***PRACTICAL: 1-A***

***AIM:*** Write a program to draw Line Plot

***CODE***

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

plt.plot(x, y, marker='o', linestyle='-', color='b', label='Line')

plt.xlabel('X-axis')

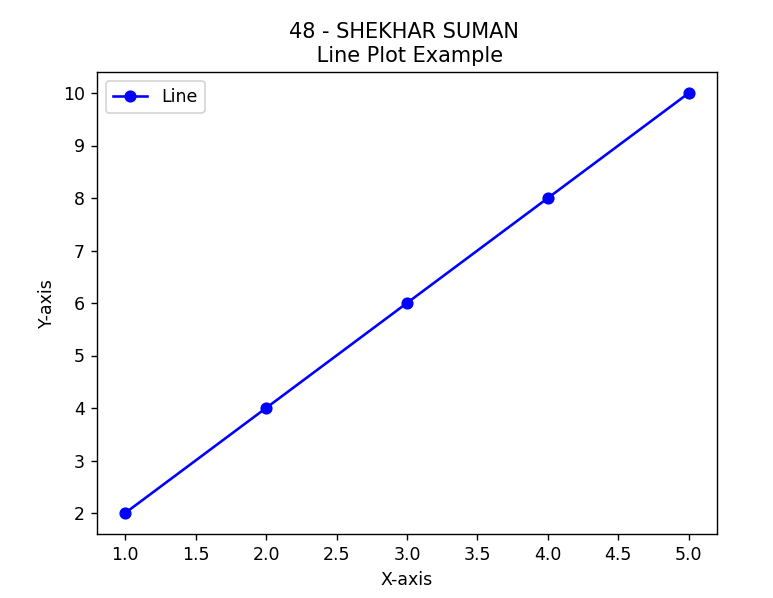
plt.ylabel('Y-axis')

plt.title('48 - SHEKHAR SUMAN \n Line Plot Example')

plt.legend()

plt.show()

***OUTPUT***



***PRACTICAL: 1-B***

***AIM:*** Write a program to Developed a Histogram chart

***CODE***

import matplotlib.pyplot as plt

import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=20, color='skyblue')

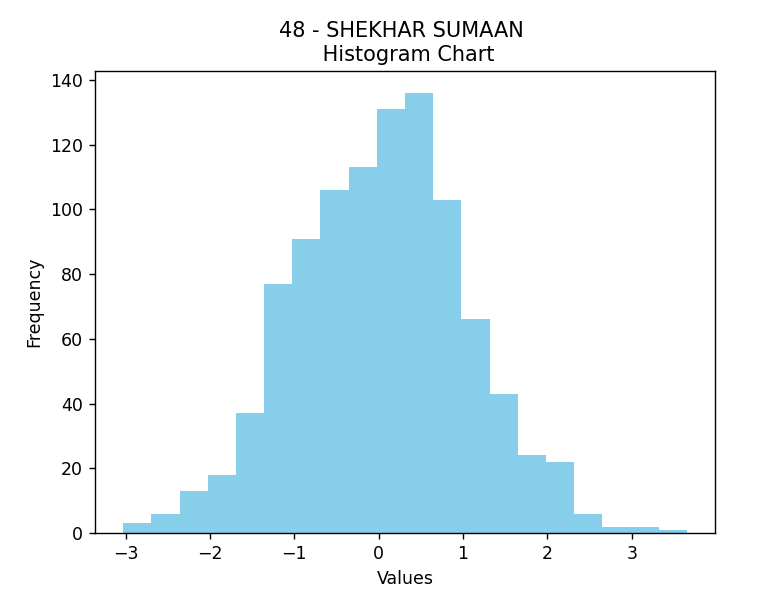
plt.xlabel('Values')

plt.ylabel('Frequency')

plt.title('48 - SHEKHAR SUMAAN \n Histogram Chart')

plt.show()

***OUTPUT***



***PRACTICAL: 1-C***

***AIM:*** Write a program to developed Pie Chart

***CODE***

import matplotlib.pyplot as plt

sizes = [25, 30, 15, 20, 10]

labels = ['A', 'B', 'C', 'D', 'E']

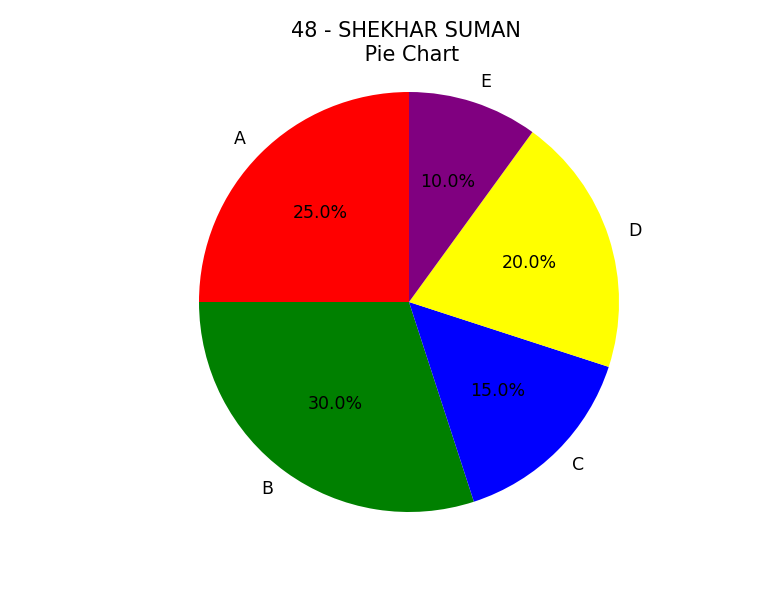
plt.pie(sizes, labels=lables, autopct='%1.1f%%', startangle=90, colors=['red', 'green', 'blue', 'yellow', 'purple'])

plt.axis('equal')

plt.title('48 - SHEKHAR SUMAN \n Pie Chart')

plt.show()

***OUTPUT***



***PRACTICAL: 2-A***

***AIM:*** Draw a co-ordinate axis at the center of the screen.

***CODE***

import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.set\_xlim(-5, 5)

ax.set\_ylim(-5, 5)

ax.axhline(0, color='black', lw=2)

ax.axvline(0, color='black', lw=2)

ax.set\_xlabel('X-axis')

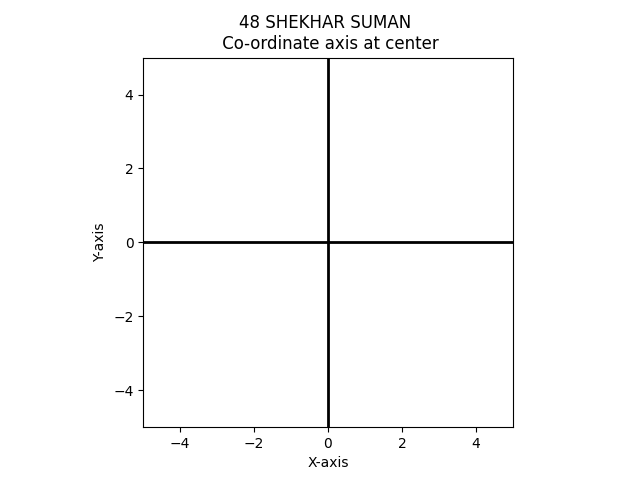
ax.set\_ylabel('Y-axis')

ax.set\_aspect('equal', adjustable='box')

plt.title('48 SHEKHAR SUMAN \n Co-ordinate axis at center')

plt.show()

***OUTPUT***



***PRACTICAL: 3-A***

***AIM:*** Write a program to draw basic shape.

***CODE***

import matplotlib.pyplot as plt

import matplotlib.patches as patches

fig, ax = plt.subplots()

circle = plt.Circle((0.5, 0.5), 0.4, edgecolor='black', facecolor='red')

ax.add\_patch(circle)

rectangle = plt.Rectangle((0.2, 0.2), 0.6, 0.4, edgecolor='black', facecolor='green')

ax.add\_patch(rectangle)

square = plt.Rectangle((0.2, 0.6), 0.4, 0.4, edgecolor='black', facecolor='blue')

ax.add\_patch(square)

circle\_outer = plt.Circle((0.8, 0.2), 0.2, edgecolor='black', facecolor='yellow')

circle\_inner = plt.Circle((0.8, 0.2), 0.1, edgecolor='black', facecolor='orange')

ax.add\_patch(circle\_outer)

ax.add\_patch(circle\_inner)

ellipse = patches.Ellipse((0.5, 0.8), 0.4, 0.2, edgecolor='black', facecolor='purple')

ax.add\_patch(ellipse)

line = plt.Line2D([0.2, 0.8], [0.8, 0.8], color='black', linewidth=2)

ax.add\_line(line)

ax.set\_xlim(0, 1)

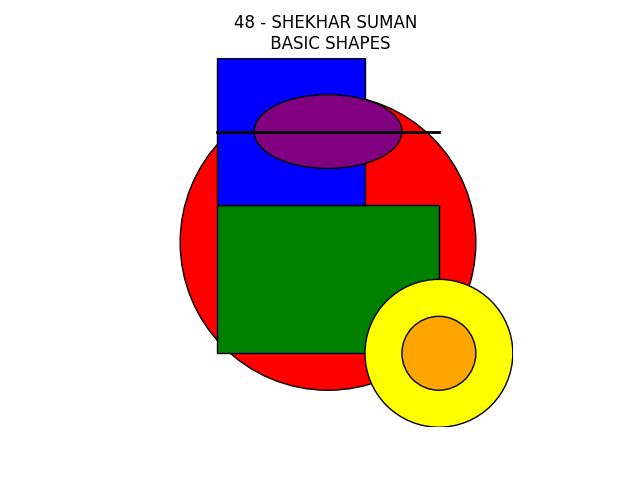
ax.set\_ylim(0, 1)

ax.set\_aspect('equal', adjustable='box')

ax.axis('off')

plt.title('48 - SHEKHAR SUMAN \n BASIC SHAPES')

plt.show()

***OUTPUT***

***PRACTICAL: 3-B***

***AIM:*** Write a program to draw a hut.

***CODE***

import matplotlib.pyplot as plt

import matplotlib.patches as patches

fig, ax = plt.subplots()

ax.add\_patch(patches.Rectangle((0.1, 0.1), 0.8, 0.6, edgecolor='black', facecolor='tan'))

ax.add\_patch(patches.Rectangle((0.35, 0.6), 0.3, 0.1, edgecolor='black', facecolor='saddlebrown'))

roof = plt.Polygon([(0.1, 0.7), (0.5, 1.0), (0.9, 0.7)], edgecolor='black', facecolor='firebrick')

ax.add\_patch(roof)

ax.add\_patch(patches.Rectangle((0.45, 0.1), 0.1, 0.3, edgecolor='black', facecolor='sienna'))

ax.add\_patch(patches.Rectangle((0.2, 0.5), 0.2, 0.2, edgecolor='black', facecolor='lightblue'))

ax.add\_patch(patches.Rectangle((0.6, 0.5), 0.2, 0.2, edgecolor='black', facecolor='lightblue'))

ax.set\_xlim(0, 1)

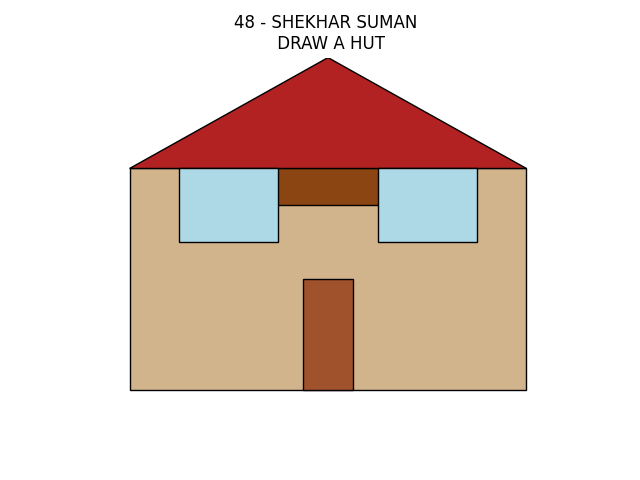
ax.set\_ylim(0, 1)

ax.axis('off')

plt.title('48 - SHEKHAR SUMAN \n DRAW A HUT')

plt.show()

***OUTPUT***



***PRACTICAL: 4-A***

***AIM:*** Write a program to developed **DDA Line drawing algorithm**

***CODE***

import matplotlib.pyplot as plt

def dda\_line(x1, y1, x2, y2):

    dx = x2 - x1

    dy = y2 - y1

    steps = max(abs(dx), abs(dy))

    x\_increment = dx / steps

    y\_increment = dy / steps

    x, y = x1, y1

    x\_points = [x]

    y\_points = [y]

    for \_ in range(steps):

        x += x\_increment

        y += y\_increment

        x\_rounded = round(x)

        y\_rounded = round(y)

        x\_points.append(x\_rounded)

        y\_points.append(y\_rounded)

    return x\_points, y\_points

x1, y1 = 2, 2

x2, y2 = 8, 5

x\_points, y\_points = dda\_line(x1, y1, x2, y2)

plt.plot(x\_points, y\_points, marker='o', linestyle='-', color='b')

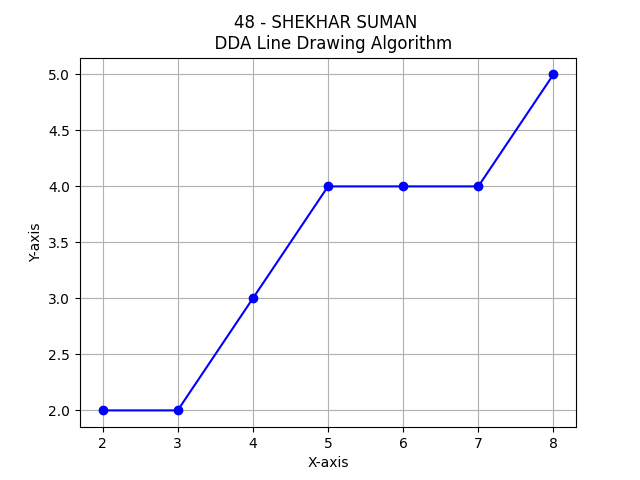
plt.title('48 - SHEKHAR SUMAN \n  DDA Line Drawing Algorithm')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.grid(True)

plt.show()

***OUTPUT***

***PRACTICAL: 4-B***

***AIM:*** Write a program to developed **Bresenham's Line Drawing Algorithm**.

***CODE***

import matplotlib.pyplot as plt

def bresenham\_line(x1, y1, x2, y2):

    dx = abs(x2 - x1)

    dy = abs(y2 - y1)

    sx = 1 if x1 < x2 else -1

    sy = 1 if y1 < y2 else -1

    x, y = x1, y1

    error = dx - dy

    x\_points = [x]

    y\_points = [y]

    while x != x2 or y != y2:

        x\_points.append(x)

        y\_points.append(y)

        error2 = 2 \* error

        if error2 > -dy:

            error -= dy

            x += sx

        if error2 < dx:

            error += dx

            y += sy

    return x\_points, y\_points

x1, y1 = 2, 2

x2, y2 = 8, 5

x\_points, y\_points = bresenham\_line(x1, y1, x2, y2)

plt.plot(x\_points, y\_points, marker='o', linestyle='-', color='b')

plt.title("48 - SHEKHAR SUMAN \n Bresenham's Line Drawing Algorithm")

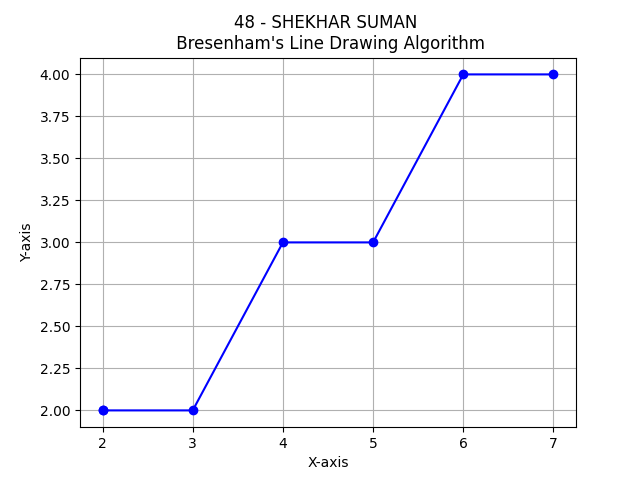
plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.grid(True)

plt.show()

***OUTPUT***



***PRACTICAL: 5-A***

***AIM:*** Write a program to developed **Midpoint Ellipse Drawing Algorithm**.

***CODE***

import matplotlib.pyplot as plt

def midpoint\_ellipse(a, b):

    x, y = 0, b

    a\_sqr = a\*\*2

    b\_sqr = b\*\*2

    d1 = (b\_sqr - (a\_sqr \* b) + 0.25 \* a\_sqr)

    dx = 2 \* b\_sqr \* x

    dy = 2 \* a\_sqr \* y

    x\_points = []

    y\_points = []

    plot\_ellipse\_points(x, y, x\_points, y\_points)

    while dx < dy:

        x += 1

        dx += 2 \* b\_sqr

        if d1 < 0:

            d1 += dx + b\_sqr

        else:

            y -= 1

            dy -= 2 \* a\_sqr

            d1 += dx - dy + b\_sqr

        plot\_ellipse\_points(x, y, x\_points, y\_points)

    d2 = b\_sqr \* (x + 0.5)\*\*2 + a\_sqr \* (y - 1)\*\*2 - a\_sqr \* b\_sqr

    while y > 0:

        y -= 1

        dy -= 2 \* a\_sqr

        if d2 > 0:

            d2 += a\_sqr - dy

        else:

            x += 1

            dx += 2 \* b\_sqr

            d2 += a\_sqr + dx - dy

        plot\_ellipse\_points(x, y, x\_points, y\_points)

    return x\_points, y\_points

def plot\_ellipse\_points(x, y, x\_points, y\_points):

    x\_points.extend([x, -x, x, -x])

    y\_points.extend([y, y, -y, -y])

a, b = 5, 3

x\_points, y\_points = midpoint\_ellipse(a, b)

plt.scatter(x\_points, y\_points, marker='.', color='b')

plt.title("48 - SHEKHAR SUMAN \n Midpoint Ellipse Drawing Algorithm")

plt.xlabel('X-axis')

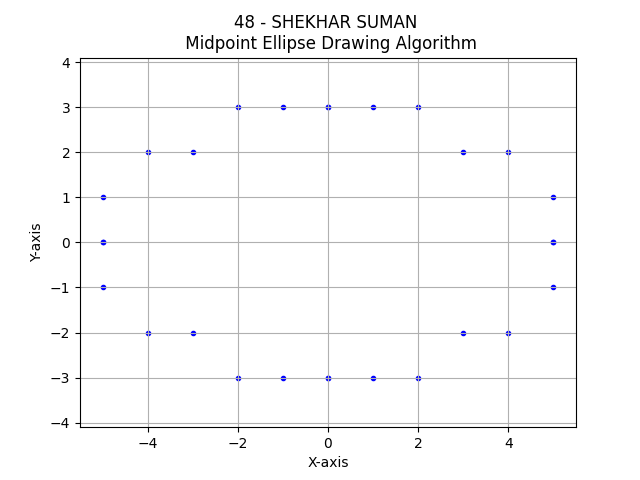
plt.ylabel('Y-axis')

plt.axis('equal')

plt.grid(True)

plt.show()

***OUTPUT***



***PRACTICAL: 6-A***

***AIM:*** Write a program to developed **2D SCALING EXAMPLE**.

***CODE***

import matplotlib.pyplot as plt

def scale\_2d(points, scale\_factor):

    scaled\_points = []

    for point in points:

        x, y = point

        scaled\_x = x \* scale\_factor

        scaled\_y = y \* scale\_factor

        scaled\_points.append((scaled\_x, scaled\_y))

    return scaled\_points

def plot\_points(points, color='b', label=None):

    x, y = zip(\*points)

    plt.scatter(x, y, color=color, label=label)

original\_points = [(1, 1), (2, 3), (3, 2), (4, 4)]

scale\_factor = 2

scaled\_points = scale\_2d(original\_points, scale\_factor)

plot\_points(original\_points, color='b', label='Original Points')

plot\_points(scaled\_points, color='r', label=f'Scaled Points (Factor: {scale\_factor})')

plt.axis('equal')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

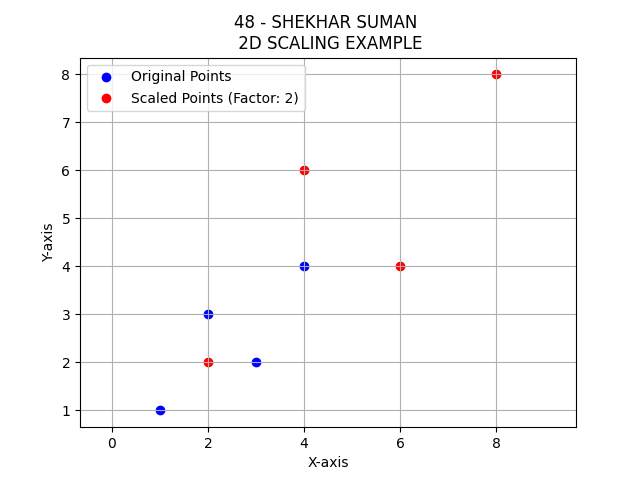
plt.title('48 - SHEKHAR SUMAN \n 2D SCALING EXAMPLE')

plt.legend()

plt.grid(True)

plt.show()

***OUTPUT***



***PRACTICAL: 7-A***

***AIM:*** Write a program to developed **2D ROTATION EXAMPLE**.

***CODE***

import matplotlib.pyplot as plt

import numpy as np

def rotate\_2d(points, angle\_degrees, rotation\_center=(0, 0)):

    angle\_radians = np.radians(angle\_degrees)

    cos\_theta = np.cos(angle\_radians)

    sin\_theta = np.sin(angle\_radians)

    rotated\_points = []

    for point in points:

        x, y = point

        x\_centered = x - rotation\_center[0]

        y\_centered = y - rotation\_center[1]

        rotated\_x = x\_centered \* cos\_theta - y\_centered \* sin\_theta

        rotated\_y = x\_centered \* sin\_theta + y\_centered \* cos\_theta

        rotated\_x += rotation\_center[0]

        rotated\_y += rotation\_center[1]

        rotated\_points.append((rotated\_x, rotated\_y))

    return rotated\_points

def plot\_points(points, color='b', label=None):

    x, y = zip(\*points)

    plt.scatter(x, y, color=color, label=label)

original\_points = [(1, 1), (2, 3), (3, 2), (4, 4)]

rotation\_angle = 45

rotation\_center = (2, 2)

rotated\_points = rotate\_2d(original\_points, rotation\_angle, rotation\_center)

plot\_points(original\_points, color='b', label='Original Points')

plot\_points(rotated\_points, color='r', label=f'Rotated Points (Angle: {rotation\_angle})')

plt.axis('equal')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('48 - SHEKHAR SUMAAN \n 2D ROTATION EXAMPLE')

plt.legend()

plt.grid(True)

plt.show()

***OUTPUT***

