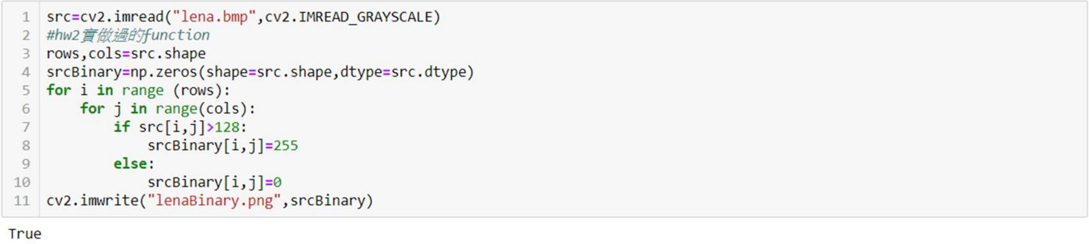
# Computer Vision HW#4

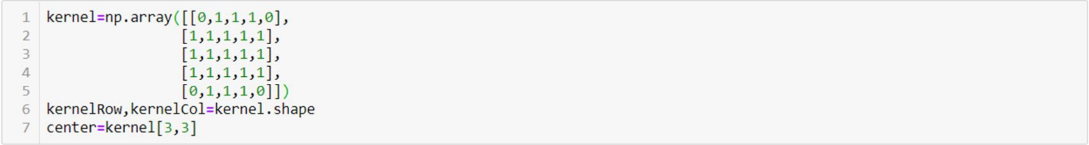
## R08942125 廖克允

1. Binarize Lena and use the 3-5-5-5-3 kernel



```
src=cv2.imread("lena.bmp",cv2.IMREAD_GRAYSCALE)
#hw2實做過的function
rows,cols=src.shape
srcBinary=np.zeros(shape=src.shape,dtype=src.dtype)
for i in range (rows):
    for j in range(cols):
        if src[i,j]>128:
            srcBinary[i,j]=255
        else:
            srcBinary[i,j]=0
cv2.imwrite("lenaBinary.png",srcBinary)
```

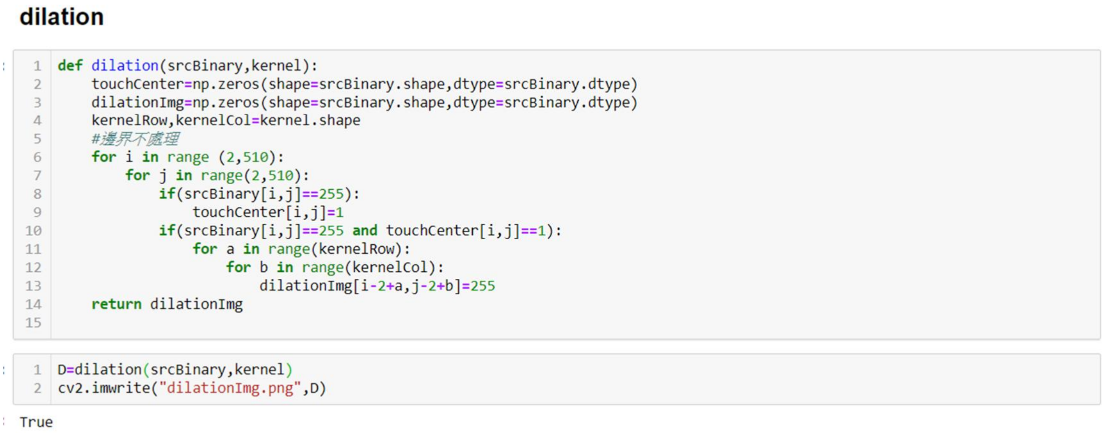
```
True
```



```
kernel=np.array([[0,1,1,1,0],
                [1,1,1,1,1],
                [1,1,1,1,1],
                [1,1,1,1,1],
                [0,1,1,1,0]])
kernelRow,kernelCol=kernel.shape
center=kernel[3,3]
```

2. Start to do morphology operation on image

(a) Dilation



### dilation

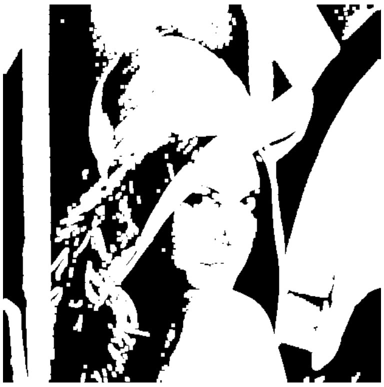
```
def dilation(srcBinary,kernel):
    touchCenter=np.zeros(shape=srcBinary.shape,dtype=srcBinary.dtype)
    dilationImg=np.zeros(shape=srcBinary.shape,dtype=srcBinary.dtype)
    kernelRow,kernelCol=kernel.shape
    #邊界不處理
    for i in range (2,510):
        for j in range(2,510):
            if(srcBinary[i,j]==255):
                touchCenter[i,j]=1
            if(srcBinary[i,j]==255 and touchCenter[i,j]==1):
                for a in range(kernelRow):
                    for b in range(kernelCol):
                        dilationImg[i-2+a,j-2+b]=255
    return dilationImg

```

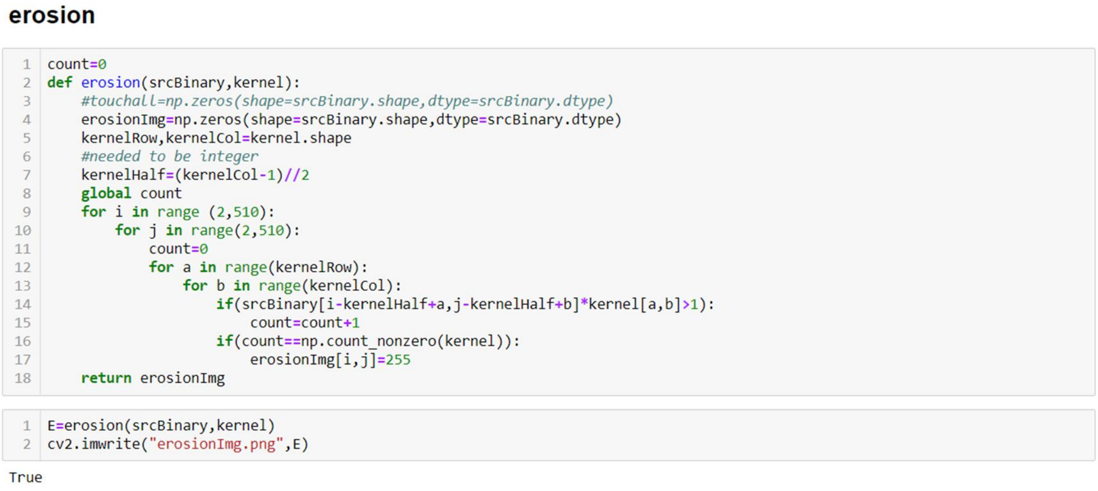
```
D=dilation(srcBinary,kernel)
cv2.imwrite("dilationImg.png",D)
```

```
True
```

Result



## (b) Erosion



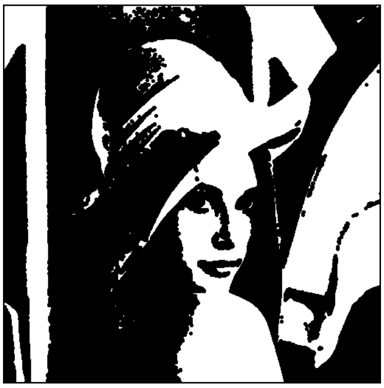
### erosion

```
count=0
def erosion(srcBinary,kernel):
    #touchall=np.zeros(shape=srcBinary.shape,dtype=srcBinary.dtype)
    erosionImg=np.zeros(shape=srcBinary.shape,dtype=srcBinary.dtype)
    kernelRow,kernelCol=kernel.shape
    #needed to be integer
    kernelHalf=(kernelCol-1)//2
    global count
    for i in range (2,510):
        for j in range(2,510):
            count=0
            for a in range(kernelRow):
                for b in range(kernelCol):
                    if(srcBinary[i-kernelHalf+a,j-kernelHalf+b]*kernel[a,b]>1):
                        count=count+1
                    if(count==np.count_nonzero(kernel)):
                        erosionImg[i,j]=255
    return erosionImg
```

```
E=erosion(srcBinary,kernel)
cv2.imwrite("erosionImg.png",E)
```

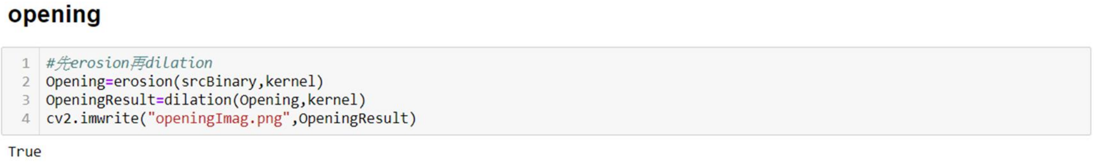
```
True
```

Result:



## (c) Opening

Opening 是先對圖片做 erosion 再做 dilation

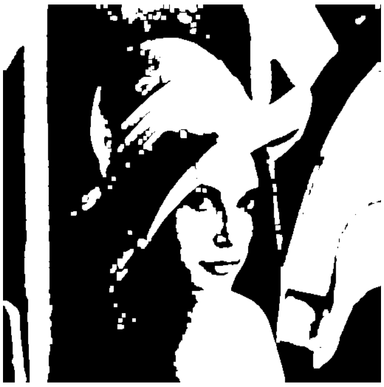


### opening

```
#先erosion再dilation
Opening=erosion(srcBinary,kernel)
OpeningResult=dilation(Opening,kernel)
cv2.imwrite("openingImag.png",OpeningResult)
```

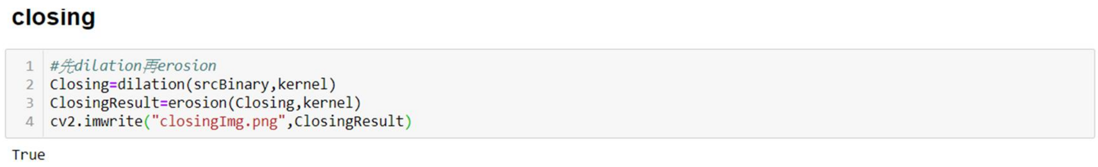
```
True
```

Result:



(d) Closing

Closing 是先對圖片做 dilation 再做 erosion

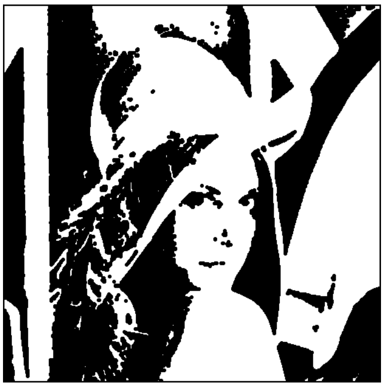


**closing**

```python
#先dilation再erosion
Closing=dilation(srcBinary,kernel)
ClosingResult=erosion(Closing,kernel)
cv2.imwrite("closingImg.png",ClosingResult)
```

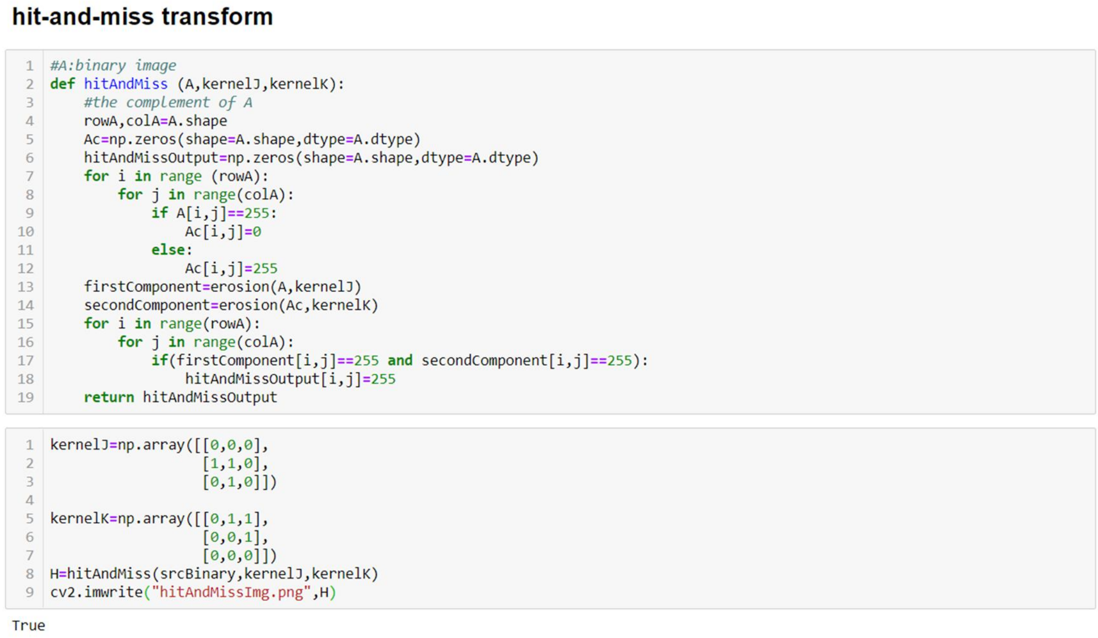
```
True
```

Result:



(e) Hit-and-miss transform

Hit-and-miss 的邏輯運算式: $A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$



**hit-and-miss transform**

```python
#A:binary image
def hitAndMiss (A,kernelJ,kernelK):
    #the complement of A
    rowA,colA=A.shape
    Ac=np.zeros(shape=A.shape,dtype=A.dtype)
    hitAndMissOutput=np.zeros(shape=A.shape,dtype=A.dtype)
    for i in range (rowA):
        for j in range(colA):
            if A[i,j]==255:
                Ac[i,j]=0
            else:
                Ac[i,j]=255
    firstComponent=erosion(A,kernelJ)
    secondComponent=erosion(Ac,kernelK)
    for i in range(rowA):
        for j in range(colA):
            if(firstComponent[i,j]==255 and secondComponent[i,j]==255):
                hitAndMissOutput[i,j]=255
    return hitAndMissOutput
```

```python
kernelJ=np.array([[0,0,0],
                  [1,1,0],
                  [0,1,0]])

kernelK=np.array([[0,1,1],
                  [0,0,1],
                  [0,0,0]])
H=hitAndMiss(srcBinary,kernelJ,kernelK)
cv2.imwrite("hitAndMissImg.png",H)
```

```
True
```

Result: