Arithmetic Operation

These are simple operations that allow us to directly add or subtract to the color instantly.

Calculate the per-element operation of two arrays. The overall effect is increasing or decreasing brightness.

In [2]:
```
import cv2, numpy as np
image = cv2.imread('my.JPG')
cv2.imshow('Original Image', image)
#Create a matrix of ones, then multiply it by a scaler of 75
#This gives a matrix with same dimension of our image with all values being 75
M1 = np.ones(image.shape, dtype = 'uint8') * 20
M2 = np.ones(image.shape, dtype = 'uint8') * 75
#We use this to add this matrix M, to our Image
#Notice the imcrease in brightness
added = cv2.add(image, M1)
cv2.imshow('Added', added)
#cv2.imwrite('Added Image increased brightness.jpeg', added)

#likewise we can also subtract
#Notice the decrease in brightness
subtracted = cv2.subtract(image, M2)
cv2.imshow('Subtracted', subtracted)
#cv2.imwrite('Subtracted Image decrease in brightness.jpeg', subtracted)
cv2.waitKey()
cv2.destroyAllWindows()
```

In [10]:
```
M2 = np.ones(image.shape, dtype = 'int32') * 100
cv2.imshow('M2', M2)
cv2.waitKey(2222)
cv2.destroyAllWindows()
```

Bitwise Operations and Masking

To demonstrate these operations let's create some simple images

In [26]:
```
import cv2, numpy as np

#If you're wondering why only two dimensions, well this is grayscale image,
#If we doing a colored image, we'd use
#rectangle * np.zeros((300, 300, 3), np.uint8)

#Making a square
square = np.zeros((300, 300), np.uint8)

cv2.rectangle(square, (50, 50), (250, 250), 255, -2)
cv2.imshow('Square', square)
cv2.waitKey()

#Making a ellipse

ellipse = np.zeros((300, 300), np.uint8)
cv2.ellipse(ellipse, (150, 150), (150, 150), 30, 0, 180, 255, -1)
cv2.imshow('Ellipse', ellipse)
cv2.imwrite('Ellipse.png', ellipse)
cv2.waitKey()
cv2.destroyAllWindows()
```

Experimenting some bitwise operation

In [32]:

```python
import cv2, numpy as np

#If you're wondering why only two dimensions, well this is grayscale image,
#If we doing a colored image, we'd use
#rectangle * np.zeros((300, 300, 3), np.uint8)

#Making a square
square = np.zeros((300, 300), np.uint8)

cv2.rectangle(square, (50, 50), (250, 250), 255, -2)
cv2.imshow('Square', square)
cv2.waitKey()

#Making a ellipse

ellipse = np.zeros((300, 300), np.uint8)
cv2.ellipse(ellipse, (150, 150), (150, 150), 30, 0, 180, 255, -1)
cv2.imshow('Ellipse', ellipse)
cv2.imwrite('Ellipse.png', ellipse)
cv2.waitKey()

#bitwise_and
#Shows only where they intersect
And = cv2.bitwise_and(square, ellipse)
cv2.imshow('bitwise square And ellipse', And)
cv2.waitKey()

##bitwise_or
bitwiseOr = cv2.bitwise_or(square, ellipse)
cv2.imshow('bitwise square OR ellipse', bitwiseOr)
cv2.waitKey()

#bitwise_xor
bitwiseXor = cv2.bitwise_xor(square, ellipse)
cv2.imshow('Bitwise square XOR ellipse', bitwiseXor)
cv2.waitKey()

#bitwise_not on square
bitwiseNot_square = cv2.bitwise_not(square)
cv2.imshow('Not square', bitwiseNot_square)
cv2.waitKey()
```

```
42
43 #bitwise_not on ellipse
44 bitwiseNot_ellipse = cv2.bitwise_not(ellipse)
45 cv2.imshow('Not ellipse', bitwiseNot_ellipse)
46 cv2.waitKey()
47 cv2.destroyAllWindows()
```

ADD TWO IMAGES

Whatsapp crop

In [26]:

```
img = cv2.imread('Subtracted Image decrease in brightness.jpeg')
img.shape
cv2.imshow('original image', img)
cv2.waitKey()
cv2.destroyAllWindows()

img2 = np.zeros((700, 300, 3), np.uint8)
cv2.imshow('Zeros image', img2)
cv2.waitKey()
cv2.destroyAllWindows()

img3 = np.concatenate((img, img2), axis = 1)
img3 = np.concatenate((img2, img3), axis = 1)
cv2.imshow('concatenate', img3)
cv2.waitKey()
cv2.destroyAllWindows()

img3.shape

img2 = np.zeros((100, 1112, 3), np.uint8)
img3 = np.concatenate((img2, img3), axis = 0)
img3 = np.concatenate((img3, img2), axis = 0)
cv2.imshow('concatenate', img3)
cv2.waitKey()
cv2.destroyAllWindows()

cv2.imwrite('wtsapp.jpeg', img3)
```