實戰一

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from google.colab import drive
from google.colab.patches import cv2_imshow
drive.mount('/content/drive')
img = cv2.imread("/content/drive/My Drive/Colab Notebooks/image_processing/Lenna.bmp")
print(img.shape)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
(512, 512, 3)
```

掛載雲端硬碟，並載入圖片

```
nr, nc = img.shape[:2]
x0 = nr // 2
y0 = nc // 2
sigma = 200
illumination = np.zeros([nr, nc], dtype = 'float32')
print(illumination.shape)
for x in range(nr):
        for y in range(nc):
                illumination[x, y] = np.exp( -(( x - x0) ** 2 + (y - y0) ** 2) / (2 * sigma * sigma))
```

```
(512, 512)
```

對圖片進行處裡

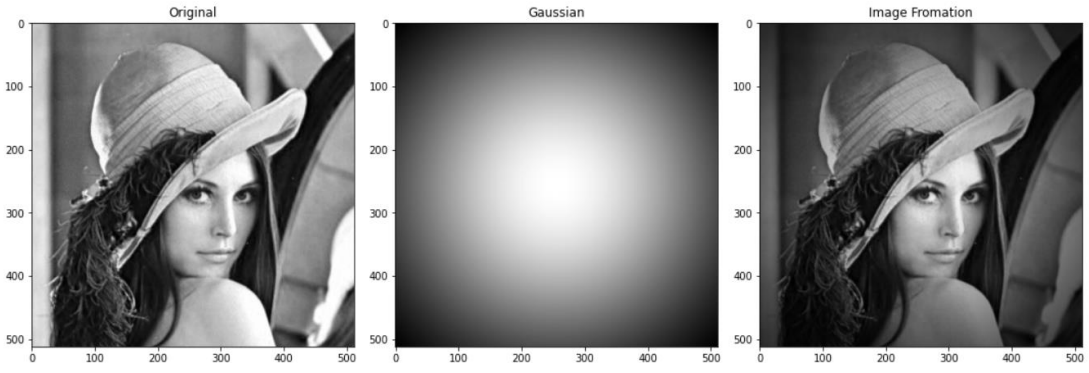
```
new_img = img.copy()
for x in range(nr):
        for y in range(nc):
                for k in range(3):
                        val = round(illumination[x, y] * img[x, y, k])
                        new_img[x, y, k] = np.uint8(val)
images = [img, illumination, new_img]
titles = ['Original', 'Gaussian', 'Image Fromation']
plt.figure(figsize = (15, 15))

for i in range(3):
        plt.subplot(1, 3, i + 1), plt.imshow(images[i], 'gray')
        plt.title(titles[i])

plt.tight_layout()
plt.show()
```

顯示三種圖片

執行結果:



實戰二

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from google.colab import drive
from google.colab.patches import cv2_imshow
drive.mount('/content/drive')
img = cv2.imread("/content/drive/My Drive/Colab Notebooks/image_processing/week3.bmp")
print(img.shape)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
(225, 225, 3)
```

掛載雲端硬碟，並載入我在網路上抓到之 bmp 檔

```
def image_quantization(img, bits):
        nr, nc = img.shape[:2]
        retImg = img.copy()
        levels = 2 ** bits
        interval = 256 / levels
        gray_level_interval = 255 / (levels)
        table = np.zeros(256)

        for k in range(256):
                for l in range(levels):
                        if k >= l * interval and l < (l + 1) * interval:
                                table[k] = round(l * gray_level_interval)
                for x in range(nr):
                        for y in range(nc):
                                retImg[x, y] = np.uint8(table[img[x, y]])

        return retImg
```

定義函式 image_quantization

```
img_5bit = image_quantization(img, 5)
img_2bit = image_quantization(img, 2)
images = [img, img_5bit, img_2bit]
titles = ['Original', 'QZ 5bits', 'QZ 2bits']
plt.figure(figsize = (25, 15))

for i in range(3):
        plt.subplot(1, 3, i + 1), plt.imshow(images[i], 'gray')
        plt.title(titles[i], fontsize = 20, color = 'r')

plt.tight_layout()
plt.show()
```

將數值帶入函式 image_quantization 並顯示結果

結果:

簡答題

Nyquist-Shammon 取樣定理 ：

取樣定理是讓取樣頻率高於原訊號最高頻率 2 倍以上，來使得取樣之結果不會失真

混疊 (Aliasing) 現象 ：

若取樣頻率不足時，即會產生混疊現象，易失真