Reading Image

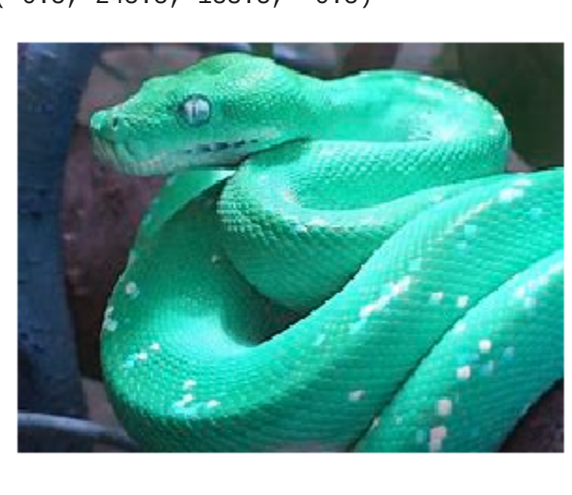
In [1]:
```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('test.jpeg')
plt.imshow(img)
```

Out[1]: `<matplotlib.image.AxesImage at 0x7f1abbf89ba8>`

Removing Axis

In [2]:
```
#to remove axis...
plt.imshow(img)
#plt.xticks([])
#plt.yticks([])
plt.axis('off')
```

Out[2]: `(-0.5, 245.5, 183.5, -0.5)`



Shape Of Image

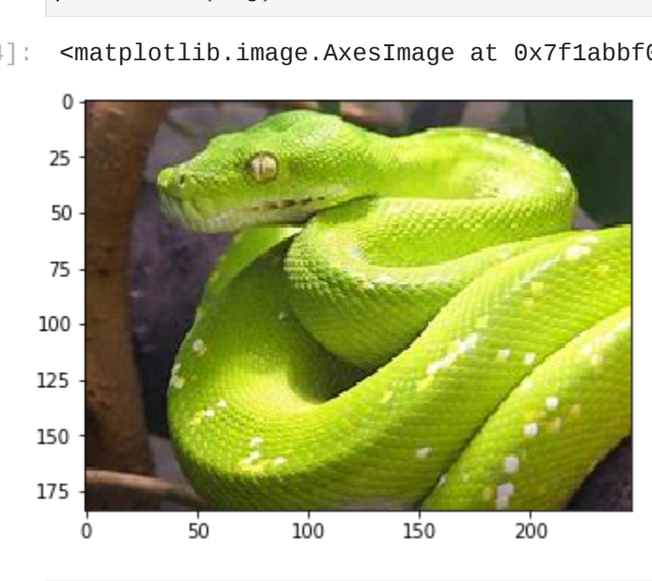
In [3]:
```
#to know the shape(resolution) of image
r,c=img.shape[:2]
print(r)
print(c)
```

```
184
246
```

BGR2RGB Conversion

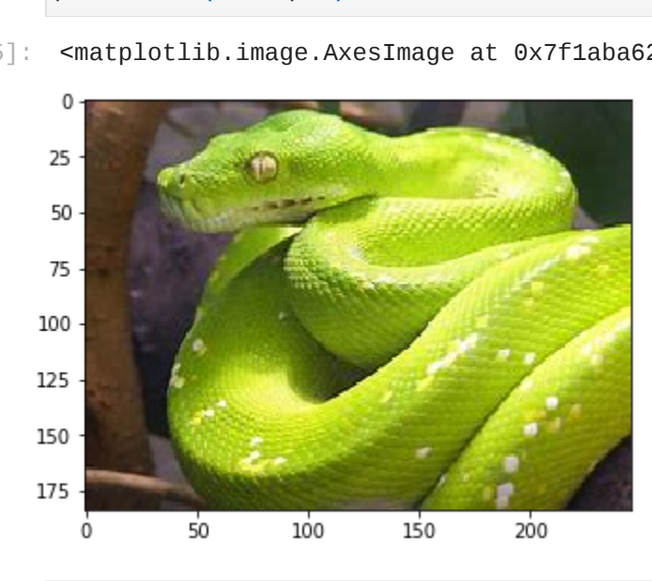
In [4]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('test.jpeg')
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

Out[4]: `<matplotlib.image.AxesImage at 0x7f1abbf044a8>`



In [5]:
```
k1 = np.array([[0,0,0],[0,1,0],[0,0,0]])
k1output = cv2.filter2D(img,-1,k1)
plt.imshow(k1output)
```

Out[5]: `<matplotlib.image.AxesImage at 0x7f1aba623eb8>`

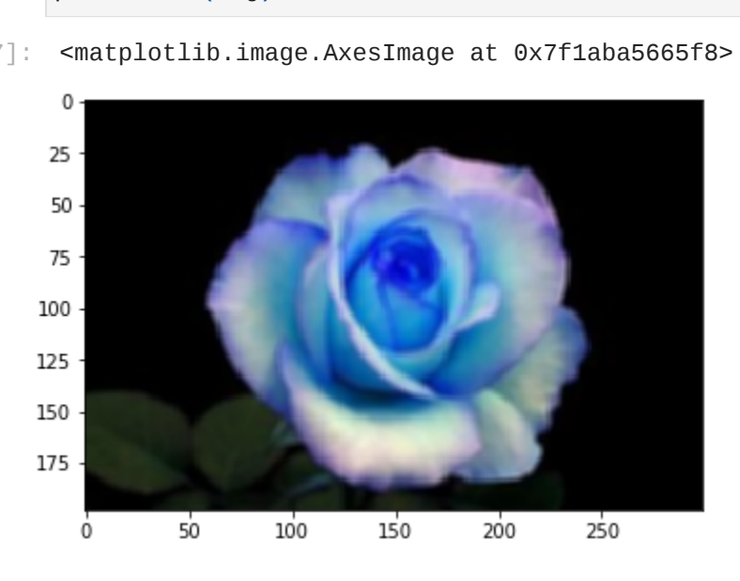


In [6]:
```
import cv2
import matplotlib.pyplot as plt
img=cv2.imread('img1.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

Out[6]: `<matplotlib.image.AxesImage at 0x7f1aba603c50>`

In [7]:
```
#Bluring
import numpy as np
k1=np.ones((4,4),np.float32)/16
img=cv2.filter2D(img,-1,k1)
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

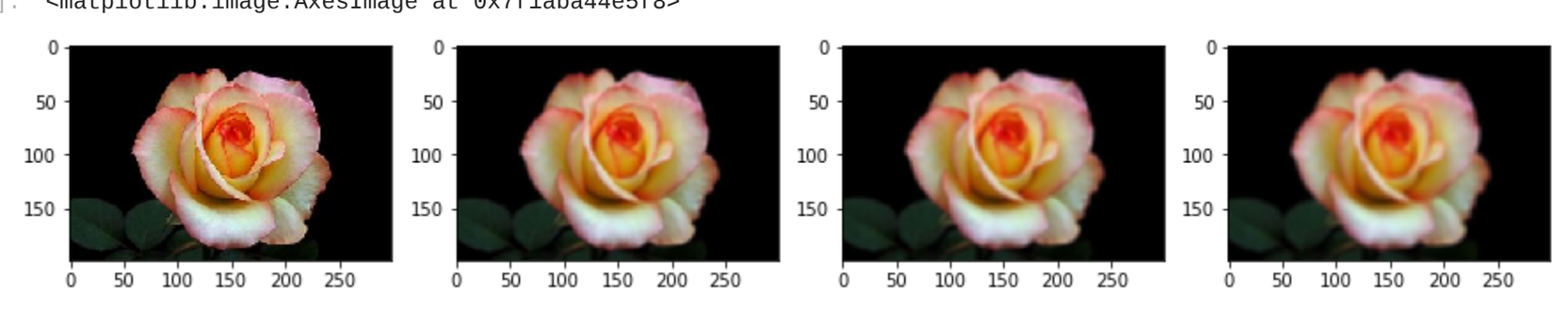
Out[7]: `<matplotlib.image.AxesImage at 0x7f1aba5665f8>`



Bluring

In [8]:
```
from matplotlib import rcParams
img=cv2.imread('img1.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k1=np.ones((4,4),np.float32)/16
k2=np.ones((5,5),np.float32)/25
k3=np.ones((6,6),np.float32)/36

img1=cv2.filter2D(img,-1,k1)
img2=cv2.filter2D(img,-1,k2)
img3=cv2.filter2D(img,-1,k3)
rcParams['figure.figsize'] = 15,8
fig,ax = plt.subplots(1,4)
ax[0].imshow(img)
ax[1].imshow(img1)
ax[2].imshow(img2)
ax[3].imshow(img3)
```

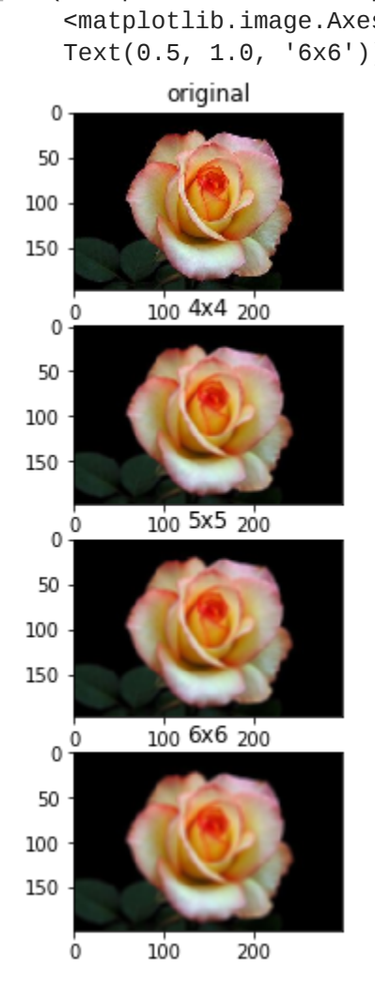
Out[8]: `<matplotlib.image.AxesImage at 0x7f1aba44e5f8>`



Bluring

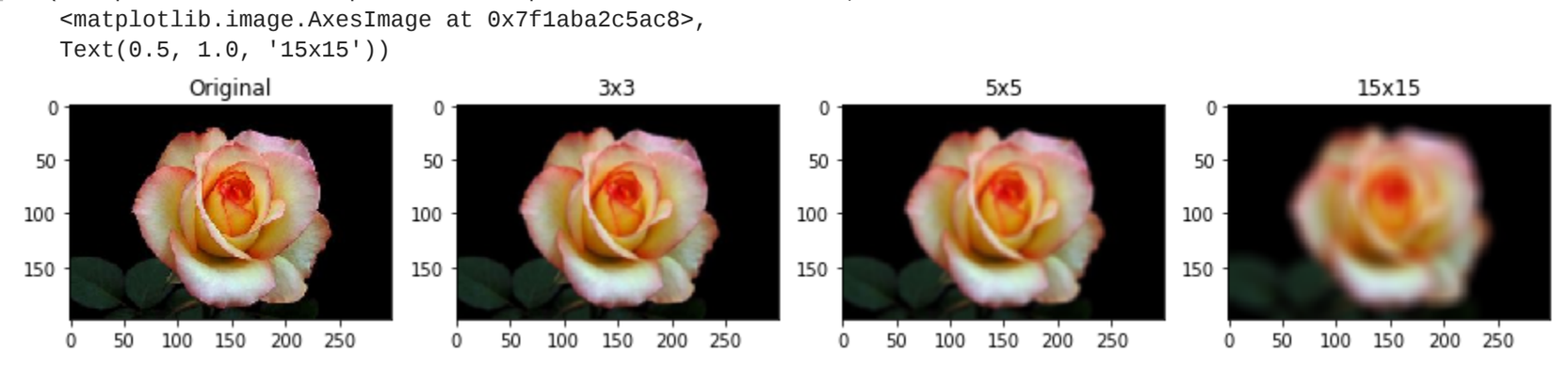
In [9]:
```
#from matplotlib import rcParams
img=cv2.imread('img1.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k1=np.ones((4,4),np.float32)/16
k2=np.ones((5,5),np.float32)/25
k3=np.ones((6,6),np.float32)/36

img1=cv2.filter2D(img,-1,k1)
img2=cv2.filter2D(img,-1,k2)
img3=cv2.filter2D(img,-1,k3)
#rcParams['figure.figsize'] = 15,8
plt.subplot(4,1,1),plt.imshow(img),plt.title('original')
plt.subplot(4,1,2),plt.imshow(img1),plt.title('4x4')
plt.subplot(4,1,3),plt.imshow(img2),plt.title('5x5')
plt.subplot(4,1,4),plt.imshow(img3),plt.title('6x6')
```

Out[9]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1aba42eeb8>,
 <matplotlib.image.AxesImage at 0x7f1aba39e550>,
 Text(0.5, 1.0, '6x6'))
```



In [10]:
```
img=cv2.imread('img1.jpeg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k3 = cv2.blur(img,(3,3))
k5 = cv2.blur(img,(5,5))
k9 = cv2.blur(img,(15,15))
plt.subplot(1,4,1),plt.imshow(img),plt.title('Original')
plt.subplot(1,4,2),plt.imshow(k3),plt.title('3x3')
plt.subplot(1,4,3),plt.imshow(k5),plt.title('5x5')
plt.subplot(1,4,4),plt.imshow(k9),plt.title('15x15')
```

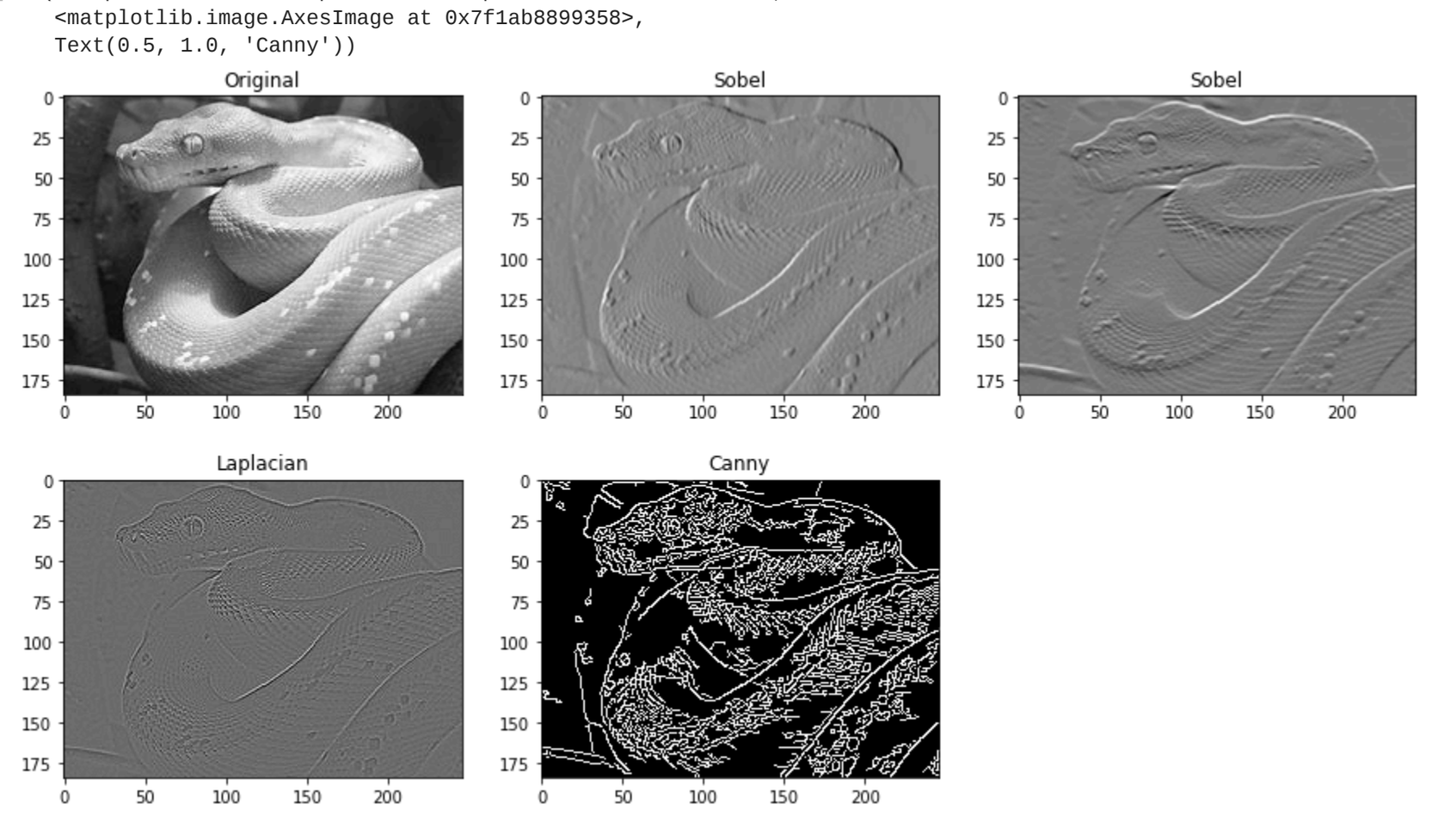
Out[10]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1aba2db438>,
 <matplotlib.image.AxesImage at 0x7f1aba2c5ac8>,
 Text(0.5, 1.0, '15x15'))
```



Edge Detaction

In [11]:
```
img = cv2.imread('test.jpeg',0)
img1 = cv2.Sobel(img,cv2.CV_64F,1,0,5)
img2 = cv2.Sobel(img,cv2.CV_64F,0,1,5)
img3 = cv2.Laplacian(img,cv2.CV_64F)
img4 = cv2.Canny(img,50,150)
plt.subplot(2,3,1),plt.imshow(img,cmap = 'gray'),plt.title('Original')
plt.subplot(2,3,2),plt.imshow(img1,cmap = 'gray'),plt.title('Sobel')
plt.subplot(2,3,3),plt.imshow(img2,cmap = 'gray'),plt.title('Sobel')
plt.subplot(2,3,4),plt.imshow(img3,cmap = 'gray'),plt.title('Laplacian')
plt.subplot(2,3,5),plt.imshow(img4,cmap = 'gray'),plt.title('Canny')
```

Out[11]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab88a8cc0>,
 <matplotlib.image.AxesImage at 0x7f1ab8899358>,
 Text(0.5, 1.0, 'Canny'))
```



Motion Blur(Horizontal)

In [12]:
```
img = cv2.cvtColor(cv2.imread('th.jpeg'),cv2.COLOR_BGR2RGB)
size=50
kernel_motion_blur = np.zeros((size, size))
kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
kernel_motion_blur = kernel_motion_blur / size
img1 = cv2.filter2D(img,-1,kernel_motion_blur)

plt.subplot(1,2,1),plt.imshow(img)
plt.subplot(1,2,2),plt.imshow(img1)
```

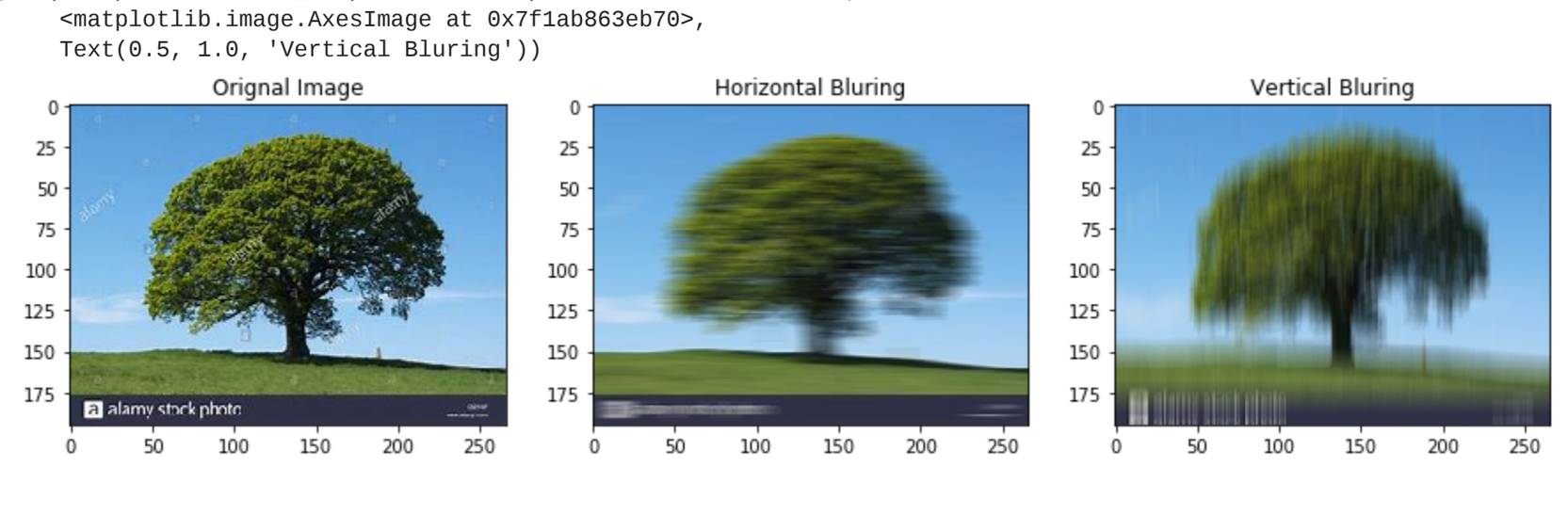
Out[12]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab8899828>,
 <matplotlib.image.AxesImage at 0x7f1ab8743630>)
```

Motion Blur(Vertical)

Motion Blur

In [13]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams
img=cv2.cvtColor(cv2.imread('tree.jpeg'),cv2.COLOR_BGR2RGB)
size=20

#Horizontal
k1=np.zeros((size,size))
k1[int(size/2),:]=np.ones(size)
k1=k1/size
img1=cv2.filter2D(img,-1,k1)

#vertical
k2=np.zeros((size,size))
k2[:,int(size/2)]=np.ones(size)
k2=k2/size
img2=cv2.filter2D(img,-1,k2)

#Increase Image size
rcParams['figure.figsize'] = 15,15

plt.subplot(1,3,1),plt.imshow(img),plt.title("Orignal Image")
plt.subplot(1,3,2),plt.imshow(img1),plt.title("Horizontal Bluring")
plt.subplot(1,3,3),plt.imshow(img2),plt.title("Vertical Bluring")
```

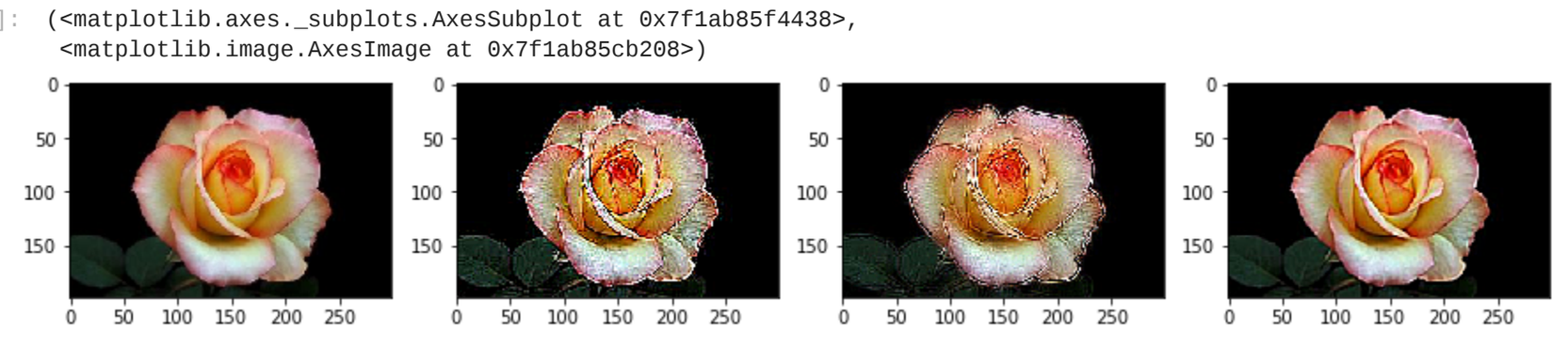
Out[13]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab86d95f8>,
 <matplotlib.image.AxesImage at 0x7f1ab863eb70>,
 Text(0.5, 1.0, 'Vertical Bluring'))
```



Sharpning

In [14]:
```
#sharpening
import cv2
import matplotlib.pyplot as plt
import numpy as np

img = cv2.cvtColor(cv2.imread('img1.jpeg'),cv2.COLOR_BGR2RGB)

k1 = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
k2 = np.array([[1,1,1],[1,-7,1],[1,1,1]])
k3 = np.array([[-1,-1,-1,-1,-1],[-1,2,2,2,-1],[-1,2,8,2,-1],[-1,2,2,2,-1],[-1,-1,-1,-1,-1]])/8.0
img1 = cv2.filter2D(img,-1,k1)
img2 = cv2.filter2D(img,-1,k2)
img3 = cv2.filter2D(img,-1,k3)

plt.subplot(1,4,1),plt.imshow(img)
plt.subplot(1,4,2),plt.imshow(img1)
plt.subplot(1,4,3),plt.imshow(img2)
plt.subplot(1,4,4),plt.imshow(img3)
```

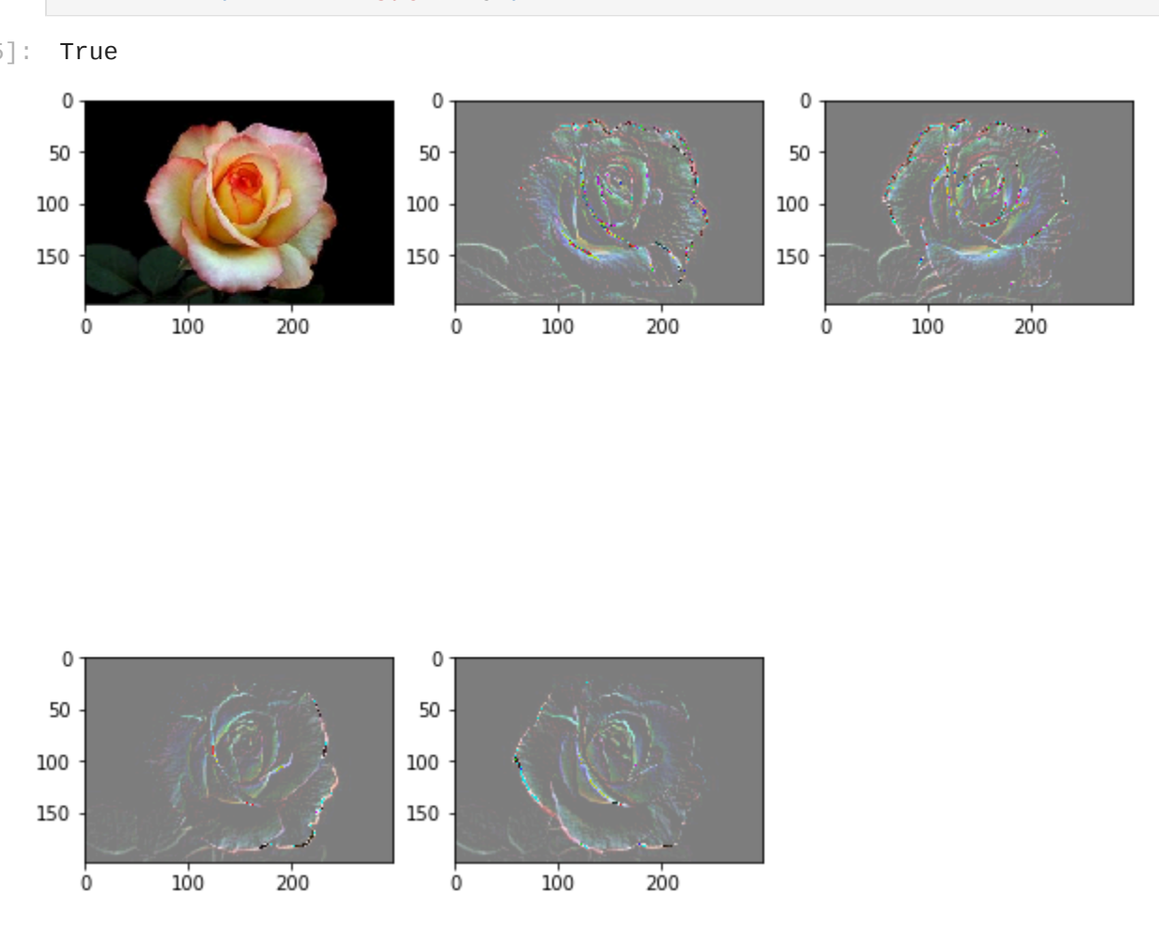
Out[14]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab85f4438>,
 <matplotlib.image.AxesImage at 0x7f1ab85cb208>)
```



Embossing

In [15]:
```
#Embossing
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams

#img = cv2.imread('img1.jpeg',0)
img = cv2.cvtColor(cv2.imread('img1.jpeg'),cv2.COLOR_BGR2RGB)

rcParams['figure.figsize']=10,10
k1 = np.array([[0,-1,-1],[1,0,-1],[1,1,0]])
k2 = np.array([[-1,-1,0],[-1,0,1],[0,1,1]])
k3 = np.array([[1,0,0],[0,0,0],[0,0,-1]])
k4 = np.array([[0,0,1],[0,0,0],[-1,0,0]])

img1 = cv2.filter2D(img,-1,k1) + 128
img2 = cv2.filter2D(img,-1,k2) + 128
img3 = cv2.filter2D(img,-1,k3) + 128
img4 = cv2.filter2D(img,-1,k4) + 128

plt.subplot(2,3,1),plt.imshow(img)
plt.subplot(2,3,2),plt.imshow(img1)
plt.subplot(2,3,3),plt.imshow(img2)
plt.subplot(2,3,4),plt.imshow(img3)
plt.subplot(2,3,5),plt.imshow(img4)

#Saving image
cv2.imwrite('emoboss3.jpg',img3)
```

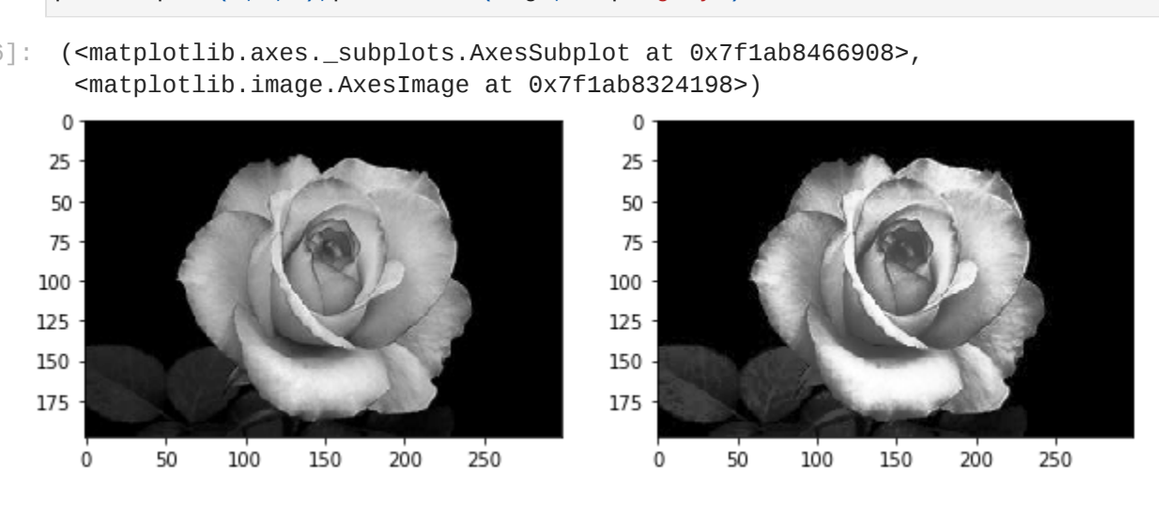
Out[15]: `True`



Contrast

In [16]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams

img = cv2.imread('img1.jpeg',0)
img1 = cv2.equalizeHist(img)

plt.subplot(1,2,1),plt.imshow(img,cmap='gray')
plt.subplot(1,2,2),plt.imshow(img1,cmap='gray')
```

Out[16]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab8466908>,
 <matplotlib.image.AxesImage at 0x7f1ab8324198>)
```

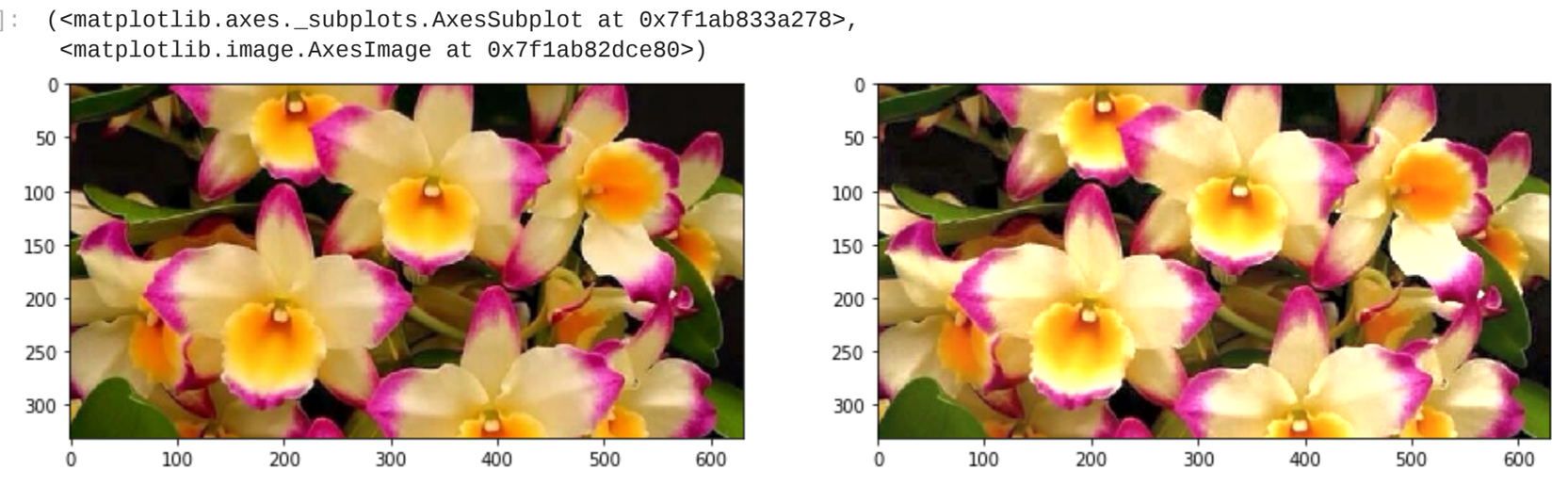


Contras Effect On Color Image

    * YUV Format is used to saperate intensity from color part im an image.

In [17]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams
img=cv2.cvtColor(cv2.imread("img.jpg"),cv2.COLOR_BGR2RGB)
img_YUV=cv2.cvtColor(img,cv2.COLOR_BGR2YUV)
img_YUV[:,:,0]=cv2.equalizeHist(img_YUV[:,:,0])
img1=cv2.cvtColor(img_YUV,cv2.COLOR_YUV2BGR)

rcParams['figure.figsize']=15,15
plt.subplot(1,2,1),plt.imshow(img)
plt.subplot(1,2,2),plt.imshow(img1)
```

Out[17]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab833a278>,
 <matplotlib.image.AxesImage at 0x7f1ab82dce80>)
```

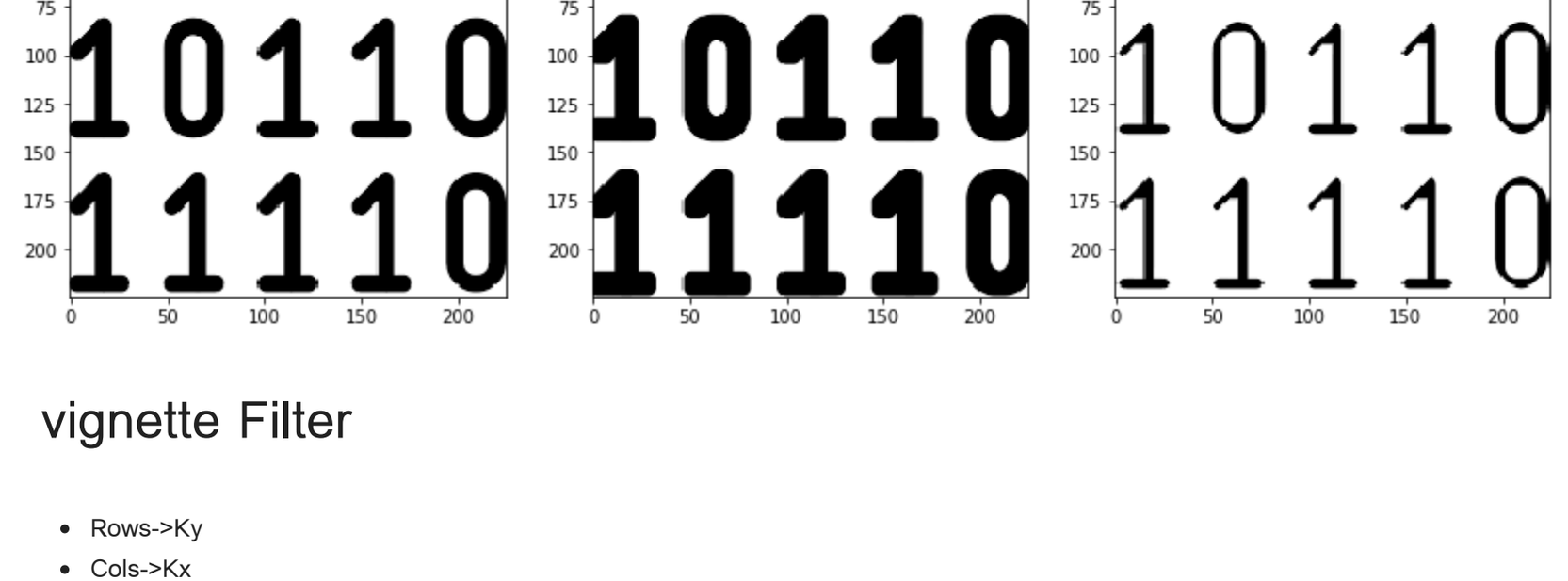


Morphological Image Processing Operations

- Erosion and dilation are morphological image processing operations. Morphological image processing basically deals with modifying geometric structures in the image.
- These operations are primarily defined for binary images, but we can also use them on grayscale images.
- Erosion basically strips out the outermost layer of pixels in a structure, where as dilation adds an extra layer of pixels on a structure.

In [18]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams

img = cv2.imread('binaryimg.png')
#img = cv2.imread('sample.jpg',0)

k1 = np.ones((5,5),np.uint8)

img1 = cv2.erode(img,k1,iterations=1)
img2 = cv2.dilate(img,k1,iterations=1)

rcParams['figure.figsize']=15,15
plt.subplot(1,3,1),plt.imshow(img),plt.title("Original")
plt.subplot(1,3,2),plt.imshow(img1),plt.title("Erosion")
plt.subplot(1,3,3),plt.imshow(img2),plt.title("Dialation")
```

Out[18]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab8260b70>,
 <matplotlib.image.AxesImage at 0x7f1ab80cdfd0>,
 Text(0.5, 1.0, 'Dialation'))
```



vignette Filter

- Rows->Ky
- Cols->Kx
- Gosian

In [19]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img=cv2.cvtColor(cv2.imread("img.jpg"),cv2.COLOR_BGR2RGB)
rows,cols=img.shape[:2]
kx=cv2.getGaussianKernel(cols,200)
ky=cv2.getGaussianKernel(rows,200)
k=ky*kx.T
mask=255*k/np.linalg.norm(k)
img1=np.copy(img)
for i in range(3):
    img1[:,:,i]=img1[:,:,i]*mask
plt.subplot(1,2,1),plt.imshow(img)
plt.subplot(1,2,2),plt.imshow(img1)
```

Out[19]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab805c2b0>,
 <matplotlib.image.AxesImage at 0x7f1ab84b5898>)
```