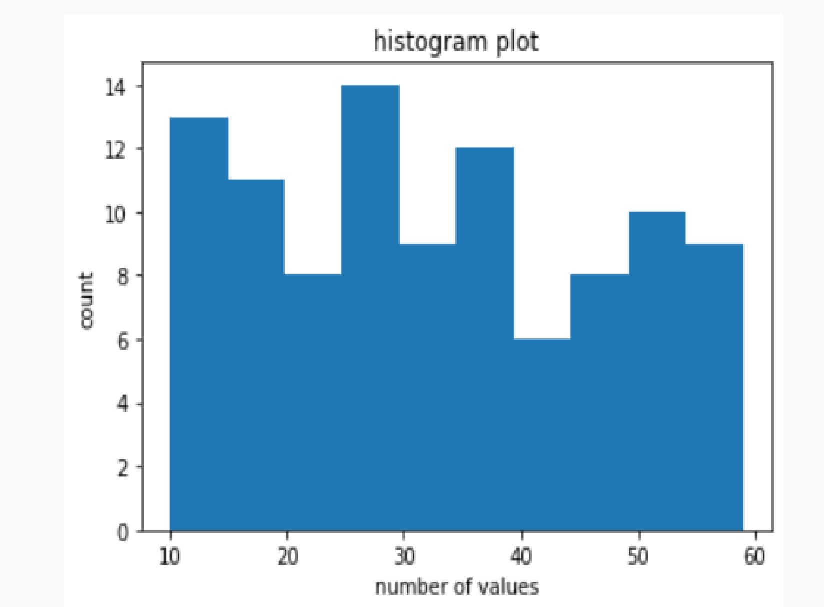


```
import matplotlib.pyplot as plt

import numpy as np


# Generate random data

x = np.random.rand(50)

y = np.random.rand(50)

colors = np.random.rand(50)

sizes = 100 * np.random.rand(50)


# Create a customized scatter plot

plt.scatter(x, y, c=colors, s=sizes, alpha=0.7, cmap='viridis')


# Add title and axis labels

plt.title("Customized Scatter Plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")


# Display color intensity scale

plt.colorbar(label='Color Intensity')


# Show the plot

plt.show()
```

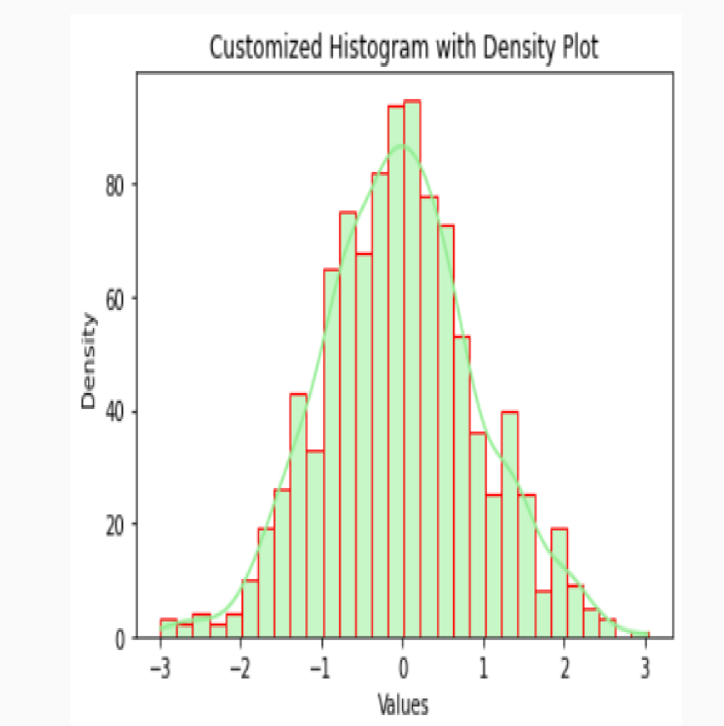


5 a) Write a Python program to Demonstrate how to Draw a Histogram Plot using Matplotlib.

```
import matplotlib.pyplot as plt

import numpy as np

import random

x=np.random.randint(10,60,(100))

print(x)

plt.hist(x)

plt.title("histogram plot")

plt.xlabel("number of values")

plt.ylabel("count")

plt.show()
```

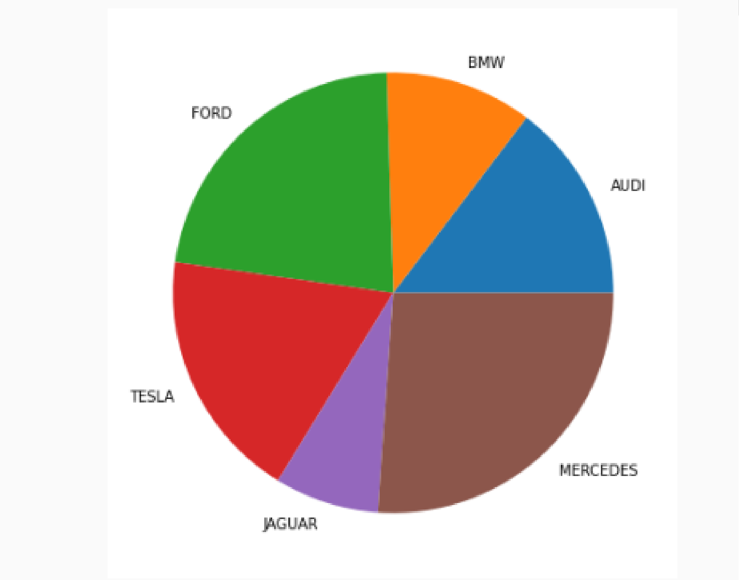


```
import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

# Generate random data for the histogram

data = np.random.randn(1000)

# Creating a customized histogram with a density plot

sns.histplot(data, bins=30, kde=True, color='lightgreen', edgecolor='red')

# Adding labels and title

plt.xlabel('Values')

plt.ylabel('Density')

plt.title('Customized Histogram with Density Plot')

plt.show()
```



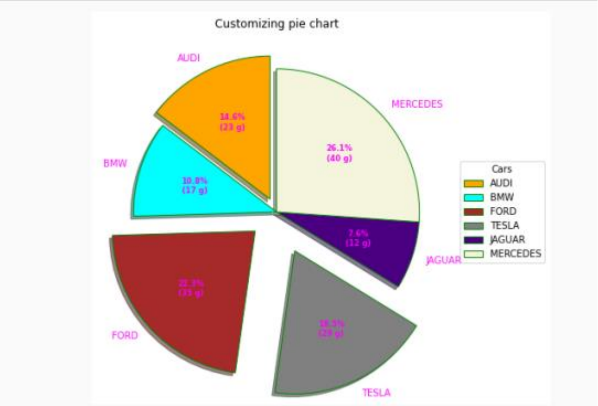
b) Write a Python program to Demonstrate how to Draw a Pie Chart using Matplotlib.

```
from matplotlib import pyplot as plt

import numpy as np


# Creating dataset
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

# Creating plot
fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = cars)

# show plot
plt.show()
```



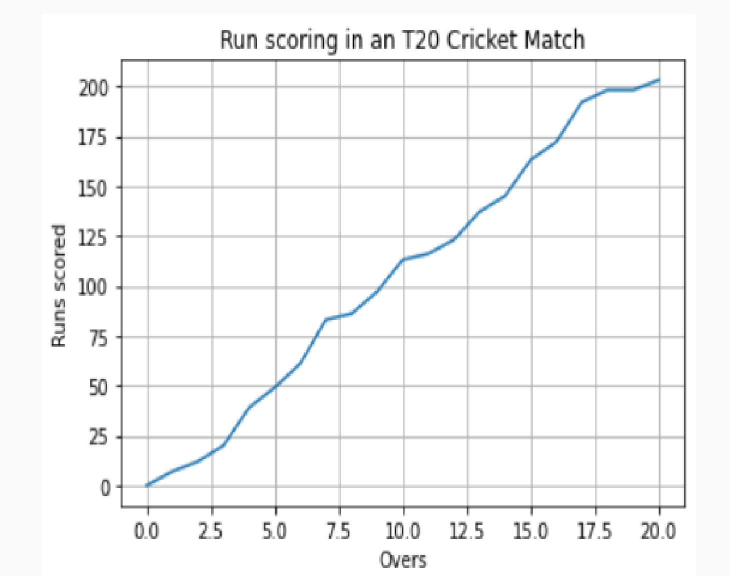
```
import numpy as np
import matplotlib.pyplot as plt
```

```
 # Creating dataset
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

 # Creating explode data
explode = (0.1, 0.0, 0.2, 0.3, 0.0, 0.0)

# Creating color parameters
colors = ( "orange", "cyan", "brown",
          "grey", "indigo", "beige")

# Wedge properties
wp = { 'linewidth' : 1, 'edgecolor' : "green" }

# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    return "{:.1f}%\n({:d} g)".format(pct, absolute)

# Creating plot
fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(data,
                                  autopct = lambda pct: func(pct, data),
                                  explode = explode,
                                  labels = cars,
                                  shadow = True,
                                  colors = colors,
                                  startangle = 90,
                                  wedgeprops = wp,
                                  textprops = dict(color ="magenta"))

# Adding legend
ax.legend(wedges, cars,
          title ="Cars",
          loc ="center left",
          bbox_to_anchor =(1, 0, 0.5, 1))
 plt.setp(autotexts, size = 8, weight ="bold")
ax.set_title("Customizing pie chart")

plt.show()
```



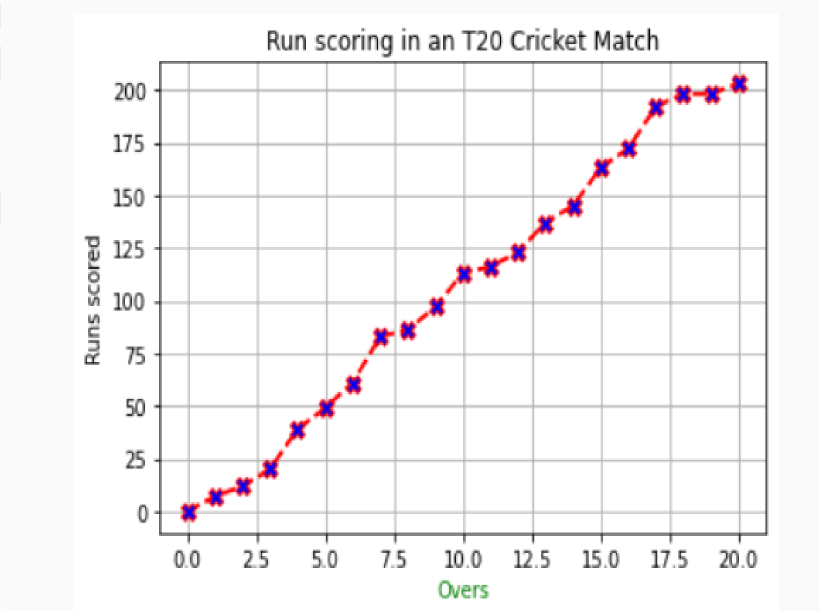
6 a) Write a Python program to illustrate Linear Plotting using Matplotlib.

```
import matplotlib.pyplot as plt


# Hypothetical data: Run rate in an T20 cricket match
overs = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
runs_scored = [0,7,12,20,39,49,61,83,86,97,113,116,123,137,145,163,172,192,198,198,203]


# Create a linear plot
plt.plot(overs, runs_scored)


# Add labels and title
plt.xlabel('Overs')
plt.ylabel('Runs scored')
plt.title('Run scoring in an T20 Cricket Match')


# Display the plot
plt.grid(True)
plt.show()
```



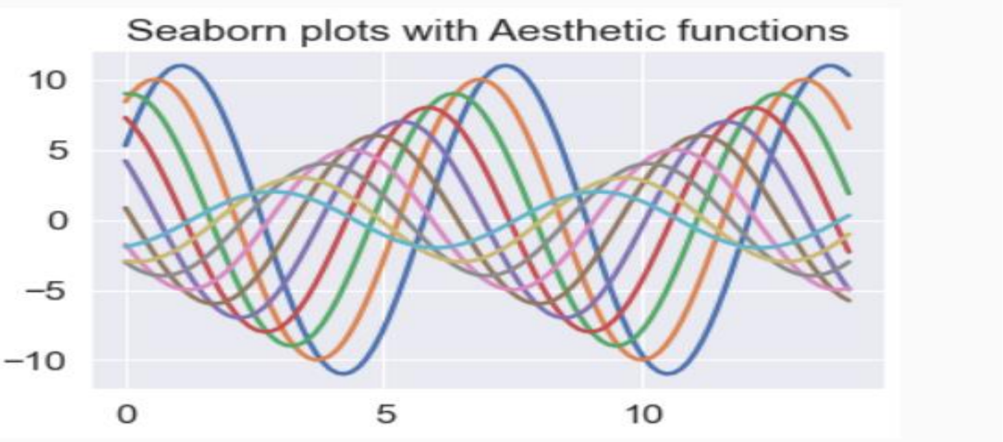
b) Write a Python program to illustrate liner plotting with line formatting using Matplotlib.

```
import matplotlib.pyplot as plt

# Hypothetical data: Run rate in an T20 cricket match

overs = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

runs_scored = [0,7,12,20,39,49,61,83,86,97,113,116,123,137,145,163,172,192,198,198,203]


# Create a linear plot

plt.plot(overs, runs_scored, marker='X', linestyle='dashed',color='red', linewidth=2, markerfacecolor='blue', markersize=8)


# Add labels and title

plt.xlabel('Overs', color = 'green')

plt.ylabel('Runs scored')

plt.title('Run scoring in an T20 Cricket Match')


# Display the plot

plt.grid(True)

plt.show()
```



7.Write a Python program which explains uses of customizing seaborn plots with Aesthetic functions.

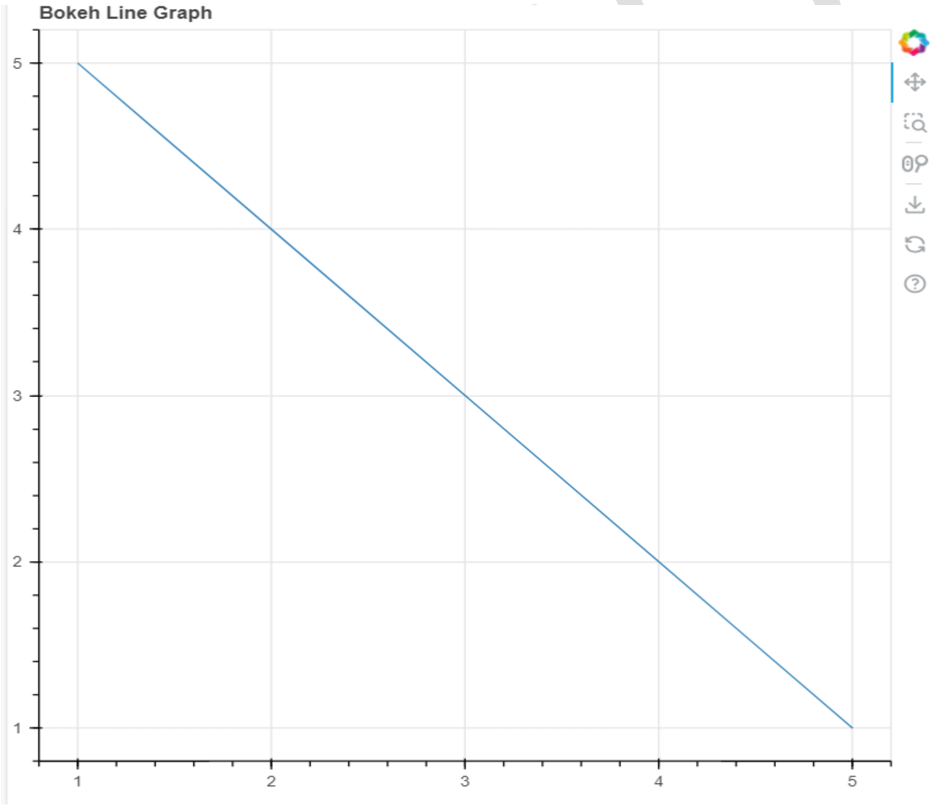
```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


def sinplot(n=10):

    x = np.linspace(0, 14, 100)

    for i in range(1, n + 1):

        plt.plot(x, np.sin(x + i * .5) * (n + 2 - i))

sns.set()

#sns.set_context("talk")

sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 2.5})


sinplot()

plt.title('Seaborn plots with Aesthetic functions')

plt.show()
```



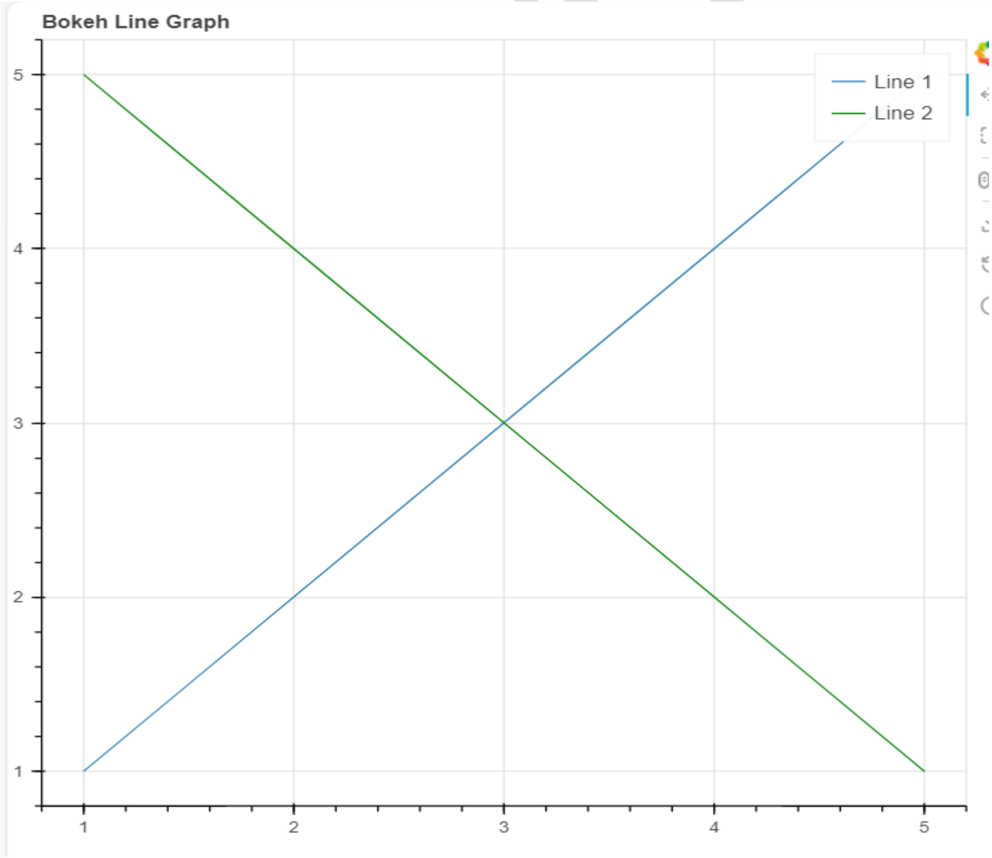
8. Write a Python program to explain working with bokeh line graph using Annotations and Legends.

a) Write a Python program for plotting different types of plots using Bokeh.

```
from bokeh.plotting import figure, output_file, show
  # instantiating the figure object
graph = figure(title = "Bokeh Line Graph")
 # the points to be plotted
x = [1, 2, 3, 4, 5]
y = [5, 4, 3, 2, 1]
# plotting the line graph
graph.line(x, y)
  # displaying the model
show(graph)
```

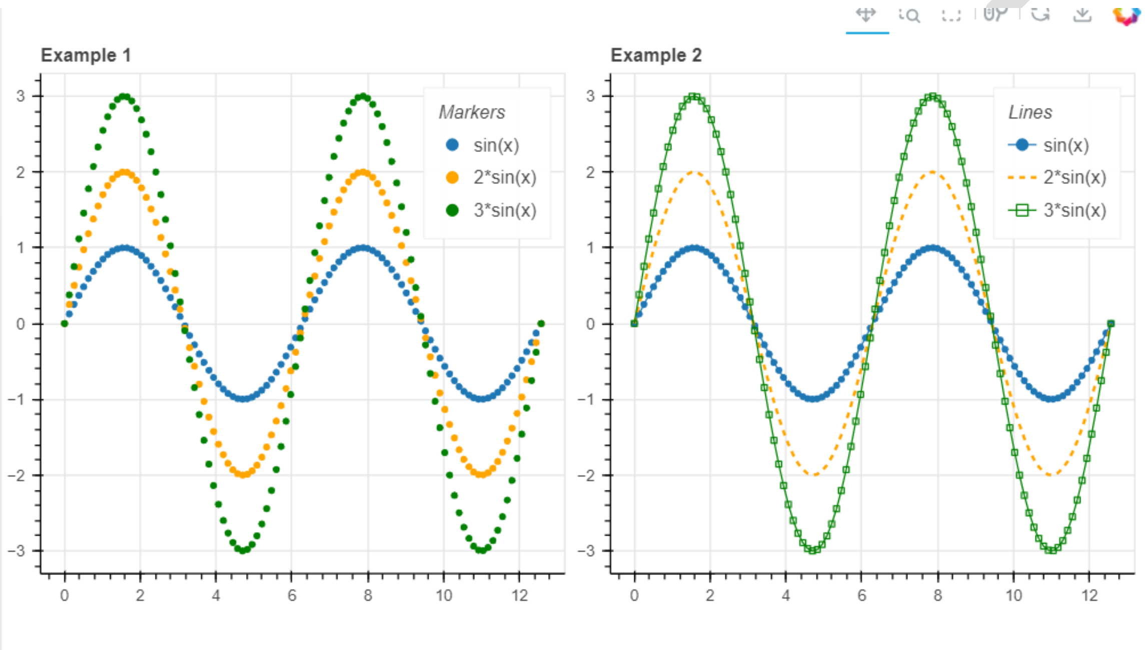


```
from bokeh.plotting import figure, output_file, show
  # instantiating the figure object
graph = figure(title="Bokeh Line Graph")
  # the points to be plotted
x = [1, 2, 3, 4, 5]
y = [5, 4, 3, 2, 1]
  # plotting the 1st line graph
graph.line(x, x, legend_label="Line 1")
  # plotting the 2nd line graph with a
# different color
graph.line(y, x, legend_label="Line 2",
          line_color="green")
  # displaying the model
show(graph)
```



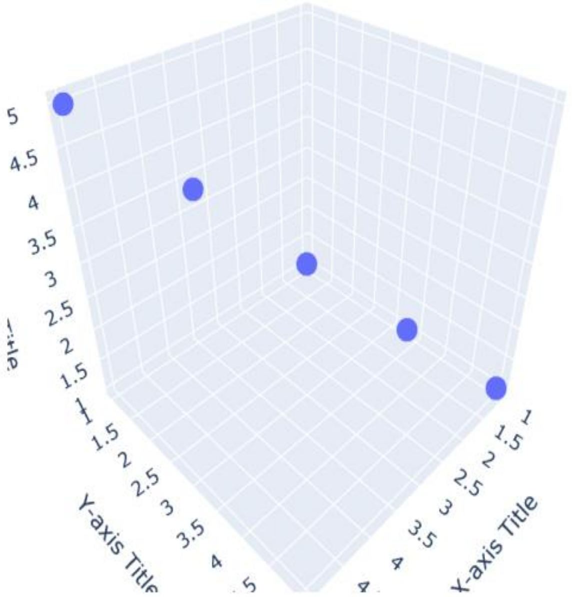
```
import numpy as np


from bokeh.layouts import gridplot
from bokeh.plotting import figure, show


x = np.linspace(0, 4*np.pi, 100)
y = np.sin(x)


TOOLS = "pan,wheel_zoom,box_zoom,reset,save,box_select"


p1 = figure(title="Example 1", tools=TOOLS)


p1.circle(x,   y, legend_label="sin(x)")
p1.circle(x, 2*y, legend_label="2*sin(x)", color="orange")
p1.circle(x, 3*y, legend_label="3*sin(x)", color="green")


p1.legend.title = 'Markers'


p2 = figure(title="Example 2", tools=TOOLS)


p2.circle(x, y, legend_label="sin(x)")
p2.line(x, y, legend_label="sin(x)")


p2.line(x, 2*y, legend_label="2*sin(x)",
        line_dash=(4, 4), line_color="orange", line_width=2)


p2.square(x, 3*y, legend_label="3*sin(x)", fill_color=None, line_color="green")
```

```
p2.line(x, 3*y, legend_label="3*sin(x)", line_color="green")


p2.legend.title = 'Lines'


show(gridplot([p1, p2], ncols=2, width=400, height=400))
```



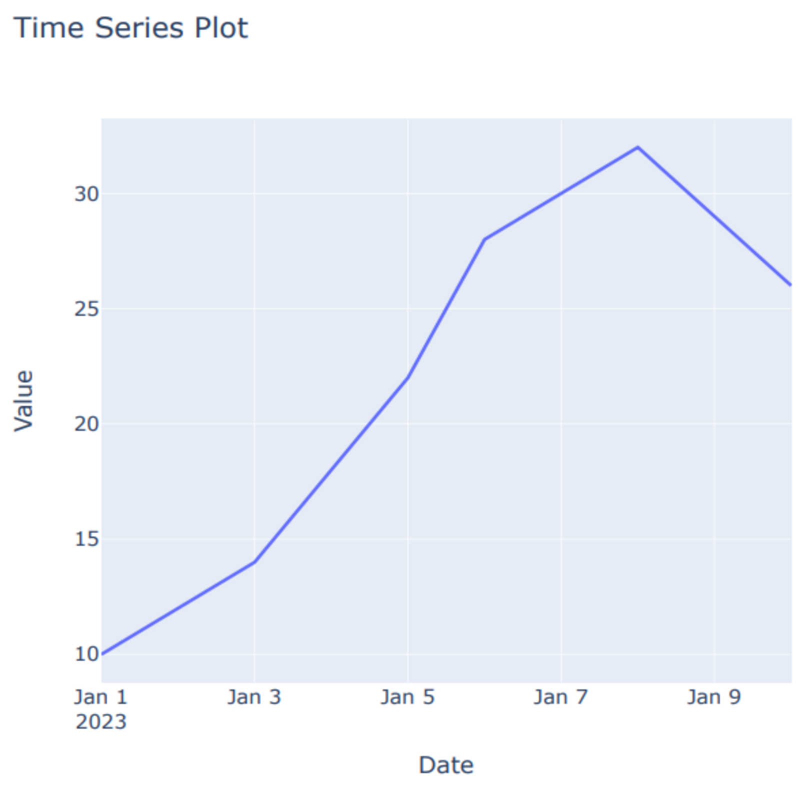
9 Write a Python program to draw 3D Plots using Plotly Libraries.

```
import plotly.express as px

# Sample data

import pandas as pd

data = pd.DataFrame({'X': [1, 2, 3, 4, 5],

'Y': [5, 4, 3, 2, 1],

'Z': [1, 2, 3, 4, 5]})

# Create a 3D scatter plot

fig = px.scatter_3d(data, x='X', y='Y', z='Z')

# Customize the plot

fig.update_layout(

scene=dict(

xaxis_title='X-axis Title',

yaxis_title='Y-axis Title',

zaxis_title='Z-axis Title'

)

)

# Show the plot

fig.show()
```



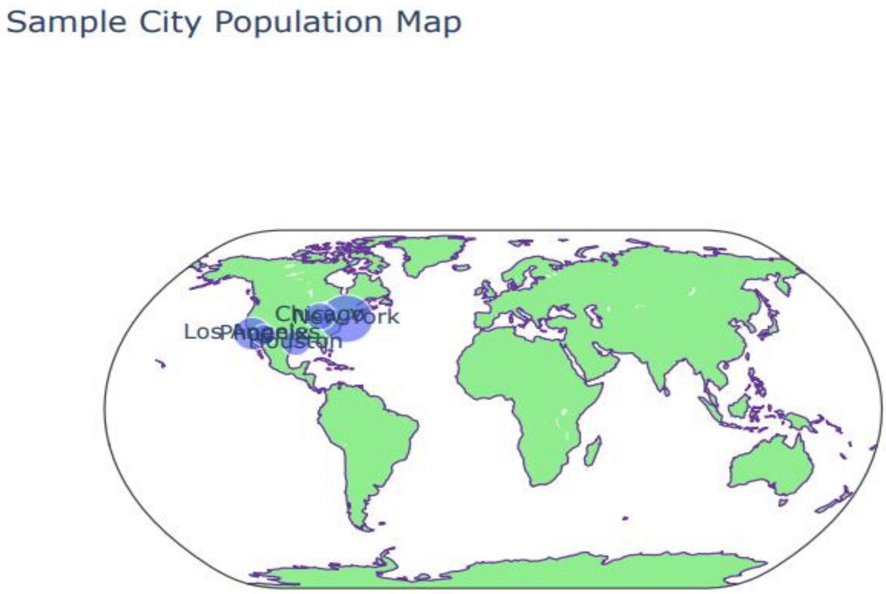
10 a) Write a Python program to draw Time Series using Plotly Libraries.

```
import plotly.express as px
import pandas as pd
# Sample time series data
data = pd.DataFrame({
'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'),
'Value': [10, 12, 14, 18, 22, 28, 30, 32, 29, 26]
})
# Create a time series plot
fig = px.line(data, x='Date', y='Value', title='Time Series Plot')
# Customize the plot
fig.update_xaxes(title_text='Date')
fig.update_yaxes(title_text='Value')
# Show the plot
fig.show()
```



b) Write a Python program for creating Maps using Plotly Libraries.

```
import plotly.express as px
# Sample data
import pandas as pd
data = pd.DataFrame({
'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix'],
'Lat': [40.7128, 34.0522, 41.8781, 29.7604, 33.4484],
'Lon': [-74.0060, -118.2437, -87.6298, -95.3698, -112.0740],
'Population': [8398748, 3990456, 2716000, 2320255, 1680992]
})
# Create a map
fig = px.scatter_geo(data, lat='Lat', lon='Lon', text='City', size='Population',
projection="natural earth", title='Sample City Population Map')
# Customize the map
fig.update_geos(
showcoastlines=True,
coastlinecolor="RebeccaPurple",
showland=True,
landcolor="LightGreen",
)
# Show the map
fig.show()
```


Sample City Population Map
Los Angeles
Phoenix
Chicago
New York
Houston