DATA VISUALIZATION USING PYTHON PROGRAMMING

Matplotlib is a popular Python library for creating static, interactive, and animated visualizations.
It provides a MATLAB-like interface and is used widely in AI, data analysis, and machine learning.

BACKGROUND: What are the differences between these **Function-Module-Packeage-Library**?

Function: A function is a block of reusable code designed to perform a specific task. It helps in organizing programs and avoiding repetition.

```
def greet(name):
    return f"Hello, {name}!"

print(greet("Students"))
print("x")
```

```
Hello, Students!
x
```

Module: A module is a Python file (.py) that contains a collection of related functions, variables, and classes. It allows you to organize your code logically and reuse it in multiple programs.

To make a module create a file, write the following add and subtract function and then save it with a name. Ex: abc.py

```
# abc.py
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

Create another python file in the same folder and then import the previous file [abc.py]. Call add/subtract functions by using the following code

```
# make sure that abc.py is saved in the local folder
import abc
print(abc.add(5, 6))
```

```
11
```

A package A package is a directory (folder) that contains multiple modules and an *_init_*.py file (which can be empty). It helps organize large projects by grouping related modules together.

Must have an *_init_*.py file (which can be empty).

```
my_package/
│
├── __init__.py
├── math_utils.py
└── string_utils.py
```

Library: A library is a collection of packages and modules that provide tools for performing specific types of tasks such as data analysis, visualization, or machine learning.

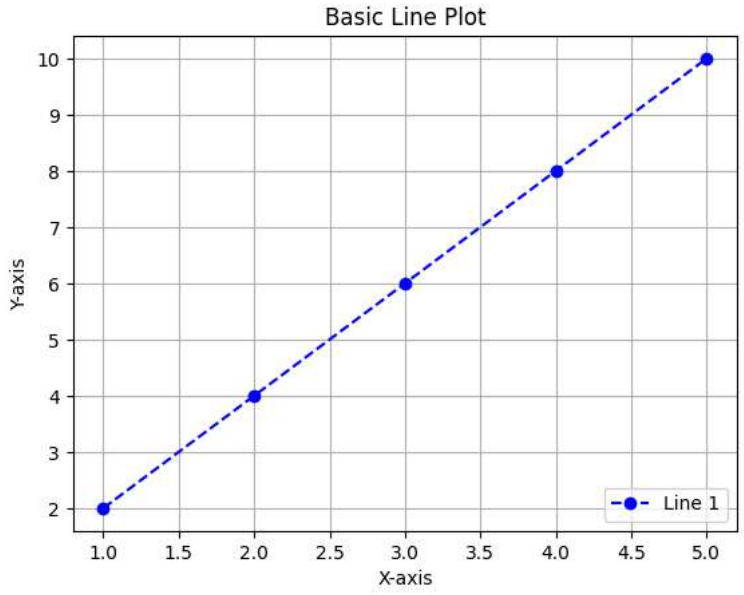
Mathematical Plots

using Matplotlib library

Line Graph

```
# ------------------------------------------
# Program: Basic Line Plot using Matplotlib
# Objective: To visualize a simple linear relationship between X and Y
# ------------------------------------------

# Importing the matplotlib library for data visualization
import matplotlib.pyplot as plt
```

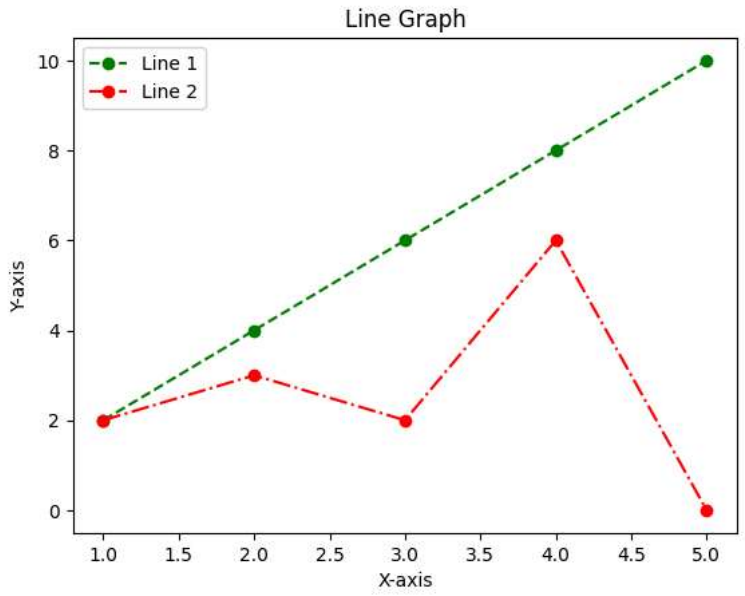
```
# Defining the data points for the X and Y axes
x = [1, 2, 3, 4, 5]          # X-axis values
y = [2, 4, 6, 8, 10]        # Corresponding Y-axis values (linear relation)

# Creating a line plot
# color     : defines the line color
# marker    : 'o' places circular markers on each data point
# linestyle : '--' makes the line dashed
plt.plot(x, y, color='blue', marker='o', linestyle='--')

# Adding a descriptive title to the graph
# loc = 'center' positions the title in the center (default)
plt.title("Basic Line Plot", loc='center')

# Labeling the X and Y axes
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

# Displaying a grid to make it easier to read the graph
plt.grid(True)

# Adding a legend to describe the plotted line
# loc = 'lower right' positions it at the bottom-right corner
plt.legend(["Line 1"], loc='lower right')

# Displaying the final plotted graph
plt.show()
```



Line Plot: More than two variables

```
# ------------------------------------------------------------
# Program: Multiple Line Plot using Matplotlib
# Objective: To plot two different line graphs on the same axes
# ------------------------------------------------------------

# Importing the matplotlib library for plotting
import matplotlib.pyplot as plt

# Defining the data values for the X-axis
x = [1, 2, 3, 4, 5]

# Defining the Y values for the first line (Line 1)
y = [2, 4, 6, 8, 10]      # This shows a linear increasing trend

# Defining the Y values for the second line (Line 2)
z = [2, 3, 2, 6, 0]       # This shows a fluctuating pattern

# Plotting Line 1
# color     → sets the line color
# marker    → 'o' adds circular markers at each point
# linestyle → '--' makes a dashed line
plt.plot(x, y, color='green', marker='o', linestyle='--')

# Plotting Line 2 with different styling
```

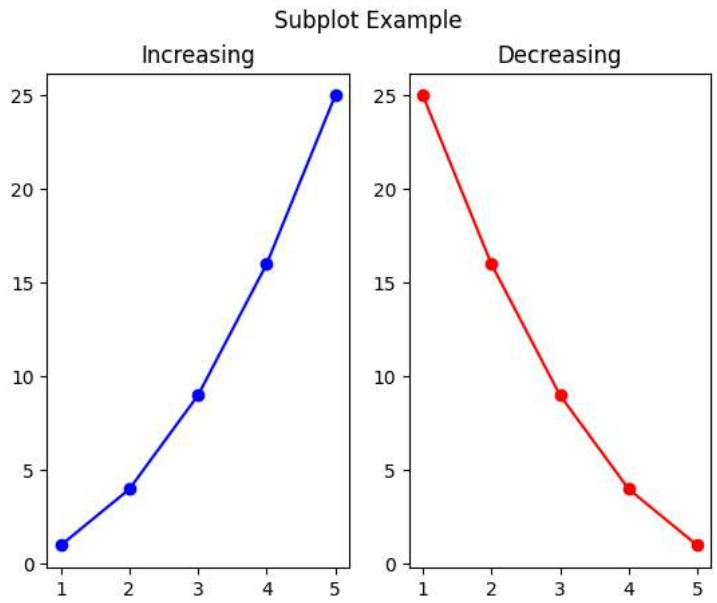
```
# linestyle '-.' is a dash-dot style
plt.plot(x, z, color='red', marker='o', linestyle='-.')

# Adding labels to the axes
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Adding a title to the graph
plt.title('Line Graph')

# Adding a legend to differentiate the two lines
# loc='upper left' places the legend at the top-left corner
plt.legend(['Line 1', 'Line 2'], loc='upper left')

# Displaying the final graph
plt.show()
```



LINE PLOT: Multiple line plot using different subplots

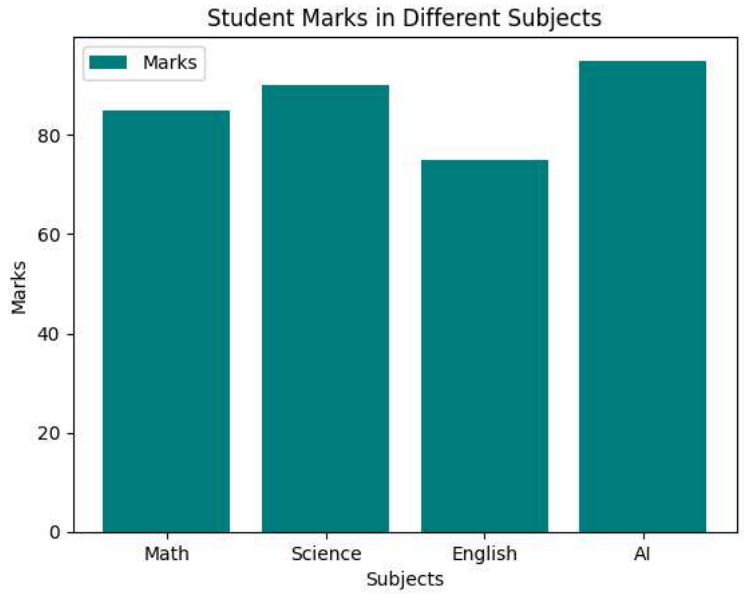
```
# ------------------------------------------------------------
# Program: Subplot Example (Two Graphs in One Figure)
# Objective: To display two line plots side-by-side using subplots
# ------------------------------------------------------------

# Importing matplotlib for plotting
import matplotlib.pyplot as plt

# Defining X values (common for both graphs)
x = [1, 2, 3, 4, 5]

# Y values for the first graph (increasing pattern)
y = [1, 4, 9, 16, 25]

# Y values for the second graph (decreasing pattern)
z = [25, 16, 9, 4, 1]

# ----------------------------
# First Subplot (1 row, 2 columns, position 1)
# ----------------------------
plt.subplot(1, 2, 1)           # Layout: 1 row × 2 columns, this is subplot 1
plt.plot(x, y, color='blue', marker='o')
plt.title("Increasing")        # Title for first graph

# ----------------------------
# Second Subplot (same layout, position 2)
# ----------------------------
plt.subplot(1, 2, 2)           # Subplot position 2
plt.plot(x, z, color='red', marker='o')
plt.title("Decreasing")        # Title for second graph

# Adding a main title for the entire figure
plt.suptitle("Subplot Example")

# Displaying both subplots together
```

```
plt.show()
```

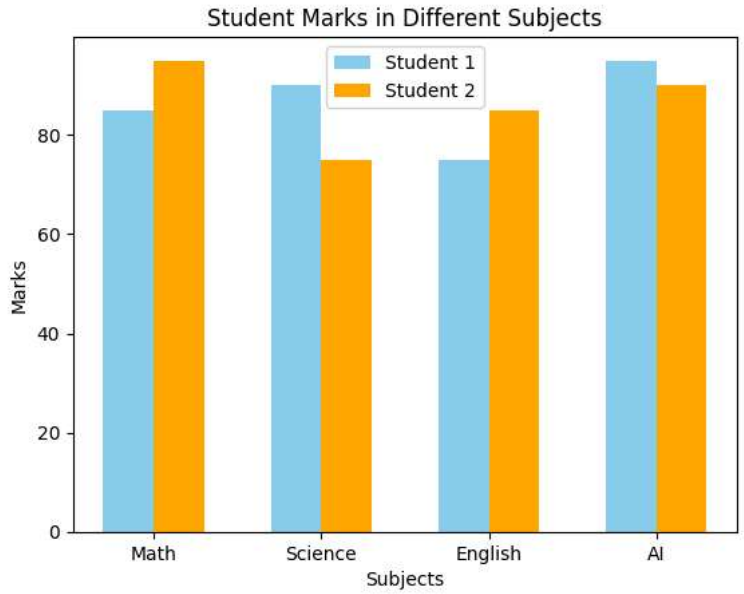


BAR GRAPH:

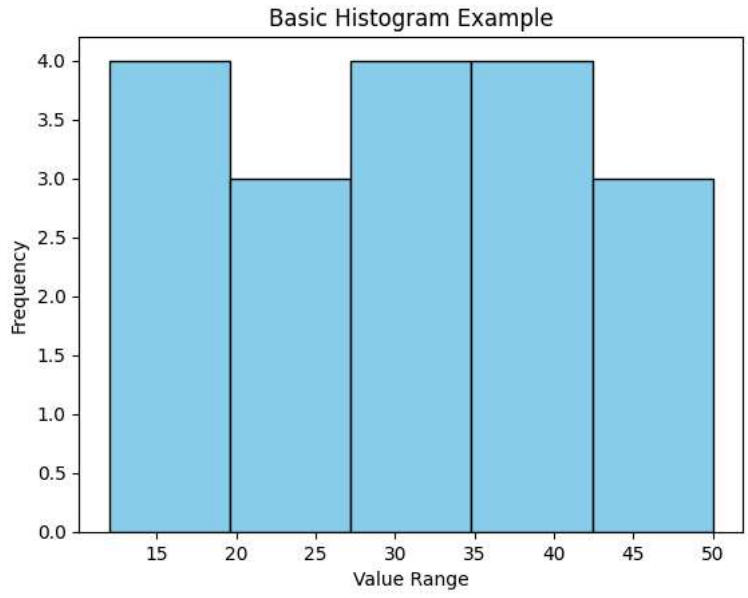
```
# ------------------------------------------------------------
# Program: Bar Graph of Student Marks
# Objective: To visualize marks scored by a student in different subjects
# ------------------------------------------------------------

# Importing matplotlib for creating the bar graph
import matplotlib.pyplot as plt

# Defining the categories (subjects) for the X-axis
subjects = ['Math', 'Science', 'English', 'AI']

# Marks scored by Student 1
marks = [85, 90, 75, 95]

# Creating the bar graph
# subjects → X-axis labels
# marks    → heights of bars
# color    → sets the bar color
plt.bar(subjects, marks, color='teal')

# Adding the graph title
plt.title("Student Marks in Different Subjects")

# Labeling X and Y axes
plt.xlabel("Subjects")
plt.ylabel("Marks")

# Adding a legend to describe the data
plt.legend(["Marks"])

# Displaying the bar graph
plt.show()
```




```
# ------------------------------------------------------------------------
# Program: Side-by-Side Bar Chart (Grouped Bar Chart)
# Objective: Compare marks of two students across different subjects
# ------------------------------------------------------------------------

import matplotlib.pyplot as plt

# Subjects (categories on X-axis)
subs = ['Math', 'Science', 'English', 'AI']

# Marks of two different students
marks = [85, 90, 75, 95]     # Student 1
marks2 = [95, 75, 85, 90]    # Student 2

# Numeric positions for bars on X-axis
x = [1, 2, 3, 4]

# Width of each bar in the grouped bar chart
width = 0.30

# Adjusting positions for the two sets of bars
# x1: left positions for Student 1
# x2: right positions for Student 2
x1 = [item - width/2 for item in x]
x2 = [item + width/2 for item in x]

# Plotting Student 1's marks
plt.bar(x1, marks, width, label='Student 1', color='skyblue')

# Plotting Student 2's marks
plt.bar(x2, marks2, width, label='Student 2', color='orange')

# Adding title and labels
plt.title("Student Marks in Different Subjects")
plt.xlabel("Subjects")
plt.ylabel("Marks")

# Setting the X-axis tick labels to subject names
plt.xticks(x, subs)

# Adding legend to identify the bars
plt.legend()

# Displaying the final chart
plt.show()

# NOTE:
# An alternative method for positioning bars:
# x = np.arange(len(subs))  # Example: array([0, 1, 2, 3])
```




```
# ------------------------------------------------------------
# Program: Histogram Example
# Objective: To visualize the frequency distribution of numerical data
# ------------------------------------------------------------

import matplotlib.pyplot as plt

# Sample numerical data
data = [12, 15, 18, 19, 22, 24, 25, 29, 30, 32, 33, 35, 38, 40, 42, 45, 47, 50]

# Creating the histogram
# data  → numerical values to group into bins
# bins  → number of intervals (groups)
# color → bar color
# edgecolor → outline color for bars
plt.hist(data, bins=5, color='skyblue', edgecolor='black')

# Adding title and axis labels
plt.title("Basic Histogram Example")
plt.xlabel("Value Range")
plt.ylabel("Frequency")

# Displaying the histogram
plt.show()
```

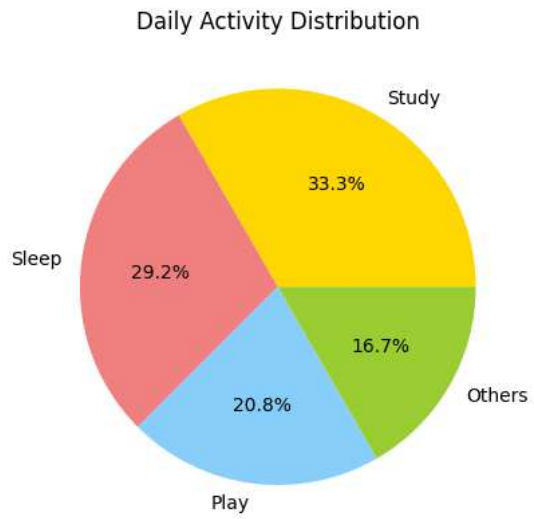


Explanation

plt.hist(data, bins=5, ...) → divides the data into 5 bins (groups) and shows how many values fall in each.

Pie Chart

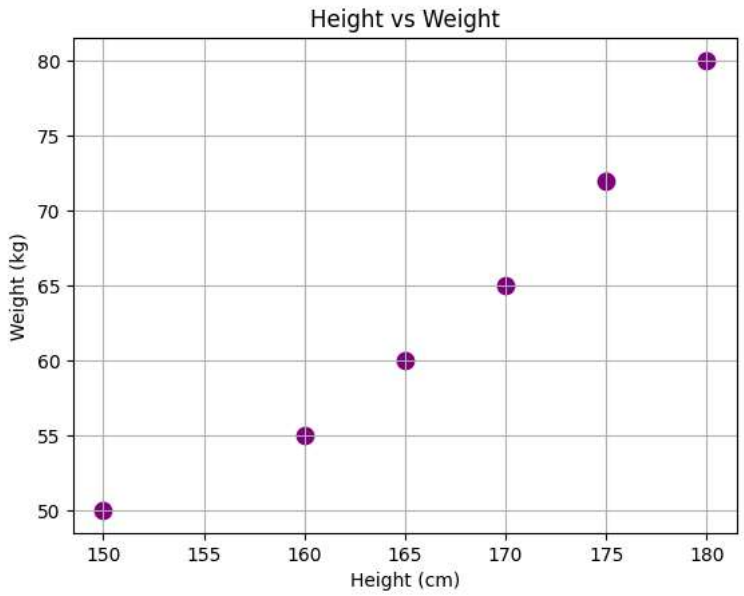
```
# ------------------------------------------------------------
# Program: Pie Chart - Daily Activity Distribution
# Objective: To show how many hours are spent on different daily activities
# ------------------------------------------------------------

import matplotlib.pyplot as plt

# Categories of daily activities
activities = ['Study', 'Sleep', 'Play', 'Others']

# Number of hours spent on each activity; numerical values that determine the size of each slice.
hours = [8, 7, 5, 4]

# Colors for each section of the pie chart
colors = ['gold', 'lightcoral', 'lightskyblue', 'yellowgreen']

# Creating the pie chart
# labels   → names of each activity
# colors   → assigns custom colors to each activity slice
# autopct  → display the percentage value on each slice
plt.pie(hours, labels=activities, colors=colors, autopct='%1.1f%%')

# Adding a title
plt.title("Daily Activity Distribution")

# Displaying the final pie chart
plt.show()
```



Scatter Plot:

```
# ------------------------------------------------------------
# Program: Scatter Plot - Height vs Weight
# Objective: To visualize the relationship between height and weight
# ------------------------------------------------------------

# Importing matplotlib for plotting
import matplotlib.pyplot as plt

# List of heights (in centimeters)
height = [150, 160, 165, 170, 175, 180]

# Corresponding list of weights (in kilograms)
weight = [50, 55, 60, 65, 72, 80]

# Creating a scatter plot
# color = 'purple' sets the dot color
# s = 80 sets the marker size (size of each dot)
plt.scatter(height, weight, color='purple', s=80)

# Adding a title to the graph
plt.title("Height vs Weight")

# Labeling the X and Y axes
plt.xlabel("Height (cm)")
```

```
plt.ylabel("Weight (kg)")

# Adding a grid for clearer visibility
plt.grid(True)

# Displaying the scatter plot
plt.show()
```



Histogram

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Close

Double-click (or enter) to edit

Start coding or generate with AI.

Start coding or generate with AI.

```
import matplotlib.pyplot as plt

subjects = ['Math', 'Science', 'English', 'AI']
marks = [85, 90, 75, 95]
marks2 = [95, 75, 85, 90]
width=0.3
plt.bar(subjects, marks, width, color='teal')
plt.bar(subjects, marks2, width, color='red')
plt.title("Student Marks in Different Subjects")
plt.xlabel("Subjects")
plt.ylabel("Marks")
plt.legend(["Per 1", "Per 2"])
plt.show()
```