import matplotlib.pyplot as plt
import numpy as np

def folding_curve(iterations, shape, fold_command_list, xlimVals=np.array([-1.5, 1.5]), ylimVals=np.array([-1.5, 1.5]), fourColors=0):
    # This function draws a Folding Curve with the following inputs:
    #   iterations = int; number of iterations
    #   xlimVals = np.array([xmin,xmax]); set the x-boundaries for the graph
    #   ylimVals = np.array([ymin,ymax]); set the y-boundaries for the graph
    #   pattern = np.array([pattern]) sequence of 1's and 0's. For each 0, the pattern
    #   will turn
    #   right. For each 1 the pattern will turn left. When the pattern reaches
    #   the end, it will repeat. For example, [0,1] will generate the standard
    #   dragon curve
    #   fourColors = 0 or 1; determines the color of the curve
    #       0 = curve is blue fading into red
    #       1 = curve is split into four segments, each a different color
    #   shape = 0 or 1; determines the starting shape (e.g. iteration 0)
    #       0 = horizontal line
    #       1 = square

    angle = np.pi/4

    if shape == 1:
        pts = np.array([[-.5, -.5], [.5, -.5], [.5, .5], [-.5, .5], [-.5, -.5]])
    elif shape == 0:
        pts = np.array([[-.5, 0], [0, 0]])

    len1 = 1
    c = np.array([0, 0, 1]);

    # Repeat for the given number of iterations

    for i in range(1, iterations + 1):
        print('starting iteration ' + str(i))
        len1 = 0.5*len1/(np.cos(angle))
        temp = np.empty([0, 0])
        dir1 = 0
        if i == iterations:
            plt.figure(figsize=(10, 10))

            # Repeat for each line segment in the curve

        for j in range(len(pts) - 1):
            theta = 0;
            # Grab the two endpoints of the curve
            pt1 = pts[j]
            pt2 = pts[j + 1]

            # Subtract one endpoint from the other so you're centered on
            # zero and can more easily calculate the angle
            pt2corr = np.array([pt2[0] - pt1[0], pt2[1] - pt1[1]])
if (0<pt2corr[0]) and (0>=pt2corr[1]):  # x pos y neg
    theta=np.pi*2-

elif (0<pt2corr[0]) and (0>=pt2corr[1]):  # x pos y neg
    theta=np.pi+np.arctan(abs(pt2corr[1]/pt2corr[0]))

elif (0<pt2corr[0]) and (0<pt2corr[1]):  # x neg y pos
    theta=np.pi-np.arctan(abs(pt2corr[1]/pt2corr[0]))

elif pt2corr[0]==0 and pt2corr[1]>=0:
    theta=np.pi/2;

elif pt2corr[0]==0 and pt2corr[1]<0:
    theta=3*np.pi/2;

else :  # (0<=pt2corr[0] and 0<=pt2corr[1]) :  # both pos
    theta=np.arctan(abs(pt2corr[1]/pt2corr[0]))

# Determine the new point (the 'midpoint') for your line segment
if fold_command_list[dir1]==0 :  # If turning RIGHT
    midpt=np.array([len1*np.cos(theta-angle)+pt1[0],
                    len1*np.sin(theta-angle)+pt1[1]])

else :  # If turning LEFT
    midpt=np.array([len1*np.cos(theta+angle)+pt1[0],
                    len1*np.sin(theta+angle)+pt1[1]])

# Determine the color based on the current iteration
if fourColors==0 :
    c=np.array([j/len(pts), 0, 1-j/len(pts)])

else :
    if j>(3/4)*len(pts) :
        c=np.array([1,0,0])
    elif j>(1/2)*len(pts) :
        c=np.array([0,1,0])
    elif j>(1/4)*len(pts) :
        c=np.array([0,0,1])
    else :
        c=np.array([0,0,0])

# Plot the new line segments
if i==iterations:
    plt.plot([pt1[0],midpt[0],pt2[0]],
              [pt1[1],midpt[1],pt2[1]],'-',color=tuple(c))

# Add the new points to the temp variable
if np.size(temp,0)==0:
    temp=pt1;
    temp.resize([1,2])
else:
    temp=np.append(temp,[pt1],axis=0);
    temp=np.append(temp,[midpt],axis=0)

# Change direction based on the fold_command_list
if dir1==len(fold_command_list)-1 :
    dir1=0
    else:
        dir1=dir1+1
        temp=np.append(temp,[pts[-1]],axis=0)
        del pts
        pts=temp.copy()

    # Clean up the display
    if i==iterations:
        plt.axis('off')
        plt.xlim(xlimVals)
        plt.ylim(ylimVals)
        plt.show()

    # Example
    folding_curve(10,0,np.array([0,0,1]))