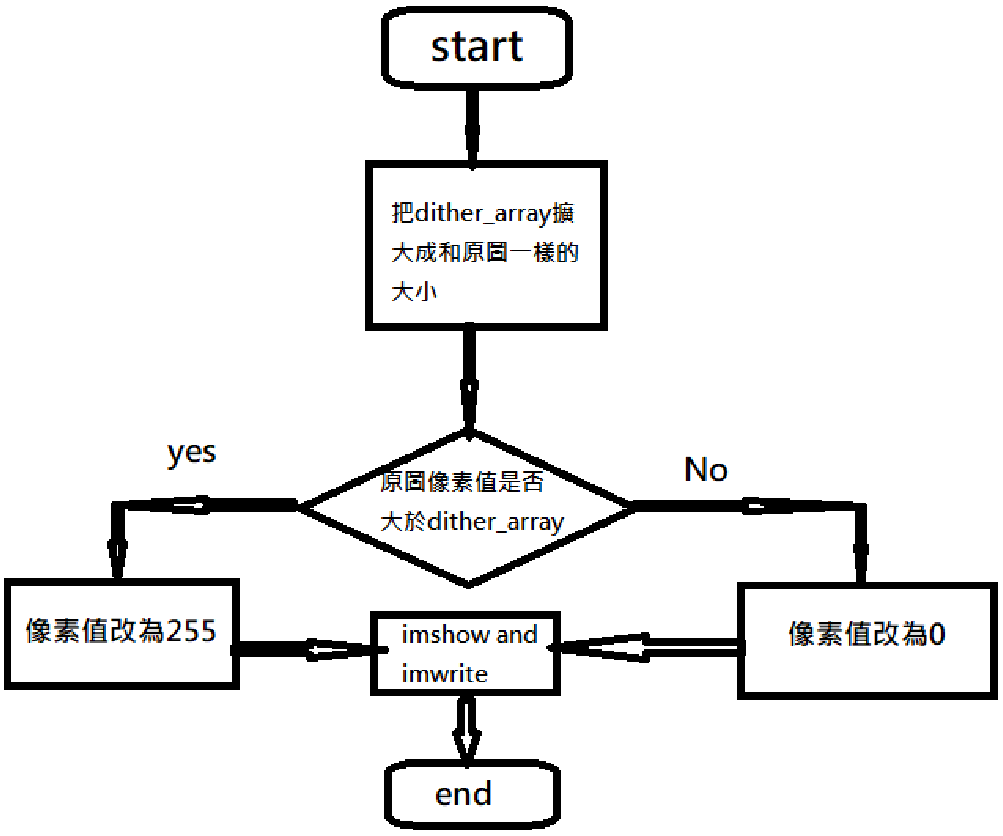
Lab2 影像混色實驗

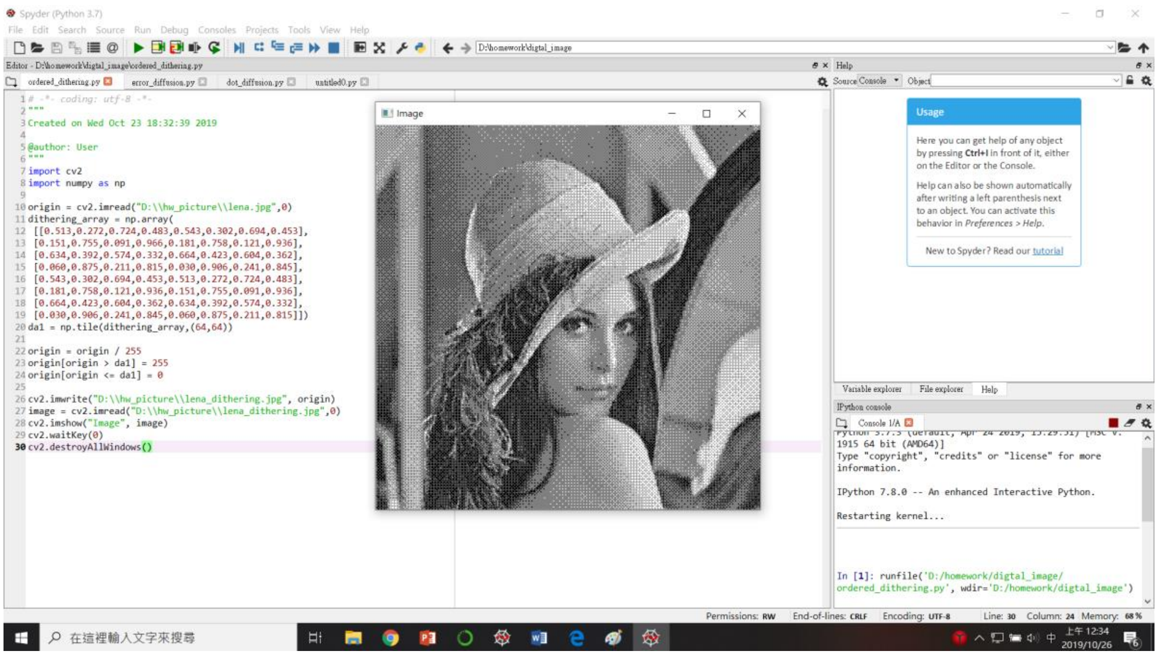
1. 有序抖動法(Ordered Dithering)


start
把dither_array擴大成和原圖一樣的大小
原圖像素值是否大於dither_array
yes
No
像素值改為255
像素值改為0
imshow and imwrite
end

程式碼:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 18:32:39 2019

@author: User
"""
import cv2
import numpy as np

origin = cv2.imread("D:\\hw_picture\\lena.jpg",0)
dithering_array = np.array(
  [[0.513,0.272,0.724,0.483,0.543,0.302,0.694,0.453],
  [0.151,0.755,0.091,0.966,0.181,0.758,0.121,0.936],
  [0.634,0.392,0.574,0.332,0.664,0.423,0.604,0.362],
  [0.060,0.875,0.211,0.815,0.030,0.906,0.241,0.845],
 [0.543,0.302,0.694,0.453,0.513,0.272,0.724,0.483],
 [0.181,0.758,0.121,0.936,0.151,0.755,0.091,0.936],
 [0.664,0.423,0.604,0.362,0.634,0.392,0.574,0.332],
 [0.030,0.906,0.241,0.845,0.060,0.875,0.211,0.815]])
da1 = np.tile(dithering_array,(64,64))

origin = origin / 255
origin[origin > da1] = 255
origin[origin <= da1] = 0

cv2.imwrite("D:\\hw_picture\\lena_dithering.jpg", origin)
image = cv2.imread("D:\\hw_picture\\lena_dithering.jpg",0)
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

執行結果:



```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 18:32:39 2019

@author: User
"""
import cv2
import numpy as np

origin = cv2.imread("D:\\hw_picture\\lena.jpg",0)
dithering_array = np.array(
 [[0.513,0.272,0.724,0.483,0.543,0.302,0.694,0.453],
 [0.151,0.755,0.091,0.966,0.181,0.758,0.121,0.936],
 [0.634,0.392,0.574,0.332,0.664,0.423,0.604,0.362],
 [0.060,0.875,0.211,0.815,0.030,0.906,0.241,0.845],
 [0.543,0.302,0.694,0.453,0.513,0.272,0.724,0.483],
 [0.181,0.758,0.121,0.936,0.151,0.755,0.091,0.936],
 [0.664,0.423,0.604,0.362,0.634,0.392,0.574,0.332],
 [0.030,0.906,0.241,0.845,0.060,0.875,0.211,0.815]])
da1 = np.tile(dithering_array,(64,64))

origin = origin / 255
origin[origin > da1] = 255
origin[origin <= da1] = 0

cv2.imwrite("D:\\hw_picture\\lena_dithering.jpg", origin)
image = cv2.imread("D:\\hw_picture\\lena_dithering.jpg",0)
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

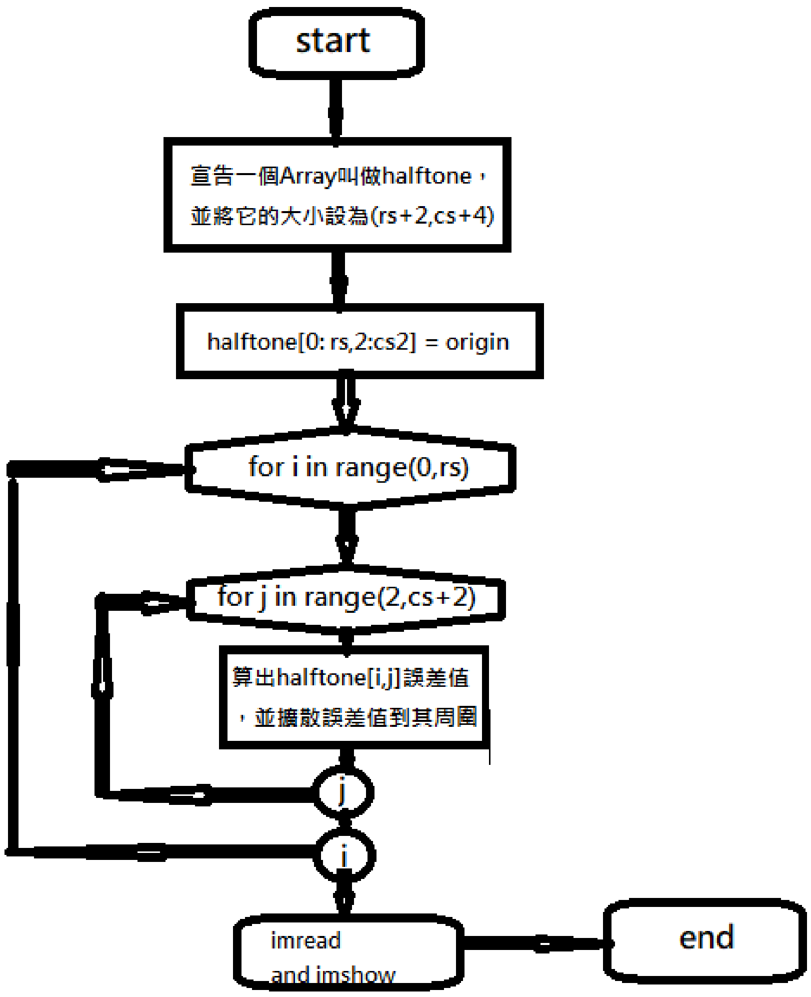
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our tutorial

```
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.
1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.8.0 -- An enhanced Interactive Python.

Restarting kernel...

In [1]: runfile('D:/homework/digtal_image/
ordered_dithering.py', wdir='D:/homework/digtal_image')
```

2.　錯誤擴散法(Error Diffusion)

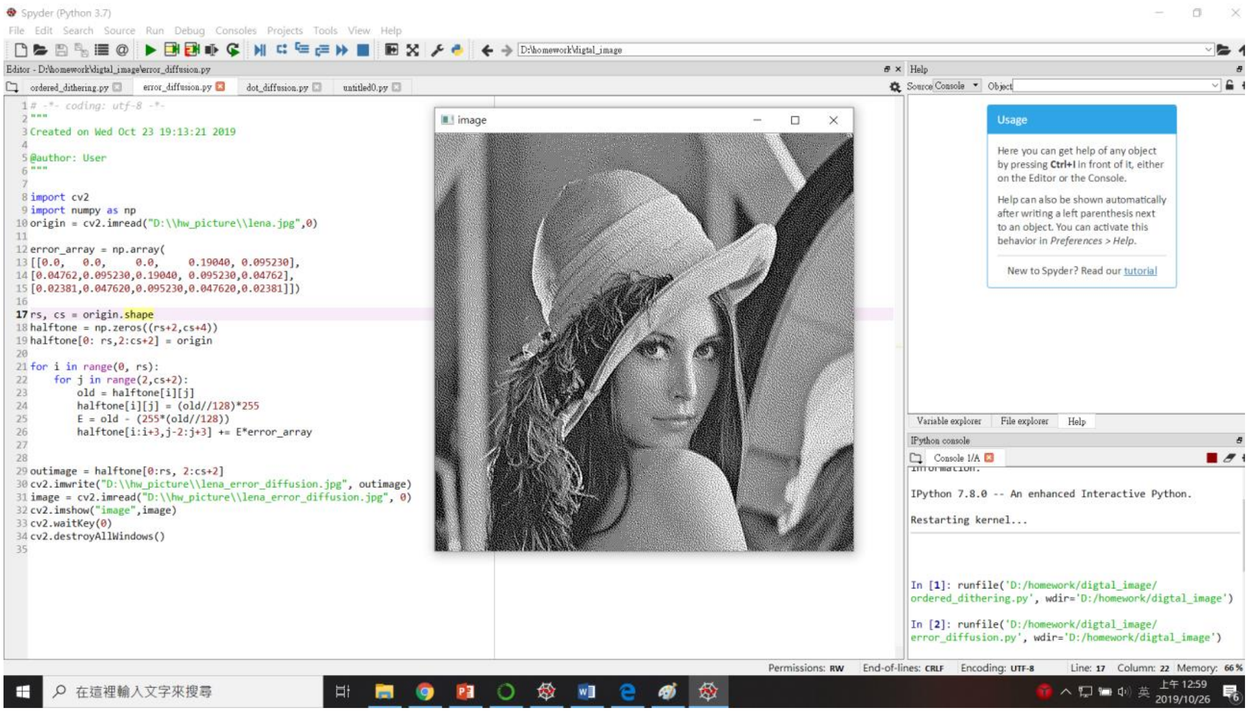


start
宣告一個Array叫做halftone，並將它的大小設為(rs+2,cs+4)
halftone[0: rs,2:cs2] = origin
for i in range(0,rs)
for j in range(2,cs+2)
算出halftone[i,j]誤差值，並擴散誤差值到其周圍
j
i
imread and imshow
end

程式碼:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 19:13:21 2019

@author: User
"""
import cv2
import numpy as np
origin = cv2.imread("D:\\hw_picture\\lena.jpg",0)

error_array = np.array(
[[0.0,      0.0,          0.0,          0.19040, 0.095230],
[0.04762,0.095230,0.19040, 0.095230,0.04762],
[0.02381,0.047620,0.095230,0.047620,0.02381]])

rs, cs = origin.shape
halftone = np.zeros((rs+2,cs+4))
halftone[0: rs,2:cs+2] = origin

for i in range(0, rs):
    for j in range(2,cs+2):
        old = halftone[i][j]
        halftone[i][j] = (old//128)*255
        E = old - (255*(old//128))
        halftone[i:i+3,j-2:j+3] += E*error_array

outimage = halftone[0:rs, 2:cs+2]
cv2.imwrite("D:\\hw_picture\\lena_error_diffusion.jpg", outimage)
image = cv2.imread("D:\\hw_picture\\lena_error_diffusion.jpg", 0)
cv2.imshow("image",image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

執行結果:



```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 19:13:21 2019

@author: User
"""

import cv2
import numpy as np
origin = cv2.imread("D:\\hw_picture\\lena.jpg",0)

error_array = np.array(
[[0.0,    0.0,      0.0,      0.19040, 0.095230],
[0.04762,0.095230,0.19040, 0.095230,0.04762],
[0.02381,0.047620,0.095230,0.047620,0.02381]])

rs, cs = origin.shape
halftone = np.zeros((rs+2,cs+4))
halftone[0: rs,2:cs+2] = origin

for i in range(0, rs):
    for j in range(2,cs+2):
        old = halftone[i][j]
        halftone[i][j] = (old//128)*255
        E = old - (255*(old//128))
        halftone[i:i+3,j-2:j+3] += E*error_array


outimage = halftone[0:rs, 2:cs+2]
cv2.imwrite("D:\\hw_picture\\lena_error_diffusion.jpg", outimage)
image = cv2.imread("D:\\hw_picture\\lena_error_diffusion.jpg", 0)
cv2.imshow("image",image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

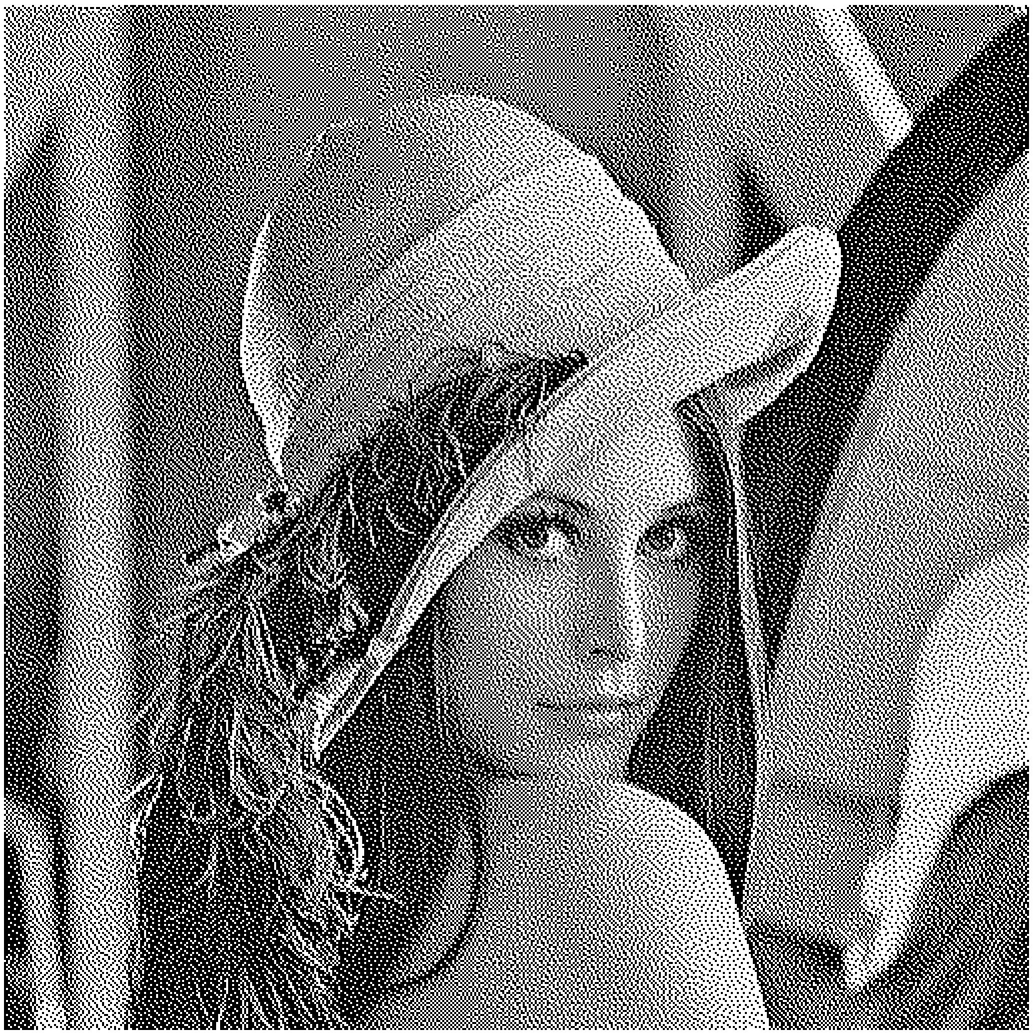
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
IPython 7.8.0 -- An enhanced Interactive Python.

Restarting kernel...



In [1]: runfile('D:/homework/digtal_image/
ordered_dithering.py', wdir='D:/homework/digtal_image')

In [2]: runfile('D:/homework/digtal_image/
error_diffusion.py', wdir='D:/homework/digtal_image')
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 17 Column: 22 Memory: 66 %

在這裡輸入文字來搜尋 上午 12:59 2019/10/26



3 點擴散法(Dot Diffusion)

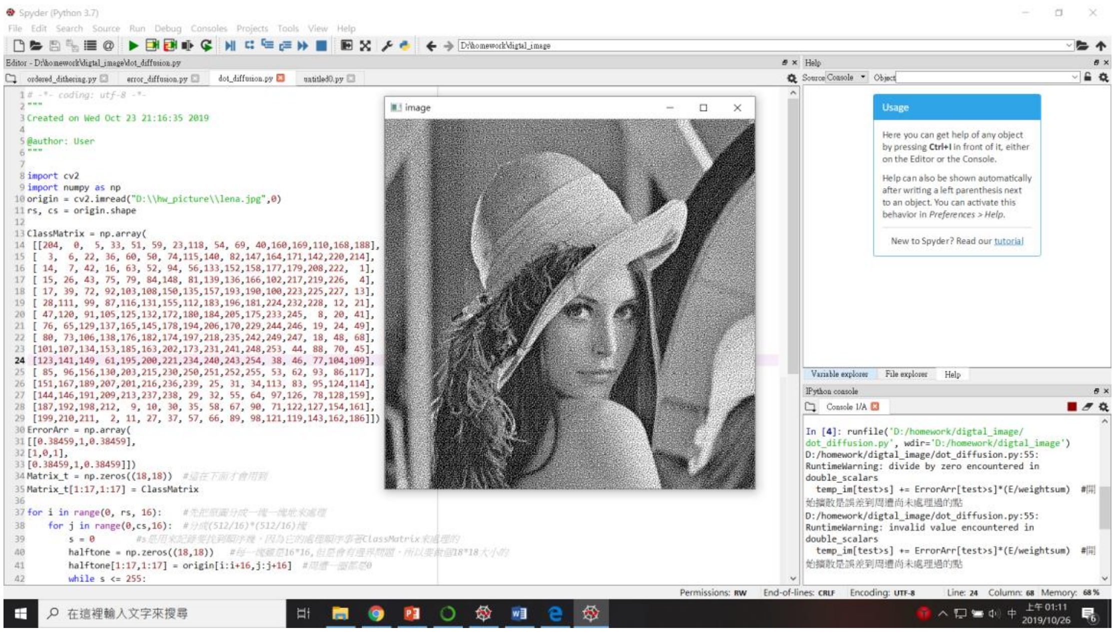
程式碼:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 21:16:35 2019

@author: User
"""
import cv2
import numpy as np
```

```
origin = cv2.imread("D:\\hw_picture\\lena.jpg",0)
rs, cs = origin.shape

ClassMatrix = np.array(
 [[204,   0,   5, 33, 51, 59, 23,118, 54, 69, 40,160,169,110,168,188],
 [   3,   6, 22, 36, 60, 50, 74,115,140, 82,147,164,171,142,220,214],
 [ 14,   7, 42, 16, 63, 52, 94, 56,133,152,158,177,179,208,222,   1],
 [ 15, 26, 43, 75, 79, 84,148, 81,139,136,166,102,217,219,226,   4],
 [ 17, 39, 72, 92,103,108,150,135,157,193,190,100,223,225,227, 13],
 [ 28,111, 99, 87,116,131,155,112,183,196,181,224,232,228, 12, 21],
 [ 47,120, 91,105,125,132,172,180,184,205,175,233,245,   8, 20, 41],
 [ 76, 65,129,137,165,145,178,194,206,170,229,244,246, 19, 24, 49],
 [ 80, 73,106,138,176,182,174,197,218,235,242,249,247, 18, 48, 68],
 [101,107,134,153,185,163,202,173,231,241,248,253, 44, 88, 70, 45],
 [123,141,149, 61,195,200,221,234,240,243,254, 38, 46, 77,104,109],
 [ 85, 96,156,130,203,215,230,250,251,252,255, 53, 62, 93, 86,117],
 [151,167,189,207,201,216,236,239, 25, 31, 34,113, 83, 95,124,114],
 [144,146,191,209,213,237,238, 29, 32, 55, 64, 97,126, 78,128,159],
 [187,192,198,212,   9, 10, 30, 35, 58, 67, 90, 71,122,127,154,161],
 [199,210,211,   2, 11, 27, 37, 57, 66, 89, 98,121,119,143,162,186]])
ErrorArr = np.array(
[[0.38459,1,0.38459],
[1,0,1],
[0.38459,1,0.38459]])
Matrix_t = np.zeros((18,18))   #這在下面才會用到
Matrix_t[1:17,1:17] = ClassMatrix


for i in range(0, rs, 16):      #先把原圖分成一塊一塊地來處理
    for j in range(0,cs,16):   #分成(512/16)*(512/16)塊
        s = 0       #s 是用來記錄要找到順序幾，因為它的處理順序按著 ClassMatrix 來處理的
        halftone = np.zeros((18,18)) #每一塊雖是 16*16,但是會有邊界問題，所以要做個 18*18 大小的
        halftone[1:17,1:17] = origin[i:i+16,j:j+16]   #周遭一圈都是 0
        while s <= 255:
            a,b = np.where(ClassMatrix == s) #np.where 是用來尋找 s 在 classMatrix 的哪裡
            a = int(a)      #在第幾 row。
            b = int(b)      #在第幾 colum， 但是因為 ClassMatrix 大小是 16*16，而要處理的是 18*18
            old =   halftone[1+a][1+b] #所以 a,b 要加 1 才是真正的位置
```

```
                E = old - (255*(old//128))    #算誤差值
                temp_im = np.zeros((3,3))      #temp_im 是個 3*3array
                temp_im[0:3,0:3] = halftone[a:a+3,b:b+3] #它是以要處理的那點為中心的 9 個像素
                test = np.zeros((3,3))
                test = Matrix_t[a:a+3,b:b+3]    #上面有先定義 Matrix_t 了

                ave = ErrorArr[test > s]    #test 是用來看該點周遭有哪個點已經先處理過了
                weightsum =ave.sum()     #將該點周遭尚未處理過的像素值都加起來
                temp_im[test>s] += ErrorArr[test>s]*(E/weightsum)
                                                        #開始擴散誤差到周遭尚未處理過的點
                temp_im[temp_im < 0] = 0;
                temp_im[temp_im > 255] = 255;
                halftone[a:a+3,b:b+3] = temp_im[0:3,0:3]       #把 temp_im 貼回 halftone 去
                halftone[1+a][1+b] = (old//128)*255 #若該點像素值>=128 則為 255,反之則為 0
                s += 1       #處理完該點，繼續處理下一個點
            origin[i:i+16,j:j+16] = halftone[1:17,1:17] #當這一塊處理完後貼回原圖，再去處理下一塊

cv2.imwrite("D:\\hw_picture\\lena_dot_diffusion.jpg",origin)
cv2.imshow("image", origin)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

執行結果:



```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 21:16:35 2019

@author: User
"""

import cv2
import numpy as np
origin = cv2.imread("D:\\hw_picture\\lena.jpg",0)
rs, cs = origin.shape

ClassMatrix = np.array(
 [[204,  0,  5, 33, 51, 59, 23,118, 54, 69, 40,160,169,110,168,188],
 [  3,  6, 22, 36, 60, 50, 74,115,140, 82,147,164,171,142,220,214],
 [ 14,  7, 42, 16, 63, 52, 94, 56,133,152,158,177,179,208,222,  1],
 [ 15, 26, 43, 75, 79, 84,148, 81,139,136,166,102,217,219,226,  4],
 [ 17, 39, 72, 92,103,108,150,135,157,193,190,100,223,225,227, 13],
 [ 28,111, 99, 87,116,131,155,112,183,196,181,224,232,228, 12, 21],
 [ 47,120, 91,105,125,132,172,180,184,205,175,233,245,  8, 20, 41],
 [ 76, 65,129,137,165,145,178,194,206,170,229,244,246, 19, 24, 49],
 [ 80, 73,106,138,176,182,174,197,218,235,242,249,247, 18, 48, 68],
 [101,107,134,153,185,163,202,173,231,241,248,253, 44, 88, 70, 45],
 [123,141,149, 61,195,200,221,234,240,243,254, 38, 46, 77,104,109],
 [ 85, 96,156,130,203,215,230,250,251,252,255, 53, 62, 93, 86,117],
 [151,167,189,207,201,216,236,239, 25, 31, 34,113, 83, 95,124,114],
 [144,146,191,209,213,237,238, 29, 32, 55, 64, 97,126, 78,128,159],
 [187,192,198,212,  9, 10, 30, 35, 58, 67, 90, 71,122,127,154,161],
 [199,210,211,  2, 11, 27, 37, 57, 66, 89, 98,121,119,143,162,186]])
ErrorArr = np.array(
[[0.38459,1,0.38459],
[1,0,1],
[0.38459,1,0.38459]])
Matrix_t = np.zeros((18,18))  #這在下面才會用到
Matrix_t[1:17,1:17] = ClassMatrix

for i in range(0, rs, 16):      #先把原圖分成一塊一塊地來處理
    for j in range(0,cs,16):  #分成(512/16)*(512/16)塊
        s = 0          #s是用來記錄要找到順序幾，因為它的處理順序要看ClassMatrix來處理的
        halftone = np.zeros((18,18))   #每一塊雖是16*16,但是會有邊界問題，所以要做個18*18大小的
        halftone[1:17,1:17] = origin[i:i+16,j:j+16]  #周遭一圈都是0
        while s <= 255:
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

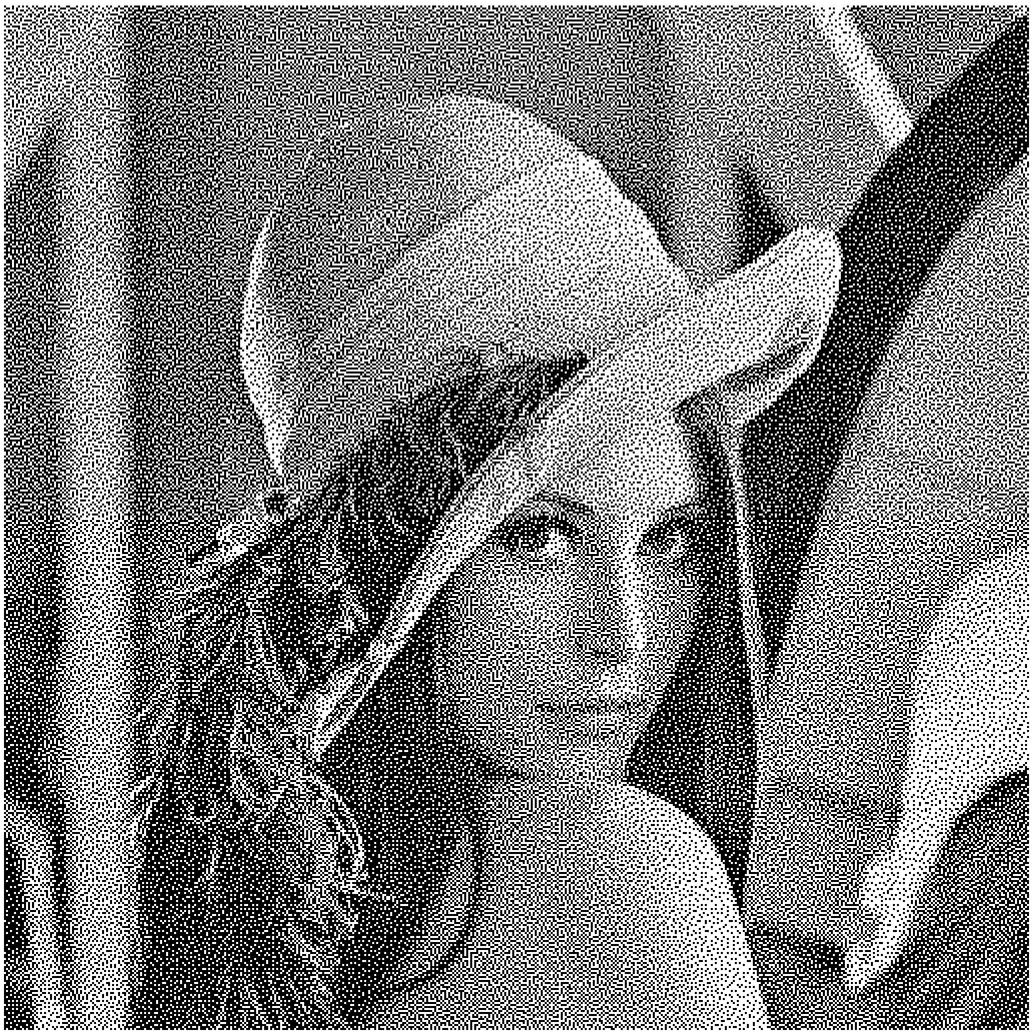
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [4]: runfile('D:/homework/digtal_image/
dot_diffusion.py', wdir='D:/homework/digtal_image')
D:/homework/digtal_image/dot_diffusion.py:55:
RuntimeWarning: divide by zero encountered in
double_scalars
  temp_im[test>s] += ErrorArr[test>s]*(E/weightsum)  #開
始擴散誤差到周遭尚未處理過的點
D:/homework/digtal_image/dot_diffusion.py:55:
RuntimeWarning: invalid value encountered in
double_scalars
  temp_im[test>s] += ErrorArr[test>s]*(E/weightsum)  #開
始擴散誤差到周遭尚未處理過的點
```

Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 24   Column: 68   Memory: 68 %

上午 01:11 2019/10/26



心得:

第一題，很快就寫完了，沒有遇到什麼問題，第二題當我把演算法看仔細後也沒麼問題，唯一的問題是是，想說有沒有不用雙重 for loop 的做法，但後來我看到課本也用雙層 for loop。

第三題我覺得是最難的，雖然當我把 Dot Diffusion 看懂後，大致上知道要怎麼寫了，但是當我真的在寫的時候還是有點卡，最後寫了 3 個多小時才寫完。

途中遇到了一些問題，像是很常發現有兩個 array 的大小不一樣也沒成比例，所以 numpy 沒辦法 boardcasting 也沒辦法運算；或是 run 的時候出現 divide by 0 的警告，但仔細想後發現因為我的寫法導致我在處理順序為 204,186,254 的點時會發生這種狀況是正常的。

　　最後我也發現，python 中，有 mutable 和 immutable 的關係，所以當我在做 什麼 = 什麼 的時候要想清楚，它是真的複製還是不是。