# Apply all the given functions in cheat sheet and describe necessary details using Markdown

In [13]:
```
import pandas as pd
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

# INPUT & OUTPUT

In [294]:
```
##1 imread will fetch the image from path and display it in a matrix form and default value will load img int
img = cv2.imread("../OpenCvDataSet/child.jpg")

## shows the image in a different window ("this value is Header of other window", this will image variable)
cv2.imshow("img",img)

## Waitkey will stop image from disappering untion
##any value is pressed (0 mean it will stop the img window untill any key is pressed or 500 for 0.5s time pau
cv2.waitKey(0)

## It will distroy all windows which are open and stop the script
cv2.destroyAllWindows()

## It will display image in our jupyter file
```

In [295]:
```
##2 it import image as it is (it does not modify or change any thing
img2 = cv2.imread("../OpenCvDataSet/child.jpg",cv2.IMREAD_UNCHANGED)

cv2.imshow("img",img2)
cv2.waitKey(0)


cv2.destroyAllWindows()
```

In [296]:
```
## IMREAD_GRAYSCALE value will load img into single-channel grayscale image
img3 = cv2.imread("../OpenCvDataSet/child.jpg",cv2.IMREAD_GRAYSCALE)

cv2.imshow("img",img3)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

# Color / Intensity

In [297]:
```
## cvtColor will convert the image in from bgr to gray
bgr_gray = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)
cv2.imshow("img",bgr_gray)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

In [298]:
```
## cvtColor will convert the image in from bgr to rgb
bgr_rbg = cv2.cvtColor(img , cv2.COLOR_BGR2RGB)
cv2.imshow("img",bgr_rbg)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

In [299]:
```
## cvtColor will convert the image in from gray to rgb
bgr_rbg = cv2.cvtColor(img3 , cv2.COLOR_GRAY2RGB)
cv2.imshow("img",bgr_rbg)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

In [300]:
```
## equalizeHist(img3) Only work with GrayScale Image Histogram equalization is a method used to
##enhance the contrast of an image by redistributing the intensity values of pixels.
i = cv2.equalizeHist(img3)
cv2.imshow("img",i)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

# Other UseFull color Spaces

In [301]:
```
## cvtColor will convert the image in from bgr to HSV
bgr_rbg = cv2.cvtColor(img , cv2.COLOR_BGR2HSV)
cv2.imshow("img",bgr_rbg)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

In [302]:
```
## cvtColor will convert the image in from bgr to LUV
bgr_rbg = cv2.cvtColor(img , cv2.COLOR_BGR2LUV)
cv2.imshow("img",bgr_rbg)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

In [303]:
```
## cvtColor will convert the image in from bgr to LAB
bgr_rbg = cv2.cvtColor(img , cv2.COLOR_BGR2LAB)
cv2.imshow("img",bgr_rbg)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

In [304]:
```
## cvtColor will convert the image in from bgr to YCrCb
bgr_rbg = cv2.cvtColor(img , cv2.COLOR_BGR2YCrCb)
cv2.imshow("img",bgr_rbg)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

# Channel Manipulation

In [305]:
```
## split the images into channel
a ,b , c= cv2.split(img)
cv2.imshow("img",a)
cv2.waitKey(1000)
cv2.imshow("img",b)
cv2.waitKey(1000)
cv2.imshow("img",c)
cv2.waitKey(1000)
cv2.destroyAllWindows()
```

In [306]:
```
## Will split image of format (RGBA) four Channel image

a,b,c,d = cv2.split(img)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[306], line 3
      1 ## Will split image of format (RGBA) four Channel image
----> 3 a,b,c,d = cv2.split(img)

ValueError: not enough values to unpack (expected 4, got 3)
```

In [307]:
```
merg = cv2.merge((a,b,c))
cv2.imshow("img",merg)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

# Arthematic Operations

In [308]:
```
## add each pixel element wise
i = cv2.add(a,b,c)
cv2.imshow("img",i)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [310]:
```
## addWeight will blend the image together

img4= cv2.resize(img4, (img.shape[1], img.shape[0]))

# Now blend the images 'img' and 'img4_resized'
mer = cv2.addWeighted(img, 0.7, img4, 0.5, 0.3)

# Display the blended image 'mer'
cv2.imshow("img", mer)
cv2.waitKey(1000)
cv2.destroyAllWindows()
```

In [311]:
```
## subtract image 1 and 4 pixel wise
sub = cv2.subtract( img , img4)
cv2.imshow("img",sub)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [312]:
```
## will take absolute difference image 1 and 4 pixel wise
sub = cv2.absdiff( img , img4)
cv2.imshow("img",sub)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

# Logical Operation

In [313]:
```
## will create a inverse image (invert each bit) create a negative image

nott = cv2.bitwise_not(img)
cv2.imshow("img",nott)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [314]:
```
## will take bitwise_and and set value to 1 of its true or 0 if its falsw(mask image)
andd = cv2.bitwise_and(img, img4)
cv2.imshow("img",andd)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [315]:
```
## Perform bitwise_or if both img and img4 is zero then 0 other wise 1

orr = cv2.bitwise_or(img, img4)
cv2.imshow("img",orr)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [316]:
```
## Perform bitwise_xor if both img and img4 is zero then 0 other wise 1

xorr = cv2.bitwise_xor(img, img4)
cv2.imshow("img",xorr)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

# Statistic

In [317]:
```
## Mean value of blue green red and alpha

mB ,mG ,mR ,mA = cv2.mean(img)
mB ,mG ,mR ,mA
```

Out[317]: (67.64423601017442, 60.65259811046512, 46.79386900436047, 0.0)

In [318]:
```
## will find mean and Standard deviation of each color channel

mean ,sds = cv2.meanStdDev(img)
mean ,sds
```

Out[318]:
```
(array([[67.64423601],
        [60.65259811],
        [46.793869  ]]),
 array([[50.63913743],
        [57.43021442],
        [56.71853188]]))
```

In [319]:
```
##

hist = cv2.calcHist([img], [0], None, [256], [0, 256])
cv2.imshow("img",hist)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

# Filtering

In [320]:
```
## Will blur the img
bl = cv2.blur(img ,(20,20))
cv2.imshow("img",bl)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [321]:
```
## Will blur the img
gb= cv2.GaussianBlur(img ,(5,5),sigmaX=10,sigmaY=0)
cv2.imshow("img",gb)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [322]:
```
## Will auto blur the img
gb1 = cv2.GaussianBlur(img ,None,sigmaX=2,sigmaY=2)
cv2.imshow("img",gb1)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [323]:
```
## This line applies the filter to the img using a custom kernel filter specified by the value 5.
## filter using cross correlation
im = cv2.filter2D(img ,-1, 5)
cv2.imshow("img",im)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [324]:
```
## This line applies the filter to the img using a custom kernel filter specified by the value 5.
##
kx = cv2.getGaussianKernel(10,-1)
cv2.imshow("img",kx)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [325]:
```
sep = cv2.sepFilter2D(img,-1,5,1)
cv2.imshow("img",sep)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [326]:
```
bl = cv2.medianBlur(img,5)
cv2.imshow("img",bl)
cv2.waitKey(1000)

cv2.destroyAllWindows()
```

In [ ]:

In [ ]: