# Data Visualization in Python

**Introduction:**

This assignment incorporates key topics such as line plots, scatter plots, and bar charts, each tailored to specific scenarios for analyzing and presenting data effectively. By mastering these techniques, you can uncover patterns, relationships, and insights that are otherwise hidden in raw data.

- **Line Plots:** Used for visualizing changes over time or continuous data trends.
- **Scatter Plots:** Ideal for analyzing relationships and correlations between two numerical variables.
- **Bar Charts:** Best suited for comparing categorical data, such as monthly performance or grouped metrics.

**Exercise 1:** (Score : 4)

Create a line plot using matplotlib pyplot that displays the population of four different cities over time. Each city should have its own line, and the x-axis should represent years (e.g. 2010, 2011, 2012, etc.) while the y-axis should represent the population.

The data for the four cities is provided below:

City A: [500000, 550000, 600000, 650000, 700000, 750000, 800000]

City B: [800000, 850000, 900000, 950000, 1000000, 1050000, 1100000]

City C: [1000000, 1050000, 1100000, 1150000, 1200000, 1250000, 1300000]

City D: [1200000, 1250000, 1300000, 1350000, 1400000, 1450000, 1500000].

a. Description: This exercise involves creating a line plot to display the population growth of four cities over several years. Each city's growth trend is represented with a unique line, facilitating a comparative temporal analysis.
b. Advantage: Line plots allow you to monitor trends, detect patterns, and observe fluctuations in continuous data over time. They are particularly effective for spotting long-term changes or irregularities.
c. Code:

```
#importing library for plotting the graph.
import matplotlib.pyplot as plt
#this is used for formatting to non-exponential.
from matplotlib import ticker

CityA = [500000, 550000, 600000, 650000, 700000, 750000, 800000]
CityB = [800000, 850000, 900000, 950000, 1000000, 1050000, 1100000]
CityC = [1000000, 1050000, 1100000, 1150000, 1200000, 1250000, 1300000]
CityD = [1200000, 1250000, 1300000, 1350000, 1400000, 1450000, 1500000]
years= [2010,2011,2012,2013,2014,2015,2016]

#Plotting the graph for different cities.
```

```
plt.plot(years,CityA)
plt.plot(years,CityB)
plt.plot(years,CityC)
plt.plot(years,CityD)

#labelling the x and y axis as per the question
plt.title("Population based on Cities")
plt.xlabel("Years")
plt.ylabel("Population")

#formatting the exponential number for clear visualization in y axis
formatter = ticker.ScalarFormatter()
formatter.set_scientific(False)
plt.gca().yaxis.set_major_formatter(formatter)
plt.show()
```

**Note:** Fig 1 population(y-axis) is in exponential format and Fig 2 is in number format.

Fig.1

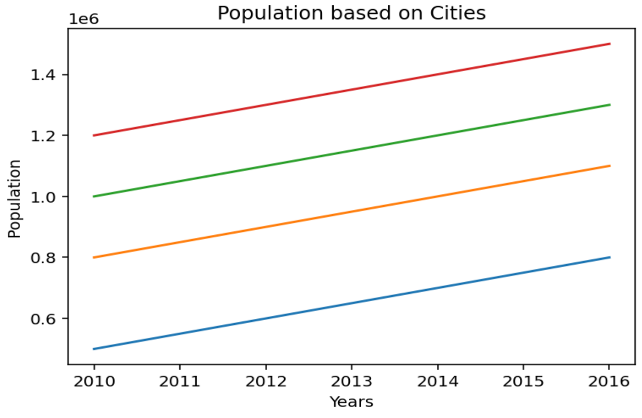
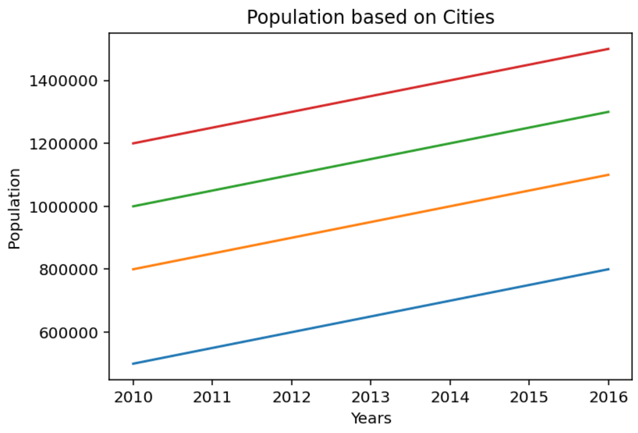
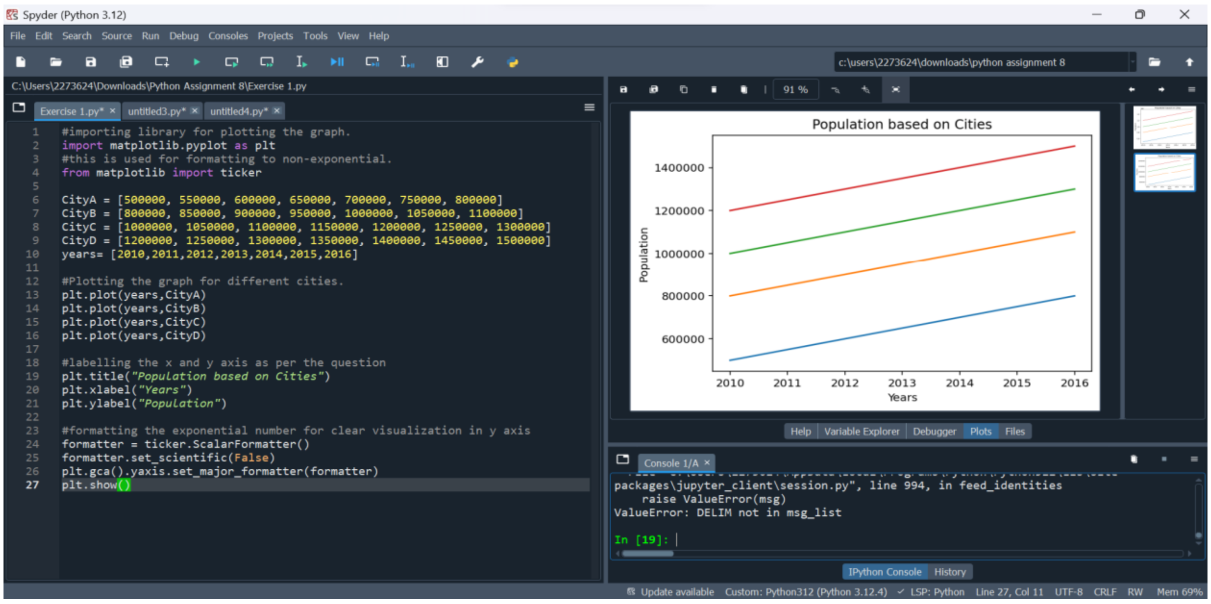


Fig.2





```
#importing library for plotting the graph.
import matplotlib.pyplot as plt
#this is used for formatting to non-exponential.
from matplotlib import ticker

CityA = [500000, 550000, 600000, 650000, 700000, 750000, 800000]
CityB = [800000, 850000, 900000, 950000, 1000000, 1050000, 1100000]
CityC = [1000000, 1050000, 1100000, 1150000, 1200000, 1250000, 1300000]
CityD = [1200000, 1250000, 1300000, 1350000, 1400000, 1450000, 1500000]
years= [2010,2011,2012,2013,2014,2015,2016]

#Plotting the graph for different cities.
plt.plot(years,CityA)
plt.plot(years,CityB)
plt.plot(years,CityC)
plt.plot(years,CityD)

#labelling the x and y axis as per the question
plt.title("Population based on Cities")
plt.xlabel("Years")
plt.ylabel("Population")

#formatting the exponential number for clear visualization in y axis
formatter = ticker.ScalarFormatter()
formatter.set_scientific(False)
plt.gca().yaxis.set_major_formatter(formatter)
plt.show()
```

```
packages\jupyter_client\session.py", line 994, in feed_identities
    raise ValueError(msg)
ValueError: DELIM not in msg_list

In [19]: 
```

**Exercise 2:** (Score: 3)

Create a scatter plot using seaborn that shows the relationship between the number of hours studied and the test scores obtained by a group of students. Use the following data:

Hours Studied: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Test Scores: [93, 57, 61, 54, 51, 53, 87, 81, 83, 85].

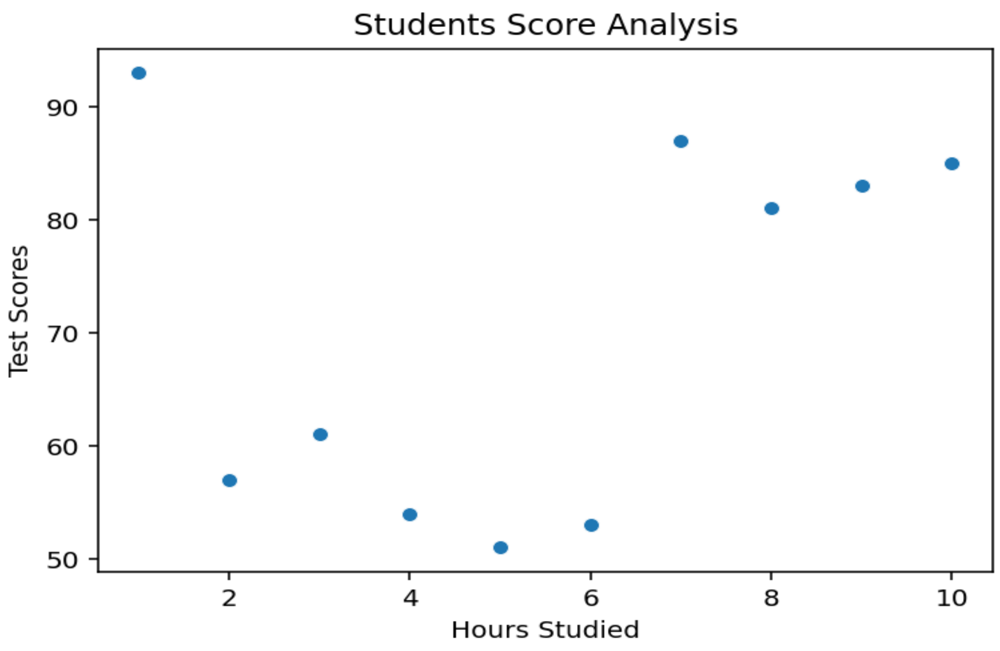
a. Description: In this exercise, a scatter plot is created to analyze the relationship between the number of hours studied and test scores achieved by students. Each point in the plot correlates the two variables, making it easy to visualize their relationship.
b. Advantage: Scatter plots reveal relationships, trends, or clusters between two variables. They are highly useful for identifying positive or negative correlations and highlighting outliers.
c. Code:

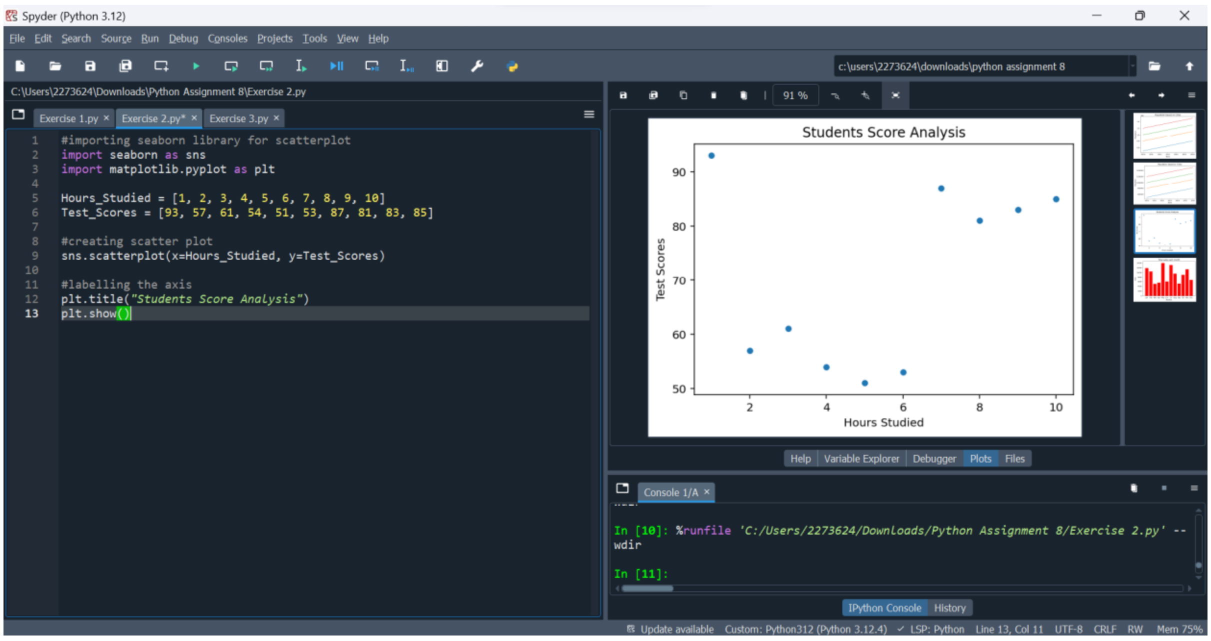
```
#importing seaborn library for scatterplot
import seaborn as sns
import matplotlib.pyplot as plt

Hours_Studied = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Test_Scores = [93, 57, 61, 54, 51, 53, 87, 81, 83, 85]

#creating scatter plot
sns.scatterplot(x=Hours_Studied, y=Test_Scores)

#labelling the axis
plt.title("Students Score Analysis")
plt.show()
```





```
#importing seaborn library for scatterplot
import seaborn as sns
import matplotlib.pyplot as plt

Hours_Studied = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Test_Scores = [93, 57, 61, 54, 51, 53, 87, 81, 83, 85]

#creating scatter plot
sns.scatterplot(x=Hours_Studied, y=Test_Scores)

#labelling the axis
plt.title("Students Score Analysis")
plt.show()
```

```
In [10]: %runfile 'C:/Users/2273624/Downloads/Python Assignment 8/Exercise 2.py' --
wdir

In [11]:
```

**Exercise 3:** (Score : 3)

Create a bar chart using matplotlib pyplot that shows the total sales for each month of the year. Use the following data:

Month: ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]

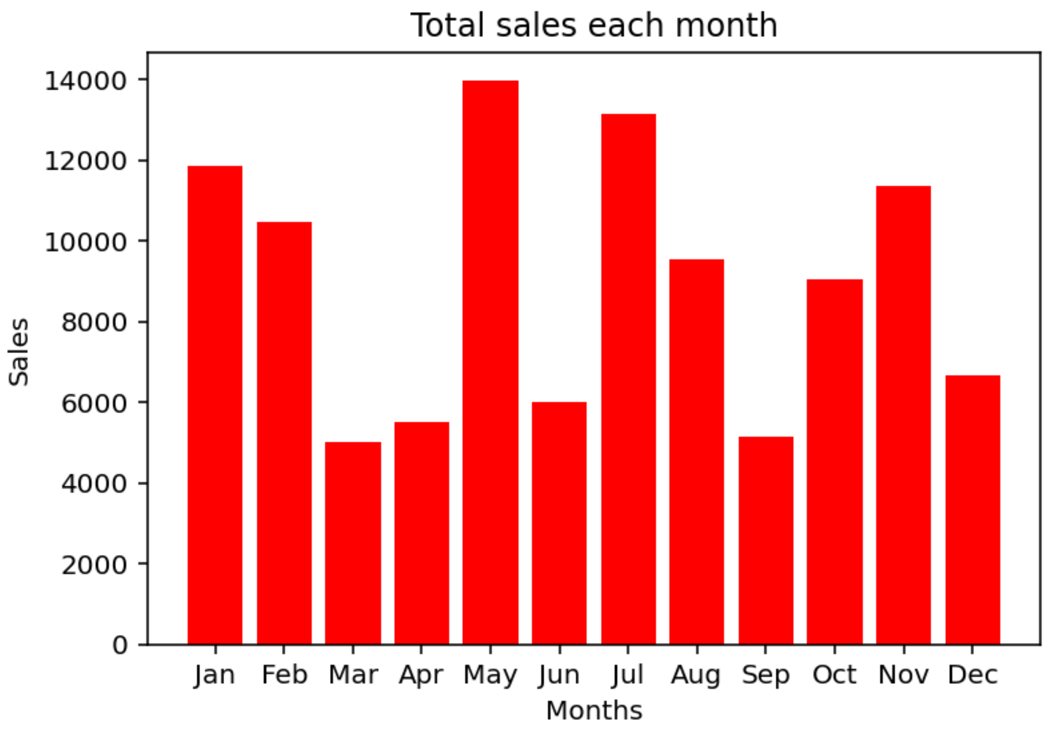
Sales: [11860, 10480, 4997, 5523, 13965, 6011, 13158, 9533, 5158, 9058, 11346, 6675]

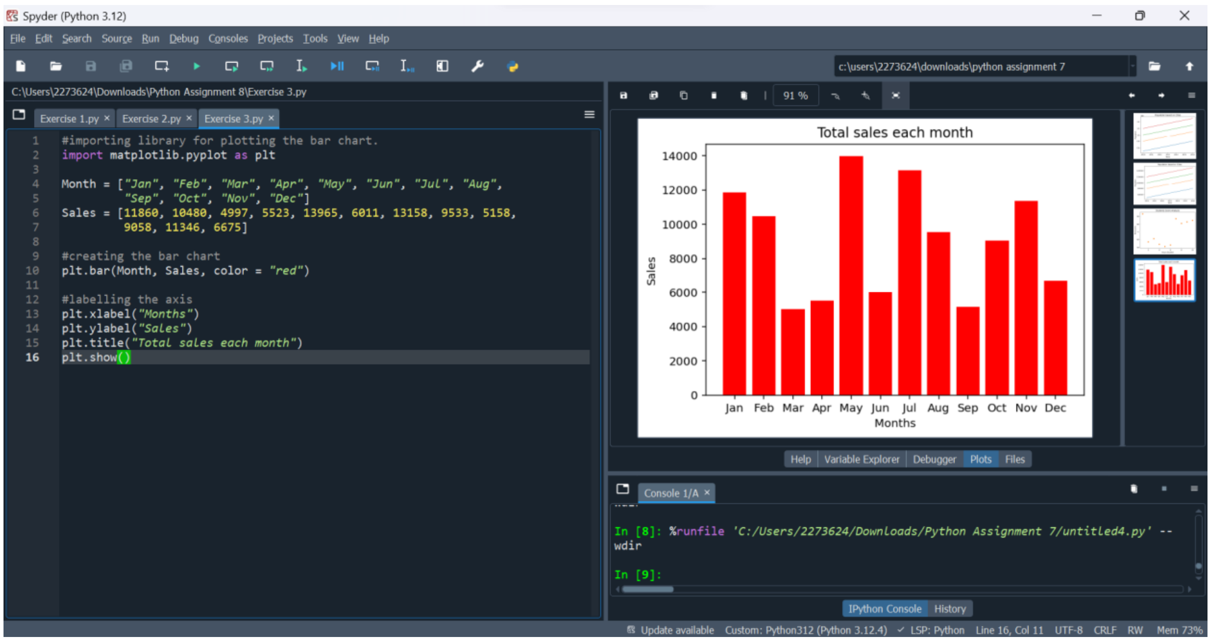
a. Description: This exercise involves creating a bar chart to illustrate total sales figures for each month of the year. Monthly sales are depicted as bars, providing a clear visual comparison of sales performance across the year.
b. Advantage: Bar charts are excellent for comparing categorical data, highlighting variations, and identifying trends or high-performing periods. They offer a clear and simple way to communicate business metrics effectively.
c. Code:

```
#importing library for plotting the bar chart.
import matplotlib.pyplot as plt

Month = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
    "Sep", "Oct", "Nov", "Dec"]
Sales = [11860, 10480, 4997, 5523, 13965, 6011, 13158, 9533, 5158,
    9058, 11346, 6675]

#creating the bar chart
plt.bar(Month, Sales, color = "red")
#labelling the axis
plt.xlabel("Months")
plt.ylabel("Sales")
plt.title("Total sales each month")
plt.show()
```





```python
#importing library for plotting the bar chart.
import matplotlib.pyplot as plt

Month = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
         "Sep", "Oct", "Nov", "Dec"]
Sales = [11860, 10480, 4997, 5523, 13965, 6011, 13158, 9533, 5158,
         9058, 11346, 6675]

#creating the bar chart
plt.bar(Month, Sales, color = "red")

#labelling the axis
plt.xlabel("Months")
plt.ylabel("Sales")
plt.title("Total sales each month")
plt.show()
```

```
In [8]: %runfile 'C:/Users/2273624/Downloads/Python Assignment 7/untitled4.py' --
wdir

In [9]:
```