LAB 2: IMPLEMENTATION OF DDA LINE DRAWING ALGORITHM (PYTHON)

Objective(s)

i) To understand the Digital Differential Analyzer (DDA) line drawing algorithm.

ii) To implement the DDA algorithm in Python and visualise the generated pixels.

iii) To compare DDA with the analytical line equation.

Software(s) Required:

Python 3, matplotlib, any IDE or Jupyter Notebook.

Theory:

Given two endpoints ($x1$, $y1$) and ($x2$, $y2$), we define: The number of steps is chosen as: In raster graphics, a straight line must be approximated using discrete pixels. The DDA algorithm is an incremental method that uses the line slope to step along the dominant axis and compute intermediate points. $dx = x2 - x1$, $dy = y2 - y1$ steps = max(|dx|, |dy|) The increment in each step is: $x$inc = dx steps , $y$inc = dy steps Starting from ($x1$, $y1$), the algorithm adds these increments repeatedly and rounds to the nearest pixel.

Algorithm: DDA Line Drawing 1. Read starting point ($x1$, $y1$) and ending point ($x2$, $y2$)

2. Compute dx,dy and number of steps.

3. Compute $x$inc and $y$inc.

4. Initialise $x = x1$, $y = y1$.

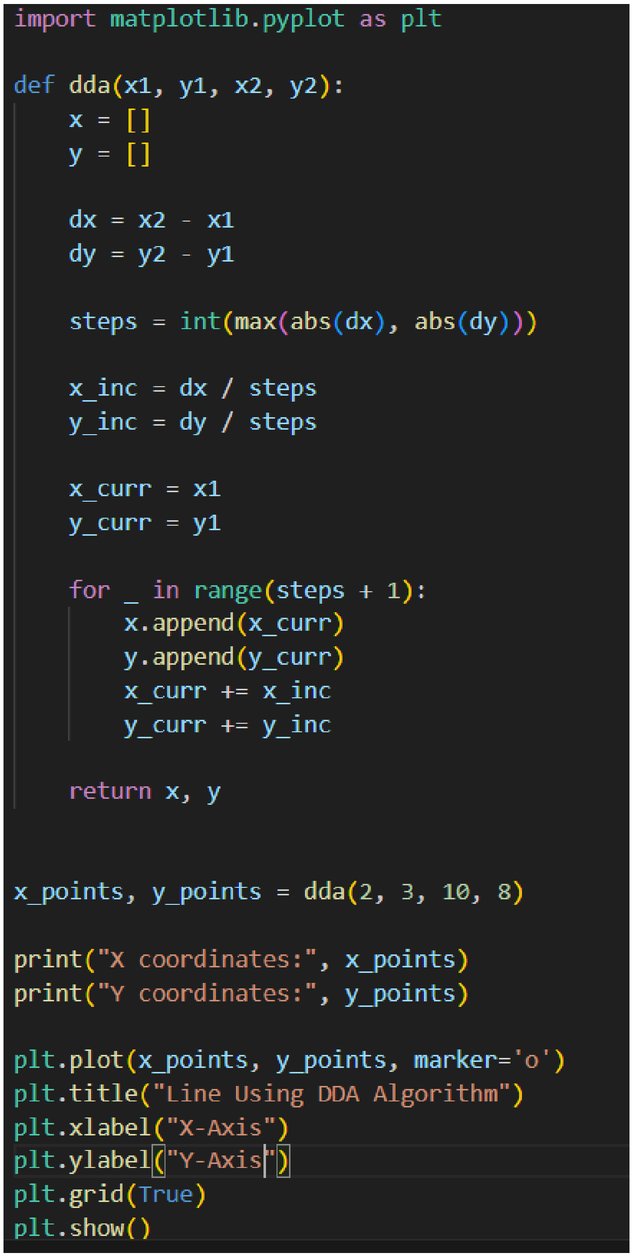
5. For k = 0to steps:

• Plot pixel at (round(x),round(y)).

• Update $x = x +x$inc, $y = y + y$inc.

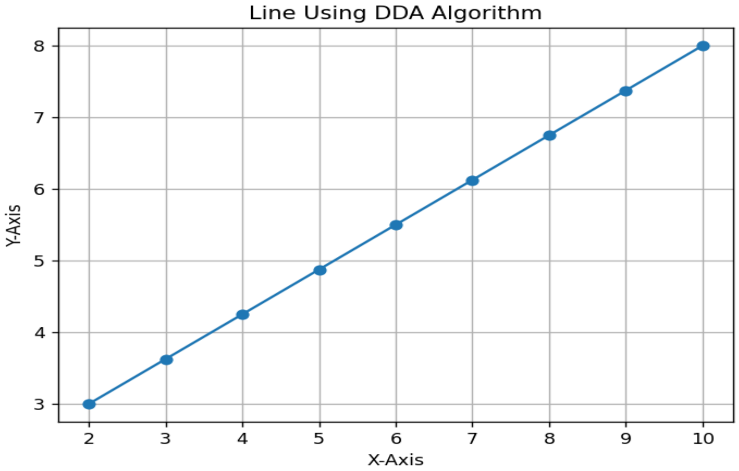
Qn1.Implement the DDA algorithm as a function that returns x and y coordinate

Code:



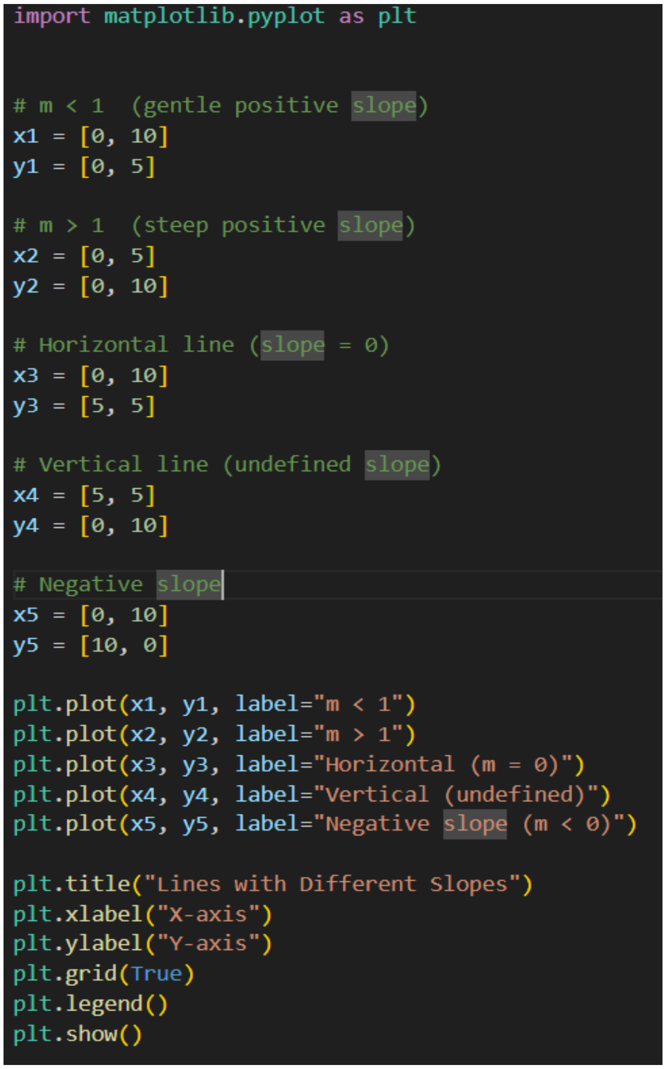
```
import matplotlib.pyplot as plt

def dda(x1, y1, x2, y2):
    x = []
    y = []

    dx = x2 - x1
    dy = y2 - y1

    steps = int(max(abs(dx), abs(dy)))

    x_inc = dx / steps
    y_inc = dy / steps

    x_curr = x1
    y_curr = y1

    for _ in range(steps + 1):
        x.append(x_curr)
        y.append(y_curr)
        x_curr += x_inc
        y_curr += y_inc

    return x, y


x_points, y_points = dda(2, 3, 10, 8)

print("X coordinates:", x_points)
print("Y coordinates:", y_points)

plt.plot(x_points, y_points, marker='o')
plt.title("Line Using DDA Algorithm")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.grid(True)
plt.show()
```

Output:



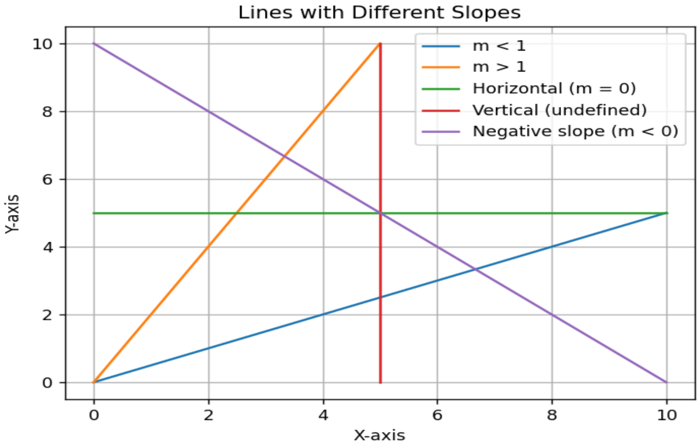
Qn2.Draw a line with the different slopes:m<1,m>1, horizontal, vertical and negative slope

Code:



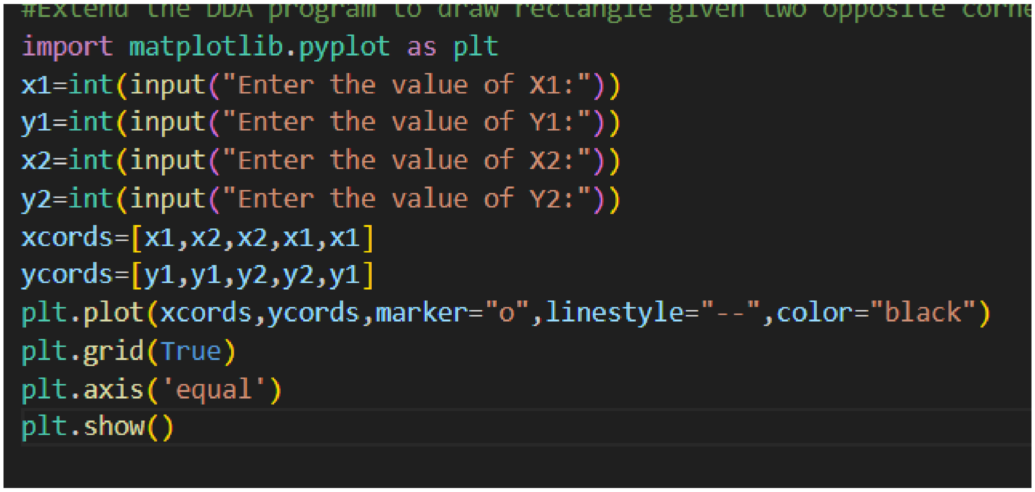
```python
import matplotlib.pyplot as plt


# m < 1  (gentle positive slope)
x1 = [0, 10]
y1 = [0, 5]

# m > 1  (steep positive slope)
x2 = [0, 5]
y2 = [0, 10]

# Horizontal line (slope = 0)
x3 = [0, 10]
y3 = [5, 5]

# Vertical line (undefined slope)
x4 = [5, 5]
y4 = [0, 10]

# Negative slope
x5 = [0, 10]
y5 = [10, 0]

plt.plot(x1, y1, label="m < 1")
plt.plot(x2, y2, label="m > 1")
plt.plot(x3, y3, label="Horizontal (m = 0)")
plt.plot(x4, y4, label="Vertical (undefined)")
plt.plot(x5, y5, label="Negative slope (m < 0)")

plt.title("Lines with Different Slopes")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid(True)
plt.legend()
plt.show()
```

Output:



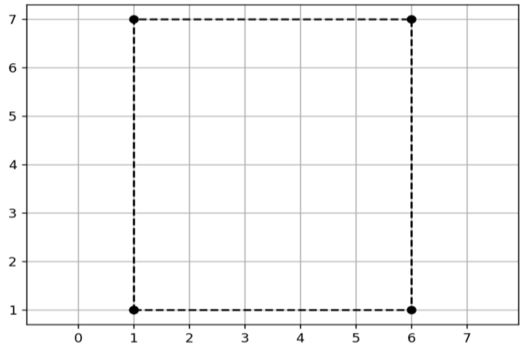
Qn3:Extend the DDA program to draw a rectangle given two opposite corners.

Code:



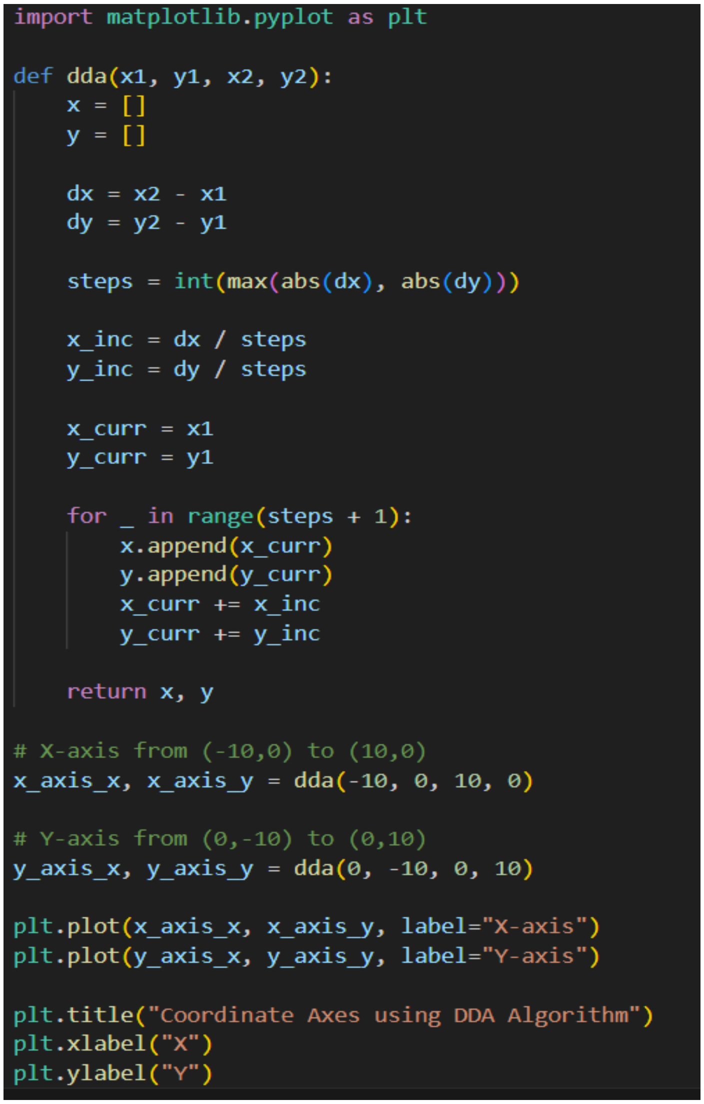
```
#Extend the DDA program to draw rectangle given two opposite corne
import matplotlib.pyplot as plt
x1=int(input("Enter the value of X1:"))
y1=int(input("Enter the value of Y1:"))
x2=int(input("Enter the value of X2:"))
y2=int(input("Enter the value of Y2:"))
xcords=[x1,x2,x2,x1,x1]
ycords=[y1,y1,y2,y2,y1]
plt.plot(xcords,ycords,marker="o",linestyle="--",color="black")
plt.grid(True)
plt.axis('equal')
plt.show()
```

Output:

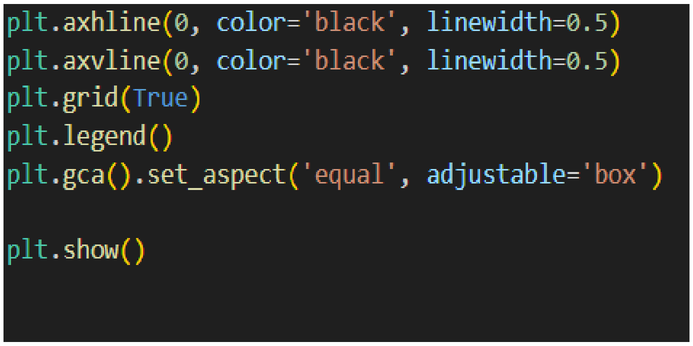


Qn4: Use DDA to draw the axes of a simple coordinate system(X and Y axes)

Code:

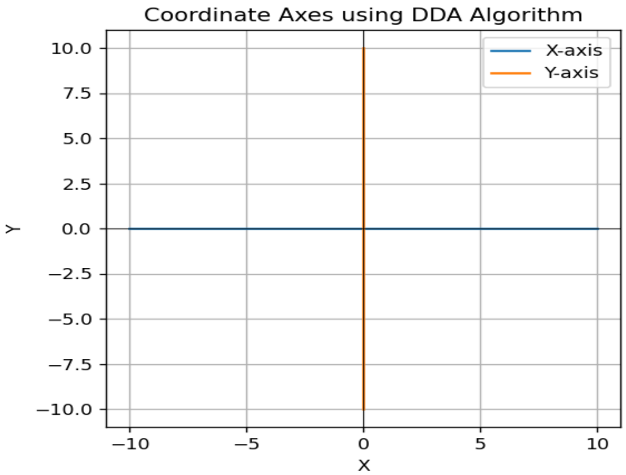


```python
import matplotlib.pyplot as plt

def dda(x1, y1, x2, y2):
    x = []
    y = []

    dx = x2 - x1
    dy = y2 - y1

    steps = int(max(abs(dx), abs(dy)))

    x_inc = dx / steps
    y_inc = dy / steps

    x_curr = x1
    y_curr = y1

    for _ in range(steps + 1):
        x.append(x_curr)
        y.append(y_curr)
        x_curr += x_inc
        y_curr += y_inc

    return x, y

# X-axis from (-10,0) to (10,0)
x_axis_x, x_axis_y = dda(-10, 0, 10, 0)

# Y-axis from (0,-10) to (0,10)
y_axis_x, y_axis_y = dda(0, -10, 0, 10)

plt.plot(x_axis_x, x_axis_y, label="X-axis")
plt.plot(y_axis_x, y_axis_y, label="Y-axis")

plt.title("Coordinate Axes using DDA Algorithm")
plt.xlabel("X")
plt.ylabel("Y")
```



```python
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True)
plt.legend()
plt.gca().set_aspect('equal', adjustable='box')

plt.show()
```

Output:)


Coordinate Axes using DDA Algorithm
X-axis
Y-axis
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
Y
−10
−5
0
5
10
X