Histograms

Histograms are most of the times plotted for a grayscale image. But can also simetimes be plottted for the other channels.
It is a plot of **Intensity VS. No of pixel**

`cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])`

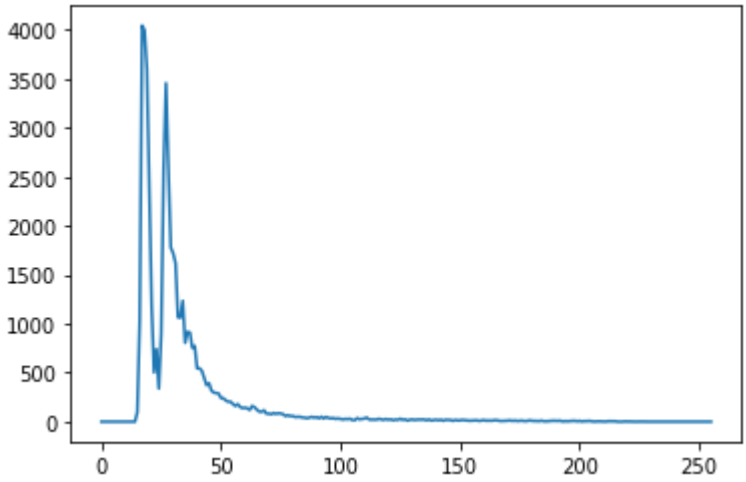
- **images** : it is the source image of type uint8 or float32. it should be given in square brackets, ie, "[img]".
- **channels** : it is also given in square brackets. It the index of channel for which we calculate histogram. For example, if input is grayscale image, its value is [0]. For color image, you can pass [0],[1] or [2] to calculate histogram of blue,green or red channel respectively.
- **mask** : mask image. To find histogram of full image, it is given as "None". But if you want to find histogram of particular region of image, you have to create a mask image for that and give it as mask. (I will show an example later.)
- **histSize** : this represents our BIN count. Need to be given in square brackets. For full scale, we pass [256].
- **ranges** : this is our RANGE. Normally, it is [0,256].

In [5]:
```
import cv2
import numpy as np

img = cv2.imread('images/moon.jpg',0)

hist = cv2.calcHist([img],[0],None,[256],[0,256])

cv2.imshow('win',img)
cv2.imshow('hist',hist)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

The output of the calcHist fucntion is not an actual imgae insted it is just a 2D array with 256 rows and each row has a value corresponding to the frquency of that pixel intensity.
So to actually plot an histogram we can use matplotlib.

In [9]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('images/moon.jpg',0)

hist = cv2.calcHist([img],[0],None,[256],[0,256])

cv2.imshow('win',img)
```

```
cv2.imshow('hist',hist)
plt.plot(hist)
plt.show()

# cv2.waitKey(0)
# cv2.destroyAllWindows()
```



Histogram Equalization

In [3]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('images/hist_equ1.jpg',0)

equ = cv2.equalizeHist(img)

cv2.imshow('win',img)
cv2.imshow('equ',equ)

# plt.plot(hist)
# plt.show()

cv2.waitKey(0)
cv2.destroyAllWindows()
```

CLAHE (Contrast Limited Adaptive Histogram Equalization)

In [4]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('images/hist_equ1.jpg',0)

equ = cv2.equalizeHist(img)

cv2.imshow('win',img)
cv2.imshow('equ',equ)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

But with adaptive histogram equalization we can get a better result

In [5]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('images/hist_equ1.jpg',0)

equ = cv2.equalizeHist(img)

clahe = cv2.createCLAHE()
clahe1 = cv2.createCLAHE(clipLimit=4.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)
cl2 = clahe.apply(img)

cv2.imshow('win',img)
cv2.imshow('cl1',cl1)
cv2.imshow('cl2',cl2)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

Home Work

1) Do histogram equalisation on the below image