# Image Arithmetic & logical Operations using OpenCV in Python

## Image Addition

In [1]:

```
# We can add two images by OpenCV function, cv2.add() or simply by numpy operation,
# res = img1 + img2. Both images should be of same depth and type, or second image can ju
st be a scalar value.

# Importing required modules
import cv2
import numpy as np

# Considering both the imgaes of indentical size 500 x 250
img1 = cv2.imread('./Img-1.png')
img2 = cv2.imread('./Img-2.png')

add = img1+img2

cv2.imshow('addition window',add)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [2]:

```
# Importing required modules
import cv2
import numpy as np

# Considering both the imgaes of indentical size 500 x 250
img1 = cv2.imread('./Img-1.png')
img2 = cv2.imread('./Img-2.png')

add = cv2.add(img1,img2)

cv2.imshow('addition window',add)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

***We can see that much of the image is very "white." This is because colors are 0-255, where 255 is "full light."***

## Image Blending

In [3]:

```
#### This is also image addition, but different weights are given to images so that it gi
ves a feeling of blending or transparency.
#### By varying alpha from 0-1, we can perform a cool transition between one image to ano
ther. Here I took two images to blend them together. First image is given a weight of 0.6
and second image is given 0.4. cv2.addWeighted() applies .
#### Here gamma is taken as zero.

# Importing required modules
import cv2
import numpy as np

# Considering both the imgaes of indentical size 500 x 250
img1 = cv2.imread('./Img-1.png')
img2 = cv2.imread('./Img-2.png')
```

```
weighted = cv2.addWeighted(img1, 0.6, img2, 0.4, 0)
# Notation: addWeighted method, the parameters are the first image, the weight, the secon
d image, that weight, and gamma, which is a measurement of light.

cv2.imshow('add weighted window',weighted)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## To show the mask inverse using threshold

In [7]:

```
# Importing required modules
import cv2
import numpy as np

# Load two images
img1 = cv2.imread('./Img-1.png')
img2 = cv2.imread('./pythonlogo.png')

# I want to put logo on top-left corner, So I create a ROI
rows,cols,channels = img2.shape
roi = img1[0:rows, 0:cols]

# Now create a mask of logo and create its inverse mask, mask generly would be to grey sc
ale
img2gray = cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)

# add a threshold
#  Threshold: it works is that, it will convert all pixels to either black or white, base
d on a threshold value.
ret, mask = cv2.threshold(img2gray, 220, 255, cv2.THRESH_BINARY_INV)

cv2.imshow('mask window', mask)

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

## Bitwise Operations

In [8]:

```
# Importing required modules
import cv2
import numpy as np

# Load two images
img1 = cv2.imread('./Img-1.png')
img2 = cv2.imread('./pythonlogo.png')

# I want to put logo on top-left corner, So I create a ROI
rows,cols,channels = img2.shape
roi = img1[0:rows, 0:cols ]

# Now create a mask of logo and create its inverse mask, mask generly would be to grey sc
ale
img2gray = cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)

# add a threshold
#  Threshold: it works is that, it will convert all pixels to either black or white, base
d on a threshold value.
ret, mask = cv2.threshold(img2gray, 220, 255, cv2.THRESH_BINARY_INV)

#This is a bitwise operation.black area of mask
mask_inv = cv2.bitwise_not(mask)

# bitwise_not : mask or not mask
# bitwise_and : when two are equal
# bitwise_or : When both values are true it runs, if only one value iss true it runs
# bitwise_xor : onle when one value is true

# we want to black out this area in the first image, and then take image 2 and replace it
's contents in that empty spot.

# Now black-out the area of logo in ROI, bitwise is a lowlevel logical operation.
img1_bg = cv2.bitwise_and(roi,roi,mask = mask_inv)

# Take only region of logo from logo image.
img2_fg = cv2.bitwise_and(img2,img2,mask = mask)

dst = cv2.add(img1_bg,img2_fg)
img1[0:rows, 0:cols ] = dst

cv2.imshow('res window', img1)
cv2.imshow('mask_inv window', mask_inv)
cv2.imshow('img1_bg window', img1_bg)
cv2.imshow('img2_fg window', img2_fg)
cv2.imshow('dst window', dst)

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

## Performing bitwise operation with out using mask inverse

In [9]:

```
# Load two images
img1 = cv2.imread('./Tower.jpg')
img2 = cv2.imread('./pythonlogo.png')

# I want to put logo on top-left corner, So I create a ROI
rows,cols,channels = img2.shape
roi = img1[0:rows, 0:cols ]

# Now create a mask of logo and create its inverse mask also
img2gray = cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)

# Now black-out the area of logo in ROI
img1_bg = cv2.bitwise_and(roi,roi,mask = mask_inv)

# Take only region of logo from logo image.
img2_fg = cv2.bitwise_and(img2,img2,mask = mask)

# Put logo in ROI and modify the main image
dst = cv2.add(img1_bg,img2_fg)
img1[0:rows, 0:cols ] = dst

cv2.imshow('res',img1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]: