Practical No. 7

Aim: Data Visualization in python using Mathplotlib & Numpy Libraries. (scatter plot, Bare graph, Histogram, Line plot & Pie chart).

Aim: Data Visualization in python using Mathplotlib & Numpy Libraries. (Scatter plot, Bar graph, Histo-gram, Line plot & Pie Chart).

Theory: Data visualization is an essential part of data analysis. It helps in interpreting complex data sets by presenting them in a visual context such as graphs or Charts, which makes the data easier to understand.

Libraries Overview:

Mathplotlib: It is a comprehensive library for creating static, animated & interactive visualizations in python. It provides an Object-oriented API for embedding plots into applications using general purpose GUI Toolkits.

Numpy: is a fundamental package for scientific computing in python. It provides support for large, multi-dimenssional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

Syntax:

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

A) Scatter plot Code :

```
#Generating data
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 5, 7, 8])
#Plotting the scatter Plot
plt.scatter(x, y, color = "blue")
plt.title('Scatter plot : Study Hrs Vs Exam Scores')
plt.xlabel('Hours of study')
plt.ylabel ('Exam Score')
plt.show ()
```

B) Bar Graph :

```
#Data
categories = ['Product A', 'Product B', 'Product c']
values = [20, 35, 30]
#Plotting the Bar graph
plt.bar(categories, values, color = ['Red', 'Blue', 'green'])
plt.title ('Bar Graph : Sales Performance')
plt.xlabel ('Product')
plt.ylabel ('Sales')
plt.show ()
```

C) Histogram :

```
#Generating random data :
data = np.random.normal(30, 5, 1000)
```

```
#Plotting the histogram
plt.hist(data, bins = 30, color = 'purple', alpha = 0.7)
plt.title('Histogram: Age Distribution of employees)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

D) line Plot :

```
# Data :
time = np.array([1, 2, 3, 4, 5])
prices = np.array([100, 102, 98, 105, 110])
#Plotting the Line Plot
plt.plot(time, prices, marker ='O', color ='green')
plt.title('Line plot : stock prices over time')
plt.xlabel('Time (Days'))
plt.ylabel('Price')
plt.show()
```

E) Pie Chart :

```
#Data :
companies = ['Company A', 'Company B', 'Company C']
shares = [40, 30, 30]
#Plotting the pie chart
plt.pie(shares, labels = Companies, autopct =
    '%1.1f%%', colors = ['gold', 'silver', 'bronze'])
```

Conclusion: Data Visualization helps simplify complex datasets enabling quick insights. Using python's Mathplotlib & Numpy, an individual can create various plots, each suited for specific types of data analysis.

```
plt.title('Pie chart : Market Share')
plt.show()
```

Conclusion: Data visualization helps simplify complex datasets, enabling quick insights. Using python's Mathplotlib & Numpy, an individual can create various plots, each suited for specific types of data analysis.