Getting Started with Images

This notebook will help you take your first steps in learning Image Processing and Computer Vision using OpenCV. You will learn some important lessons using some simple examples. In this notebook, you will learn the following:

- Reading an image
- Check image attributes like data type, shape, size, max, min, and mean
- Check image attributes like format and Palette
- Matrix representation of an image in Numpy
- Color Images and splitting/merging image channels
- Displaying images using matplotlib
- Saving images

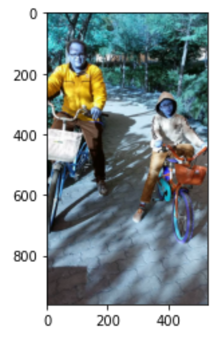
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Sample Image Read and Display

```
# Method-1: imread function, orginal color not showing

import cv2
import matplotlib.pyplot as plt

img = cv2.imread('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images\
/shocchoCycle.jpg')

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fa61c3a86d0>
```

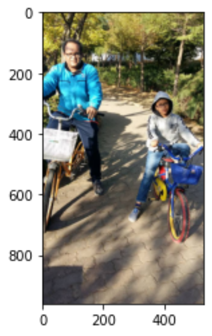


```
# Method-2: use imread function and solve color issue using flag (x, y, i):

import cv2
import matplotlib.pyplot as plt

img = cv2.imread('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images\
/shocchoCycle.jpg')

#[Image Filp VD (x) ::-1, Image Filp HD (y) ::-1, Image Color Index  (I) ::-1]
img1 = img[::-1, :, ::-1]
plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7fc06b1ddf10>
```

```
# Method-3: PIL and open function, no color issue

from PIL import Image
import matplotlib.pyplot as nur

img = Image.open('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoCycle.jpg')

nur.imshow(img)
#plt.imshow(img, cmap = 'gray')
```

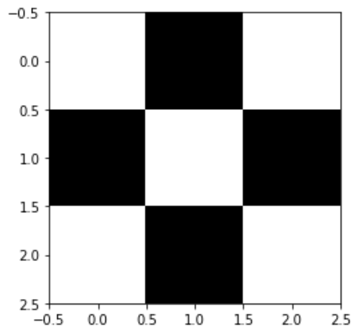
```
<matplotlib.image.AxesImage at 0x7fc06b0e72b0>
```



How Create Custom Image and fixed color using cmap function

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

a = [[100, 0, 100], [0, 100, 0], [100, 0, 100]]

plt.imshow(a, cmap='gray')
```

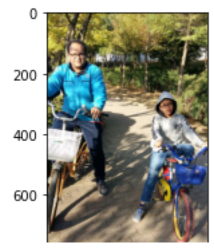
```
<matplotlib.image.AxesImage at 0x7fa4846b14f0>
```



Check image attributes like shape, size, max, min, mean

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

img = cv2.imread('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoCycle.jpg')
img1 = img[:, :, ::-1]

plt.imshow(img1)

print("image shape:",img.shape)
print(img.size)
print(img.dtype)
print("Minimum Value:", np.min(img))
print("Maximum Value:", np.max(img))
print("mean value:", np.mean(img))
```

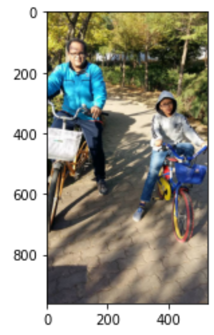
```
image shape: (960, 528, 3)
1520640
uint8
Minimum Value: 0
Maximum Value: 255
mean value: 109.86646214751684
```



Check image attributes like format and Palette

```
from PIL import Image

import cv2
import numpy as np
import matplotlib.pyplot as plt

img = Image.open('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoCycle.jpg')

plt.imshow(img)

print("Filename: ", img.filename)
print("Format: ", img.format)
print("Image Patette:", img.palette)
print("Mode: ", img.mode)
print("Size: ", img.size)
print("Width: ", img.width)
print("Height: ", img.height)
print("Is Animated: ", (getattr(img, "is_animated", False)))
```

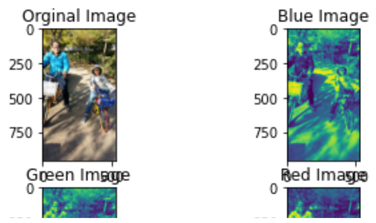
```
Filename:  /content/drive/MyDrive/my_Lecture_Note/Computer Vision/opencv-python-free-cours
Format:  JPEG
Image Patette: None
Mode:  RGB
Size:  (528, 960)
Width:  528
Height:  960
Is Animated:  False
```



Display plot, subplot, and title

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoCycle.jpg')
img1 = img[:, :, ::-1]

plt.subplot(221), plt.imshow(img1), plt.title('Orginal Image')
plt.subplot(222), plt.imshow(img1[:, :, 0]), plt.title('Blue Image')
plt.subplot(223), plt.imshow(img1[:, :, 1]), plt.title('Green Image')
plt.subplot(224), plt.imshow(img1[:, :, 2]), plt.title('Red Image')
```

```
(<AxesSubplot:title={'center':'Red Image'}>,
 <matplotlib.image.AxesImage at 0x7fa48455e340>,
 Text(0.5, 1.0, 'Red Image'))
```



Python Pillow – Create Image

Pillow Link

Draw a simple image with one color

To create a new image using Python Pillow PIL library, use Image.new() method. The syntax of Image.new() method is new(mode, size, color=0) where

mode:

mode is the image mode. For example RGB, RGBA, CMYK, etc.

Size:

size is the tuple with width and height of the image as elements. Width and height values are in pixels.

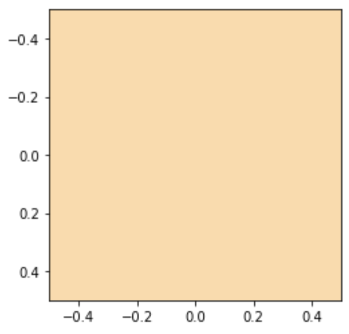
Color:

color is for painting all the pixels with. Based on the mode, values can be provided for each band, altogether as a tuple. color parameter is optional and the default value of color is 0.

Search google engine to "RGB Color Picker"

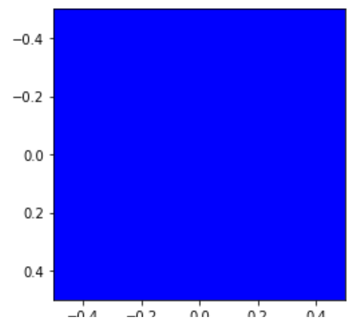
```
from PIL import Image

import cv2
import numpy as np
import matplotlib.pyplot as plt

#img = Image.new('RGB', (3, 3), color = 'green')
img = Image.new('CMYK', (1, 1), color = (0, 31, 77, 6))

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fa61c47b1f0>
```



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

a = cv2.imread('/content/drive/MyDrive/20220407FBAI/cb.jpg')
plt.imshow(a)

print(a.size)
print(a)
```

```
3
[[[  0   0 254]]]
```



Import Libraries

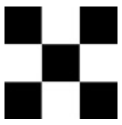
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from IPython.display import Image
```

Display Image Directly

We will use the following as our sample images. We will use the ipython image function to load and display the image.

```
# Display 18x18 pixel image.
Image(filename= '/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/checkerboard_18x18.png')
```

```
# Display 84x84 pixel image.
Image(filename='/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/checkerboard_84x84.jpg')
```



Reading images using OpenCV

OpenCV allows reading different types of images (JPG, PNG, etc). You can load grayscale images, color images or you can also load images with Alpha channel. It uses the **`cv2.imread()`** function which has the following syntax:

Function Syntax

```
retval = cv2.imread( filename, flags )
```

`retval`: Is the image if it is successfully loaded. Otherwise it is `None`. This may happen if the filename is wrong or the file is corrupt.

The function has **1 required input argument** and one optional flag:

1. `filename`: This can be an **absolute** or **relative** path. This is a **mandatory argument**.
2. `Flags`: These flags are used to read an image in a particular format (for example, grayscale/color/with alpha channel). This is an **optional argument** with a default value of `cv2.IMREAD_COLOR` or `1` which loads the image as a color image.

Before we proceed with some examples, let's also have a look at some of the `flags` available.

Flags

1. **`cv2.IMREAD_GRAYSCALE`** or **`0`**: Loads image in grayscale mode
2. **`cv2.IMREAD_COLOR`** or **`1`**: Loads a color image. Any transparency of image will be neglected. It is the default flag.
3. **`cv2.IMREAD_UNCHANGED`** or **`-1`**: Loads image as such including alpha channel.

OpenCV Documentation

`Imread`: https://docs.opencv.org/4.5.1/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56

`ImreadModes`: https://docs.opencv.org/4.5.1/d8/d6a/group__imgcodecs__flags.html#ga61d9b0126a3e57d9277ac48327799c80

```
# Read image as gray scale, Image = checkerboard_18x18.png.
cb_img = cv2.imread("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/checkerboard_18x18.png", 0)

# Print the image data (pixel values), element of a 2D numpy array.
# Each pixel value is 8-bits [0,255]
print(cb_img)
```

```
[[  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [255 255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255 255]
 [255 255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255 255]
 [255 255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255 255]
 [255 255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255 255]
 [255 255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255 255]
 [255 255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255 255]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]
 [  0   0   0   0   0   0 255 255 255 255 255 255   0   0   0   0   0   0]]
```

Display Image attributes

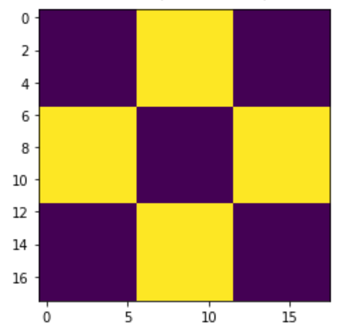
```
# print the size  of image
print("Image size is ", cb_img.shape)

# print data-type of image
print("Data type of image is ", cb_img.dtype)
```

```
Image size is  (18, 18)
Data type of image is  uint8
```

Display Images using Matplotlib

```
# Display image.
plt.imshow(cb_img)
```

```
<matplotlib.image.AxesImage at 0x7f5324be6c90>
```

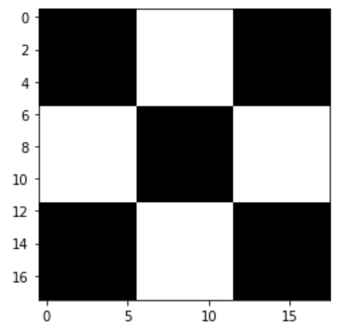


What happened?

Even though the image was read in as a gray scale image, it won't necessarily display in gray scale when using `imshow()`. matplotlib uses different color maps and it's possible that the gray scale color map is not set.

```
# Set color map to gray scale for proper rendering.
plt.imshow(cb_img, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f5324b55b10>
```



Another example

```
# Read image as gray scale, Image = checkerboard_fuzzy_18x18.jpg.
cb_img_fuzzy = cv2.imread("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/checkerboard_fuzzy_18x18.jpg",0)

# print image
print(cb_img_fuzzy)

# Display image.
plt.imshow(cb_img_fuzzy,cmap='gray')
```

```
[[  0   0  15  20   1 134 233 253 253 253 255 229 130   1  29   2   0   0]
 [  0   1   5  18   0 137 232 255 254 247 255 228 129   0  24   2   0   0]
 [  7   5   2  28   2 139 230 254 255 249 255 226 128   0  27   3   2   2]
 [ 25  27  28  38   0 129 236 255 253 249 251 227 129   0  36  27  27  27]
 [  2   0   0   4   2 130 239 254 254 254 255 230 126   0   4   2   0   0]
```

Working with Color Images

Until now, we have been using gray scale images in our discussion. Let us now discuss color images.

```
 [254 255 255 252 255 226 104   0   1   1   0 120 229 255 255 254 255 255]
```

```
# Read and display Coca-Cola logo.
Image("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/coca-cola-logo.png")
```

Read and display color image

Let us read a color image and check the parameters. Note the image dimension.

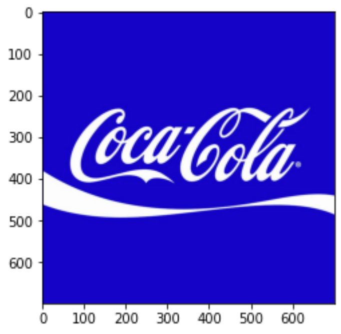
```
# Read in image
coke_img = cv2.imread("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/coca-cola-logo.png",1)

# print the size  of image
print("Image size is ", coke_img.shape)

# print data-type of image
print("Data type of image is ", coke_img.dtype)

print("")
```

```
Image size is  (700, 700, 3)
Data type of image is  uint8
```

Display the Image

```
plt.imshow(coke_img)
#  What happened?
```

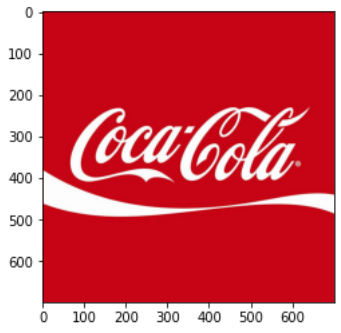
```
<matplotlib.image.AxesImage at 0x7f5324d296d0>
```



The color displayed above is different from the actual image. This is because matplotlib expects the image in RGB format whereas OpenCV stores images in BGR format. Thus, for correct display, we need to reverse the channels of the image. We will discuss about the channels in the sections below.

```
coke_img_channels_reversed = coke_img[:, :, ::-1]
plt.imshow(coke_img_channels_reversed)
```

```
<matplotlib.image.AxesImage at 0x7f5324c47f90>
```



Splitting and Merging Color Channels

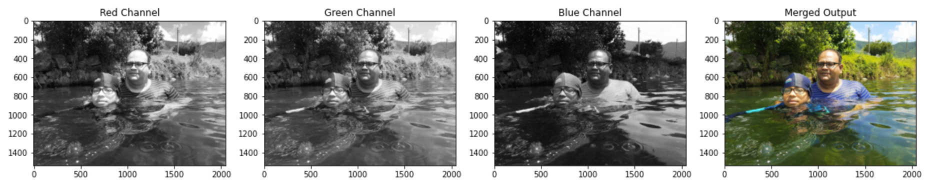
`cv2.split()` Divides a multi-channel array into several single-channel arrays.

`cv2.merge()` Merges several arrays to make a single multi-channel array. All the input matrices must have the same size.

OpenCV Documentation

https://docs.opencv.org/4.5.1/d2/de8/group__core__array.html#ga0547c7fed86152d7e9d0096029c8518a

```
# Split the image into the B,G,R components
img_NZ_bgr = cv2.imread("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoWater.jpg",cv2.IMREAD_COLOR)
b,g,r = cv2.split(img_NZ_bgr)

# Show the channels
plt.figure(figsize=[20,5])
plt.subplot(141);plt.imshow(r,cmap='gray');plt.title("Red Channel");
plt.subplot(142);plt.imshow(g,cmap='gray');plt.title("Green Channel");
plt.subplot(143);plt.imshow(b,cmap='gray');plt.title("Blue Channel");

# Merge the individual channels into a BGR image
imgMerged = cv2.merge((b,g,r))
# Show the merged output
plt.subplot(144);plt.imshow(imgMerged[:,:,::-1]);plt.title("Merged Output");
```



Converting to different Color Spaces

`cv2.cvtColor()` Converts an image from one color space to another. The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.

Function Syntax

```
dst = cv2.cvtColor( src, code )
```

`dst` : Is the output image of the same size and depth as `src` .

The function has **2 required arguments**:

1. `src` input image: 8-bit unsigned, 16-bit unsigned (CV_16UC...), or single-precision floating-point.
2. `code` color space conversion code (see ColorConversionCodes).

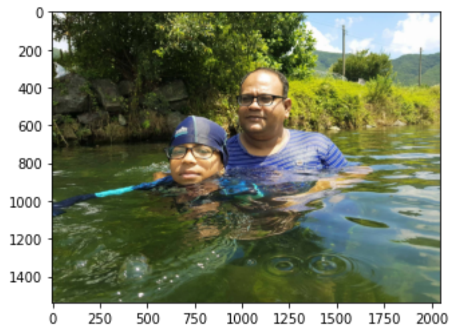
OpenCV Documentation

`cv2.cvtColor`: https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab
`ColorConversionCodes`:
https://docs.opencv.org/4.5.1/d8/d01/group__imgproc__color__conversions.html#ga4e0972be5de079fed4e3a10e24ef5ef0

Changing from BGR to RGB

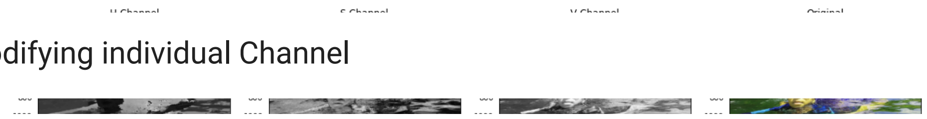
```
# OpenCV stores color channels in a differnet order than most other applications (BGR vs RGB).
img_NZ_rgb = cv2.cvtColor(img_NZ_bgr, cv2.COLOR_BGR2RGB)
plt.imshow(img_NZ_rgb)
```

```
<matplotlib.image.AxesImage at 0x7f53240fd210>
```



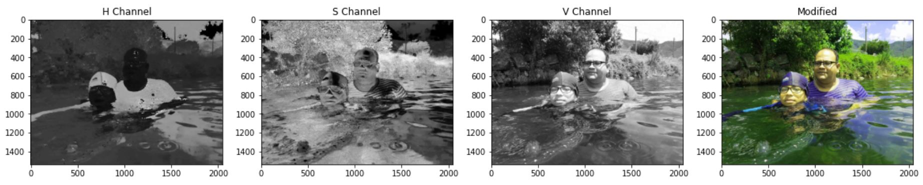
Changing to HSV color space

```
img_hsv = cv2.cvtColor(img_NZ_bgr, cv2.COLOR_BGR2HSV)
# Split the image into the B,G,R components
h,s,v = cv2.split(img_hsv)

# Show the channels
plt.figure(figsize=[20,5])
plt.subplot(141);plt.imshow(h,cmap='gray');plt.title("H Channel");
plt.subplot(142);plt.imshow(s,cmap='gray');plt.title("S Channel");
plt.subplot(143);plt.imshow(v,cmap='gray');plt.title("V Channel");
plt.subplot(144);plt.imshow(img_NZ_rgb);plt.title("Original");
```



Modifying individual Channel

```
h_new = h+10
img_NZ_merged = cv2.merge((h_new,s,v))
img_NZ_rgb = cv2.cvtColor(img_NZ_merged, cv2.COLOR_HSV2RGB)

# Show the channels
plt.figure(figsize=[20,5])
plt.subplot(141);plt.imshow(h,cmap='gray');plt.title("H Channel");
plt.subplot(142);plt.imshow(s,cmap='gray');plt.title("S Channel");
plt.subplot(143);plt.imshow(v,cmap='gray');plt.title("V Channel");
plt.subplot(144);plt.imshow(img_NZ_rgb);plt.title("Modified");
```



Display Modifyed Images Intensity Values

```
print(h)
print(h+10)
```

```
[[120 120   0 ... 103 102 102]
 [120   0   0 ... 103 103 102]
 [  0   0   0 ... 104 104 103]
 ...
 [ 40  40  40 ...  42  41  41]
 [ 40  40  40 ...  42  41  41]
 [ 40  40  40 ...  42  41  41]]
[[130 130  10 ... 113 112 112]
 [130  10  10 ... 113 113 112]
 [ 10  10  10 ... 114 114 113]
 ...
 [ 50  50  50 ...  52  51  51]
 [ 50  50  50 ...  52  51  51]
 [ 50  50  50 ...  52  51  51]]
```

Saving Images

Saving the image is as trivial as reading an image in OpenCV. We use the function **`cv2.imwrite()`** with two arguments. The first one is the filename, second argument is the image object.

The function imwrite saves the image to the specified file. The image format is chosen based on the filename extension (see cv::imread for the list of extensions). In general, only 8-bit single-channel or 3-channel (with 'BGR' channel order) images can be saved using this function (see the OpenCV documentation for further details).

Function Syntax

```
cv2.imwrite( filename, img[, params] )
```

The function has **2 required arguments**:

1. `filename` : This can be an **absolute** or **relative** path.
2. `img` : Image or Images to be saved.

OpenCV Documentation

`Imwrite:` https://docs.opencv.org/4.5.1/d4/da8/group__imgcodecs.html#gabbc7ef1aa2edfaa87772f1202d67e0ce
`ImwriteFlags:` https://docs.opencv.org/4.5.1/d8/d6a/group__imgcodecs__flags.html#ga292d81be8d76901bff7988d18d2b42ac

```
# save the image
cv2.imwrite("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoWater_SAVED.png", img_NZ_bgr)
```

```
#Image(filename='/content/drive/MyDrive/my_Lecture_Note/Computer Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shc

import cv2
grey_img = cv2.imread('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoBay.jpg', cv2.IMREAD_GRAYSCALE)

# save image
status = cv2.imwrite('/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/shocchoBay.png',grey_img)

print("Image written to file-system : ",status)
```

```
Image written to file-system :  True
```

```
# read the image as Color
img_NZ_bgr = cv2.imread("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/New_Zealand_Lake_SAVED.png", cv2.IMREAD_COLOR)
print("img_NZ_bgr shape is: ", img_NZ_bgr.shape)

# read the image as Grayscaled
img_NZ_gry = cv2.imread("New_Zealand_Lake_SAVED.png", cv2.IMREAD_GRAYSCALE)
print("img_NZ_gry shape is: ", img_NZ_gry.shape)
```

```
img_NZ_bgr shape is:  (600, 840, 3)
img_NZ_gry shape is:  (1536, 2048)
```

Create custom image using two different sources

```
import cv2
import matplotlib.pyplot as plt

cb_img = cv2.imread("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/checkerboard_color.png")
coke_img = cv2.imread("/content/drive/MyDrive/my_Lecture_Note/Computer \
Vision/opencv-python-free-course-code/01_Getting_Started_with_Images/coca-cola-logo.png")

# Use matplotlib imshow()
plt.imshow(cb_img)
plt.title("matplotlib imshow")
plt.show()

# Use OpenCV imshow(), display for 8 sec
window1 = cv2.namedWindow("w1")
cv2.imshow(window1, cb_img)
cv2.waitKey(2000)
cv2.destroyWindow(window1)

# Use OpenCV imshow(), display for 8 sec
window2 = cv2.namedWindow("w2")
cv2.imshow(window2, coke_img)
cv2.waitKey(8000)
cv2.destroyWindow(window2)

# Use OpenCV imshow(), display until any key is pressed
window3 = cv2.namedWindow("w3")
cv2.imshow(window3, cb_img)
cv2.waitKey(0)
cv2.destroyWindow(window3)

window4 = cv2.namedWindow("w4")

Alive = True
while Alive:
    # Use OpenCV imshow(), display until 'q' key is pressed
    cv2.imshow(window4, coke_img)
    keypress = cv2.waitKey(1)
    if keypress == ord('q'):
        Alive = False
cv2.destroyWindow(window4)

cv2.destroyAllWindows()
stop = 1
```

✓ 2s completed at 2:38 PM ● ×