An-Najah National University
Faculty of Engineering and IT
Electrical and Computer Engineering
Department

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات
دائرة الهندسة الكهربائية والحاسوب

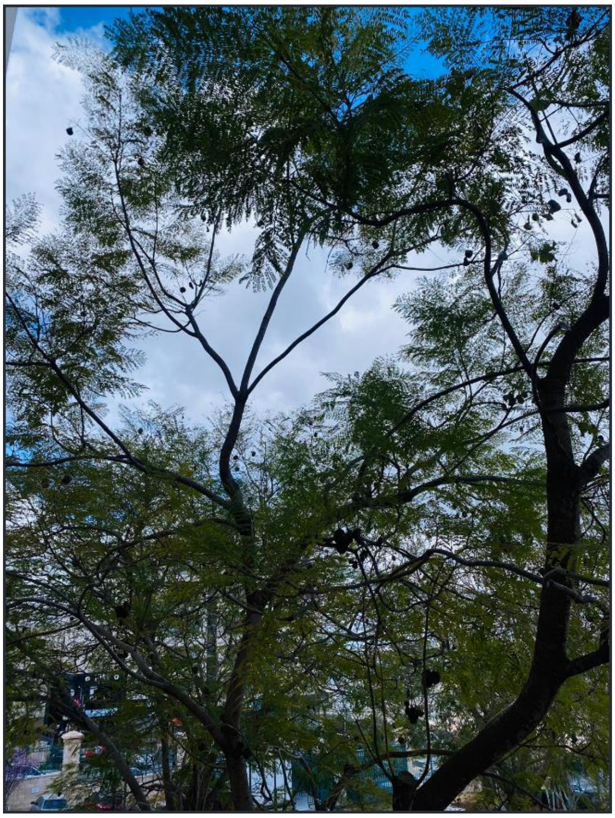
# Digital Image Processing HW.01

| | |
|---|---|
| **Student Name** | Mohammad Abd-Allateef Alawneh |
| **Student ID** | 12028067 |
| **Course Name** | Digital Image Processing |
| **Instructor Name** | Dr. Anas Toma |
| **Performed In** | 11.03.2023 |

## Contents

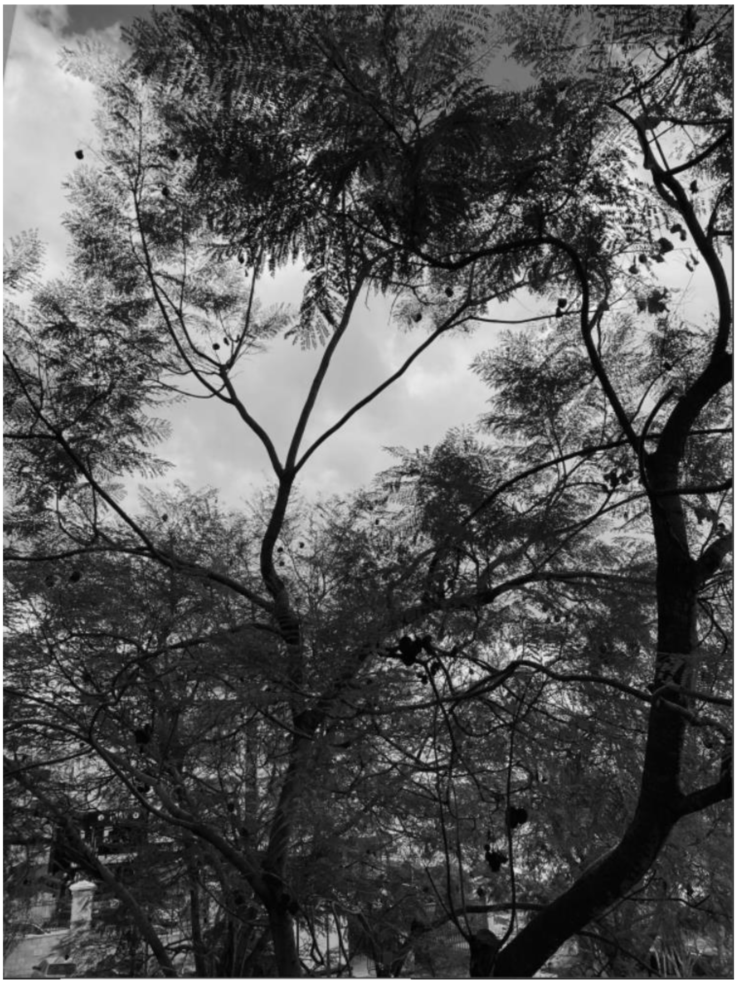
## Images and Histograms

- **Input image**

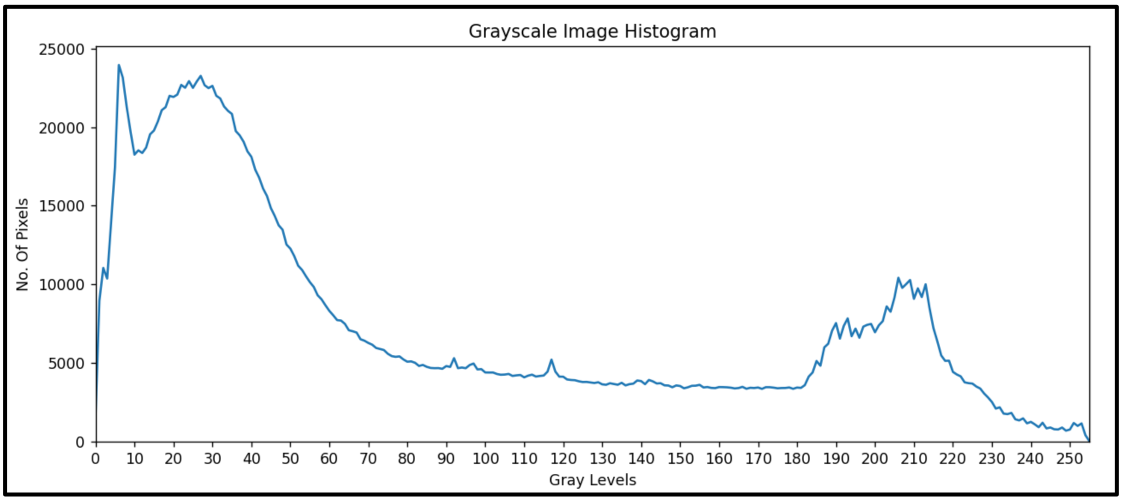


Input image

- **Input image in the grayscale and its histogram**

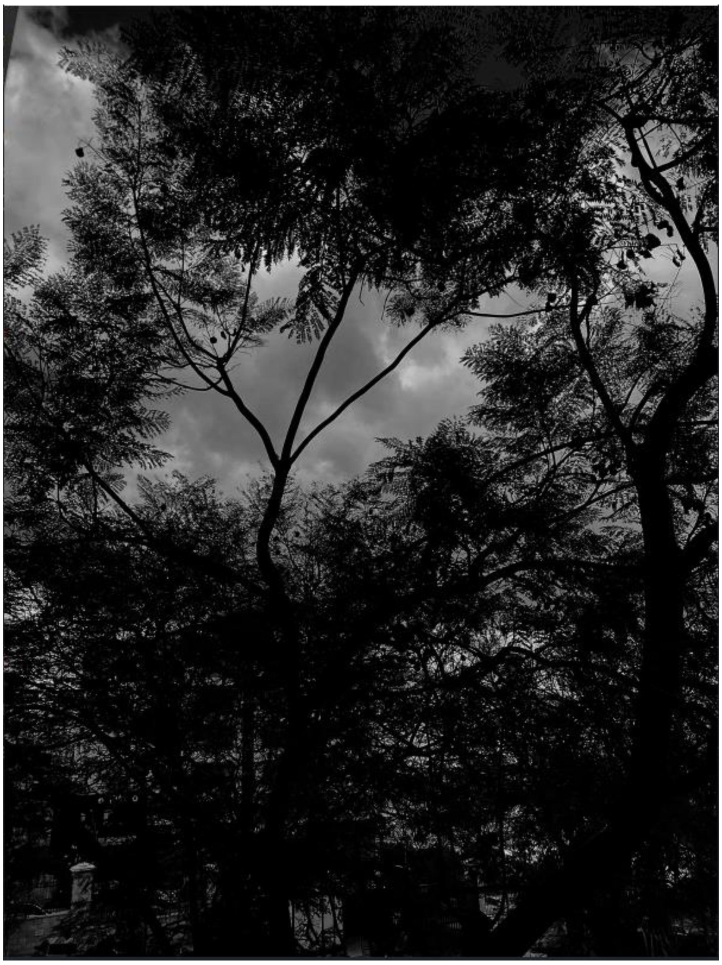


Input image in the grayscale

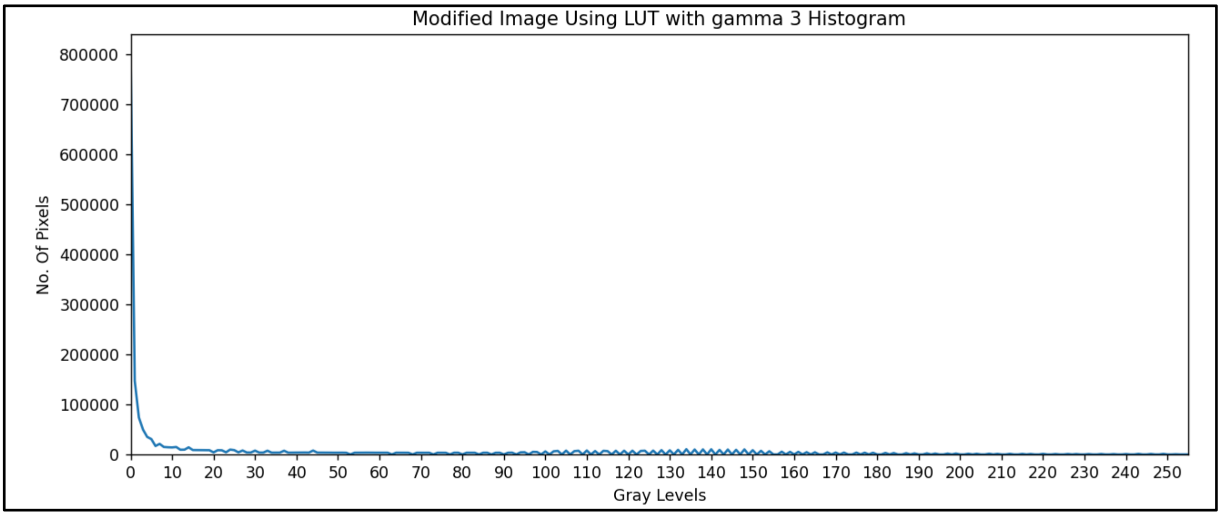


Histogram of the input image in the grayscale

- **Gamma correction using look-up table**

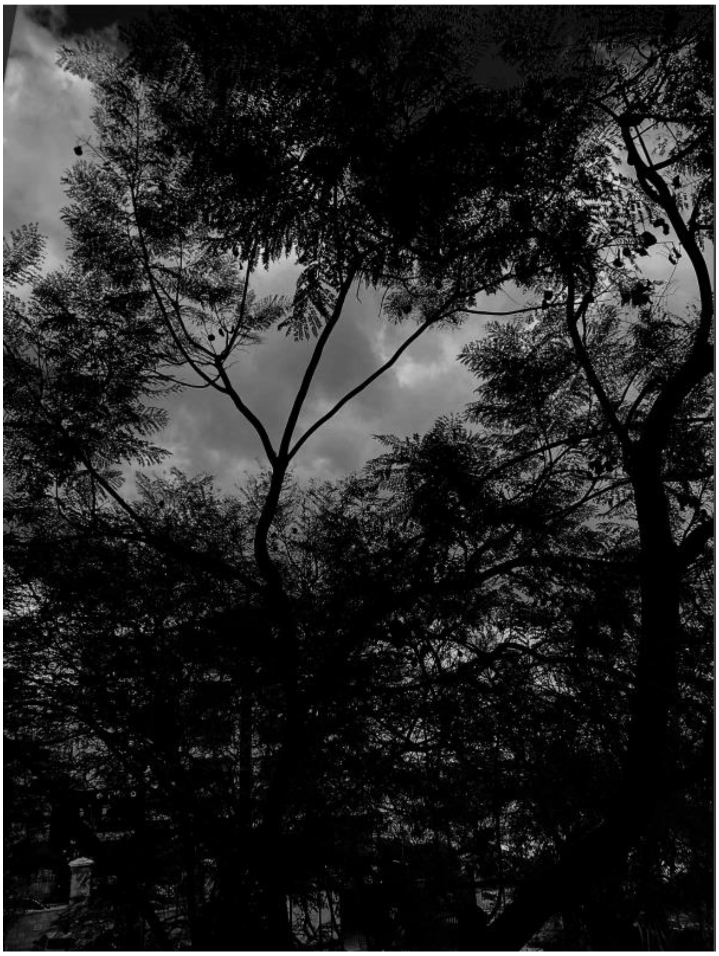


Grayscale Input image after gamma correction (γ = 3) using LUT

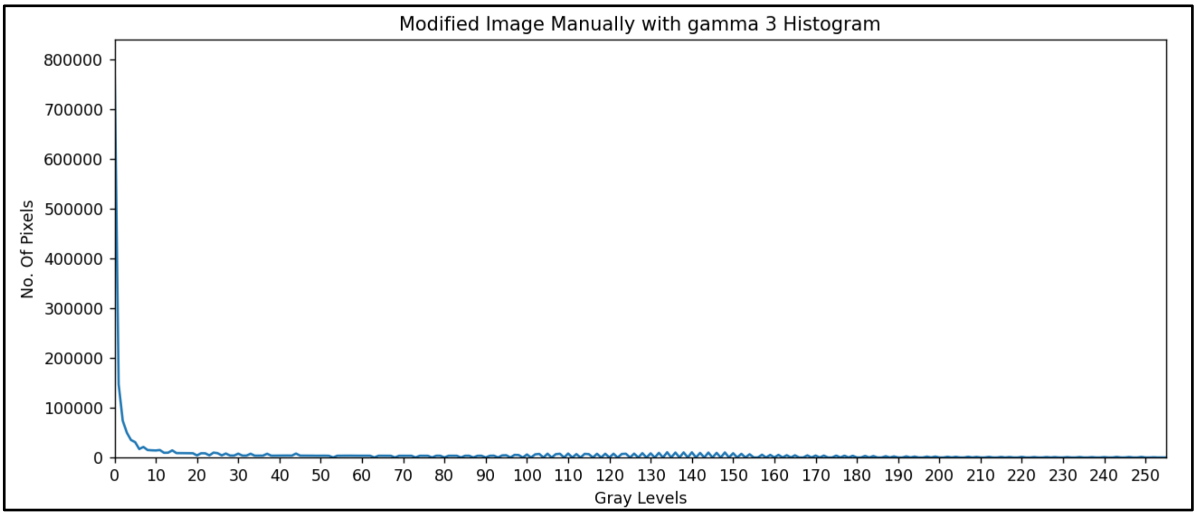


Histogram of the grayscale input image after gamma correction (γ = 3) using LUT

- **Gamma correction pixel by pixel**



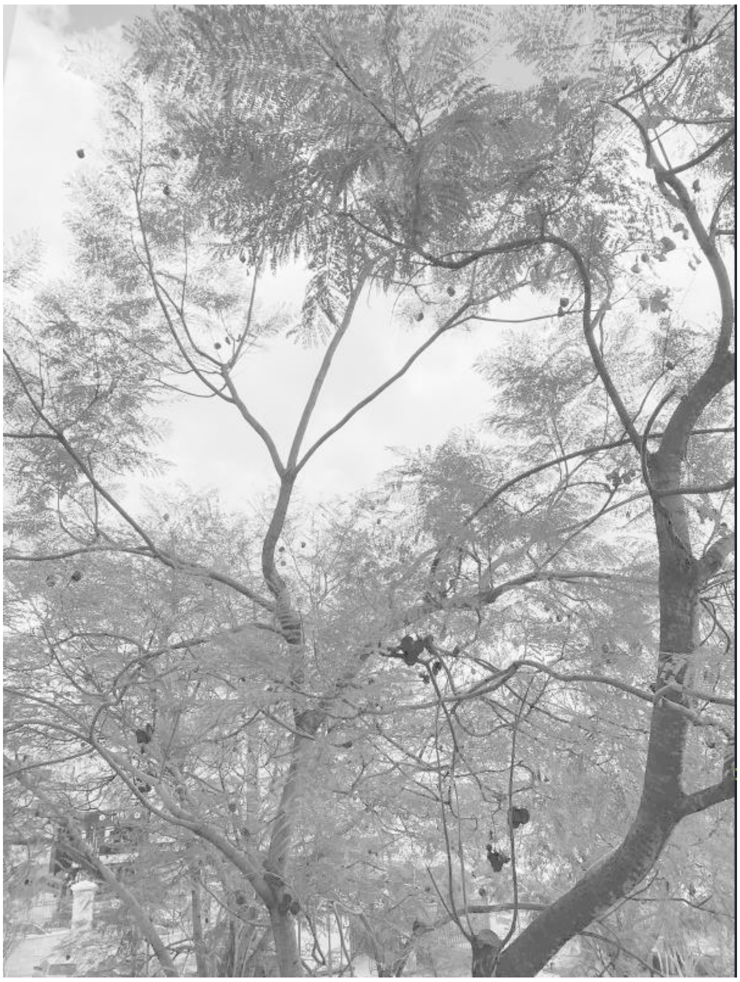
Grayscale Input image after gamma correction (γ = 3) pixel by pixel



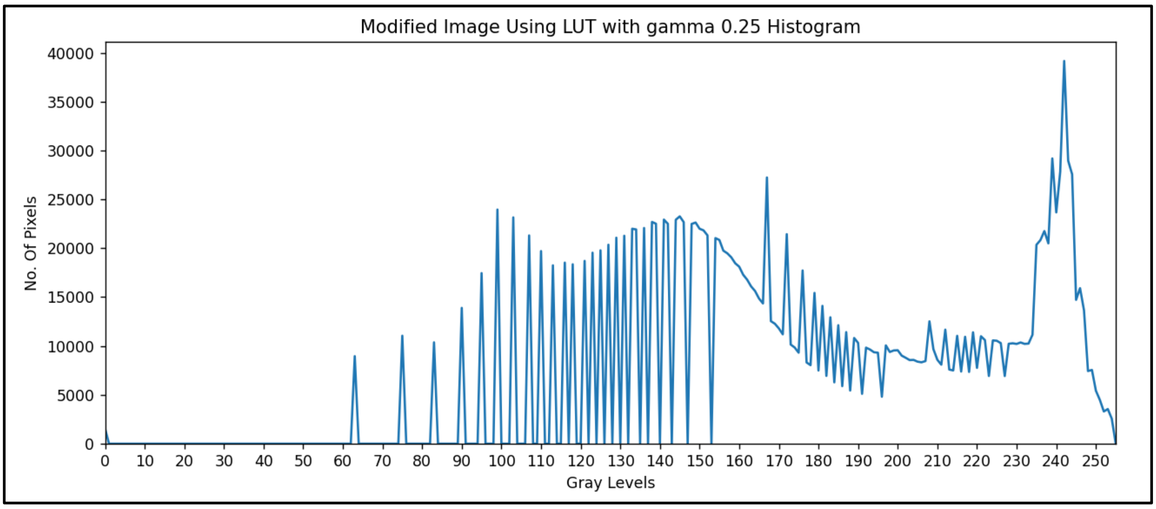
Histogram of the grayscale input image after gamma correction (γ = 3) pixel by pixel

**NOTE:** the two methods generated two identical histograms as i expected 😊 .

- **Extra part: (gamma correction with γ = 0.25)**

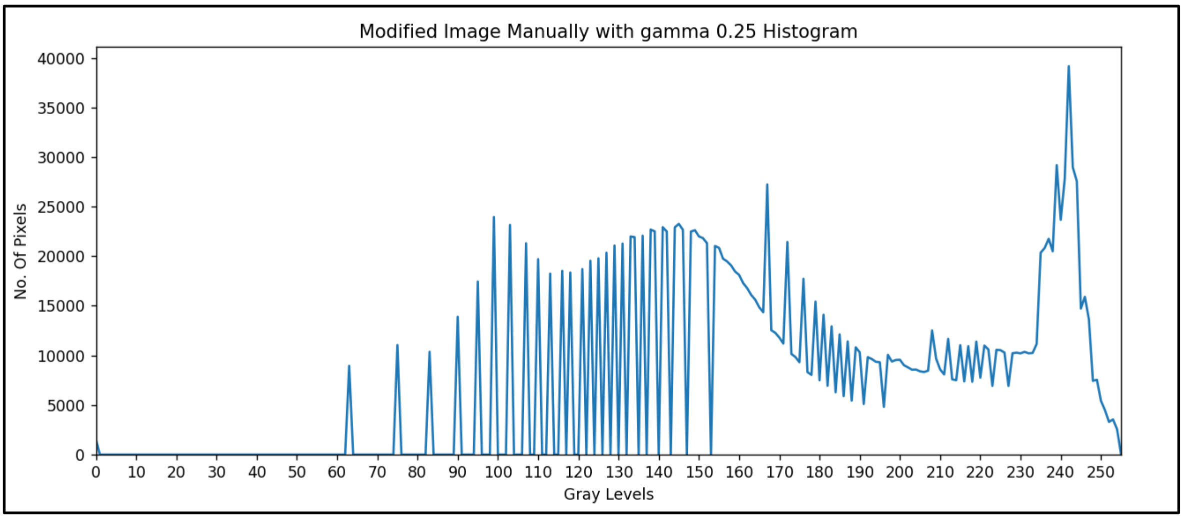


**Using LUT**



**Histogram for the previous image**

Pixel by pixel



Histogram for the previous image

## Execution Time Comparison

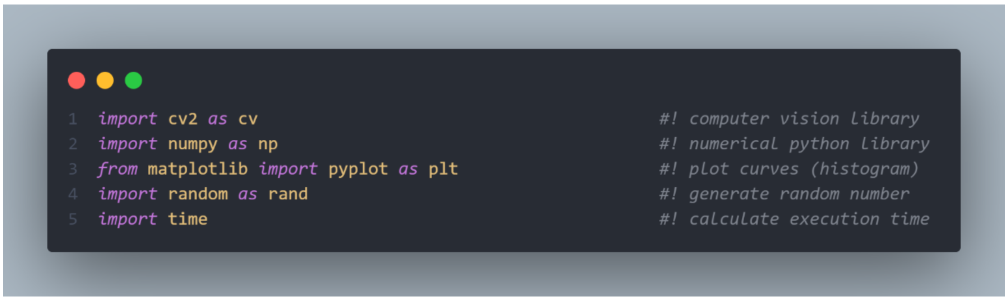
| Case | Execution Time (seconds) |
|---|---|
| **Lookup Table (LUT)** | 0.002012491226196289 |
| **Pixel by pixel** | 5.947009563446045 |

$$Speedup = \frac{5.947009563446045}{0.002012491226196289} = 2955.04869$$

The usage of LUT approach, gives a very large speedup.

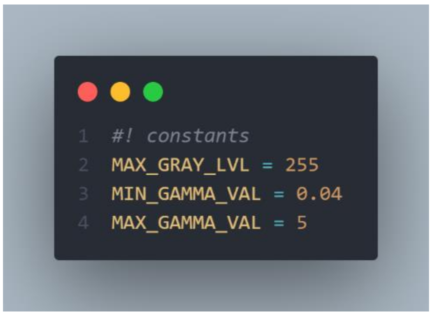
## Code Segments

- **Required Libraries**



```python
import cv2 as cv                                      #! computer vision library
import numpy as np                                    #! numerical python library
from matplotlib import pyplot as plt                  #! plot curves (histogram)
import random as rand                                 #! generate random number
import time                                           #! calculate execution time
```

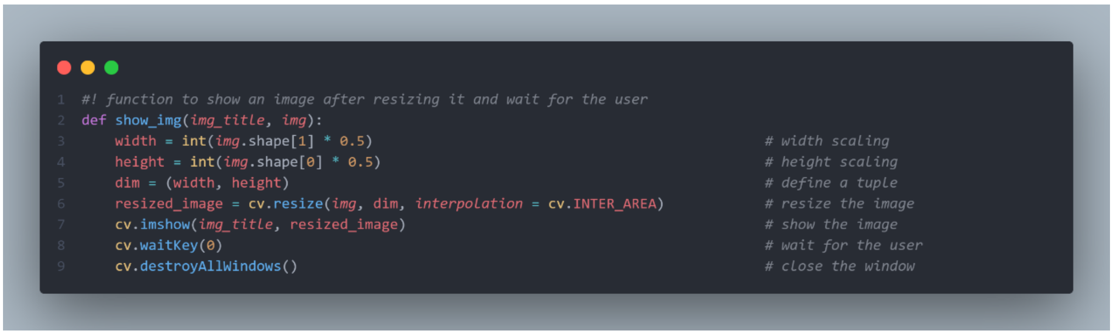
- **Some Constants**



```python
#! constants
MAX_GRAY_LVL  = 255
MIN_GAMMA_VAL  = 0.04
MAX_GAMMA_VAL  = 5
```

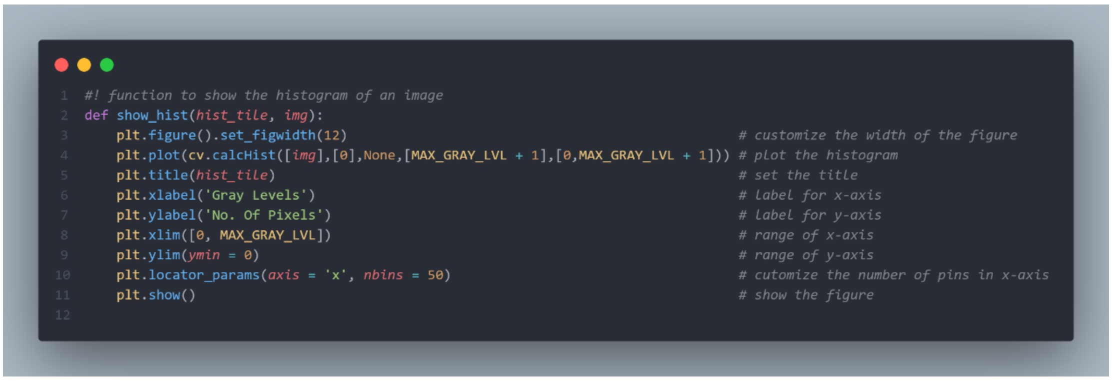
- **Two functions**

**First: takes a string and an image to display the image with a title after resizing it with a scaling factor 0.5**



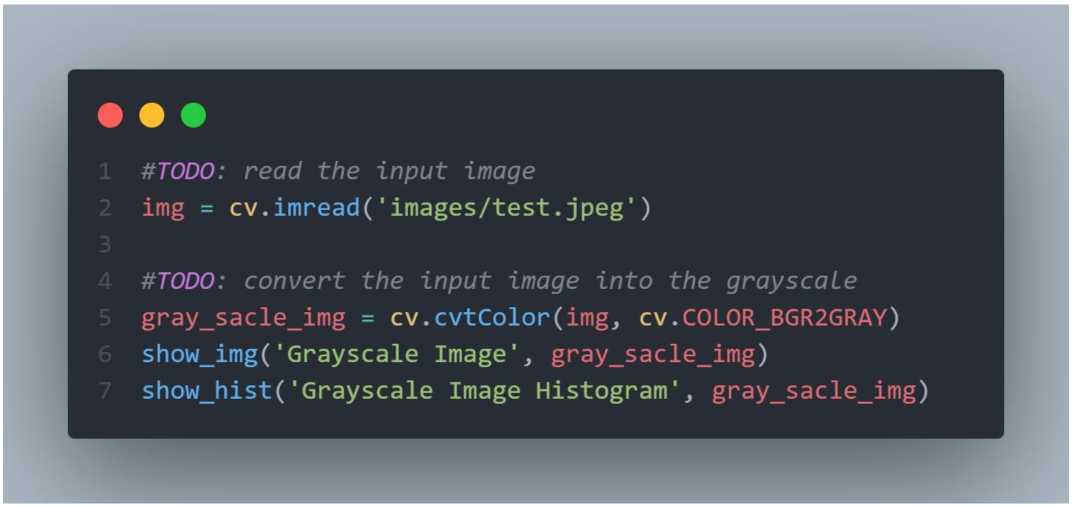
```python
#! function to show an image after resizing it and wait for the user
def show_img(img_title, img):
    width = int(img.shape[1] * 0.5)                                        # width scaling
    height = int(img.shape[0] * 0.5)                                       # height scaling
    dim = (width, height)                                                  # define a tuple
    resized_image = cv.resize(img, dim, interpolation = cv.INTER_AREA)     # resize the image
    cv.imshow(img_title, resized_image)                                    # show the image
    cv.waitKey(0)                                                          # wait for the user
    cv.destroyAllWindows()                                                 # close the window
```

### Second: takes a string and an image to display the histogram of the image with a title



```python
#! function to show the histogram of an image
def show_hist(hist_tile, img):
    plt.figure().set_figwidth(12)                                                # customize the width of the figure
    plt.plot(cv.calcHist([img],[0],None,[MAX_GRAY_LVL + 1],[0,MAX_GRAY_LVL + 1])) # plot the histogram
    plt.title(hist_tile)                                                         # set the title
    plt.xlabel('Gray Levels')                                                    # label for x-axis
    plt.ylabel('No. Of Pixels')                                                  # label for y-axis
    plt.xlim([0, MAX_GRAY_LVL])                                                  # range of x-axis
    plt.ylim(ymin = 0)                                                           # range of y-axis
    plt.locator_params(axis = 'x', nbins = 50)                                   # cutomize the number of pins in x-axis
    plt.show()                                                                   # show the figure

```

- **Read the input image and convert it into the gray scale**



```python
#TODO: read the input image
img = cv.imread('images/test.jpeg')

#TODO: convert the input image into the grayscale
gray_sacle_img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
show_img('Grayscale Image', gray_sacle_img)
show_hist('Grayscale Image Histogram', gray_sacle_img)
```

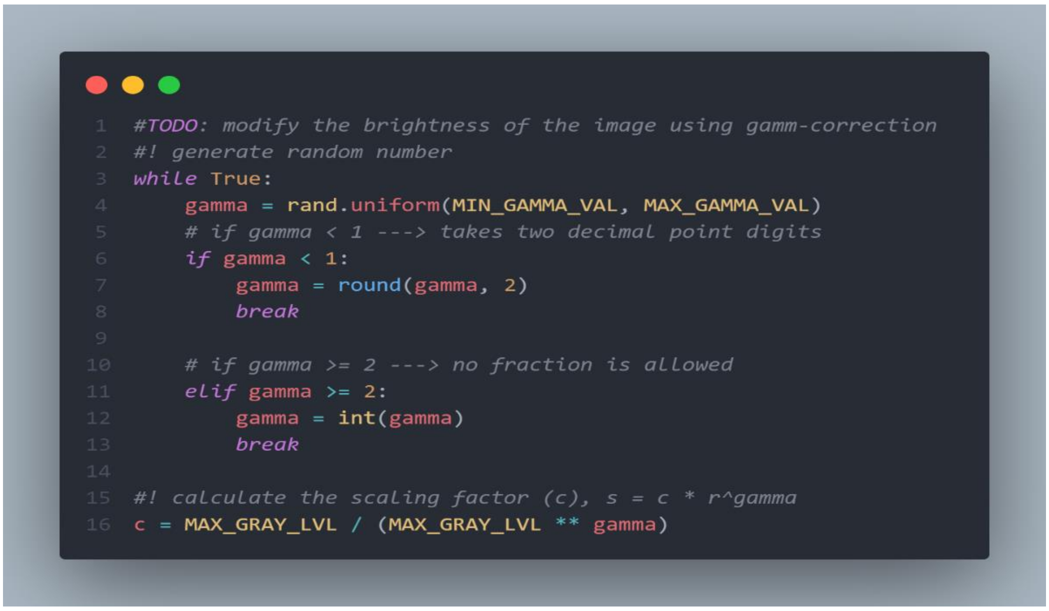
- **Generate random number for gamma, and calculate the scaling factor c**
-

**NOTE:** this code generates a random gamma value for the gamma correction -see the following steps-, you can update it if you want to test it under a certain gamma value.

1. Generate a random number between **0.04** and **5**
2. If the generated number is **less than 1**, takes only two decimal point digits.
3. If it's **greater than or equal to 2**, convert it into integer value -no fractions-.
4. Values in the interval **[1,2)** are not acceptable.

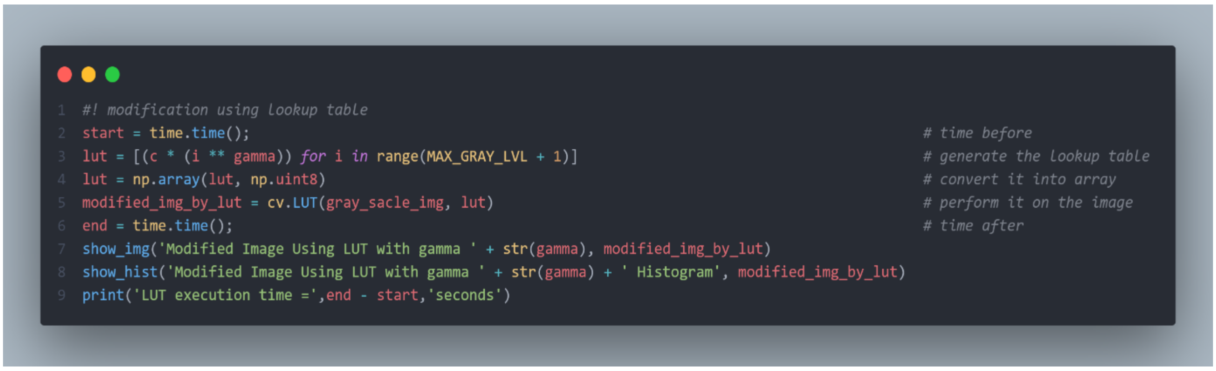
**To calculate the scaling factor:**

$$c = \frac{MAX_{GRAY_{LEVEL}}}{\left(MAX_{GRAY_{LEVEL}}\right)^{gamma}}$$



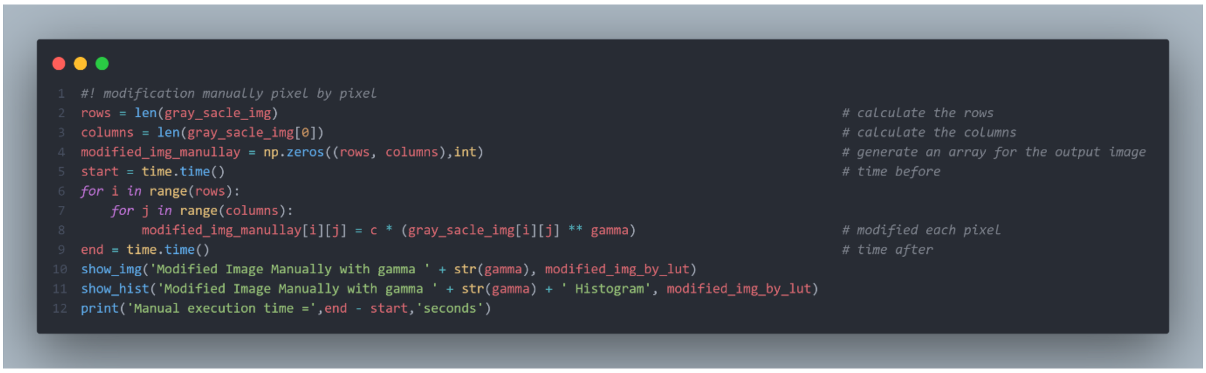
```
#TODO: modify the brightness of the image using gamm-correction
#! generate random number
while True:
    gamma = rand.uniform(MIN_GAMMA_VAL, MAX_GAMMA_VAL)
    # if gamma < 1 ---> takes two decimal point digits
    if gamma < 1:
        gamma = round(gamma, 2)
        break

    # if gamma >= 2 ---> no fraction is allowed
    elif gamma >= 2:
        gamma = int(gamma)
        break

#! calculate the scaling factor (c), s = c * r^gamma
c = MAX_GRAY_LVL / (MAX_GRAY_LVL ** gamma)
```

- **Modification using the lookup table**



```
#! modification using lookup table
start = time.time();                                                                # time before
lut = [(c * (i ** gamma)) for i in range(MAX_GRAY_LVL + 1)]                         # generate the lookup table
lut = np.array(lut, np.uint8)                                                       # convert it into array
modified_img_by_lut = cv.LUT(gray_sacle_img, lut)                                   # perform it on the image
end = time.time();                                                                  # time after
show_img('Modified Image Using LUT with gamma ' + str(gamma), modified_img_by_lut)
show_hist('Modified Image Using LUT with gamma ' + str(gamma) + ' Histogram', modified_img_by_lut)
print('LUT execution time =',end - start,'seconds')
```

- **Modification using the manual way -pixel by pixel-**



```
#! modification manually pixel by pixel
rows = len(gray_sacle_img)                                                     # calculate the rows
columns = len(gray_sacle_img[0])                                               # calculate the columns
modified_img_manullay = np.zeros((rows, columns),int)                          # generate an array for the output image
start = time.time()                                                            # time before
for i in range(rows):
    for j in range(columns):
        modified_img_manullay[i][j] = c * (gray_sacle_img[i][j] ** gamma)      # modified each pixel
end = time.time()                                                              # time after
show_img('Modified Image Manually with gamma ' + str(gamma), modified_img_by_lut)
show_hist('Modified Image Manually with gamma ' + str(gamma) + ' Histogram', modified_img_by_lut)
print('Manual execution time =',end - start,'seconds')
```

# THE END!

# THANK YOU!