Welcome to the tutorial on Arithmetic and Logical Operations

In this document, we'll talk about the following:

- Arithmetic operations in Image Processing
- Logical operations in Image Processing

Arithmetic operations

Image arithmetic applies one of the standard arithmetic operations on two or more images.
The operators are applied in a pixel-by-pixel way, i.e. the value of a pixel in the output image depends only on the values of the corresponding pixels in the input images. Hence, the **images must be of the same size**.

Let's import the necessary packages and import an image in **RGB**.

In [1]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

We'll create a helper function which will help us view the input and output images together.

In [2]:
```
def plotter(img1, img2, final_img):
    fig=plt.figure(figsize=(10,4))
    rows = 1
    columns = 3

    fig.add_subplot(rows, columns, 1)
    plt.imshow(img1); plt.axis('off'); plt.title("Image 1")

    fig.add_subplot(rows, columns, 2)
    plt.imshow(img2); plt.axis('off'); plt.title("Image 2")

    fig.add_subplot(rows, columns, 3)
    plt.imshow(final_img); plt.axis('off'); plt.title("After Operation")
```

We'll understand the various operations by combining various solid colors.

In [3]:
```
black_100 = np.zeros((300, 100, 3), np.uint8)

white_100 = np.zeros((300, 100, 3), np.uint8)
white_100[:] = (255, 255, 255)

red_200 = np.zeros((300, 200, 3), np.uint8)
red_200[:] = (255, 0, 0)

blue_200 = np.zeros((300, 200, 3), np.uint8)
blue_200[:] = (0, 0, 255)
```

`np.hstack(arrays)`

Horizontally stacks the given arrays

Parameters

arrays: sequence of numpy arrays

Returns

Returns horizontally stacked input array

In [4]:
```
img1 = np.hstack([red_200, black_100])
img2 = np.hstack([black_100, blue_200])
```

`cv2.add(src1, src2 )`

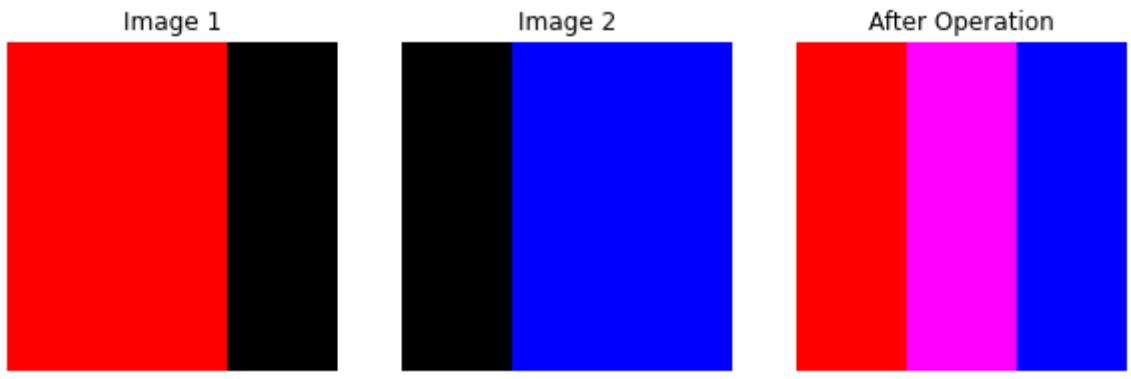
Parameters

src1: First Image
src2: Second Image

Returns

Returns the addition of the two images

In [5]:
```
img3 = cv2.add(img1, img2)
plotter(img1, img2, img3)
```



Often in reality we don't want to just add the images, we also want to control how much of each image is added. For this we have a sepearate function.

`cv2.addWeighted(src1, alpha, src2, beta, gamma )`

Parameters

src1: First Image
alpha: weight of the First image **src2**: Second Image
beta: weight of the Second image
gamma: scalar added to each sum

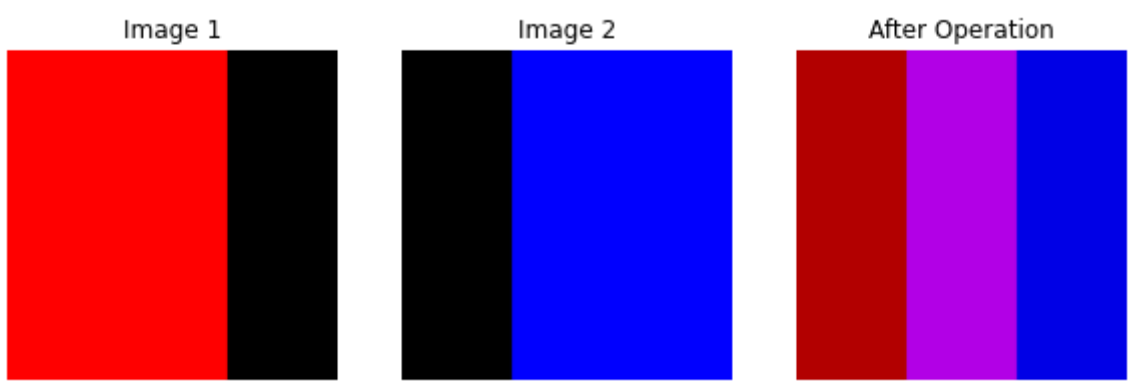
Returns

Returns the weighted addition of the two images as per the following formula:

src1*alpha + src2*beta + gamma

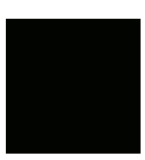
In [6]:
```
alpha = 0.7
beta = 0.9
gamma = 0

img3 = cv2.addWeighted(img1, alpha, img2, beta, gamma)
plotter(img1, img2, img3)
```



Some other Arithmetic operations possible are:

- Subtraction
- Multiplication
- Division
- Log
- Exponentional, etc

Operation	Function used	Image 1	Image 2	Final
Subtraction	`cv2.subtract(src1, src2)`			
Multiplication	`cv2.multiply(src1, src2 [,scale])`			
Division	`cv2.divide(src1, src2 [,scale])`			

Logical operations

Logical operators are often used to combine two (mostly binary) images. The logical operations possible are:

- AND
- OR
- NOT
- XOR

Operation	Function used	Image 1	Image 2	Final
AND	cv2.bitwise_and(src1, src2)			
OR	cv2.bitwise_or(src1, src2)			
NOT	cv2.bitwise_not(src)		-