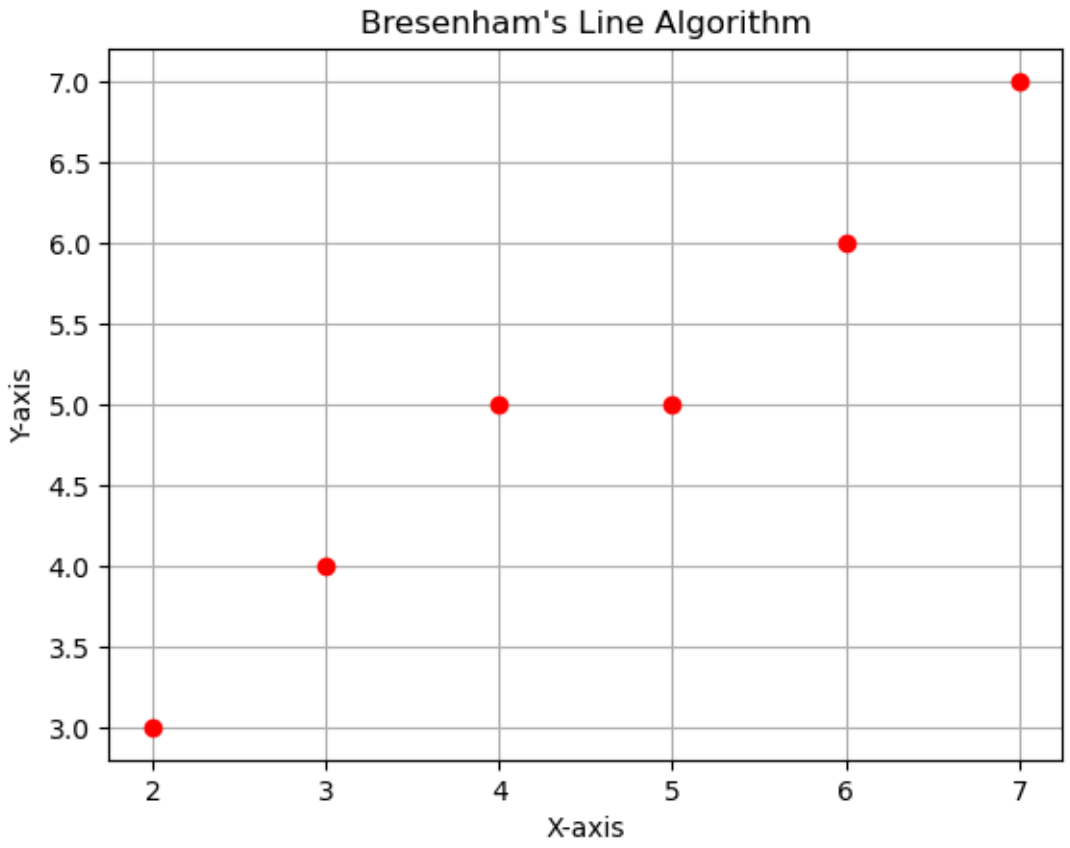
# g-hw

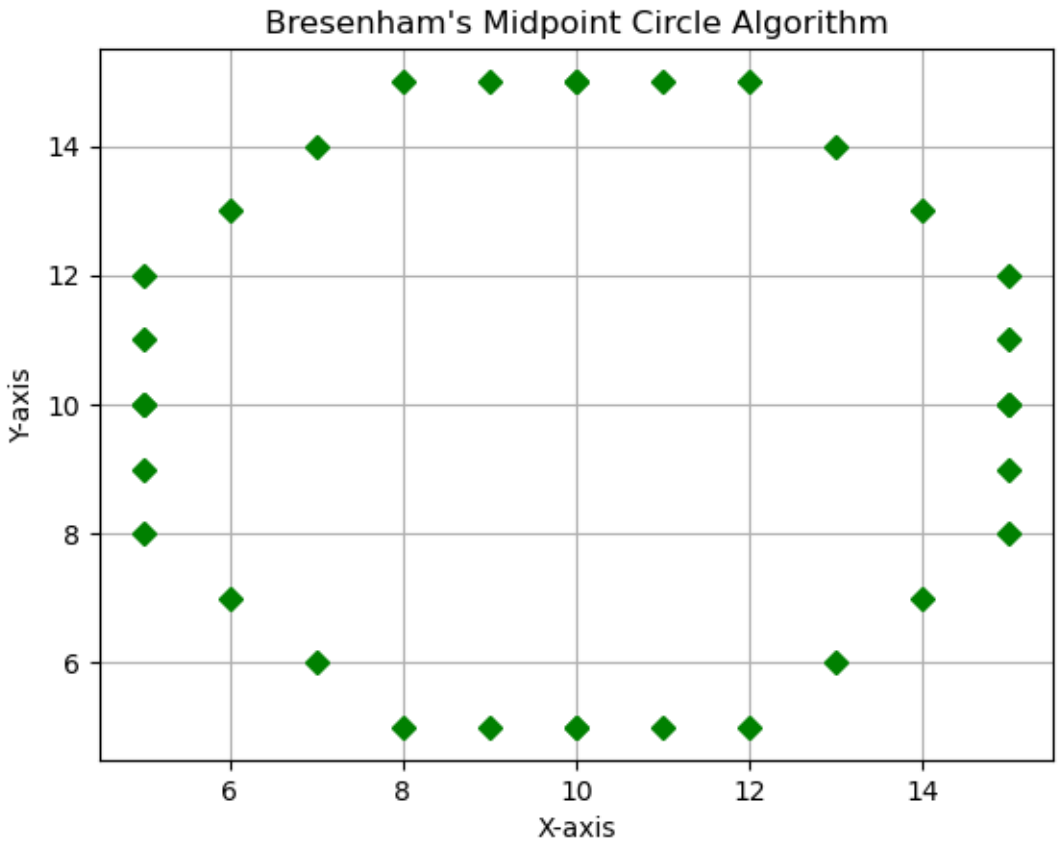
December 3, 2023

[2]:
```
# Bresenham's Line Algorithm

import matplotlib.pyplot as plt

def B_L_A(x0, y0, x1, y1):
    dx = abs(x1 - x0)
    dy = abs(y1 - y0)
    sx = 1 if x0 < x1 else -1
    sy = 1 if y0 < y1 else -1
    err = dx - dy

    x, y = x0, y0
    plt.plot(x, y, marker='o', color='red')

    while (x, y) != (x1, y1):
        err_2 = 2 * err

        if err_2 > -dy:
            err -= dy
            x += sx

        if err_2 < dx:
            err += dx
            y += sy

        plt.plot(x, y, marker='o', color='red')

    plt.title("Bresenham's Line Algorithm")
    plt.xlabel("X-axis")
    plt.ylabel("Y-axis")
    plt.grid(True)
    plt.show()

#     x0,y0, x1,y1
B_L_A(2, 3, 7, 7)
```



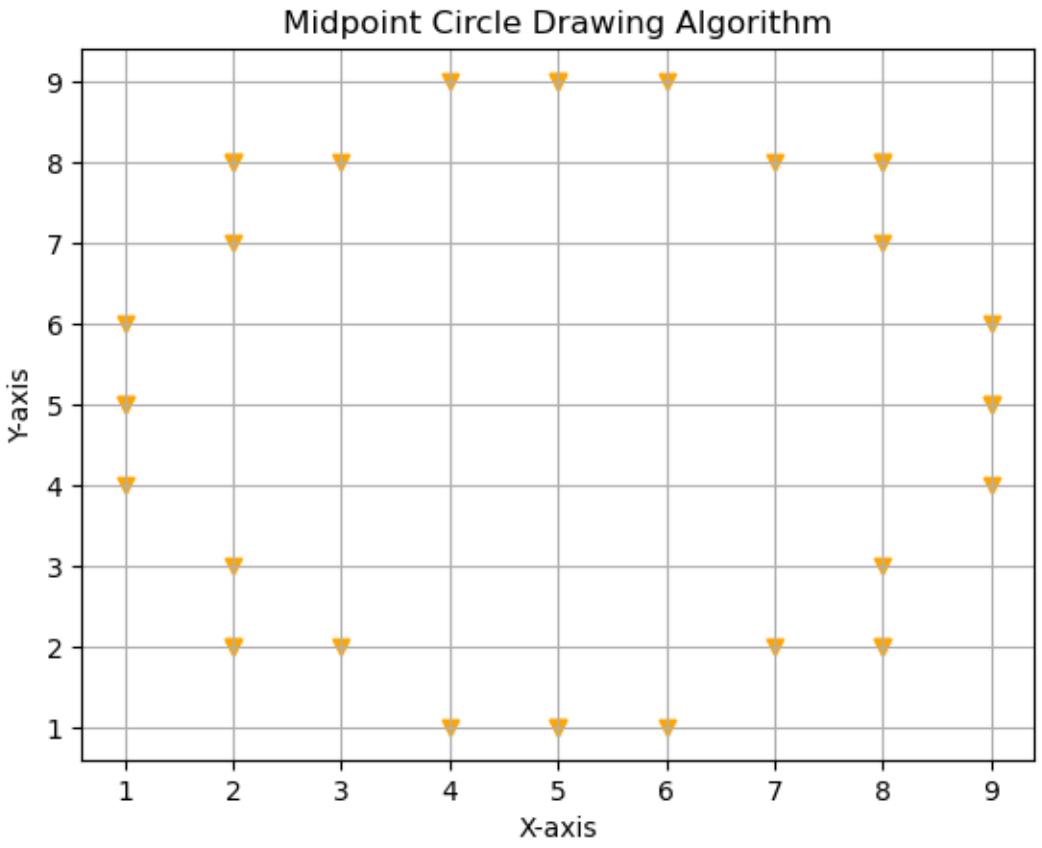

[2]:
```
# Bresenham's Midpoint Circle Algorithm

def B_M_C_A(x_center, y_center, radius):
    y = 0
    x = radius
    p = 1 - radius

    points = []

    while x >= y:

        points.extend([
            (x + x_center, y + y_center),(-x + x_center, y + y_center),
            (x + x_center, -y + y_center),(-x + x_center, -y + y_center),
            (y + x_center, x + y_center),(-y + x_center, x + y_center),
            (y + x_center, -x + y_center),(-y + x_center, -x + y_center)])
        y += 1

        if p <= 0:
            p = p + 2 * y + 1
```

```
        else:
            x -= 1
            p = p + 2 * y - 2 * x + 1

    for point in points:
        plt.plot(point[0], point[1], marker='D', color='green')

    plt.title("Bresenham's Midpoint Circle Algorithm")
    plt.xlabel("X-axis")
    plt.ylabel("Y-axis")
    plt.grid(True)
    plt.show()

# (x_center, y_center, radius)
B_M_C_A(10, 10, 5)
```




[11]:
```
# Midpoint Circle Drawing Algorithm

def M_C_D_A(x_center, y_center, radius):
    x = radius
```

```
    y = 0
    p = 5/4 - radius
    points = []

    while x >= y:

        points.extend([
            (x + x_center, y + y_center),(-x + x_center, y + y_center),
            (x + x_center, -y + y_center),(-x + x_center, -y + y_center),
            (y + x_center, x + y_center), (-y + x_center, x + y_center),
            (y + x_center, -x + y_center),(-y + x_center, -x + y_center)])
        y += 1

        if p <= 0:
            p = p + 2 * y + 1
        else:
            x -= 1
            p = p + 2 * y - 2 * x + 1

    for point in points:
        plt.scatter(point[0], point[1], color='orange', marker='v')

    plt.title("Midpoint Circle Drawing Algorithm")
    plt.xlabel("X-axis")
    plt.ylabel("Y-axis")
    plt.grid(True)
    plt.show()

# (x_center, y_center, radius)
M_C_D_A(5, 5, 4)
```


Midpoint Circle Drawing Algorithm
Y-axis
X-axis
1
2
3
4
5
6
7
8
9