```python
import numpy as np
import cv2 as cv


img = cv.imread('img/F1.bmp',0)
# Apply thresholding to convert to binary
_, img = cv.threshold(img, 127, 255, cv.THRESH_BINARY)

se = np.array([
    [1,1,1,1,1],
    [1,1,1,1,1],
    [1,1,1,1,1],
    [1,1,1,1,1],
    [1,1,1,1,1]
    ])
# se = np.ones((5))

def Dilation(img , se):

    Dilation_img = np.zeros_like(img)

    r , c = img.shape
    k_r , k_c = se.shape

    for i in range(k_r//2, r-k_r//2):
        for j in range(k_c//2, c-k_c//2):

            image_patch = img[i-k_r//2 :(i+k_r//2)+1 , j-k_c//2 :(j+k_c//2)+1 ]


            apply_or = np.logical_and(image_patch , se)
            Dilation_img[i,j] = np.max(apply_or)


    return Dilation_img


def erosion(img , se):

    erosion_img = np.zeros_like(img)

    r , c = img.shape
    k_r , k_c = se.shape

    for i in range(k_r//2, r-k_r//2):
        for j in range(k_c//2, c-k_c//2):

            image_patch = img[i-k_r//2 :(i+k_r//2)+1 , j-k_c//2 :(j+k_c//2)+1 ]
            apply_and = np.logical_and(image_patch , se)
            erosion_img[i,j] = np.min(apply_and)


    return erosion_img

def Open(img,se):
    return Dilation(erosion(img, se), se)



cv.imshow('orgin',img )
cv.imshow('Dilation img',Open(img,se) * 255) # open-cv can not show binary image so I muliply 1s by 255 to make the pixsl white in gray
cv.waitKey(0)
```