+ Code + Text

```
'''
cp467
A5 v2
thinning
nov, 2022

Write a program to thin an image,
show the input image and the output image and
explain your program and your method in your report. Make any necessarily assumptions.
'''

#code:
#https://scikit-image.org/docs/dev/auto_examples/edges/plot_skeleton.html
#https://rosettacode.org/wiki/Zhang-Suen_thinning_algorithm#Python

#concepts:
#https://medium.com/analytics-vidhya/skeletonization-in-python-using-opencv-b7fa16867331
#https://nayefreza.wordpress.com/2013/05/11/zhang-suen-thinning-algorithm-java-implementation



def neighbours(x, y, image):
    '''Return 8-neighbours of point p1 of picture, in order'''

    x1, y1= x+1, y-1
    x_1, y_1 = x-1, y+1

    p2 = image[y1][x]
    p3= image[y1][x1]

    p4 = image[y][x1]
    p5=image[y_1][x1]

    p6=image[y_1][x]
    p7=image[y_1][x_1]

    p8 = image[y][x_1]
    p9 = image[y1][x_1]

    n = [p2,p3,p4,p5,p6,p7,p8,p9]

    return n

def transitions(neighbours):
    n = neighbours + neighbours[0:1]    # P2, ... P9, P2

    s = 0 #sum
    for n1, n2 in zip(n, n[1:]):
        if (n1, n2) == (0, 1) : #white -> black
          s+=1
    return s
```

```
        return s

        #return sum((n1, n2) == (0, 1) for n1, n2 in zip(n, n[1:]))

    def zhangSuen(image):

        changing1 = changing2 = [(-1, -1)]
        while changing1 or changing2:

            #====================
            # Step 1
            changing1 = []

            for y in range(1, len(image) - 1): #traversal
                for x in range(1, len(image[0]) - 1):

                    #neighbours
                    P2,P3,P4,P5,P6,P7,P8,P9 = n = neighbours(x, y, image)


                    #Conditions
                    cond0=(image[y][x] == 1)
                    cond1 = (2 <= sum(n) <= 6)
                    cond2 =(transitions(n) == 1)

                    cond3 =(P2 * P4 * P6 == 0)
                    cond4 =(P4 * P6 * P8 == 0)


                    if (cond0 and cond1 and  cond2 and cond3 and cond4):
                        changing1.append((x,y))

            for x, y in changing1: #set to white
              image[y][x] = 0

            #====================
            # Step 2
            changing2 = []

            for y in range(1, len(image) - 1):#traversal
                for x in range(1, len(image[0]) - 1):

                    #neighbours
                    P2,P3,P4,P5,P6,P7,P8,P9 = n = neighbours(x, y, image)


                    #Conditions
                    cond0=(image[y][x] == 1)
                    cond1 = (2 <= sum(n) <= 6)
                    cond2 =(transitions(n) == 1)

                    cond3 =(P2 * P4 * P8 == 0 )
```

```
                cond4 =(P2 * P6 * P8 == 0)


                if (cond0 and cond1 and  cond2 and cond3 and cond4):
                    changing2.append((x,y))


        for x, y in changing2: #set to white
          image[y][x] = 0


    return image


from skimage.morphology import skeletonize
from skimage import data
import matplotlib.pyplot as plt

def main(blobs):

    #display
    figure, axes = plt.subplots(1, 3, figsize=(8, 5), sharex=True, sharey=True)
    display = axes.ravel()

    #display orig
    display[0].imshow(blobs, cmap=plt.cm.gray)
    display[0].set_title('original blobs')
    display[0].axis('off')

    #============================
    #thinning -version 1 - with api #(does not mutate blobs var)
    thinned_image_1 = skeletonize(blobs)

    #display thinned 1
    display[1].imshow(thinned_image_1, cmap=plt.cm.gray)
    display[1].set_title('thinned_image_1')
    display[1].axis('off')

    #============================
    #thinning -version 2 - with function #(does  mutate blobs var)
    thinned_image_2 =zhangSuen(blobs)

    #display thinned 2
    display[2].imshow(thinned_image_2, cmap=plt.cm.gray)
    display[2].set_title('thinned_image_2')
    display[2].axis('off')



#3 example inputs
print("3 examples:\n\n")
```

```
blobs = data.binary_blobs(100, blob_size_fraction=.2,  volume_fraction=.40, seed=15)
main(blobs)


blobs = data.binary_blobs(100, blob_size_fraction=.2, volume_fraction=.35, seed=1)
main(blobs)

blobs = data.binary_blobs(50, blob_size_fraction=.2, volume_fraction=.35, seed=2)
main(blobs)

# blobs = data.binary_blobs(50, blob_size_fraction=.25,  volume_fraction=.35, seed=1)

# #orig info
# print("\nblobs\n")
# print(blobs)
# print(type(blobs))
# print(blobs.shape)

# main(blobs)
```

3 examples: