import numpy as np  
import cv2  
import copy  
  
def sharpenFilter(img, imgS, ker, filS):  
 imgTempf = copy.deepcopy(img)  
 imgAfter = copy.deepcopy(img)  
 for i in range(imgS[0]): #use i as index of image to determine which pixels need to be filtered  
 for j in range(imgS[1]):  
 imgAfter = conv(imgTempf, imgAfter, imgS, ker, filS, i, j)  
  
 return imgAfter  
  
  
def conv(imgOrignal, imgNew, imgSS, ker, filSS, ii, jj): #imgOrignal - Original Image, imgNew - New Image, ker - filter  
 valFromFilter = [] #imgSS - imageSize, filSS - Filter Size, ii - X index, jj - Y index  
 for l in range(filSS[0]): #do the dot product by x direction  
 if ii + filSS[0] < imgSS[0]: #check the location whether will exceed the image size  
 for m in range(filSS[1]):  
 if jj + filSS[1] < imgSS[1]:  
 valFromFilter.append(imgOrignal[ii + l][jj + m] \* ker[l][m]) #Dot product of filter and image  
 else:  
 break  
 else:  
 break  
  
 valSum = sum(valFromFilter) #sum of all dot product  
 if ii + filSS[0] < imgSS[0] and jj + filSS[1] < imgSS[1]:  
 imgNew[ii + 1][jj + 1] = valSum #assign sum value to image by the location of filter center  
 valFromFilter.clear()  
 return imgNew  
  
  
image = cv2.imread('lenna\_rgb.png')  
imgTemp = copy.deepcopy(image)  
imgGray = cv2.cvtColor(imgTemp, cv2.COLOR\_BGR2GRAY)  
arr = np.array(imgGray)  
imgSize = arr.shape  
  
  
kernel = [[0, -1, 0], [-1, 5, -1], [0, -1, 0]]  
  
fil = np.array(kernel)  
filSize = fil.shape  
print('filter: ', kernel)  
print('filterSize: ', filSize)  
  
imgTemp2 = sharpenFilter(imgGray, imgSize, kernel, filSize)  
  
print('imSize: ', imgSize)  
print('end')  
  
cv2.imshow('Before', imgGray)  
cv2.imshow('After', imgTemp2)  
cv2.waitKey(0)