Contents:

Loading and displaying an image

```
import cv2
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving flower2.jpg to flower2.jpg

```
image = cv2.imread("flower2.jpg")
from google.colab.patches import cv2_imshow
cv2_imshow(image)
```

```
print("Image Properties")
print("- Number of Pixels: " + str(image.size))
print("- Shape/Dimensions: " + str(image.shape))
```

```
print("- Shape/Dimensions: " + str(image.shape))
sz=image.size
print("- Size={}".format(sz))
(h, w, d) = image.shape
print("width={}, height={}, depth={}".format(w, h, d))
```

```
Image Properties
- Number of Pixels: 245760
- Shape/Dimensions: (256, 320, 3)
- Size=245760
width=320, height=256, depth=3
```

Accessing individual pixels

In OpenCV color images in the RGB (Red, Green, Blue) color space have a 3-tuple associated with each pixel: (B, G, R) Each value in the BGR 3-tuple has a range of [0, 255] . Color possibilities are for each pixel in an RGB image in OpenCV== 256 * 256 * 256 = 16777216 .

```
(B, G, R) = image[100, 50] #access the RGB pixel located at x=50, y=100
print("R={}, G={}, B={}".format(R, G, B))
```

```
R=239, G=141, B=242
```

Array slicing and cropping

Extract a 100x100 pixel square ROI (Region of Interest) from the input image starting at x=50,y=40 at ending at x=220,y=150

image[startY:endY, startX:endX]

```
roi = image[40:150, 50:220]
cv2_imshow(roi)
```

Resizing images

```
resized = cv2.resize(image, (200, 150))
```

```
cv2_imshow(image)
cv2_imshow(resized)
```

```
# fixed resizing distorts aspect ratio so let's resize the width to be 300px but compute
#  the new height based on the aspect ratio

r = 500.0 / w
dim = (500, int(h * r))
resized = cv2.resize(image, dim)
cv2_imshow(resized)
```

Computing the aspect ratio each time we want to resize an image is a bit tedious, so we can use function within imutils.

```
import imutils as im
resized = im.resize(image, width=200)
cv2_imshow(resized)
```

Rotating an image

Using imutils rotating becomes very easy

```
rotated = im.rotate(image, -45)
cv2_imshow(rotated)
```

```
# rotated image is clipped after rotation
# so use other method of imutils to keep the entire image in view.
rotated = im.rotate_bound(image, 45)
cv2_imshow(rotated)
```

Converting an image to grayscale

```
# convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray)
```

Drawing on an image

To draw a rectangle, circle, and line on an input image and also overlay text on an image Syntax: cv2.rectangle(image, start_point, end_point, color, thickness)

Draw a 2px thick red rectangle surrounding the face

```
output = image.copy()
cv2.rectangle(output, (75, 70), (270,220), (40, 100, 255), 2)
cv2_imshow(output)
```

```
#Syntax: cv2.circle(image, center_coordinates, radius, color, thickness)
# draw a blue 20px (filled in) circle on the image centered at
# x=300,y=150
output = image.copy()
cv2.circle(output, (140, 90), 20, (255, 0, 0), -1)
cv2_imshow(output)
```

```
# Syntax: cv2.line(image, start_point, end_point, color, thickness)
# draw a 5px thick red line from x=60,y=20 to x=400,y=200
output = image.copy()
```

```
cv2.line(output, (60, 20), (400, 350), (0, 255, 0), 4)
cv2_imshow(output)
```

```
# Syntax: cv2.putText(image, text, org, font, fontScale, color[, thickness[, lineType[, botto
# draw green text on the image
output = image.copy()
cv2.putText(output, "Beautiful!!!", (100, 25),  cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, 0.8, (0, 0,
cv2_imshow(output)
```


```
import numpy as np
img1=np.flipud(image) # flip vertically
cv2_imshow(img1)
```

```
img2=np.fliplr(image) #flip horizontally
cv2_imshow(img2)
```