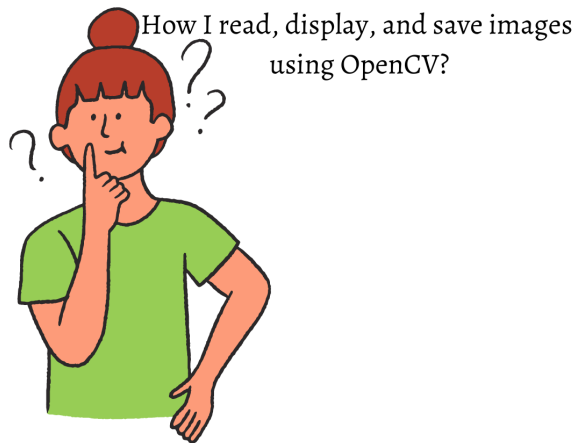


How to read, display, and save images with OpenCV in Python




Hi, you are ready to learn OpenCV Library basic operations such as reading images and saving images in your filesystem. And also you don't know That part is very exciting, Why because you enjoy most of the time real coding experience. Q: You are happy to go with me?, A: Ok let's go.


Why do most computer vision experts Use OpenCV?

Computer Vision, Machine Learning, and Image Processing all of the area using a huge open-source library named OpenCV. Important parts of the library achieve real-time operation which is the most important today in our system. That library also supports to integration various of libraries such as NumPy.



Download code — You follow this article to first step download all material including images and notebook. [Link here](#) 

In this article, I cover this topic

- Install the OpenCV library and most other important libraries.
- Read and display images in your notebook.
- Read Images and Display pixel level of data
- Data type and shape of the image you see.
- Saving images in the file manager. 

Install the OpenCV library and most other important libraries.

```
!pip install opencv-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting opencv-python
  Downloading opencv_python-4.6.0.66-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (60.9 MB)
    |████████████████████████████████████████| 60.9 MB 11.4 MB/s
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python) (1.21.6)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.6.0.66
```

If you use google colab so don't install OpenCV Library, because already installed.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
# Tells notebook to render figures in-page.
%matplotlib inline
from IPython.display import Image
```

Read and display images in your notebook.

There are multiple ways to display images dependent on where we are working on. In this notebook, we are using two methods for displaying images. One is the direct get file manager and display images in Jupiter Notebook. Another option is to Read the image into Numpy Array using OpenCV and Display the image with matplotlib help full method using the name **imshow()**. This option is often helpful and simple to display images.

Note: OpenCV also has **imshow()** method, But That is recommended for using python script, not for Jupiter notebook.

Displaying images directly in the notebook

Let's start with displaying two different sizes of images. We will use **IPython Image()** function to achieve this. That is a very opportunity way to read and display images with a single command. When this function displays the image directly into the notebook. using this method actual size of the image is.

```
# Display 18 x 18 pixel image.
# This file path is not for my actual file manager. this is a google drive path because I am using google colab

Image(filename="drive/MyDrive/OpenCV-Article/module_1/black_and_white_18x18.png")
```



```
# Display 70 x 70 pixel image
Image(filename="drive/MyDrive/OpenCV-Article/module_1/black_and_white_70x70.png")
```



Reading Images with OpenCV

We are now using OpenCV to help full function to read images name **imread()**. This function allows us to read different types of images such as (PNG, JPG, etc...). You can read color, grayscale, and an alpha channel image.

Function Syntax : `retval = cv.imread(filename[, flags])`

In this function 1 required argument and one optional flag:

1. **filename:** That is an absolute location where your image is stored. This is a mandatory argument. if you put the wrong file path that time function return provides you **None**.
2. **flag:** These values are used in your image's particular format for example (**grayscale, and color with alpha channel**). This is an optional argument with a default value **cv2.IMREAD_COLOR** or **1** which loads image color image.

If you learn more about this function and flag so go to OpenCV Documentation: [Imread\(\)](#), [ImreadModes\(\)](#).

Note — This function Reads in BGR format. When you read a color image that time load as BGR format means your image and that image are different. Always make sure when you display the image to convert the actual format. That thing helps as the [cv2.cvtColor\(\)](#) function in OpenCV.

Read Images and Display pixel level of data

Let's read 18x18 color images and display our notebook pixel-level data. These data are associated with each pixel's representation of the pixel intensity. Most of the image is stored in an 8-bit unsigned integer(**uint8**) So our intensity value range is 0 to 255. 0 represents pure black and 255 represents pure white.

```
# Read as GrayScale Image
gray_img = cv2.imread("drive/MyDrive/OpenCV-Article/module_1/black_and_white_18x18.png", cv2.IMREAD_GRAYSCALE)

# Print the image data (pixel value) of a 2D-Numpy Array
print(gray_img)
```

```
[
  [255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 254 198 170 170 169 168 168 168 169 170 171 175 235 255 255 255],
  [255 255 253 167 137 152 150 136 124 129 145 152 144 137 224 255 255 255],
  [255 255 253 167 129 132 131 130 129 130 132 133 129 133 224 255 255 255],
  [255 255 253 167 129 131 130 130 128 128 132 133 126 133 224 255 255 255],
  [255 255 253 167 139 135 131 130 129 129 133 130 119 130 224 255 255 255],
  [255 255 252 172 140 131 131 129 129 130 132 130 126 125 224 255 255 255],
  [255 255 253 170 128 122 121 129 129 132 128 121 127 133 224 255 255 255],
  [255 255 253 165 135 138 133 128 128 130 138 141 127 132 224 255 255 255],
  [255 255 252 170 141 133 134 127 128 131 126 129 114 130 224 255 255 255],
  [255 255 253 173 132 113 114 131 129 131 124 114 127 123 224 255 255 255],
  [255 255 253 167 126 144 144 130 129 128 139 147 131 132 224 255 255 255],
  [255 255 253 167 123 121 121 126 129 129 120 122 122 134 224 255 255 255],
  [255 255 253 166 130 121 122 131 128 129 128 118 130 134 224 255 255 255],
  [255 255 253 167 128 141 139 129 128 128 134 140 130 132 224 255 255 255],
  [255 255 253 168 130 137 137 133 130 131 135 138 133 134 224 255 255 255],
  [255 255 253 179 149 156 155 147 139 143 152 156 153 151 228 255 255 255],
  [255 255 255 249 245 244 243 243 243 243 243 244 244 246 253 255 255 255]]
```

Data type and shape of the image you see.

```
print("Image Shape is: ",gray_img.shape)
print("Image data type is: ",gray_img.dtype)

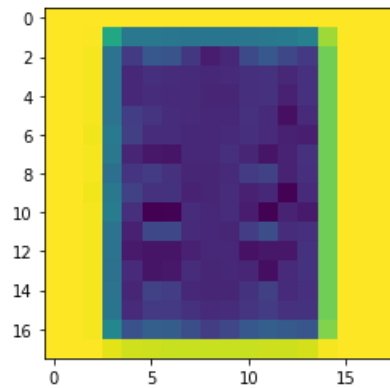
"""
OUTPUT >>> Image Shape is: (18, 18)
           Data type is: uint8
"""
```

Matplotlib using to display image data

Now our image is stored in NumPy 2d-array we can use the matplotlib **imshow()** function to display the image. There down below the image looked at and noted down a couple of things.

1. Image color representation is unexpected.
2. This Image represent is mathematically (essentially plots the image data). This image shape is 18x18 pixels as indicated matplotlib plot axis. It's the physical size of the browser much larger. This is very suitable so we can display images of different sizes at a proper scale viewing. Down below plot size show configurable using `plt.figure(figsize=[width, height])`.

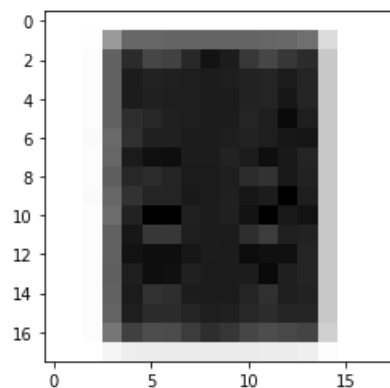
```
# Display the image
plt.imshow(gray_img);
```



What happen to change my image color?

Everyone knows the image was read as a grayscale image. When you use the matplotlib **imshow()** function that won't necessarily to a grayscale image show. matplotlib randomly picks different color maps to convert pixel intensity. we can get a proper image as grayscale so specify a color map. You can see our down below image set to the proper color map is gray one optional argument in **imshow()** function under and see what looks like.

```
plt.imshow(gray_img, cmap="gray")
```



Saving Images in the file manager.

Image saving is a very simple and straightforward part because OpenCV helps with one function **imwrite()**. This will be saved in a specified format in the current directory. If you learn more about this function so go to OpenCV library [imwrite\(\)](#).

Function Syntax: cv2.imwrite(filename, image)

This function 2 Required the argument:

- **filename:** This file name represent a string. must be file name included file extension such as **.jpg, .png, etc.**

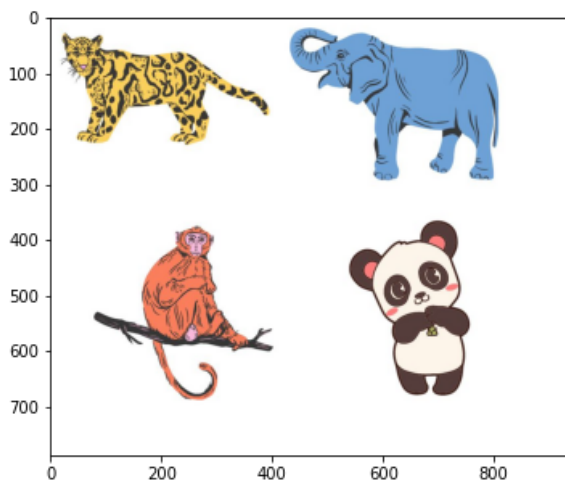
- **image:** which image you save that image put this argument.
- **Return:** This Function Return True when successfully saving your image.

Now time sees one example to clearly understand what this function is.

Read New Image

```
# Read New Images and Display
img_bgr = cv2.imread("drive/MyDrive/OpenCV-Article/module_1/open-cv-module1-imwrite.jpg",cv2.IMREAD_COLOR)
# convert BGR to RGB
img_rgb = cv2.cvtColor(img_bgr,cv2.COLOR_BGR2RGB)
# Another way to convert BGR to RGB
# img_rgb = img_bgr[:,::-1] # if you go that way so uncomment that line and comment above code

plt.figure(figsize=(10,5))
plt.imshow(img_rgb);
```



Saving images in the file manager. 📁

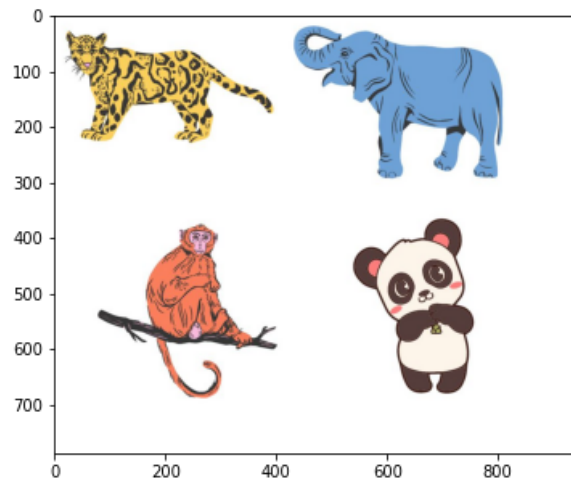
```
# Save the image in the filesystem
cv2.imwrite("animal.png",img_bgr) # I put BGR format image this function convert as RGB then save it.

# OUTPUT >>> True
```

Read the saved image and display our notebook

```
# Read the saved image
save_img = cv2.imread("animal.png",cv2.IMREAD_COLOR)

plt.figure(figsize=(10,5))
plt.imshow(save_img[:,::-1]) # image show and convert BGR to RGB
```



Thanks for reading. If you have any questions comment now below.