Question 1

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

Python Code:

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

Palindrome Check & Digit Occurrence Count

1b. 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.

Python Code :

```
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');
```

Output

Enter a value: 987654

Not Palindrome

4 appears 1 times

5 appears 1 times

6 appears 1 times

7 appears 1 times

8 appears 1 times

9 appears 1 times

Enter a value: 123321

Palindrome

1 appears 2 times

2 appears 2 times

3 appears 2 times

Question 2

Fibonacci Sequence

2a. 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.

Python Code :

```
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 :

```
Enter a number: 6
fn(6) = 5
Enter a number: -3
Input should be greater than 0
Enter a number: abc
Try with numeric value
```

Binary to Decimal & Octal to Hexadecimal Conversion

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

Python Code :

```
def bin2Dec(val):
return int(val, 2)
def oct2Hex(val):
dec=int(val, 8)
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
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 :

```
Enter a binary number: 101010
42
Enter an octal number: 755
1ED
Enter a binary number: 11011a
Invalid literal in input with base 2
Enter an octal number: 1298
Invalid literal in input with base 8
```

Question 3

Sentence Statistics

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

Python Code :

```
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 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 :

```
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
Enter a sentence: Python is Fun!
```

This sentence has 3 words

This sentence has 0 digits

3 uppercase letters

9 lowercase letters

Enter a sentence: Hello, World! 123

This sentence has 3 words

This sentence has 3 digits

1 uppercase letters

12 lowercase letters

String Similarity :

3b. Write a Python program to find the string similarity between two given strings.

Python Code :

```
from difflib import SequenceMatcher

str1 = input("Enter String 1 : ")

str2 = input("Enter String 2 : ")

sim = SequenceMatcher(None, str1, str2).ratio()

print("Similarity between strings \"" + str1 + "\" and \"" + str2 + "\" is : ",sim)
```

----------EOP----------

Here's a brief overview:

User Input:

The script prompts the user to enter two strings.

String Comparison:

The script then iterates through characters at corresponding positions and counts the matches.

Similarity Ratio:

The similarity between the two strings is calculated as the ratio of the count of matching characters to the length of the longer string.

Solution :

The script includes an alternative solution using the SequenceMatcher class from the difflib library, demonstrating a different approach to calculating string similarity.

This script offers a straightforward way to measure the similarity between two strings and presents an alternative solution using Python libraries for a comparative understanding. It's a useful tool for users interested in comparing the likeness of textual data.

Output :

```
Enter String 1 : Python Exercises

Enter String 2 : Python Exercise

Similarity between strings "Python Exercises" and "Python Exercise" is :
0.967741935483871

Enter String 1 : Python Exercises

Enter String 2 : Python Exercises

Similarity between strings "Python Exercises" and "Python Exercises" is : 1.0
```

Question 4 :

Bar Plot using Matplotlib :

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

Python Code :

```
import matplotlib.pyplot as plt
# Sample data for demonstration
categories = ['0-10', '10-20', '20-30', '30-40', '40-50']
values = [55, 48, 25, 68, 90]
# Create a bar plot
plt.bar(categories, values, color='skyblue')
# Add labels and title
plt.xlabel('Overs')
plt.ylabel('Runs')
plt.title('Bar Plot Showing Runs scored in an ODI Match')
# Display the plot
plt.show()
```

----------EOP----------

------**A brief Overview**

The provided Python script utilizes the matplotlib library to create a bar plot showcasing runs scored in an ODI (One Day International) cricket match. Here's a concise overview:

Sample Data:

The script uses sample data representing runs scored in specific overs during an ODI cricket match.

Matplotlib Plotting:

It utilizes the matplotlib.pyplot module to create a bar plot.

The bar function is used to plot the data, where categories represent overs, and values represent runs scored.

Labels and Title:

The script adds labels to the x-axis (Overs) and y-axis (Runs).

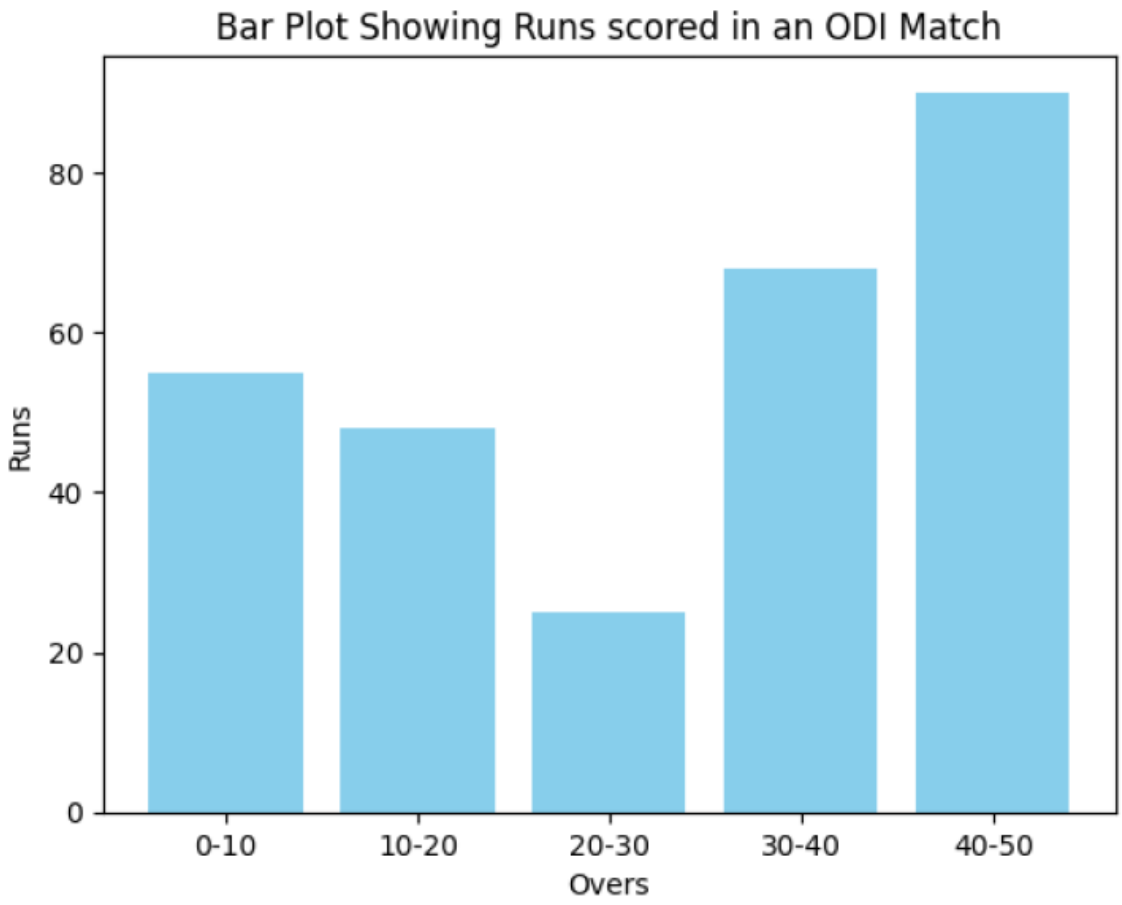
It includes a title, 'Bar Plot Showing Runs scored in an ODI Match,' to provide context to the plot.

Display:

The show function is called to display the generated bar plot.

This script is a straightforward example of using matplotlib to visualize data in a bar plot. It's a valuable resource for individuals interested in creating basic data visualizations, particularly in the context of cricket statistics.

Output:



Scatter Plot using Matplotlib

4b. Write a Python program to Demonstrate how to Draw a Scatter Plot using Matplotlib.

Python Code :

```
import matplotlib.pyplot as plt

import numpy as np

# BRICS nations data (hypothetical)

countries = ['Brazil', 'Russia', 'India', 'China', 'South Africa']

population = [213993437, 145912025, 1393409038, 1444216107, 61608912] # Population in 2021

per_capita_income = [9600, 11600, 2300, 11000, 6500] # Per capita income in USD

# Scale the population for circle size

circle_size = [pop / 1000000 for pop in population] # Scaling down for better visualization

# Assign different colors based on index

colors = np.arange(len(countries))

# Create a scatter plot with varying circle sizes and colors

scatter = plt.scatter(population, per_capita_income, s=circle_size, c=colors, cmap='viridis', alpha=0.7, label='BRICS Nations')

# Annotate each point with the country name

for i, country in enumerate(countries):

plt.annotate(country, (population[i], per_capita_income[i]), textcoords="offset points", xytext=(0,5), ha='center')

# Add colorbar

plt.colorbar(scatter, label='Index')

# Add labels and title

plt.xlabel('Population')

plt.ylabel('Per Capita Income (USD)')

plt.title('Population vs Per Capita Income of BRICS Nations')

# Display the plot

plt.show()
```

A Brief Overview:

The provided Python script employs the matplotlib library, along with numpy, to create a scatter plot visualizing population against per capita income for BRICS nations. Here's a concise overview:

Sample Data:

The script uses hypothetical data for BRICS nations, including population and per capita income.

Scaling for Visualization:

Circle sizes are scaled down from population values to enhance visualization.

Color Assignment:

Different colors are assigned based on the index of each country.

Scatter Plot:

The scatter function is used to create a scatter plot with varying circle sizes and colors.

Annotations:

Each point on the plot is annotated with the country name for clarity.

Colorbar:

A colorbar is added to the plot, providing a reference for the color index.

Labels and Title:

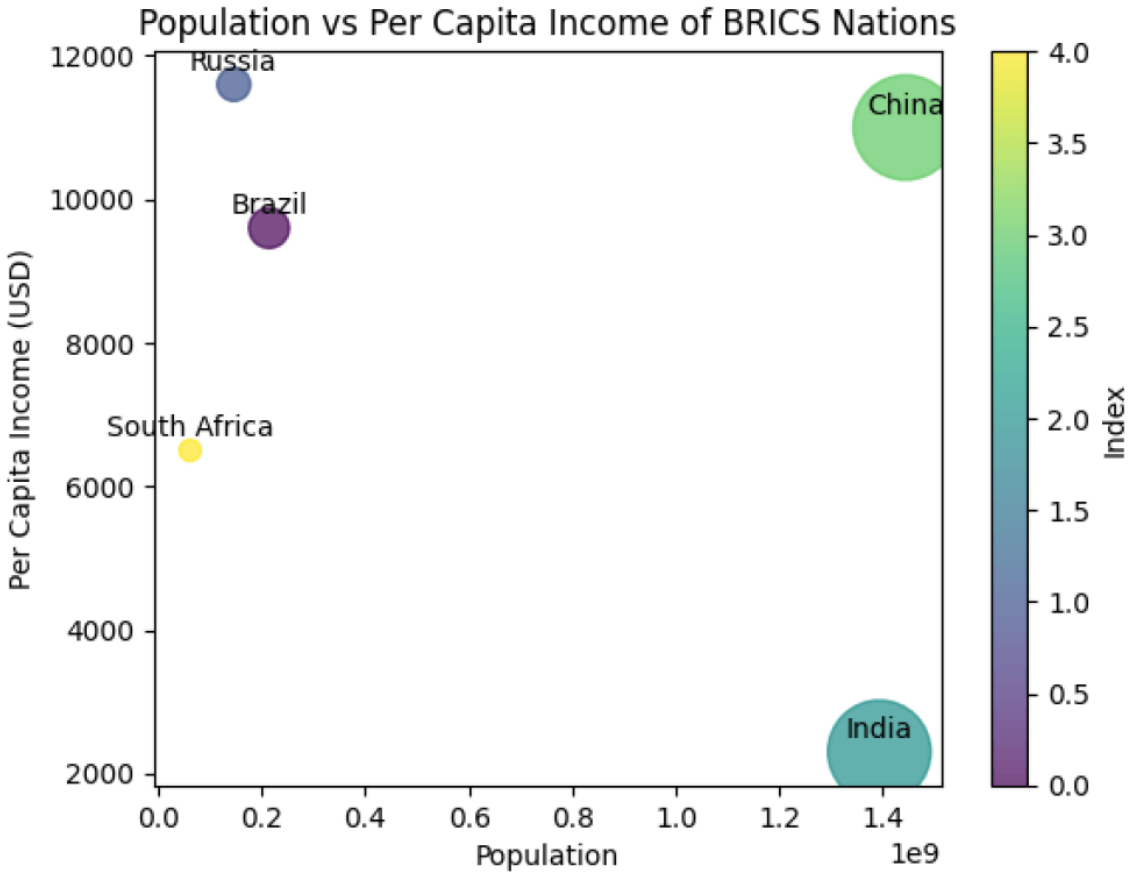
Labels for the x-axis, y-axis, and a title are included to provide context.

Display:

The show function is called to display the generated scatter plot.

This script serves as an excellent example of using matplotlib for creating informative and visually appealing scatter plots, especially for comparing socio-economic indicators among different countries. It can be helpful for readers interested in data visualization and analysis.

Output:


Population vs Per Capita Income of BRICS Nations
Russia
China
Brazil
South Africa
India
Per Capita Income (USD)
12000
10000
8000
6000
4000
2000
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e9
Population
Index
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Question 5

Histogram Plot using Matplotlib :

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

Python Code :

```
import matplotlib.pyplot as plt

import numpy as np

# Generate random student scores (example data)

np.random.seed(42)

student_scores = np.random.normal(loc=70, scale=15, size=100)

# Create a histogram plot

plt.hist(student_scores, bins=20, color='skyblue', edgecolor='black')

# Add labels and title

plt.xlabel('Student Scores')

plt.ylabel('Frequency')

plt.title('Distribution of Student Scores')

# Display the plot

plt.show()
```

----------EOP----------

----**A Brief Overview**

The provided Python script utilizes the matplotlib library and numpy to generate a histogram plot illustrating the distribution of student scores. Here's a concise overview:

Random Data Generation:

The script generates example data representing student scores using a normal distribution (np.random.normal).

Histogram Plot:

It creates a histogram plot using the hist function, where student_scores is the data array, bins determines the number of bins, and color sets the fill color while edgecolor sets the color of bin edges.

Labels and Title:

The script adds labels to the x-axis (Student Scores) and y-axis (Frequency).

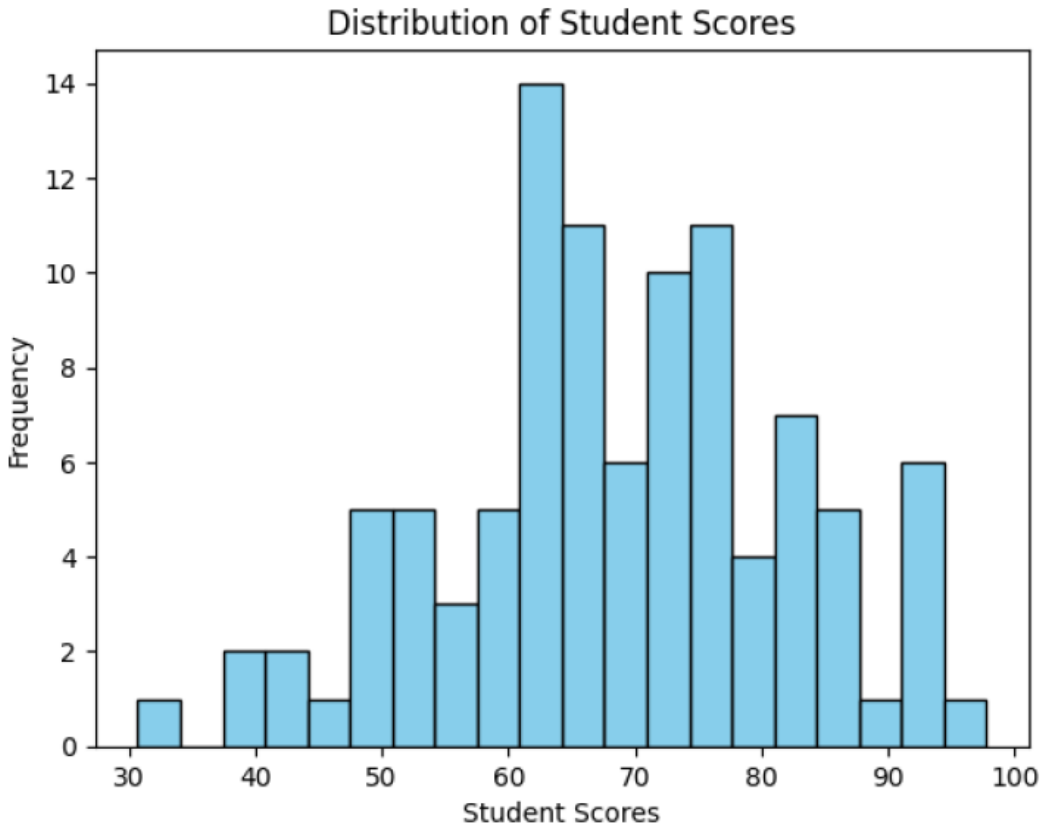
A title, ‘Distribution of Student Scores,’ is included to provide context to the plot.

Display:

The show function is called to display the generated histogram plot.

This script serves as a simple yet effective demonstration of using matplotlib to visualize the distribution of a dataset, making it suitable for readers interested in introductory data visualization techniques. It’s especially valuable for those learning to interpret histograms in the context of student scores or similar datasets.

Output:



Pie Chart using Matplotlib

5b. Write a Python program to Demonstrate how to Draw a Pie Chart using Matplotlib.

Python Code

```
import matplotlib.pyplot as plt

#Number of FIFA World Cup wins for different countries
countries = ['Brazil', 'Germany', 'Italy', 'Argentina', 'Uruguay', 'France', 'England', 'Spain']
wins = [5, 4, 4, 3, 2, 2, 1, 1] # Replace with actual data

# Colors for each country
colors = ['yellow', 'magenta', 'green', 'blue', 'lightblue', 'blue', 'red', 'cyan']

plt.pie(wins, labels=countries, autopct='%1.1f%%', colors=colors, startangle=90,
explode=[0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2], shadow=True)

# Add title
plt.title('FIFA World Cup Wins by Country')

# Display the plot
plt.axis('equal') # Equal aspect ratio ensures that the pie chart is circular.
plt.show()
```

----------EOP----------

The provided Python script utilizes the matplotlib library to create a visually appealing pie chart representing the number of FIFA World Cup wins for different countries. Here's a concise overview:

Data Representation:

The script uses hypothetical data representing the number of FIFA World Cup wins for different countries.

Pie Chart Creation:

It creates a pie chart using the pie function, where wins is the data array, labels are country names, and autopct formats the percentage display.

Colors and Styling:

Different colors are assigned to each country using the colors parameter.

The pie chart is enhanced with features such as a start angle, explode effect, and shadow.

Title:

The script adds a title to the pie chart, enhancing the overall context.

Display:

The show function is called to display the generated pie chart.

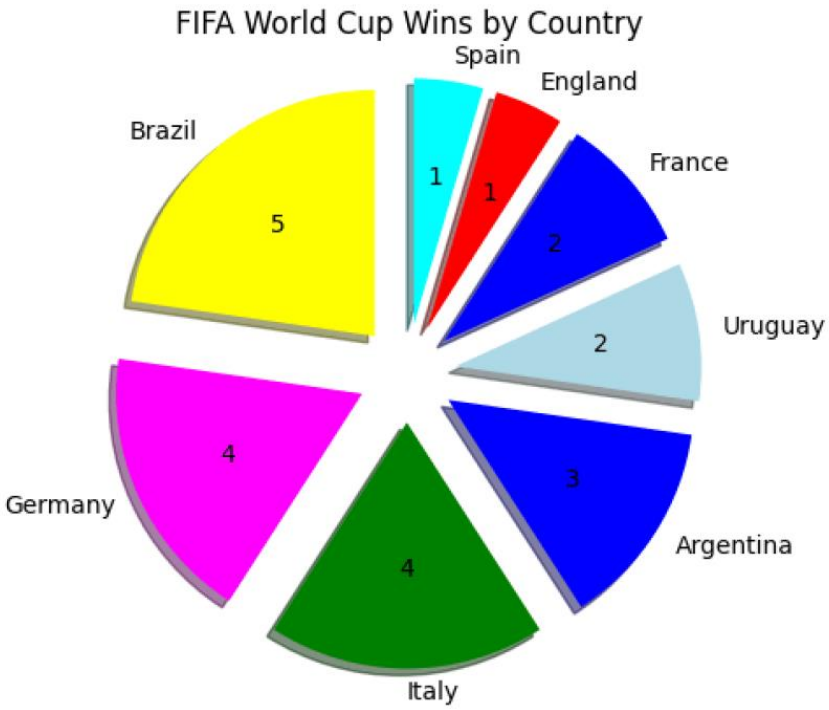
This script serves as a clear and concise example of using matplotlib to create visually engaging pie charts, making it suitable for readers interested in representing categorical data, such as FIFA World Cup wins, in a graphical format.

Output :

5b. Display values instead of percentages

Python Code:

```
import matplotlib.pyplot as plt

#Number of FIFA World Cup wins for different countries
countries = ['Brazil', 'Germany', 'Italy', 'Argentina', 'Uruguay', 'France', 'England', 'Spain']
wins = [5, 4, 4, 3, 2, 2, 1, 1] # Replace with actual data

# Colors for each country
colors = ['yellow', 'magenta', 'green', 'blue', 'lightblue', 'blue', 'red', 'cyan']

def make_autopct(values):
def my_autopct(pct):
total = sum(values)
val = int(round(pct*total/100.0))
return '{v:d}'.format(v=val)
return my_autopct

# Create a pie chart
plt.pie(wins, labels=countries, autopct=make_autopct(wins), colors=colors,
startangle=90, explode=[0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2], shadow=True)

# Add title
plt.title('FIFA World Cup Wins by Country')

# Display the plot
plt.axis('equal') # Equal aspect ratio ensures that the pie chart is circular.
plt.show()
```


FIFA World Cup Wins by Country
Spain
England
France
Uruguay
Argentina
Italy
Germany
Brazil
1
1
2
2
3
4
4
5

Question 6

Linear Plotting using Matplotlib

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

Python Code :

```
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()
```

----------EOP----------

The provided Python script utilizes the matplotlib library to create a linear plot representing the run rate in a hypothetical T20 cricket match. Here's a concise overview:

Hypothetical Data:

The script uses hypothetical data representing the number of runs scored in each over of a T20 cricket match.

Linear Plot:

It creates a linear plot using the plot function, where overs is on the x-axis and runs_scored is on the y-axis.

Labels and Title:

The script adds labels to the x-axis (Overs) and y-axis (Runs scored).

A title, 'Run Scoring in a T20 Cricket Match,' provides context to the plot.

Grid:

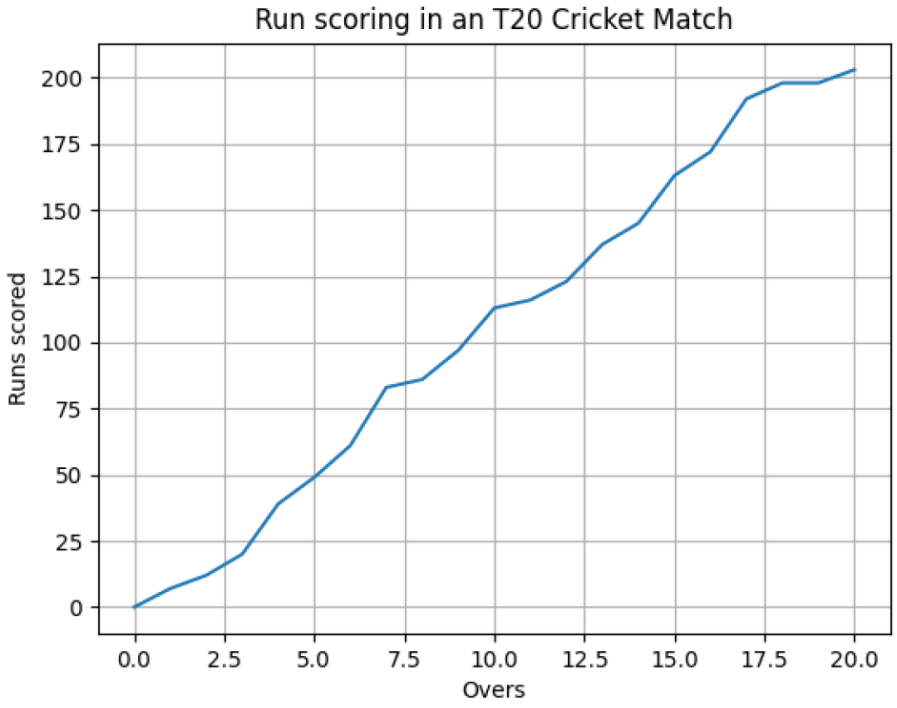
The plot includes a grid for better readability.

Display:

The show function is called to display the generated linear plot.

This script serves as a straightforward example of using matplotlib to visualize run scoring trends in a T20 cricket match, making it suitable for readers interested in representing time-dependent data in a graphical format.

Output:



6b. Write a Python program to illustrate liner plotting with line formatting using Matplotlib.

Python Code :

```
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()
```

The provided Python script, an extension of the previous T20 cricket match run rate plot, customizes the appearance of the plot with specific markers, line styles, colors, and label styles. Here's a concise overview:

Customized Plot Appearance:

The plot function is customized with parameters such as marker, linestyle, color, linewidth, markerfacecolor, and markersize to control the appearance of the plot.

Labels and Title Styling:

The script adds labels to the x-axis (Overs) and y-axis (Runs scored) with specific color styling.

The title, 'Run Scoring in a T20 Cricket Match,' maintains clarity.

Grid:

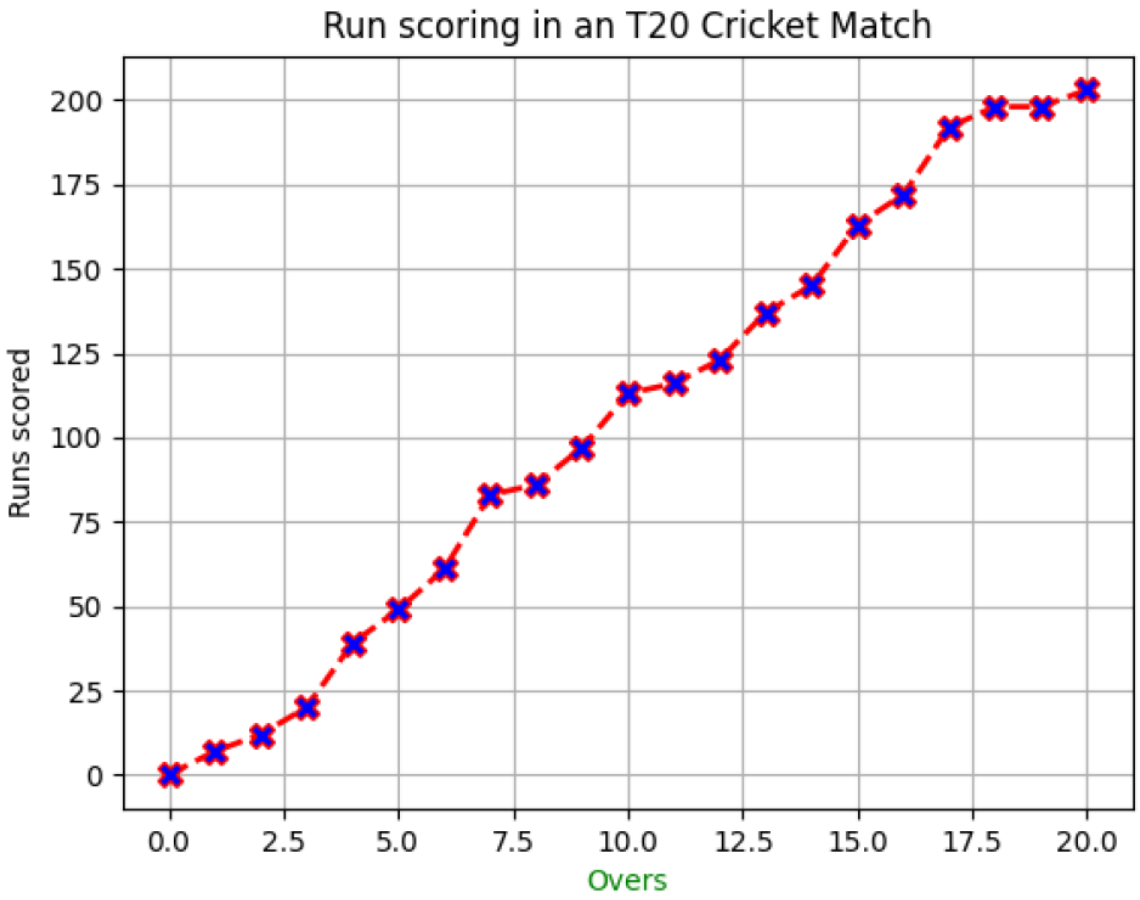
The plot includes a grid for better readability.

Display:

The show function is called to display the generated customized linear plot.

This script is an excellent example for readers looking to customize plot aesthetics in matplotlib for a more visually appealing representation of data. It's especially helpful for those interested in enhancing the clarity and style of their data visualizations.

Output:



Question 7

Seaborn plots with Aesthetic functions

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

Python Code :

```
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_theme()

#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()
```

The provided Python script utilizes the seaborn library, in conjunction with numpy and matplotlib, to create a series of sine wave plots with customized aesthetics. Here's a concise overview:

Data Generation:

The script uses numpy to generate a series of sine wave plots.

Seaborn Integration:

seaborn is imported and configured with a default theme (set_theme).

The context is set to "notebook" with customized font scaling and line width (set_context).

Customized Aesthetics:

The sinplot function generates multiple sine wave plots with varying frequencies and amplitudes.

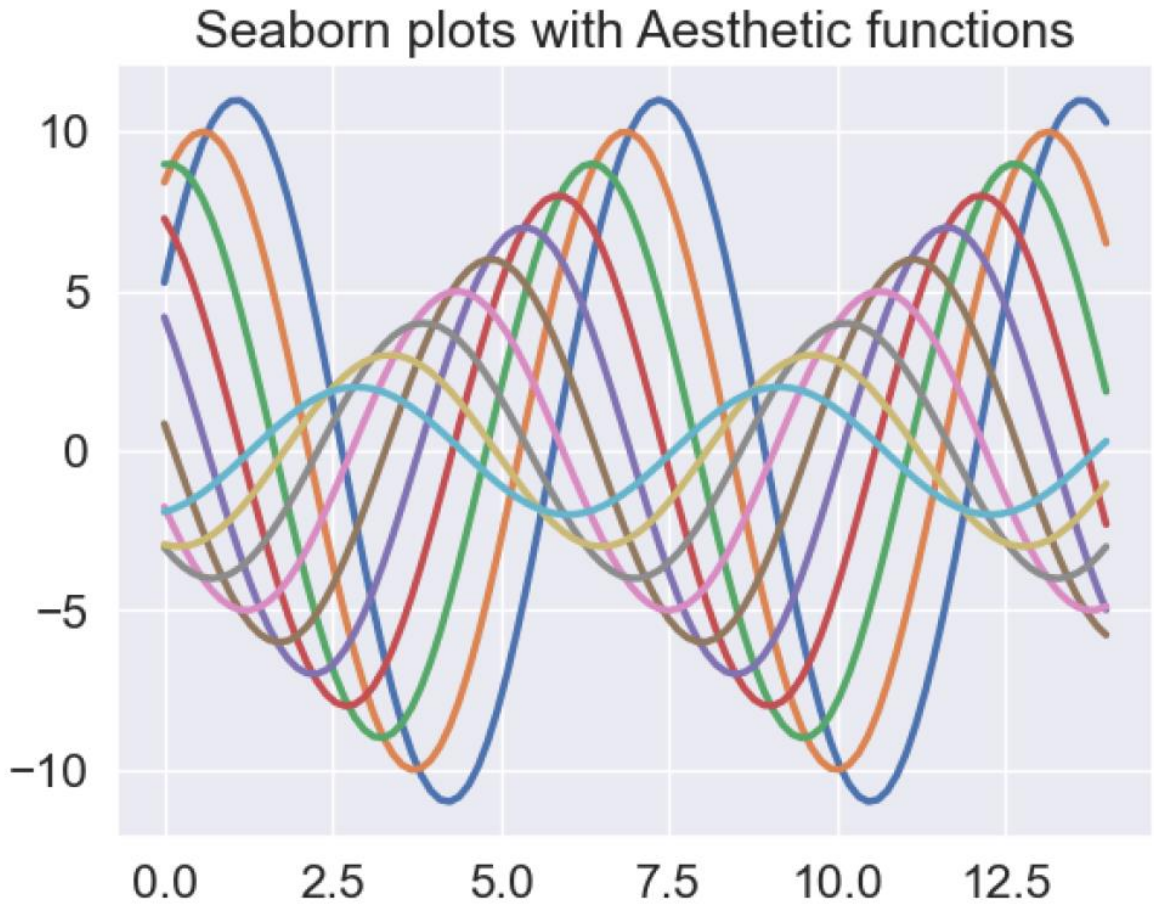
Title and Display:

The script adds a title to the plot, ‘Seaborn Plots with Aesthetic Functions.’

The show function is called to display the generated plots.

This script serves as an illustrative example of how seaborn can be used to enhance the aesthetics of data visualizations, providing readers with insights into customizing plot styles and themes for more visually appealing results. It’s particularly useful for those looking to leverage seaborn for improved aesthetics in their data analysis and visualization workflows.

Output:



Question 8

Bokeh line graph using Annotations and Legends

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.**

Python Code :

```
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))
```

The provided Python script demonstrates the use of the Bokeh library to create interactive data visualizations with multiple plots. Here's a concise overview:

Data Generation:

The script generates example data (x and y) using NumPy to represent sine waves.

Interactive Tools:

Bokeh's interactive tools (TOOLS) are enabled for features like pan, zoom, reset, and save.

Multiple Plots:

Two separate plots (p1 and p2) are created with different visualizations, including circles, lines, and markers.

Legend and Titles:

Legends are added to distinguish between different elements in the plots.

Titles are provided for each plot.

Grid Layout:

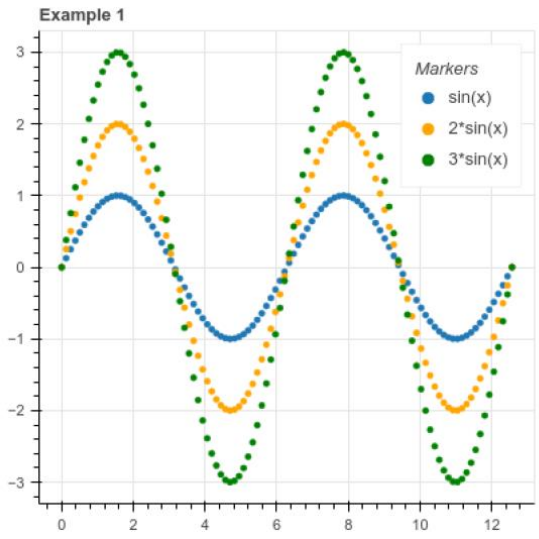
The gridplot function is used to arrange the plots in a grid layout.

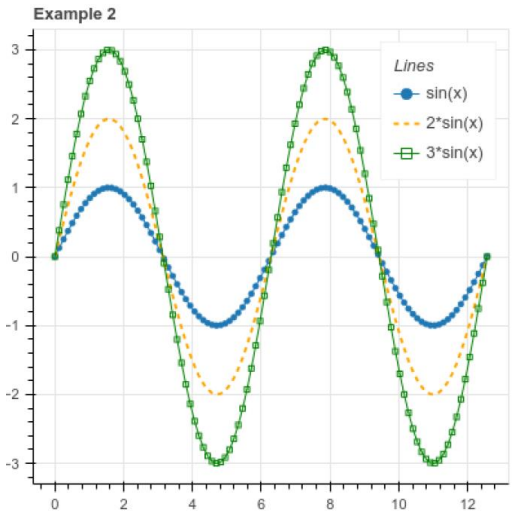
Interactive Display:

The show function is called to display the grid layout, enabling interactive exploration.

This script serves as an introduction to using Bokeh for creating interactive visualizations with multiple plots, making it suitable for readers interested in interactive data exploration and visualization techniques.

Output:





Question 9

3D Plots using Plotly Libraries

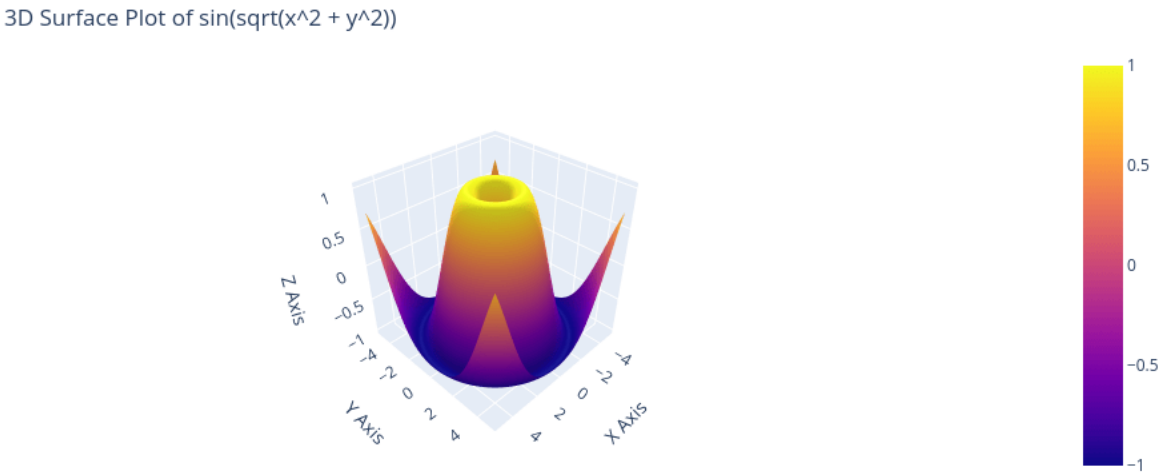
Write a Python program to draw 3D Plots using Plotly Libraries.

Python Code :

```
import plotly.graph_objects as go
import numpy as np
# Generate sample 3D data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))
# Create a 3D surface plot
fig = go.Figure(data=[go.Surface(z=z, x=x, y=y)])
# Customize layout
fig.update_layout(scene=dict(
xaxis_title='X Axis',
yaxis_title='Y Axis',
zaxis_title='Z Axis'),
margin=dict(l=0, r=0, b=0, t=40),
title='3D Surface Plot of sin(sqrt(x^2 + y^2))')
# Display the 3D surface plot
fig.show()
```

This program generates a 3D surface plot of the function z = sin(sqrt(x2+y2)). You can modify the function or provide your own data to create different types of 3D plots. The visualization will be interactive, allowing you to rotate and explore the plot.

Output:



Another Example :

```
import plotly.express as px

df = px.data.gapminder().query("continent=='Asia'")

fig = px.line_3d(df, x="gdpPercap", y="pop", z="year", color='country', title='Economic Evolution of Asian Countries Over Time')

fig.show()
```

In this Python program, we leverage the power of Plotly Express to visualize the economic evolution of Asian countries over time. The dataset used is Gapminder, a comprehensive collection of global development indicators. The focus is specifically on the Asian continent.

Import Libraries:

We start by importing the necessary libraries, including plotly.express for interactive visualizations.

Data Loading:

We load the Gapminder dataset and filter it to include only Asian countries.

3D Line Plot:

The key visualization is a 3D line plot created using px.line_3d.

The x-axis represents the GDP per capita (gdpPercap), the y-axis represents the population (pop), and the z-axis represents the year (year).

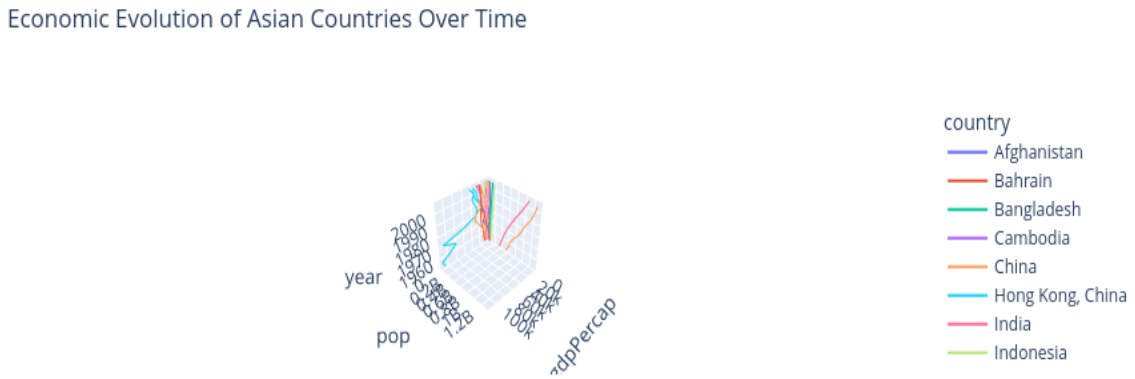
Each line corresponds to a different country, differentiated by color.

Interactive Exploration:

The resulting plot is interactive, allowing users to zoom, pan, and hover over data points to explore specific details.

Users can observe how GDP per capita and population have changed over the years for various Asian countries. The color-coded lines help distinguish between different nations.

Output:


Economic Evolution of Asian Countries Over Time
year
pop
gdpPercap
country
Afghanistan
Bahrain
Bangladesh
Cambodia
China
Hong Kong, China
India
Indonesia

Question 10

Time Series using Plotly Libraries

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

Python Code -Example 1

```
import pandas as pd
import plotly.express as px
dollar_conv = pd.read_csv('CUR_DLR_INR.csv')
fig = px.line(dollar_conv, x='DATE', y='RATE', title='Dollar vs Rupee')
fig.show()
```

CUR_DLR_INR.csv

The provided Python script showcases the use of the Plotly Express library to create an interactive line plot depicting the exchange rate between the US Dollar and the Indian Rupee over time. Here's a concise overview:

Data Import:

The script uses the Pandas library to read currency conversion data from a CSV file ('CUR_DLR_INR.csv'). You can download the csv file given above.

Plotly Express:

Plotly Express (px) is employed to create an interactive line plot with the exchange rate data.

Line Plot:

The line function from Plotly Express is used to generate a line plot.

The x-axis represents dates ('DATE'), and the y-axis represents exchange rates ('RATE').

Title:

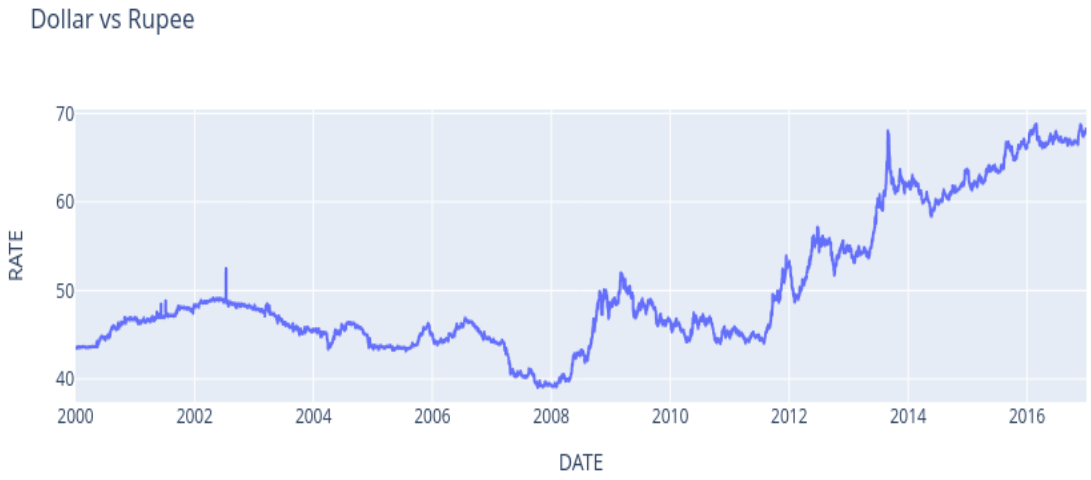
The plot is given a title, 'Dollar vs Rupee,' for context.

Interactive Display:

The show method is called on the figure (fig) to display the interactive plot.

This script provides a quick and effective demonstration of using Plotly Express to visualize time-series data, making it suitable for readers interested in creating interactive and visually appealing line plots for financial or currency-related datasets.

Output :



Python Code -Example 2

```
import pandas as pd

import plotly.express as px

runs_scored = pd.read_csv('AusVsInd.csv')

fig = px.line(runs_scored, x='Overs', y=['AUS', 'IND'], markers=True)

fig.update_layout(title='Australia vs India ODI Match', xaxis_title='OVERS',
yaxis_title='RUNS', legend_title='Country')

fig.show()
```

AusVsInd.csv File

The provided Python script utilizes the Plotly Express library to create an interactive line plot comparing the runs scored by Australia (AUS) and India (IND) over a series of overs in an ODI cricket match. Here's a concise overview:

Data Import:

The script uses Pandas to read runs scored data from a CSV file ('AusVsInd.csv').

Plotly Express:

Plotly Express (px) is employed to create an interactive line plot.

The x-axis represents overs ('Overs'), and the y-axis represents runs scored by Australia and India.

Markers:

Markers are added to the plot for each data point to enhance visibility.

Customized Layout:

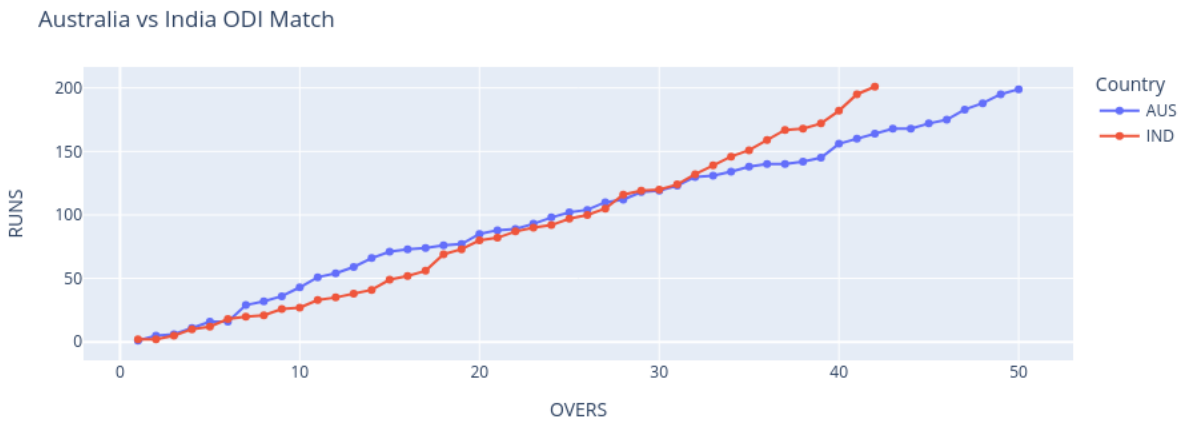
The layout is customized with a title ('Australia vs India ODI Match'), x-axis label ('OVERS'), y-axis label ('RUNS'), and legend title ('Country').

Interactive Display:

The show method is called on the figure (fig) to display the interactive plot.

This script serves as an excellent example for readers interested in using Plotly Express for comparing and visualizing data, particularly in the context of sports analytics or cricket match statistics. The interactive features make it easy for users to explore the data interactively.

Output:


Australia vs India ODI Match
RUNS
OVERS
0
50
100
150
200
0
10
20
30
40
50
Country
AUS
IND

Example 3:

Bar Graph for runs scored every over

#**Bar Graph of Runs scored every Over**

```
import pandas as pd
import plotly.express as px
runs_scored = pd.read_csv('AusVsInd.csv')
fig = px.bar(runs_scored, x='Overs', y=['AUS_RPO', 'IND_RPO'], barmode='group')
fig.update_layout(title='Australia vs India ODI Match', xaxis_title='OVERS', yaxis_title='RUNS', legend_title='Country')
fig.show()
```

The provided Python script uses the Plotly Express library to create an interactive grouped bar graph comparing the runs per over (RPO) scored by Australia (AUS) and India (IND) in an ODI cricket match. Here's a concise overview:

Data Import:

The script uses Pandas to read runs scored data from a CSV file ('AusVsInd.csv').

Plotly Express:

Plotly Express (px) is employed to create an interactive grouped bar graph.

The x-axis represents overs ('Overs'), and the y-axis represents runs per over scored by Australia and India.

Grouped Bars:

Bars are grouped for each over, and the bar mode is set to 'group' to display bars side by side.

Customized Layout:

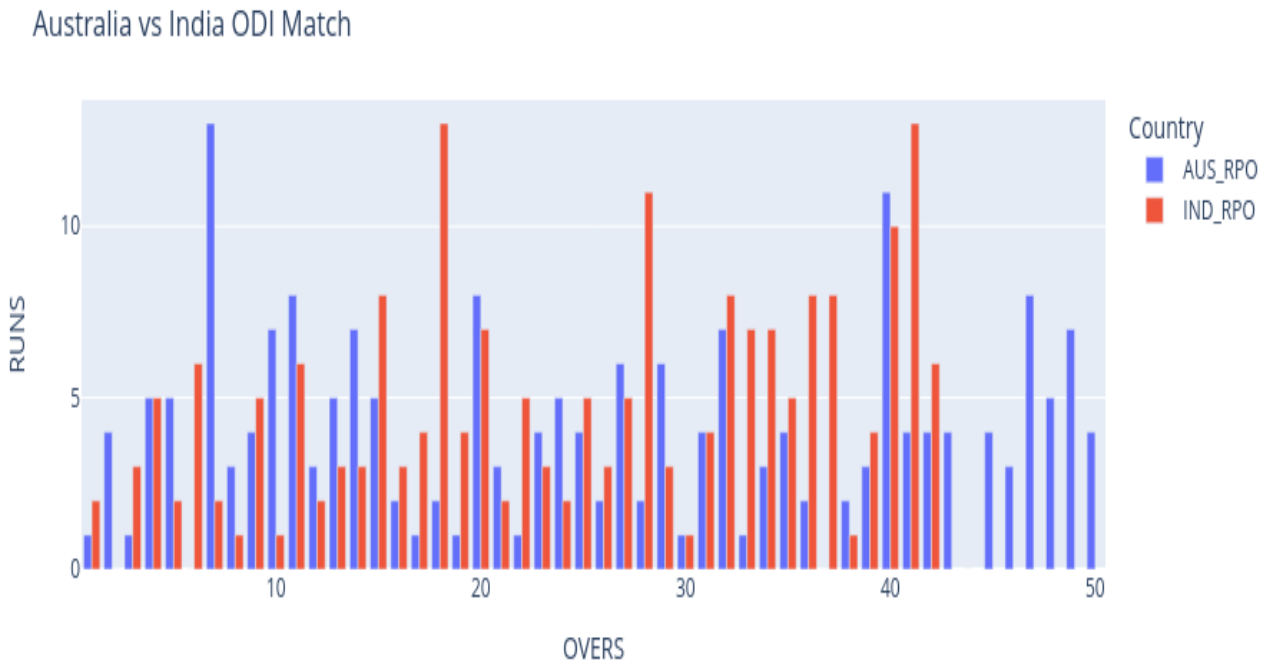
The layout is customized with a title (‘Australia vs India ODI Match’), x-axis label (‘OVERS’), y-axis label (‘RUNS’), and legend title (‘Country’).

Interactive Display:

The show method is called on the figure (fig) to display the interactive grouped bar graph.

This script serves as an illustrative example for readers interested in using Plotly Express to visualize and compare runs scored per over by different teams in a cricket match. The interactive nature of the graph allows users to explore the data interactively.

Output:



Maps using Plotly Libraries

10b. Write a Python program for creating Maps using Plotly Libraries.

Python Code -Example 1

```
import plotly.express as px

import pandas as pd

# Import data from GitHub

data=pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/gapminder_with_codes.csv')

# Create basic choropleth map

fig = px.choropleth(data, locations='iso_alpha', color='gdpPercap', hover_name='country',

projection='natural earth', title='GDP per Capita by Country')

fig.show()
```

In this Python program, we utilize Plotly Express to create an interactive choropleth map visualizing GDP per Capita by country. The dataset used is sourced from Gapminder, providing a comprehensive view of economic indicators globally.

Import Libraries:

We start by importing the necessary libraries, including plotly.express for easy and interactive visualizations.

Data Loading:

The program fetches data from a CSV file hosted on GitHub using pd.read_csv. The dataset includes information about countries, their ISO codes, and GDP per Capita.

Choropleth Map:

The choropleth map is created using px.choropleth.

Key parameters include:

locations: ISO codes of countries.

color: GDP per Capita, determining the color intensity on the map.

hover_name: Country names appearing on hover.

projection: 'natural earth' projection for a global view.

title: The title of the map.

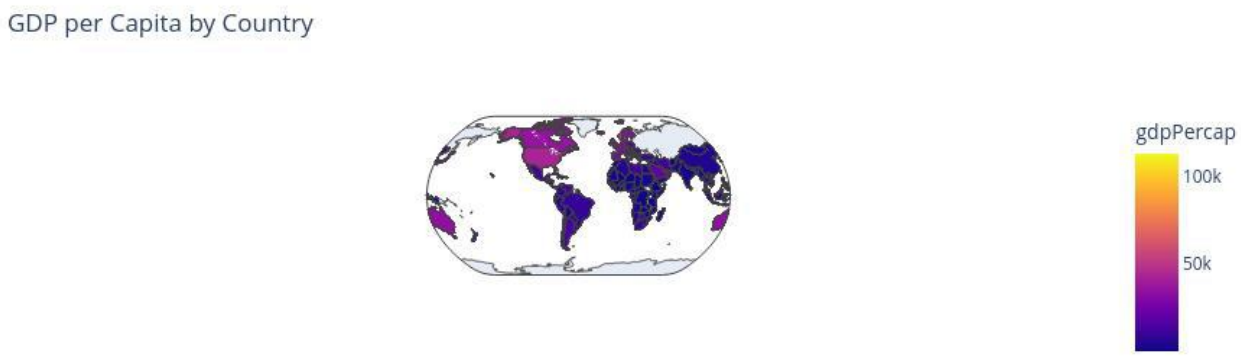
Interactive Exploration:

The resulting choropleth map is interactive, enabling users to hover over countries to see GDP per Capita values.

Users can explore and compare GDP per Capita across different countries. Darker colors indicate higher GDP per Capita. This program demonstrates the simplicity and power of Plotly Express for creating data-driven visualizations. The choropleth map offers an intuitive way to understand global economic

disparities. Feel free to customize the description based on additional details you'd like to highlight or any specific insights you've gained from the visualization.

Output :



Python Code -Example 2

```
import json
import numpy as np
import pandas as pd
import plotly.express as px
#Uncomment below lines to render map on your browser
#import plotly.io as pio
#pio.renderers.default = 'browser'
india_states = json.load(open("states_india.geojson", "r"))
df = pd.read_csv("india_census.csv")
state_id_map = {}
for feature in india_states["features"]:
feature["id"] = feature["properties"]["state_code"]
state_id_map[feature["properties"]["st_nm"]] = feature["id"]
df = pd.read_csv("india_census.csv")
df["Density"] = df["Density[a]"].apply(lambda x: int(x.split("/")[0].replace(",", "")))
df["id"] = df["State or union territory"].apply(lambda x: state_id_map[x])
#print(df.head())
fig = px.choropleth(
df,
locations="id",
geojson=india_states,
color="Population",
hover_name="State or union territory",
hover_data=["Density", "Sex ratio", "Population"],
title="India Population Statewise",
)
```

```
fig.update_geos(fitbounds="locations", visible=False)

fig.show()
```

Program Overview:

In this Python program, we leverage Plotly Express to create an insightful choropleth map that visualizes key demographic indicators across states and union territories in India. The data is sourced from India's census, providing a comprehensive overview of population distribution, density, and sex ratio.

Import Libraries:

We begin by importing necessary libraries, including json for handling GeoJSON data, numpy for numerical operations, pandas for data manipulation, and plotly.express for creating interactive visualizations.

Load GeoJSON and Census Data:

The GeoJSON file representing Indian states is loaded, and census data is read from a CSV file containing information about population, density, and sex ratio.

Data Preparation:

We create a mapping between state names and their corresponding IDs for seamless integration with GeoJSON features.

Additional data preprocessing includes converting density values to integers and creating a unique identifier (id) for each state.

Choropleth Map:

The choropleth map is generated using px.choropleth. Key parameters include:

locations: State IDs for mapping.

geojson: GeoJSON data for Indian states.

color: Population, determining color intensity on the map.

hover_name: State names for hover information.

hover_data: Additional information displayed on hover, including density, sex ratio, and population.

title: Title of the map.

Interactive Exploration:

The resulting choropleth map is interactive, allowing users to hover over states to explore population demographics.

Users can explore and compare population distribution, density, and sex ratio across different states and union territories in India. This program demonstrates the power of

Plotly Express for creating meaningful and interactive visualizations. The choropleth map provides valuable insights into the demographic landscape of India.

Output: