

Python影像處理：基本篇

使用pip安裝Python Library

執行CMD→輸入“ pip install 欲安裝的library名稱”

練習:

pip install numpy

pip install scipy

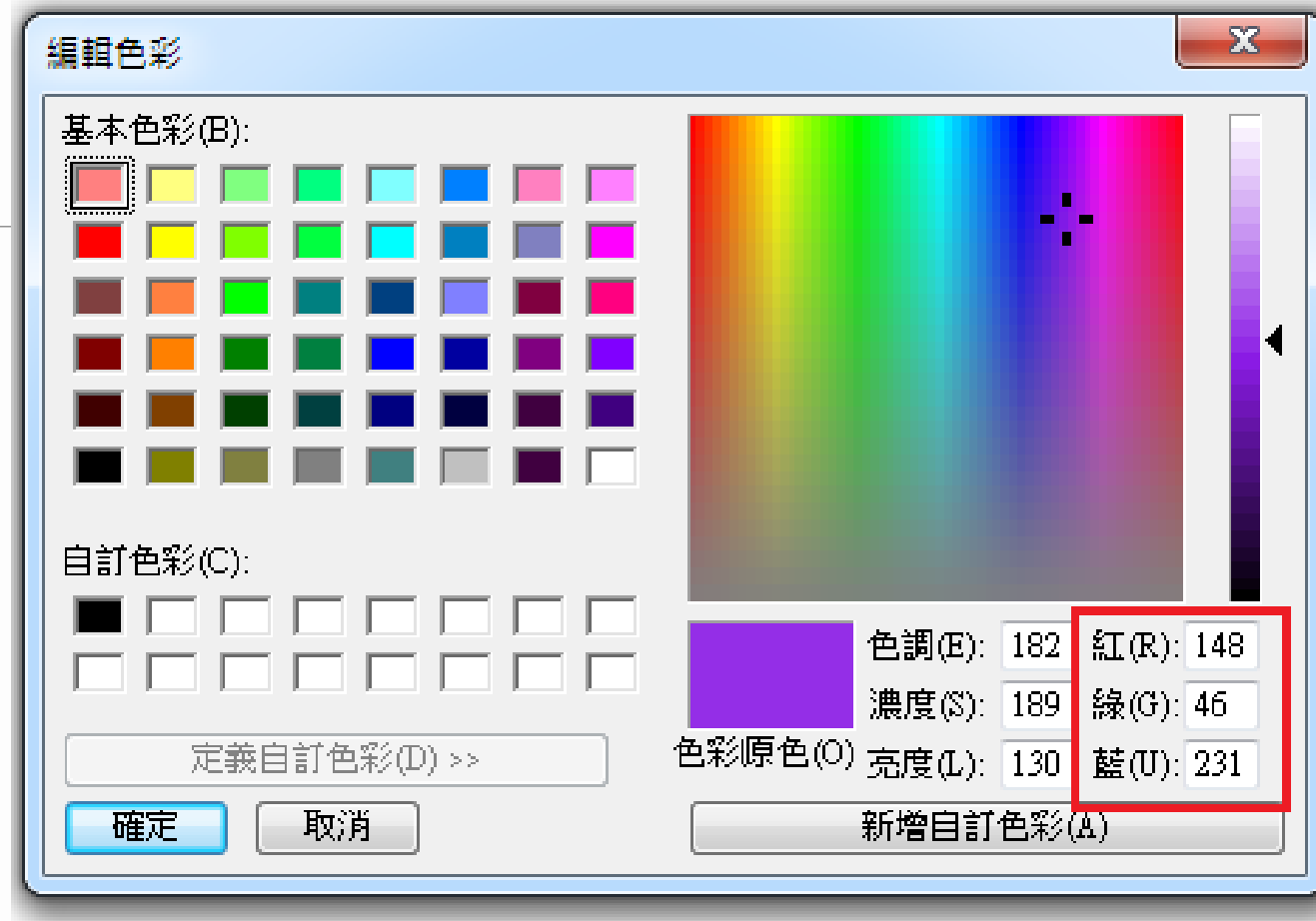
pip install opencv-python

pip install pillow

影像數位化

如同所有類型的訊號，需要在電腦裡處理的影像必須經過數位化。一般經過數位相機等電子產品拍攝出來的影像即為數位化完成的成品了，但若是傳統底片產生的相片則不是。

經過數位化的影像可以拆解成許多「像素」，像素是一個影像裡最小的面積單位。每個像素都有一組編碼來代表它呈現的顏色，常見的影像格式以紅綠藍三色組成，每個顏色分別用8位元紀錄，代表0~255的值，一個像素總共需要24位元來表示。



在「小畫家」的調色工具中
使用的就是紅綠藍三色各0~255的24位元格式。

影像處理

數位化的影像儲存在電腦裡可以看作一個矩陣。比如一個像素為 1920×1080 的影像，由紅綠藍三色表示的話，就是一個 $1920 \times 1080 \times 3$ 的矩陣，矩陣中的每一個元素代表一個畫素的紅綠藍其中一個色調的值，其值在 $0 \sim 255$ 之間。

當影像用矩陣來表示時，我們可以把它拿來進行一些數學運算來改變矩陣中的值，進而調整影像內容，這就是影像處理的基本概念。

範例一：匯入影像

匯入Lenna.jpg

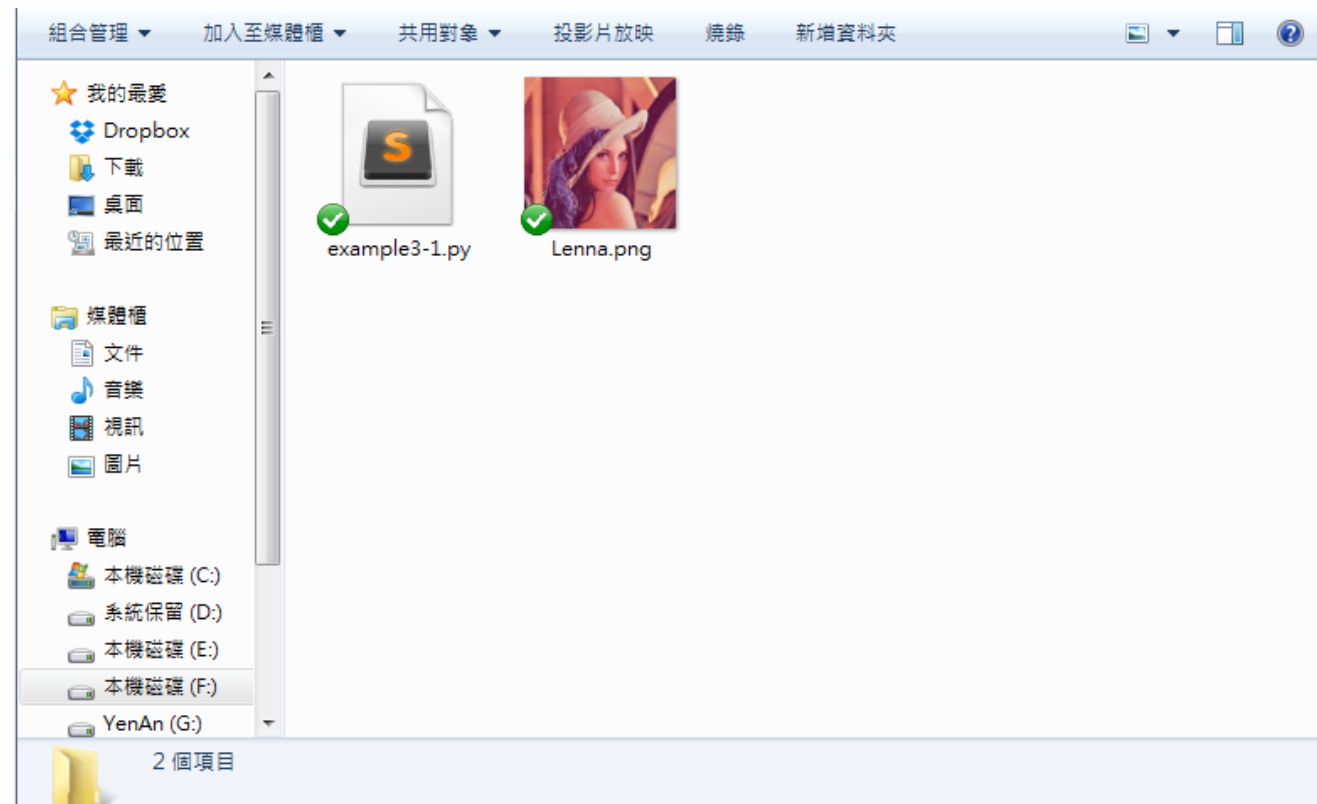


Lenna 圖歷史:

<https://zh.wikipedia.org/wiki/%E8%90%8A%E5%A8%9C%E5%9C%96>

範例一：匯入影像

打開資料夾位置



範例一：匯入影像

選擇Sublime Text，如果沒有，選擇其他應用程式



範例一：匯入影像

尋找此電腦上的其他應用程式，到C槽下選取Sublime text

您要如何開啟此檔案？



PotPlayer 播放專用 (64 位元)



Windows Media Player



Word 2016



WordPad



小畫家



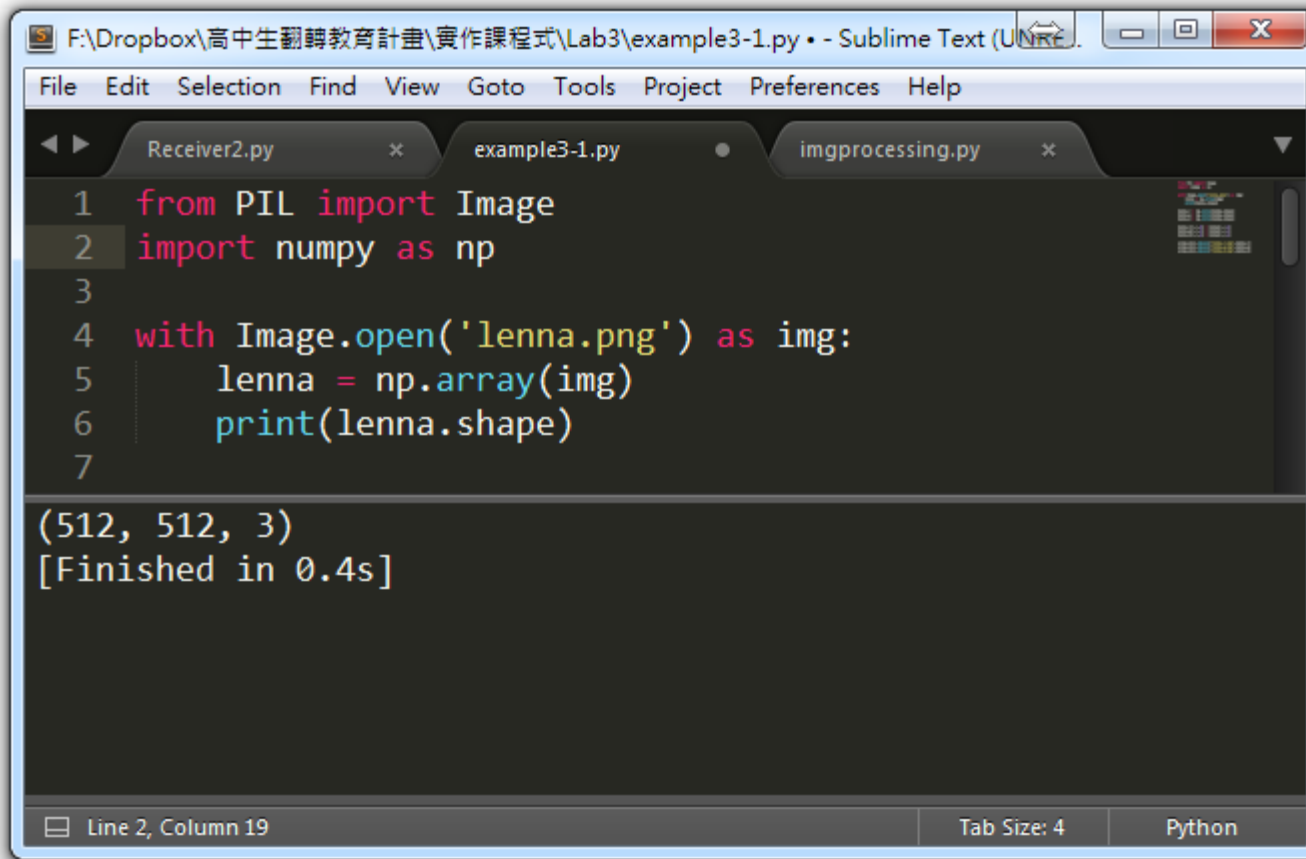
記事本

尋找此電腦上的其他應用程式



一律使用此應用程式來開啟 .py 檔案

範例一：匯入影像



```
F:\Dropbox\高中生翻轉教育計畫\實作課程\Lab3\example3-1.py • - Sublime Text (UNTITLED)
File Edit Selection Find View Goto Tools Project Preferences Help
Receiver2.py example3-1.py imgprocessing.py
1 from PIL import Image
2 import numpy as np
3
4 with Image.open('lenna.png') as img:
5     lenna = np.array(img)
6     print(lenna.shape)
7
(512, 512, 3)
[Finished in 0.4s]
Line 2, Column 19 Tab Size: 4 Python
```

Line1: 匯入PIL library中的Image函數。

Line2: 匯入numpy，並以“np”代稱。

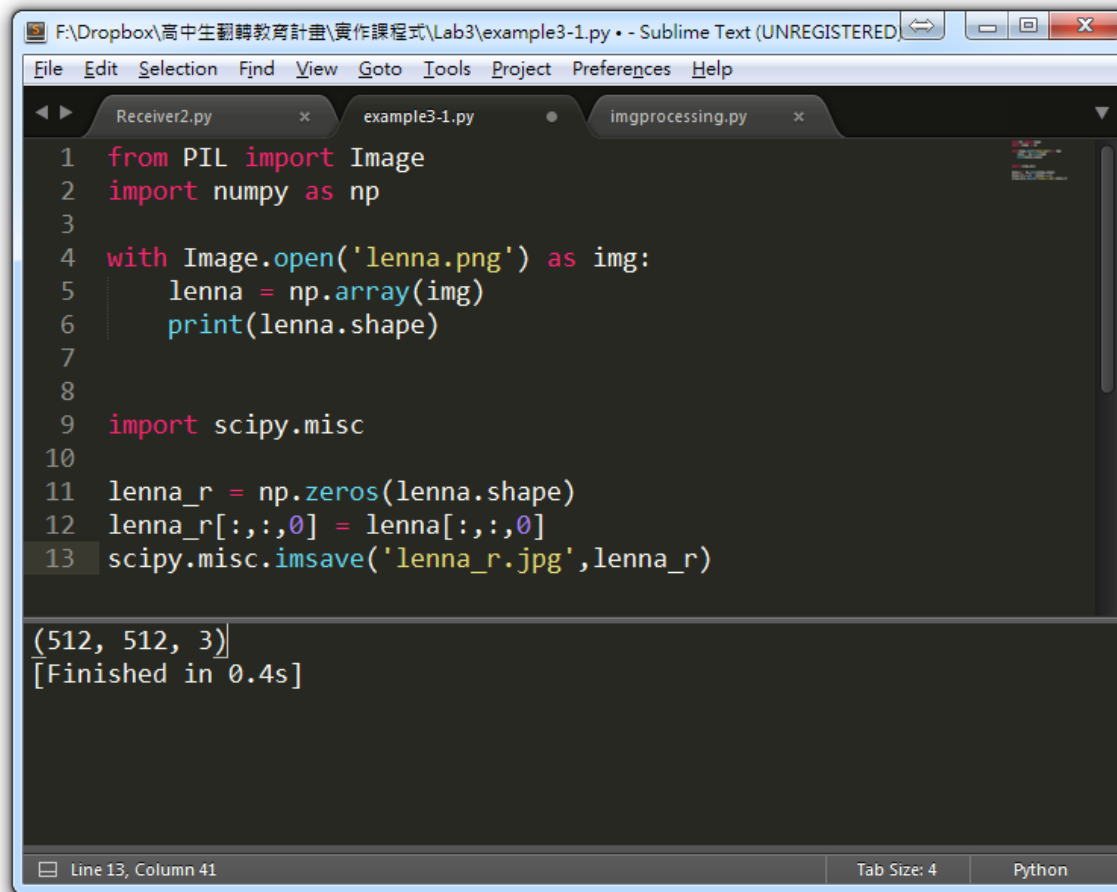
Line4: 使用Image.open()匯入影像至程式。

Line5: 將影像轉換成numpy array。

Line6: 顯示此numpy array的大小。

影像大小為512x512x3(RGB三色)

範例二：匯出RGB三色(接續前頁)



```
F:\Dropbox\高中生物轉教育計畫\實作課程\Lab3\example3-1.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Receiver2.py x example3-1.py imgprocessing.py x
1 from PIL import Image
2 import numpy as np
3
4 with Image.open('lenna.png') as img:
5     lenna = np.array(img)
6     print(lenna.shape)
7
8
9 import scipy.misc
10
11 lenna_r = np.zeros(lenna.shape)
12 lenna_r[:, :, 0] = lenna[:, :, 0]
13 scipy.misc.imsave('lenna_r.jpg', lenna_r)

(512, 512, 3)
[Finished in 0.4s]

Line 13, Column 41 Tab Size: 4 Python
```

Line 9: 匯入scipy librar中的misc，需使用pip install scipy安裝。

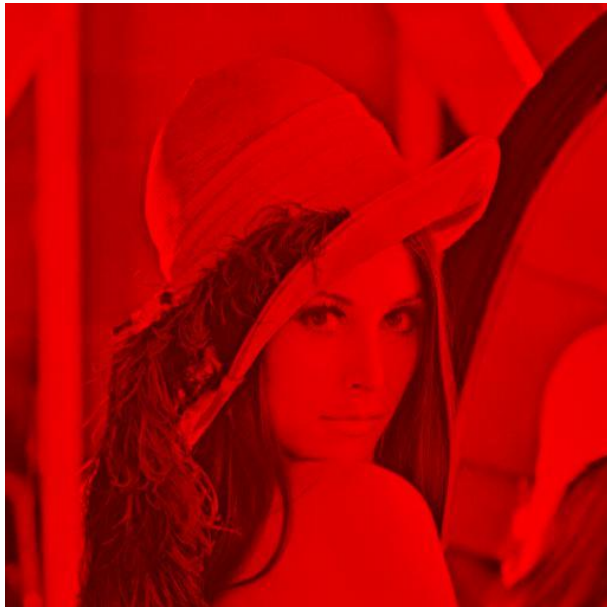
Line 11: 建立一個大小和原圖相同的空白numpy array。

Line 12: 將原圖紅色的部分複製至新的array。

Line 13: 將原圖紅色的部分儲存為" lenna_r.jpg"

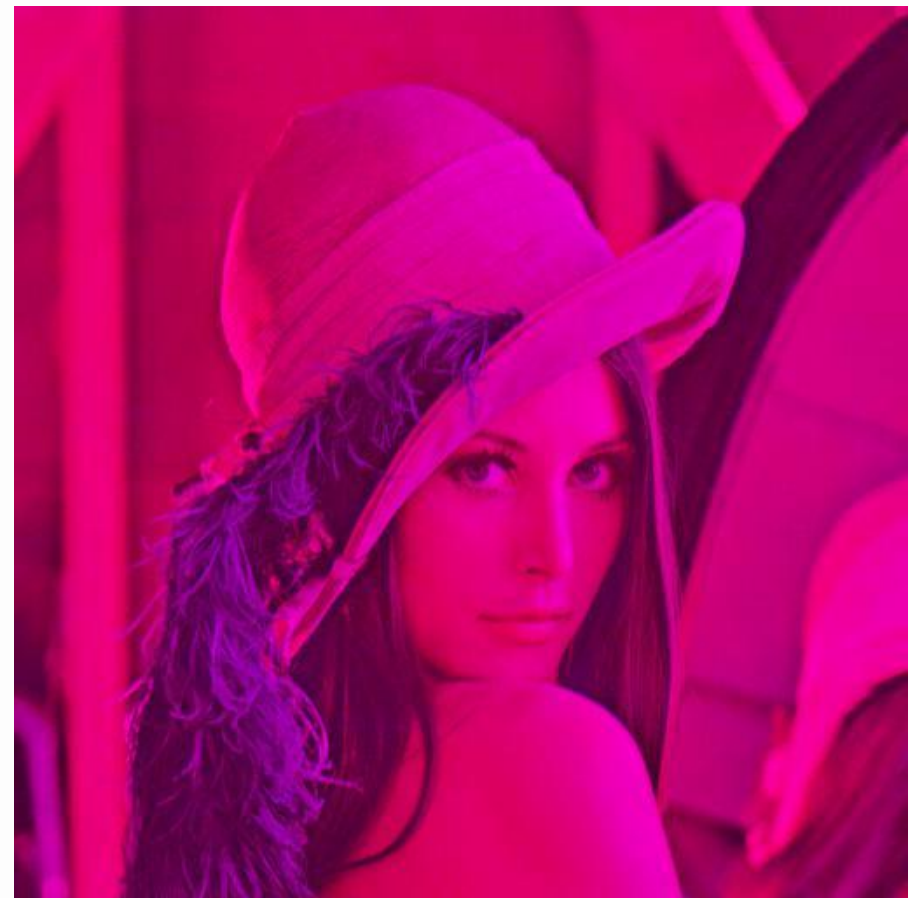
綠色和藍色的部分可循相同步驟完成。

匯出影像結果



練習一：合成兩顏色

將範例二紅藍兩色的圖片疊圖

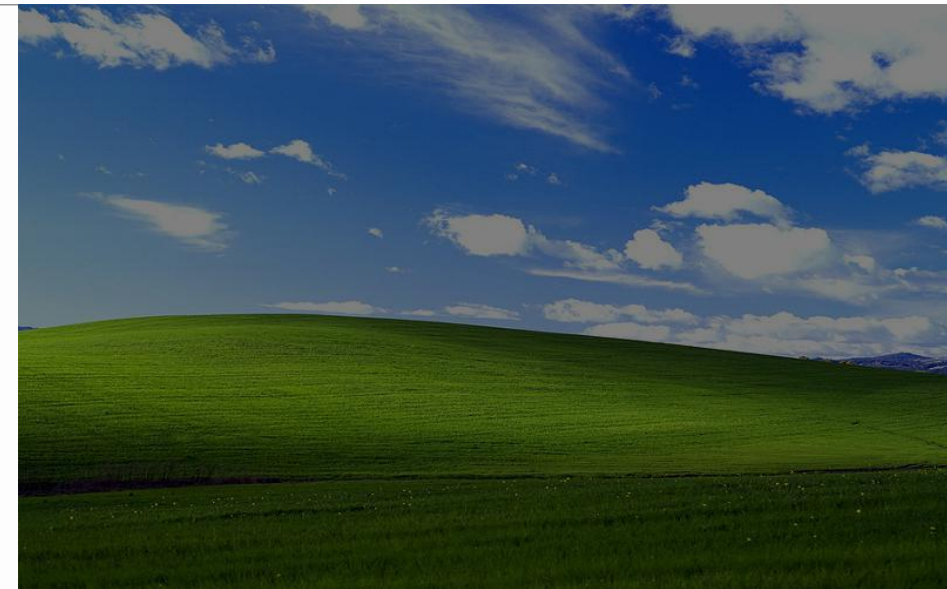


練習二：簡易的色調處理

改變色調與灰階~

以下示範：

簡易影像處理——變暗



左圖是將右圖中的每個像素的紅綠藍值都除以二的結果，平均的調低可以保留原本的色調，但使整體亮度都降低。

簡易影像處理—黑白化



將一個像素中的紅綠藍按照比例加總成單一數值可以產生如右圖的灰階影像，加成的比例和人眼對於色彩亮度的感知有關，並不是各佔三分之一。紅綠藍的比例大約為0.299:0.587:0.114，三者總和為1，所以灰階影像的數值一樣是介於0~255之間。

練習二：簡易的色調處理

1. 由RGB分別的影像看起來，這張圖片的紅色調比重較高。試著將整張圖的紅色數值除以2並輸出成一張新的jpg圖，並比較和原圖的差異。
2. 將原圖的RGB以0.299:0.587:0.144的比例放入紅藍綠三個塗層，轉換成灰階影像並輸出。

色調修改前/後比較



彩色/灰階影像

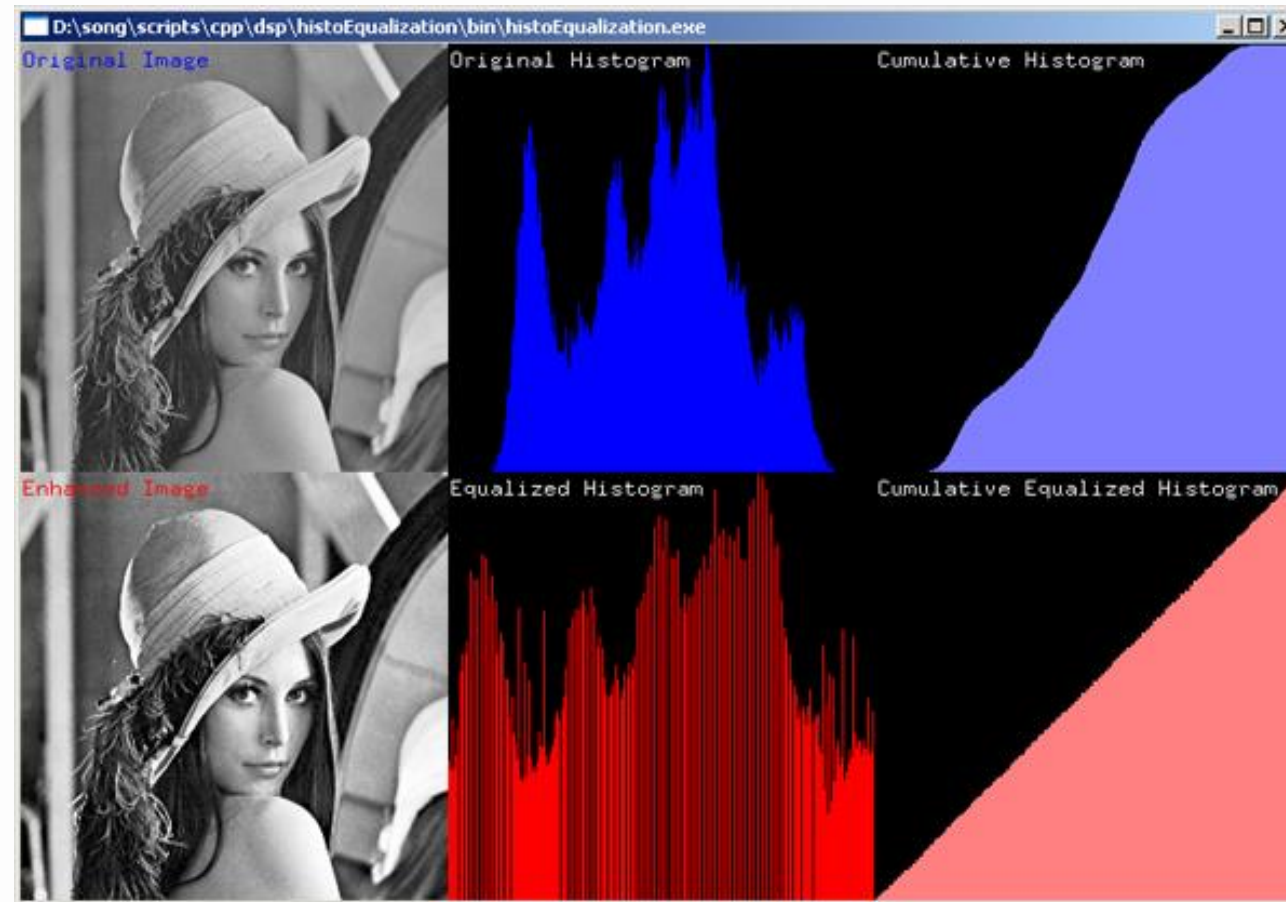


直方圖等化

直方圖等化是用來修正對比度的影像處理演算法，套用在對比度過低的影像可以使畫面變得更鮮明。其原理是將畫面中的所有像素的值做排序，灰階影像可以直接排序，彩色影像則可用RGB或YUV分別排序。YUV是另一種表示影像的系統，分別代表明亮度、色度、濃度，使用此系統的好處是處理完的結果在人眼看來會比較和諧。

按照排序的結果將每個像素的值重新分配，使得每個畫素的值都可以按照原排序平均分佈在最小值到最大值之間。以8位元灰階影像為例，處理完的影像中從亮度0~255之間的256個值都佔有同樣的像素數量。

直方圖等化



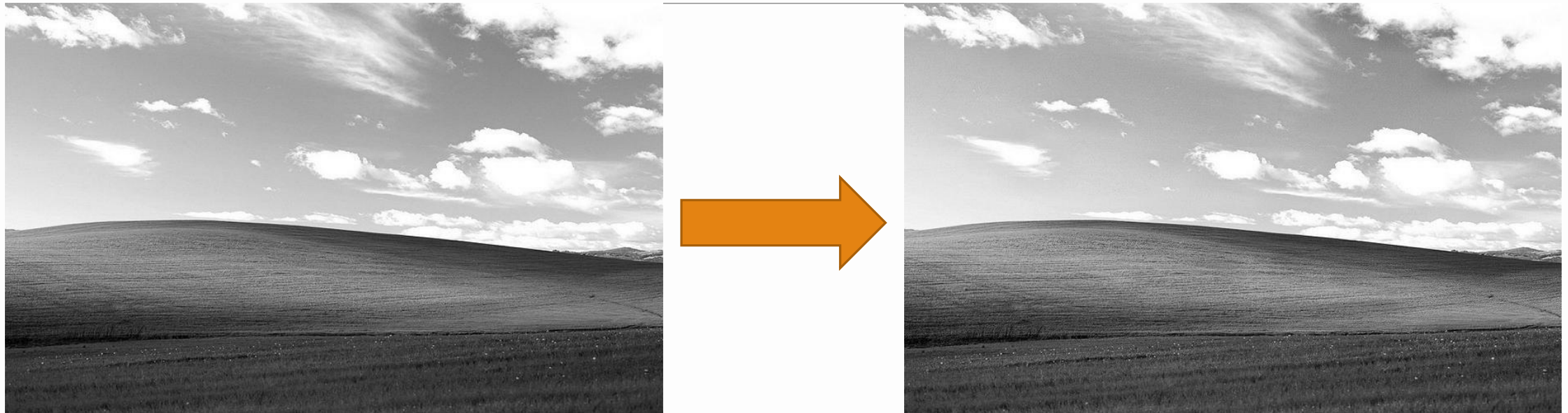
直方圖等化



局部直方圖等化

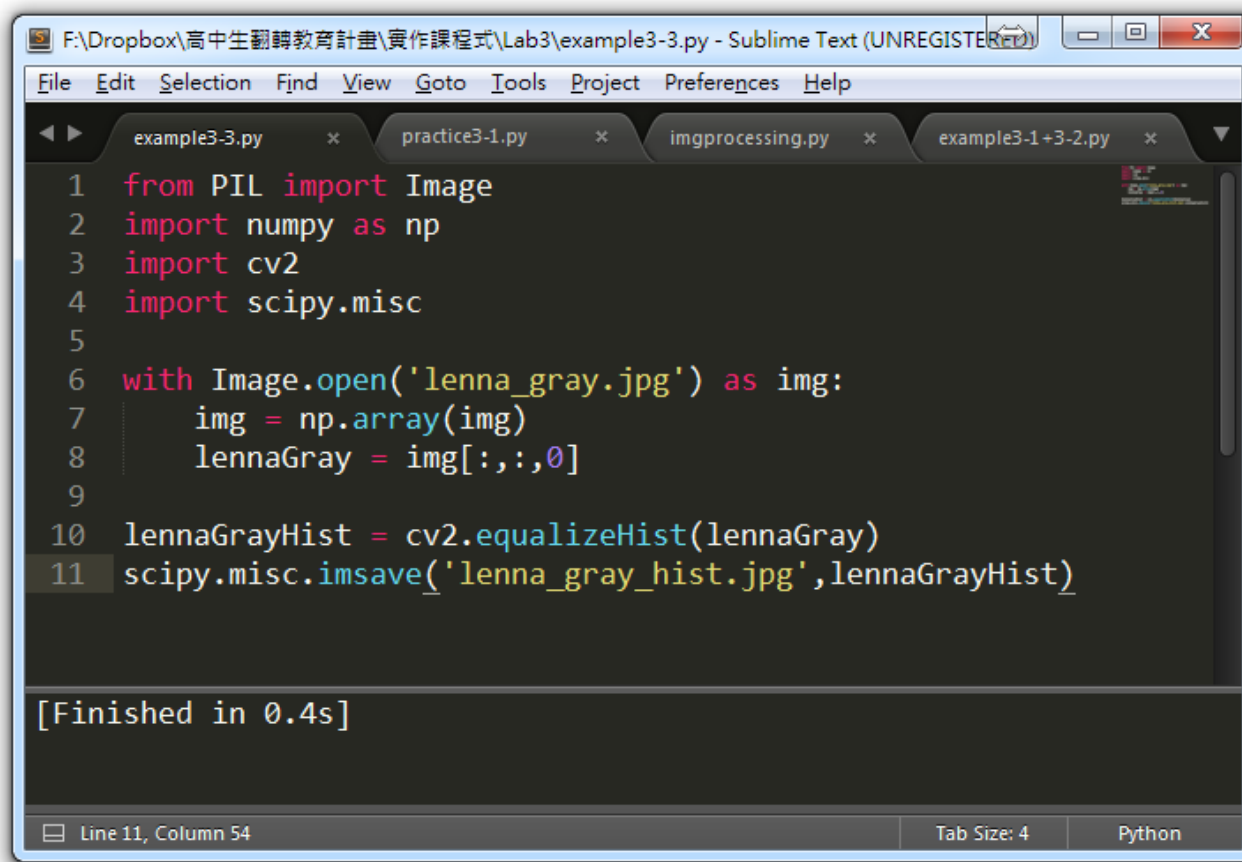
由上例可以看出來在大範圍深色區域與大範圍淺色區域同時存在圖像中時，直方圖等化的效果並不太好。這時可以應用局部直方圖等化來解決這個問題，局部直方圖等化的原理和直方圖等化相同，只是取的範圍較小，整張圖像會分好幾次計算，每次都分別調整小範圍的值，這樣就能在大範圍深色或淺色區域達到局部的修正。

局部直方圖等化



使用局部直方圖等化的影像更容易看出局部的細節。

範例三：套用OpenCV進行直方圖等化



```
F:\Dropbox\高中生翻轉教育計畫\實作課程式\Lab3\example3-3.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
example3-3.py x practice3-1.py x imgprocessing.py x example3-1+3-2.py x
1 from PIL import Image
2 import numpy as np
3 import cv2
4 import scipy.misc
5
6 with Image.open('lenna_gray.jpg') as img:
7     img = np.array(img)
8     lennaGray = img[:, :, 0]
9
10 lennaGrayHist = cv2.equalizeHist(lennaGray)
11 scipy.misc.imsave('lenna_gray_hist.jpg', lennaGrayHist)

[Finished in 0.4s]
Line 11, Column 54 Tab Size: 4 Python
```

Line3: 匯入OpenCV Library

Line6: 讀取剛剛製作的灰階影像。

Line8: 因為要做灰階影像的處理，只需取一層。

Line10: 進行直方圖等化。

Line11: 儲存直方圖等化後的結果。

直方圖等化前/後比較

