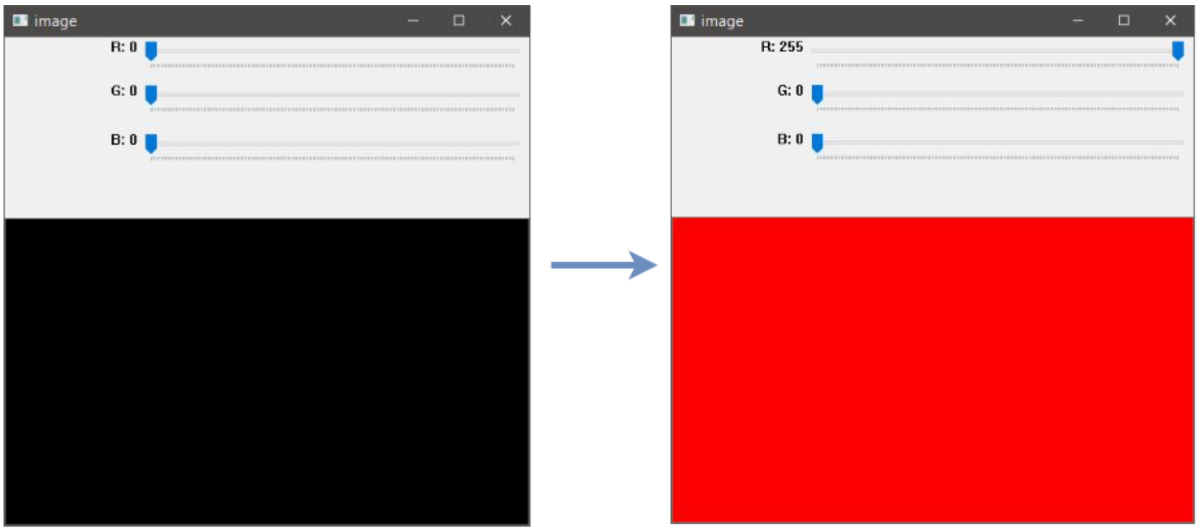
# Creating TrackBars

**TrackBars are just a fancy name for sliders in OpenCV**

Similar to MouseCallBack Function trackbars also require us to have a predefined window

In [2]:

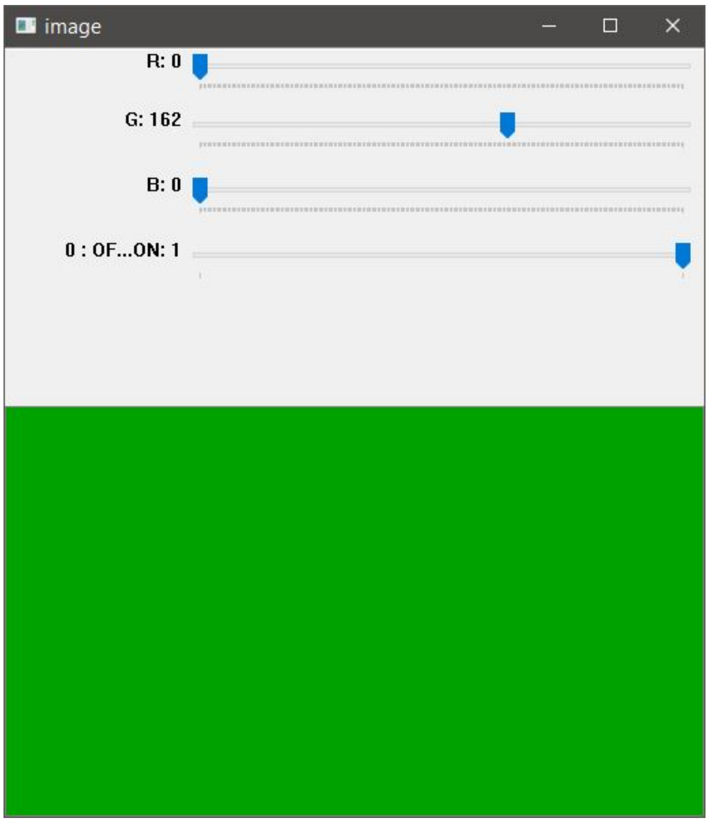
```python
import cv2
import numpy as np

def nothing(x):
    pass

# Create a black image, a window
img = np.zeros((300,512,3), np.uint8)
cv2.namedWindow('image')

# create trackbars for color change
cv2.createTrackbar('R','image',0,255,nothing)
cv2.createTrackbar('G','image',0,255,nothing)
cv2.createTrackbar('B','image',0,255,nothing)

while True:

    # get current positions of four trackbars
    r = cv2.getTrackbarPos('R','image')
    g = cv2.getTrackbarPos('G','image')
    b = cv2.getTrackbarPos('B','image')

    img[:] = [b,g,r]

    cv2.imshow('image',img)

    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break

cv2.destroyAllWindows()
```



## Practice Problem : Add an Additional TrackBar which will act as a switch

In [3]:

```
import cv2
import numpy as np

def nothing(x):
    pass

# Create a black image, a window
img = np.zeros((300,512,3), np.uint8)
cv2.namedWindow('image')

# create trackbars for color change
cv2.createTrackbar('R','image',0,255,nothing)
cv2.createTrackbar('G','image',0,255,nothing)
cv2.createTrackbar('B','image',0,255,nothing)

# create switch for ON/OFF functionality
switch = '0 : OFF \n1 : ON'
cv2.createTrackbar(switch, 'image',0,1,nothing)

while(1):
    cv2.imshow('image',img)
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break

    # get current positions of four trackbars
    r = cv2.getTrackbarPos('R','image')
    g = cv2.getTrackbarPos('G','image')
    b = cv2.getTrackbarPos('B','image')
    s = cv2.getTrackbarPos(switch,'image')

    if s == 0:
        img[:] = 0
    else:
        img[:] = [b,g,r]

cv2.destroyAllWindows()
```



## Practice Problem : Using mouse as paint brush with variable size

In [ ]:

```
import cv2
import numpy as np


r,g,b,rad = 0,0,0,0
drawing = False
s = [0]

# Create a black image, a window
img = np.zeros((250,512,3), np.uint8)
img_copy = np.copy(img)
cv2.namedWindow('image')


def nothing(x):
    pass

def mouse_call(event,x,y,flag,s):
    global drawing,r,g,b,rad

    if event == cv2.EVENT_LBUTTONDOWN:
        print(r,g,b,rad,s[0])
        drawing = True

    if event == cv2.EVENT_MOUSEMOVE:
        if drawing and s[0]==1:
            cv2.circle(img,(x,y),rad,(b,g,r),-1)

    if event == cv2.EVENT_LBUTTONUP:
        drawing = False

cv2.setMouseCallback('image',mouse_call,s)

# create trackbars for color change
cv2.createTrackbar('R','image',0,255,nothing)
```

```
cv2.createTrackbar('G','image',0,255,nothing)
cv2.createTrackbar('B','image',0,255,nothing)
cv2.createTrackbar('Radius','image',5,50,nothing)

# create switch for ON/OFF functionality
switch = '0 : OFF \n1 : ON'
cv2.createTrackbar(switch, 'image',0,1,nothing)

while(1):
    cv2.imshow('image',img)
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break

    if k == ord('c'):
        img = np.copy(img_copy)
    # get current positions of four trackbars
    r = cv2.getTrackbarPos('R','image')
    g = cv2.getTrackbarPos('G','image')
    b = cv2.getTrackbarPos('B','image')
    rad = cv2.getTrackbarPos('Radius','image')
    s[0] = cv2.getTrackbarPos(switch,'image')

#     if s == 0:
#         img[:] = 0
#     else:
#         img[:] = [b,g,r]

cv2.destroyAllWindows()
```

```
In [ ]:
```