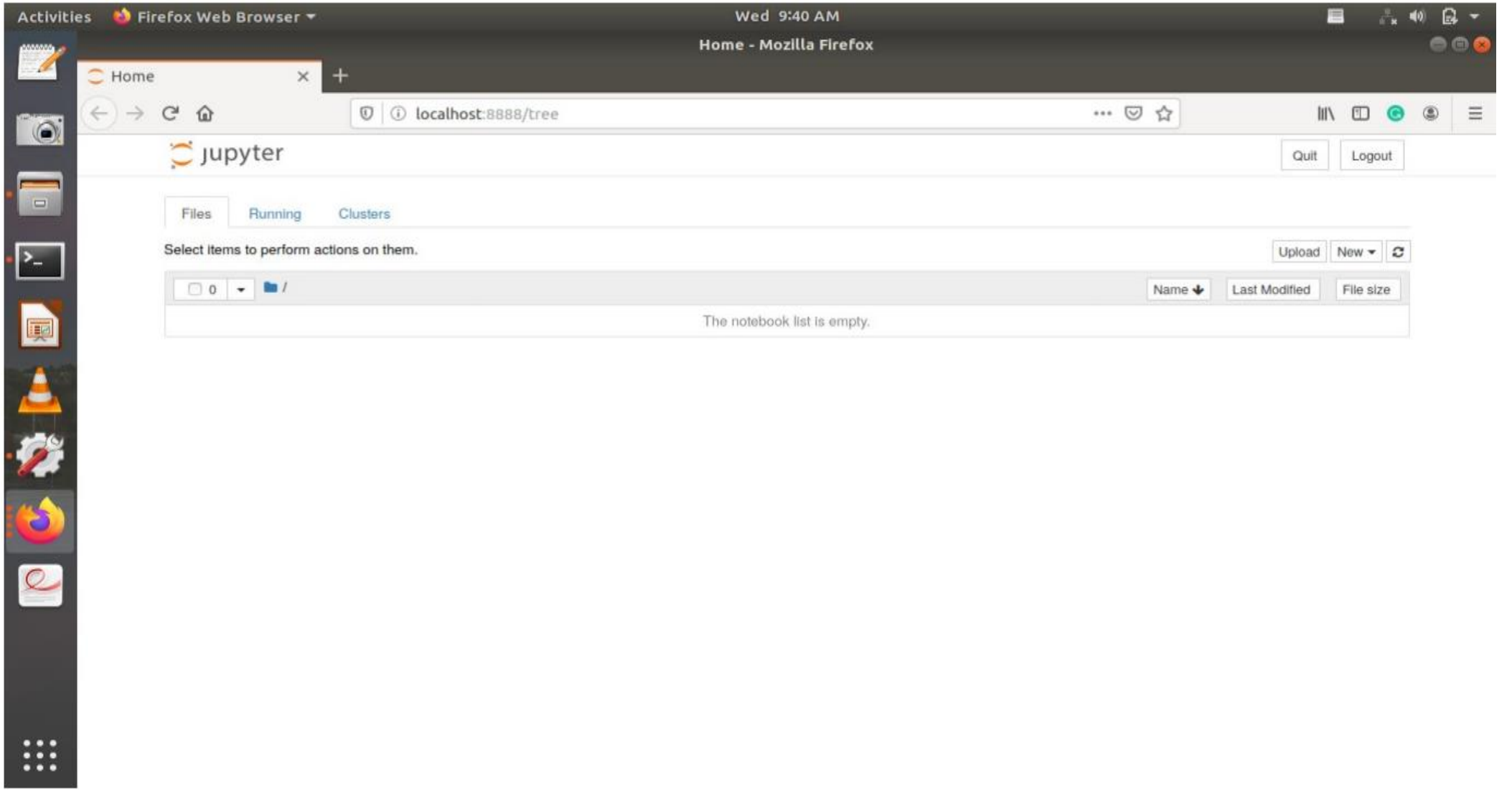
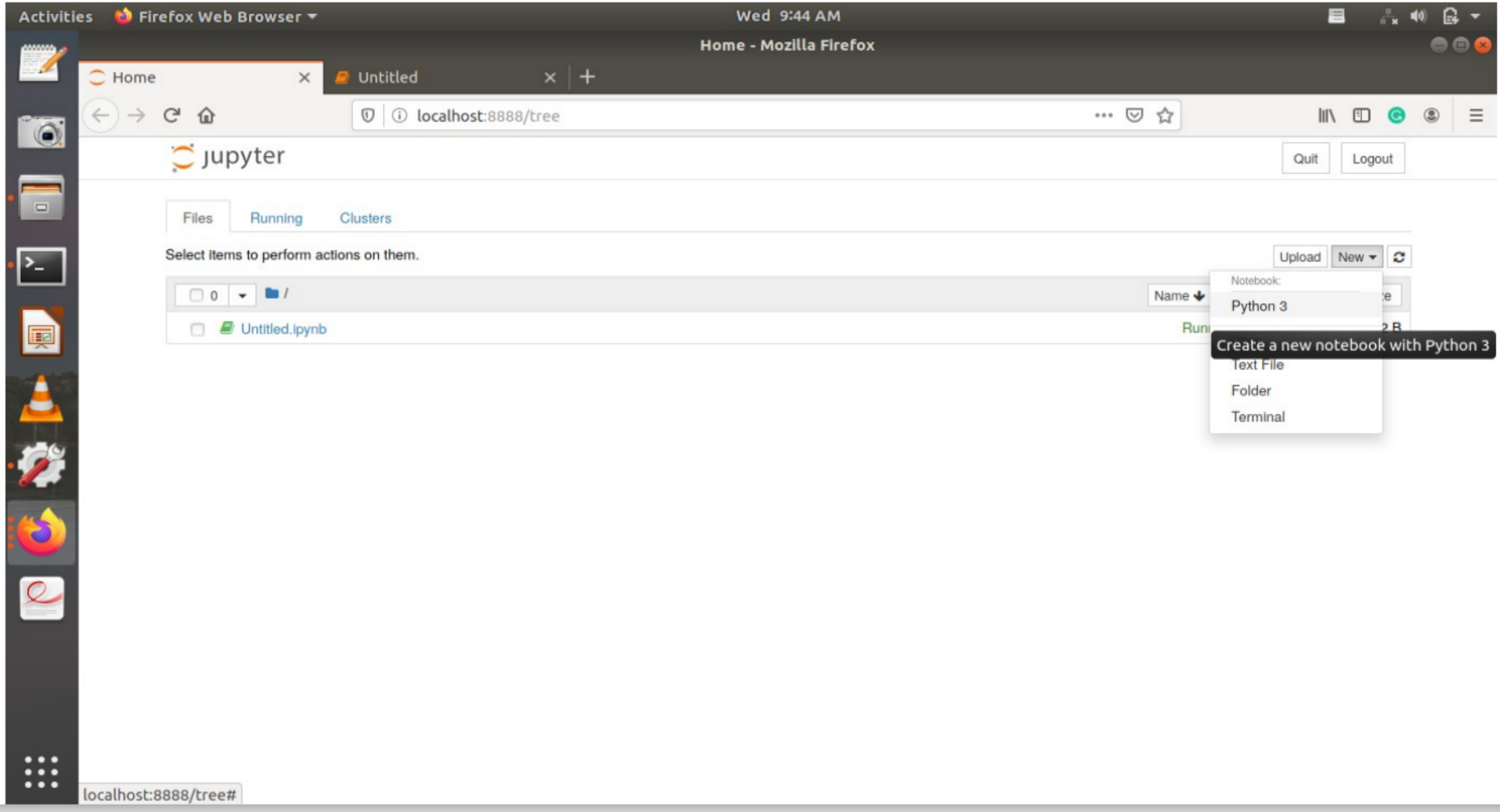
Open CV(Open Source Computer Vision)
for
Digital Image Processing

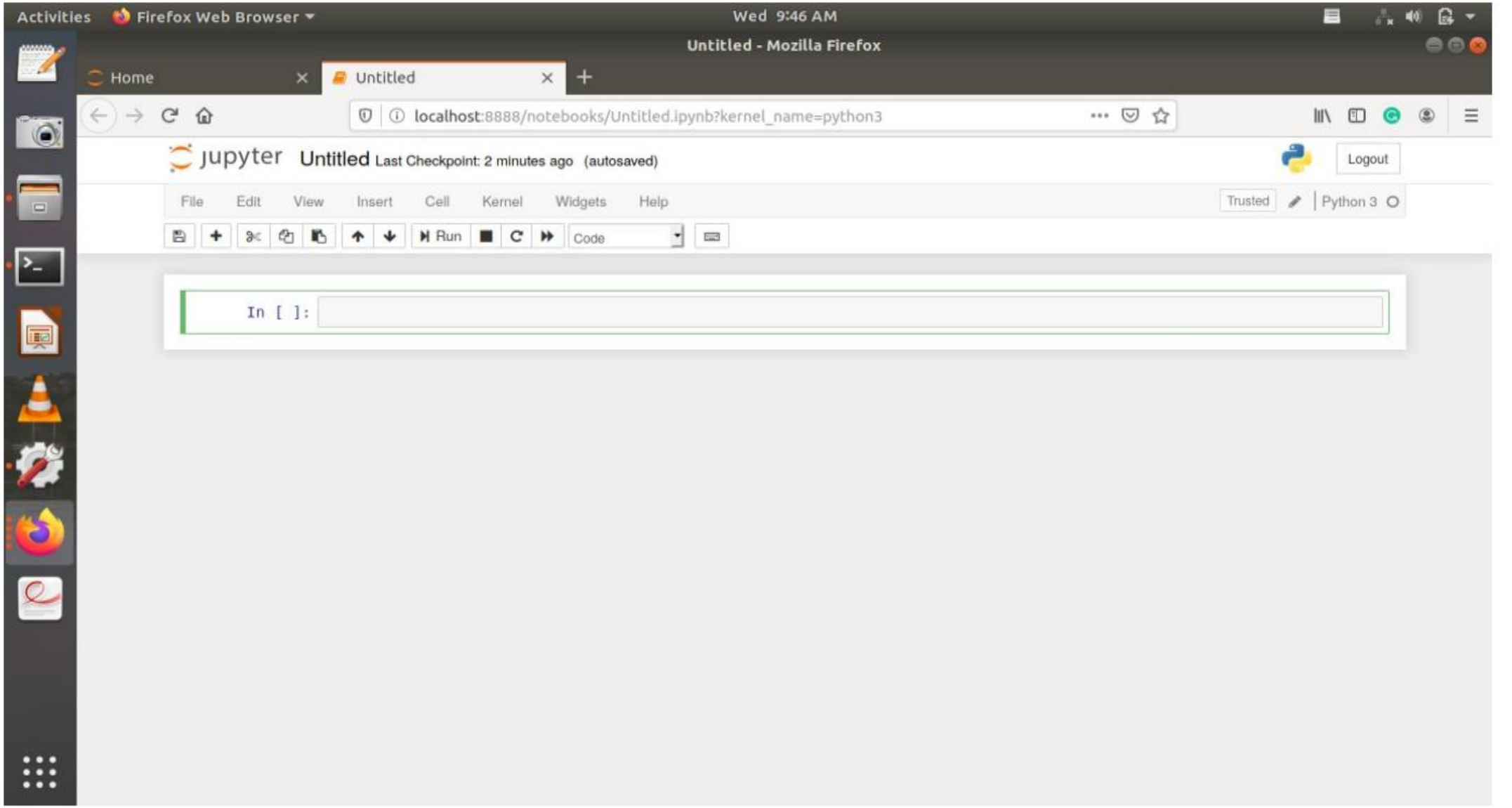
Jupyter


Activities
Firefox Web Browser
Wed 9:40 AM
Home - Mozilla Firefox
Home
localhost:8888/tree
jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/
Name
Last Modified
File size
The notebook list is empty.

Creating a python file



Jupyter Notebook



Installing OpenCV

- Install package python-opencv :

  pip install opencv-python

- Test:

  ```
  import cv2
  print(cv2.__version__)
  ```

Reading and Displaying an image

Method 1

```
import cv2

img = cv2.imread(“street.jpg")

 cv2.imshow('Original Image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

localhost:8888/notebooks/Desktop/DIP_2023/OpenCV_for_DIP.ipynb

jupyter OpenCV_for_DIP Last Checkpoint: 10 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [1]:
```
import cv2
print(cv2.__version__)
```

```
4.5.5
```

Reading and Displaying an image

In [*]:
```
# Reading Method 1
import cv2
#To read image from disk,  cv2.imread function is used
img = cv2.imread("street.jpg")
# Creating GUI window to display an image on screen
cv2.imshow("image", img)
#To hold the window on screen, use cv2.waitKey method
cv2.waitKey(0)
#For removing/deleting created GUI window from screen and memory
cv2.destroyAllWindows()
```

Reading Method 2

```
path = "C:/Users/sukesh babu/Desktop/DIP_2023/street.jpg"
img = cv2.imread(path)
cv2.imshow("image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [*]:

```
# Reading Method 2
path = "C:/Users/sukesh babu/Desktop/DIP_2023/street.jpg"
img = cv2.imread(path)
cv2.imshow("image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Displaying an image: Method II

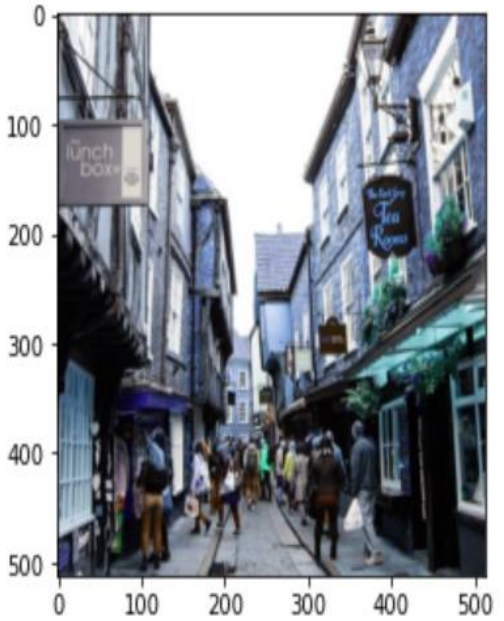
```
import numpy as np

import matplotlib.pyplot as plt

img1=cv2.imread("street.jpg")

plt.imshow(img1)
```

picture in BGR format.

```
plt.show()
```

In [2]:

```
# Displaying an image: Method II
import numpy as np
import matplotlib.pyplot as plt
#Matplotlib library uses RGB color format to read a colored image
img1=cv2.imread("street.jpg")
#Displaying image using plt.imshow() method
plt.imshow(img1)
#Image read using cv2 and displaying using matplotlib will display picture in BGR format.
plt.show()
```



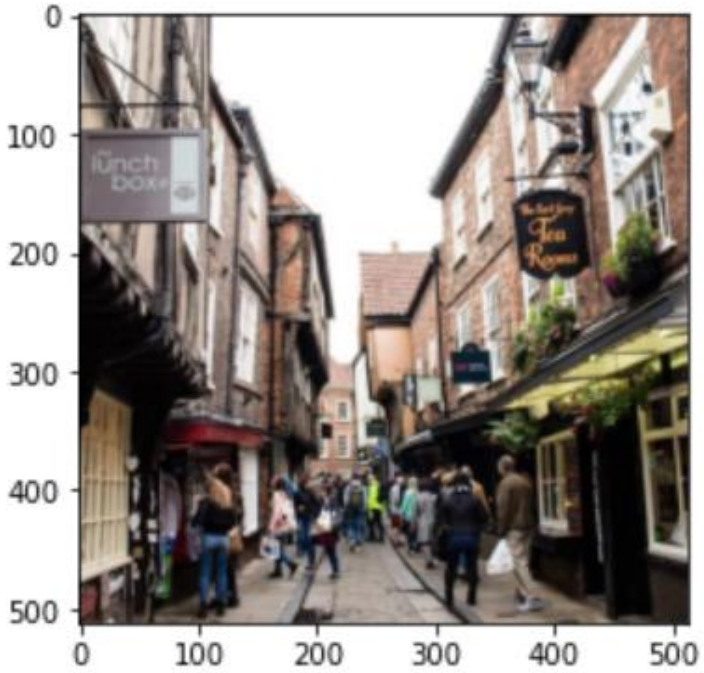
Converting BGR color to RGB color format

```
RGB_img = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)

#Displaying image using plt.imshow() method

plt.imshow(RGB_img)
```

In [3]:

```
#Image read using cv2 and displaying using matplotlib will display picture in BGR format.
# Converting BGR color to RGB color format
RGB_img = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
#Displaying image using plt.imshow() method
plt.imshow(RGB_img)
```

Out[3]: <matplotlib.image.AxesImage at 0x16ab92b13d0>



Size and shape of an image

```
print(img.size) #Total no of pixels

print(img.shape) # Width, height and number of channels

print(img[0,0]) #RGB values at pixel (0,0)

print(img) #RGB values of complete pixel
```

Size and shape of an image

In [70]:

```
print(img.size)#Total no of pixels
print(img.shape)# Width, height and number of channels
print(img[0,0])#RGB values at pixel (0,0)
print(img)#RGB values of complete pixel
```

```
786432
(512, 512, 3)
[46 53 68]
[[[ 46  53  68]
  [ 69  76  91]
  [126 132 145]
  ...
  [112 126 148]
  [ 83  97 119]
  [ 98 115 136]]

 [[ 60  67  82]
  [ 67  74  89]
  [118 124 137]
  ...
  [ 90 104 126]
  [ 84 100 123]
  [ 77  94 115]]

 [[ 61  66  81]
```

RGB to Gray Color Conversion - Method 1

```
import cv2

# Reading color image as grayscale

gray = cv2.imread("street.jpg",0)

cv2.imshow("Grayscale Image", gray)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

jupyter OpenCV_for_DIP Last Checkpoint: 10 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

```
 [ 31  37  44]
 [ 31  37  44]
 [ 30  36  43]]

[[ 11  15  44]
 [ 27  31  59]
 [ 51  56  77]
 ...
 [ 28  34  41]
 [ 30  36  43]
 [ 31  37  44]]]
```

RGB to Gray Color Conversion - Method 1

In [*]:

```
1 import cv2
2 # Reading color image as grayscale
3 gray = cv2.imread("street.jpg",0)
4 # Showing grayscale image
5 cv2.imshow("Grayscale Image", gray)
6 cv2.waitKey(0)
7 cv2.destroyAllWindows()
8
```

RGB to Gray Color Conversion - Method 2

```
import cv2
img = cv2.imread("street.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("Original Image",img)
cv2.imshow("Converted Image",gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

jupyter OpenCV_for_DIP Last Checkpoint: 10 hours ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

RGB to Gray Color Conversion - Method 1

In [4]:

```
import cv2
# Reading color image as grayscale
gray = cv2.imread("street.jpg",0)
# Showing grayscale image
cv2.imshow("Grayscale Image", gray)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

RGB to Gray Color Conversion - Method 2

In [*]:

```
import cv2
img = cv2.imread("street.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("Original Image",img)
cv2.imshow("Converted Image",gray)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

Colored image to binary image: Thresholding

```
img = cv2.imread("street.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
ret, thresh1 = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(gray, 120, 255,
cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(gray, 120, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(gray, 120, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(gray, 120, 255,
cv2.THRESH_TOZERO_INV)
```

To display the images

```
titles = ['Original Image','Gray Scale','BINARY’,
'BINARY_INV’, 'TRUNC’, 'TOZERO’, 'TOZERO_INV']
images = [imgRGB,gray,thresh1, thresh2, thresh3, thresh4,
thresh5]

for i in range(7):
    plt.subplot(2,4,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

- **THRESH_BINARY**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_BINARY_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$$

- **THRESH_TRUNC**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

Convert a colored image to a binary image: Thresholding

In [6]:

```
# Convert a color image to a binary image and  display the original, grayscale and binary images by applying
# load the input image
img = cv2.imread("street.jpg")

# convert the input image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# convert BGR to RGB to display using matplotlib
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# applying different thresholding techniques on the input image to convert grayscale to binary image
# all pixels value above 120 will  be set to 255 and below 120 will be set to 0
ret, thresh1 = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(gray, 120, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(gray, 120, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(gray, 120, 255, cv2.THRESH_TOZERO_INV)
#To display the images
titles = ['Original Image','Gray Scale','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [imgRGB,gray,thresh1, thresh2, thresh3, thresh4, thresh5]

for i in range(7):
    plt.subplot(2,4,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Writing an image

```
img = cv2.imread("street.jpg")

# Converting color image to grayscale image

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imwrite('gray_street.jpg',gray)
```

Geometric Transformation

Resize (Scaling) Method 1:

```
import cv2
img = cv2.imread('street.jpg')
height, width = img.shape[:2]
res = cv2.resize(img,(2*width, 2*height), interpolation =
cv2.INTER_CUBIC)
cv2.imshow("Origimal Image",img)
cv2.imshow("Resized Image",res)
cv2.waitKey(0); cv2.destroyAllWindows()
```

Low level image processing

Image sharpening:

```
import cv2
import numpy as np
img = cv2.imread("street.jpg")
kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])
image_sharp = cv2.filter2D(src=img, ddepth=-1, kernel=kernel)
cv2.imshow('Original Image', img)
cv2.imshow('Sharpened Image', image_sharp)
cv2.waitKey()
cv2.destroyAllWindows()
```

Image Sharpening

```
import cv2
import numpy as np
img = cv2.imread("street.jpg")
kernel = np.array([[0, -1, 0],
                   [-1, 5,-1],
                   [0, -1, 0]])
image_sharp = cv2.filter2D(src=img, ddepth=-1, kernel=kernel)
cv2.imshow('Original Image', img)

cv2.imshow('Sharpened Image', image_sharp)
cv2.waitKey()
cv2.destroyAllWindows()
```


the lunch box
The Earl Grey
Tea
Rooms


the lunch box
The Earl Grey
Tea
Rooms

Geometric Transformation

Resize : Method 1:

In [*]:

```
import cv2
img = cv2.imread('street.jpg')
height, width = img.shape[:2]
res = cv2.resize(img,(2*width, 2*height), interpolation = cv2.INTER_CUBIC)
cv2.imshow("Origimal Image",img)
cv2.imshow("Resized Image",res)
cv2.waitKey(0); cv2.destroyAllWindows()

```

Resize : Method 2:

```
import cv2
img = cv2.imread('street.jpg')
res = cv2.resize(img,None,fx=0.5, fy=0.5, interpolation =
cv2.INTER_CUBIC)
cv2.imshow("Original Image",img)
cv2.imshow("Resized Image",res)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resize : Method 2:

In [77]:

```
1  import cv2
2  img = cv2.imread('street.jpg')
3  res = cv2.resize(img,None,fx=0.5, fy=0.5, interpolation = cv2.INTER_CUBIC)
4  cv2.imshow("Original Image",img)
5  cv2.imshow("Resized Image",res)
6  cv2.waitKey(0)
7  cv2.destroyAllWindows()
8
```

Geometric Transformation

Rotation : Method 1:

```
import cv2
img = cv2.imread('street.jpg',0)
rows,cols = img.shape
M = cv2.getRotationMatrix2D((cols/2,rows/2),60,1)
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow("Original Image",img)
cv2.imshow("Rotated Image",dst); cv2.waitKey(0);
cv2.destroyAllWindows()
```

Rotation : Method 1:

In [78]:

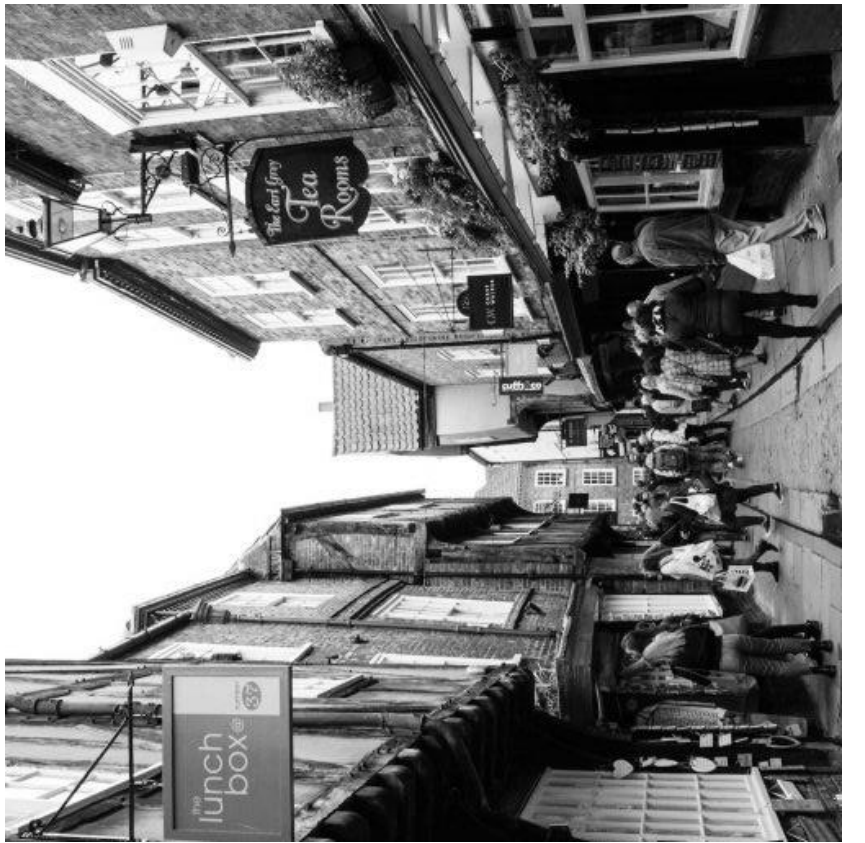
```
1  import cv2
2  img = cv2.imread('street.jpg',0)
3  rows,cols = img.shape
4  M = cv2.getRotationMatrix2D((cols/2,rows/2),60,1)
5  dst = cv2.warpAffine(img,M,(cols,rows))
6  cv2.imshow("Original Image",img)
7  cv2.imshow("Rotated Image",dst); cv2.waitKey(0); cv2.destroyAllWindows()
8
```

Rotation : Method 2:

```
import cv2

img = cv2.imread('street.jpg',0)

R = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)

cv2.imshow("Original Image",img)

cv2.imshow("Rotated Image",R)

cv2.waitKey(0); cv2.destroyAllWindows()
```

Rotation : Method 2: ¶

In [79]:
```
1 import cv2
2 img = cv2.imread('street.jpg',0)
3 R = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
4 cv2.imshow("Original Image",img)
5 cv2.imshow("Rotated Image",R)
6 cv2.waitKey(0); cv2.destroyAllWindows()
7
```



Adding Gaussian Noise

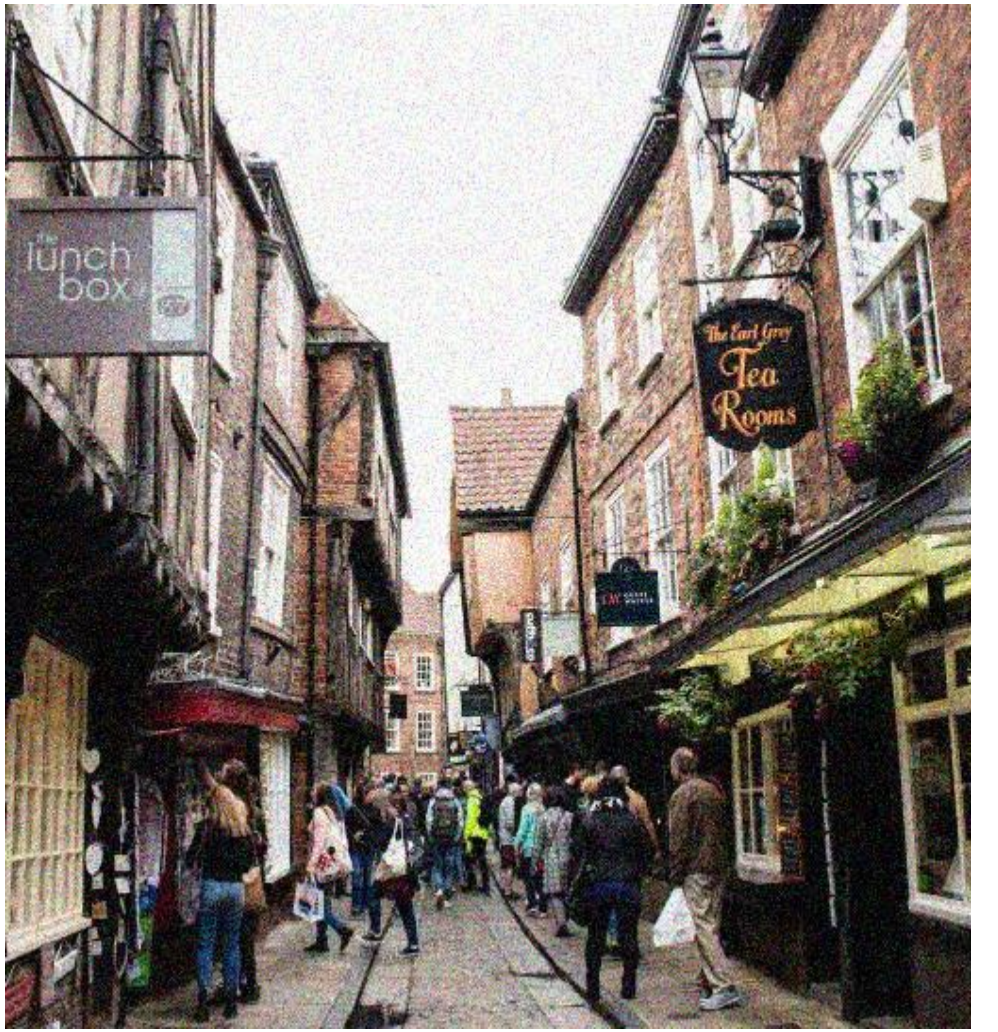
```
import cv2
import numpy as np
from skimage.util import random_noise
img = cv2.imread("street.jpg")
noise_img = random_noise(img,mode='gaussian',mean=0,var=0.01)
noise_img = np.array(255*noise_img, dtype = 'uint8')
cv2.imshow('Original Image', img)
cv2.imshow('Noisy Image',noise_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Adding noise to image ¶

*]:

```
1  import cv2
2  import numpy as np
3  from skimage.util import random_noise
4  img = cv2.imread("street.jpg")
5  noise_img = random_noise(img,mode='gaussian',mean=0,var=0.01)
6  noise_img = np.array(255*noise_img, dtype = 'uint8')
7  cv2.imshow('Original Image', img)
8  cv2.imshow('Noisy Image',noise_img)
9  cv2.waitKey(0)
10 cv2.destroyAllWindows()
11
```


the lunch box
The Earl Grey
Tea
Rooms


the lunch box
The Earl Grey
Tea
Rooms

Adding Salt and Pepper Noise

```
import cv2
import numpy as np
from skimage.util import random_noise
img = cv2.imread("street.jpg")
noise_img = random_noise(img, mode='s&p',amount=0.1)
noise_img = np.array(255*noise_img, dtype = 'uint8')
cv2.imshow('Original Image', img)
cv2.imshow('Noisy Image',noise_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


the lunch box
The Earl Grey
Tea
Rooms


the lunch box
The Earl Grey
Tea
Rooms

Smoothening: Median filter

```
import cv2
from matplotlib import pyplot as plt
img = cv2.imread("street.jpg")
median = cv2.medianBlur(img,5)
cv2.imshow('Original Image', img)
cv2.imshow('Smoothend Image',median)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Median Filter

```
]:
```

```python
import cv2
from matplotlib import pyplot as plt
img = cv2.imread("street.jpg")
median = cv2.medianBlur(img,5)
cv2.imshow('Original Image', img)
cv2.imshow('Smoothend Image',median)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


the lunch box
The Earl Grey
Tea
Rooms

Gaussian Filter

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('street.jpg')
blur = cv2.GaussianBlur(img,(11,11),0)
cv2.imshow('Original Image', img)
cv2.imshow('Smoothend Image',blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Gaussian Filter

*]:

```
1  import cv2
2  import numpy as np
3  from matplotlib import pyplot as plt
4  img = cv2.imread('street.jpg')
5  blur = cv2.GaussianBlur(img,(11,11),0)
6  cv2.imshow('Original Image', img)
7  cv2.imshow('Smoothend Image',blur)
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
```


the lunch box
The Earl Grey
Tea
Rooms

Geometric Transformation
Translation:

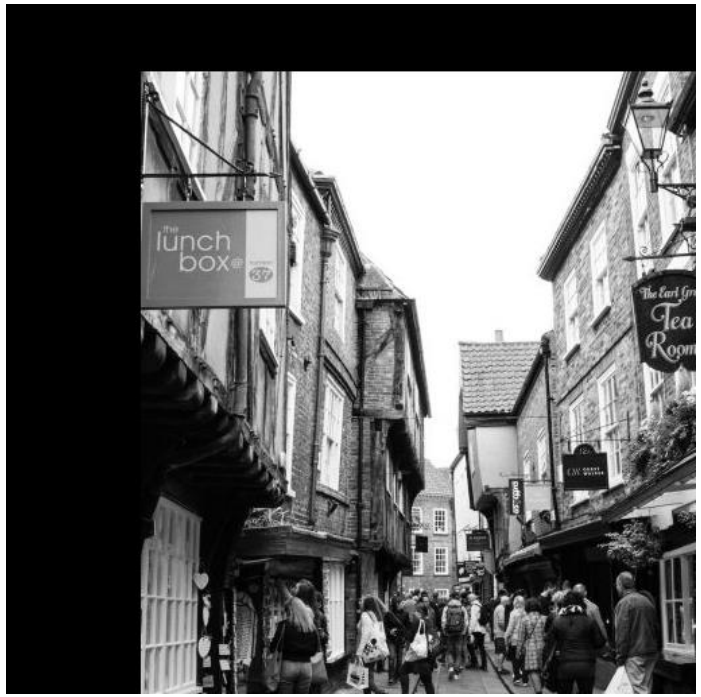
```
import cv2
import numpy as np
img = cv2.imread('street.jpg',0)
rows,cols = img.shape
M = np.float32([[1,0,100],[0,1,50]])
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow('Original image',img)
cv2.imshow('Translated image',dst); cv2.waitKey(0);
cv2.destroyAllWindows()
print(img.size)
print(img.shape)
print(img)
```

Translation:

In [80]:

```
import cv2
import numpy as np
img = cv2.imread('street.jpg',0)
rows,cols = img.shape
M = np.float32([[1,0,100],[0,1,50]])
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow('Original image',img)
cv2.imshow('Translated image',dst); cv2.waitKey(0); cv2.destroyAllWindows()
print(img.size)
print(img.shape)
print(img)

```

```
262144
(512, 512)
[[ 57  80 135 ... 131 102 119]
 [ 71  78 127 ... 109 105  98]
 [ 70  73 111 ... 103 106 107]
 ...
 [ 52  36  21 ...  41  39  36]
 [ 20  23  35 ...  38  38  37]
 [ 23  39  62 ...  35  37  38]]
```



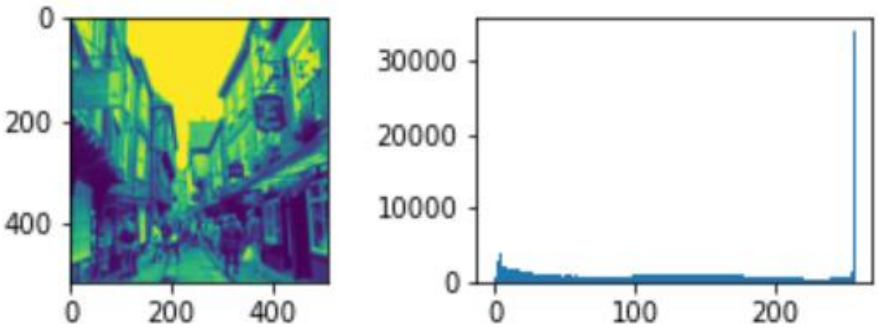
Histogram

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('street.jpg',0)

plt.subplot(221),plt.imshow(img)

plt.subplot(222),plt.hist(img.ravel(),256,[0,256]); plt.show()
```

Histogram

In [81]:

```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4 img = cv2.imread('street.jpg',0)
5 plt.subplot(221),plt.imshow(img)
6 plt.subplot(222),plt.hist(img.ravel(),256,[0,256]); plt.show()
7
```



Basic operations on image

Image complementation or Negation

```
import cv2
im = cv2.imread('Lena.png',0)
img = 255-im
cv2.imshow('Original', im)
cv2.imshow('Complement', img)
cv2.waitKey(0); cv2.destroyAllWindows()
```

Complement of images:

$$g(x, y) = (L - 1) - f(x, y)$$

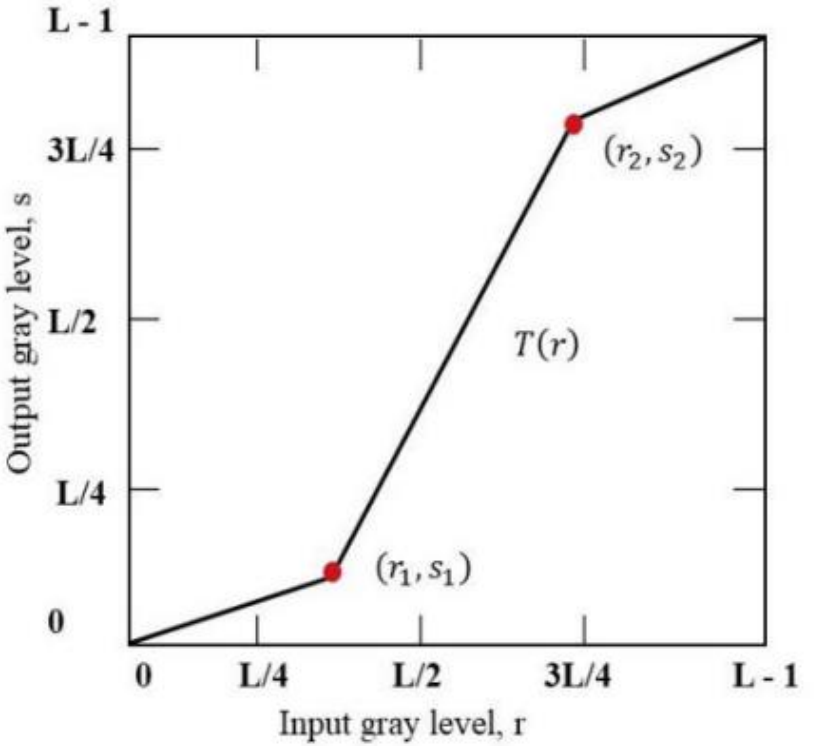
where L-1 is the maximum gray value of f.

Image complementation or Negation

[*]:

```
1  import cv2
2  # Read Image1
3  im = cv2.imread('Lena.png',0)
4  # Find complement
5  img = 255-im
6
7  # Show the image
8  cv2.imshow('Original', im)
9  cv2.imshow('Complement', img)
10
11 cv2.waitKey(0); cv2.destroyAllWindows()
```

Contrast Stretching (Normalization)



(a) Transformation function

```
import cv2
import numpy as np
 # Function to map each intensity level to output intensity level.
def pixelVal(pix, r1, s1, r2, s2):
    if (0 <= pix and pix <= r1):
        return (s1 / r1)*pix
    elif (r1 < pix and pix <= r2):
        return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1
    else:
        return ((255 - s2)/(255 - r2)) * (pix - r2) + s2

# Open the image.
img = cv2.imread('Lena.png',0)
```

```
r1 = 70
s1 = 0
r2 = 140
s2 = 255
 # Vectorize the function to apply it to each value in the Numpy array.
pixelVal_vec = np.vectorize(pixelVal)
contrast_stretched = pixelVal_vec(img, r1, s1, r2, s2)
cv2.imshow('Original', img)
cv2.imshow('Contrast streched', contrast_stretched)
  cv2.waitKey(0); cv2.destroyAllWindows()
```

Contrast Stretching (Normalization):

33]:

```
import cv2
import numpy as np

# Function to map each intensity level to output intensity level.
def pixelVal(pix, r1, s1, r2, s2):
    if (0 <= pix and pix <= r1):
        return (s1 / r1)*pix
    elif (r1 < pix and pix <= r2):
        return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1
    else:
        return ((255 - s2)/(255 - r2)) * (pix - r2) + s2

# Open the image.
img = cv2.imread('Lena.png',0)

# Define parameters.
r1 = 70
s1 = 0
r2 = 140
s2 = 255

# Vectorize the function to apply it to each value in the Numpy array.
pixelVal_vec = np.vectorize(pixelVal)

```

Logarithmic Transformation (LOG):

$S = C \log(1 + r)$where C is any constant and r, s is input and output pixel values.

```
import cv2
import numpy as np
img = cv2.imread('Lena.png',0)
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)
log_transformed = np.array(log_transformed, dtype =
np.uint8)
cv2.imshow('Original', img)
cv2.imshow('log_transformed', log_transformed)
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

Logarithmic Transformation (LOG):

In [84]:

```
import cv2
import numpy as np

# Open the image.
img = cv2.imread('Lena.png',0)

# Apply log transform.
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)

# Specify the data type.
log_transformed = np.array(log_transformed, dtype = np.uint8)
cv2.imshow('Original', img)
cv2.imshow('log_transformed', log_transformed)
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

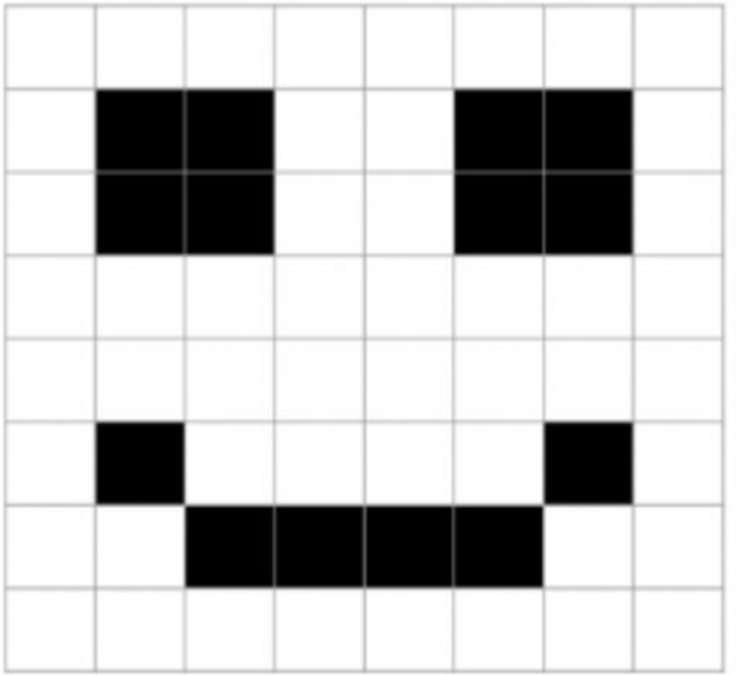
Image addition

```
import cv2
im1= cv2.imread('Lena.png', 1)
im2 = cv2.imread('b.png', 1)
img = cv2.add(im1,im2)
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of addition', img)
 cv2.waitKey(0); cv2.destroyAllWindows()
```

Image addition

In [85]:

```
import cv2
# Read Image1
im1= cv2.imread('Lena.png', 1)

# Read image2
im2 = cv2.imread('b.png', 1)

# Add the images
img = cv2.add(im1,im2)
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of addition', img)


cv2.waitKey(0); cv2.destroyAllWindows()

```



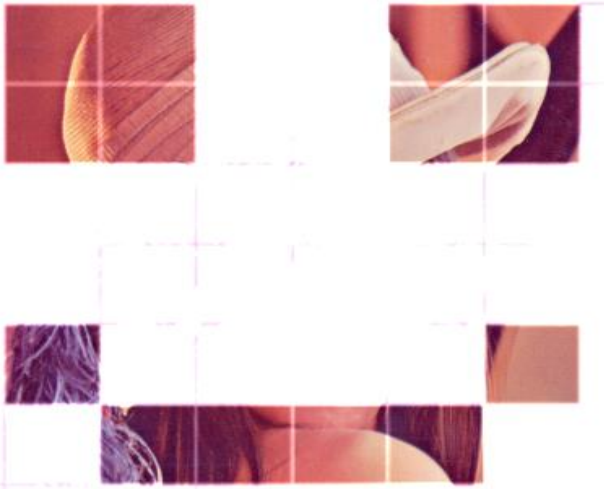


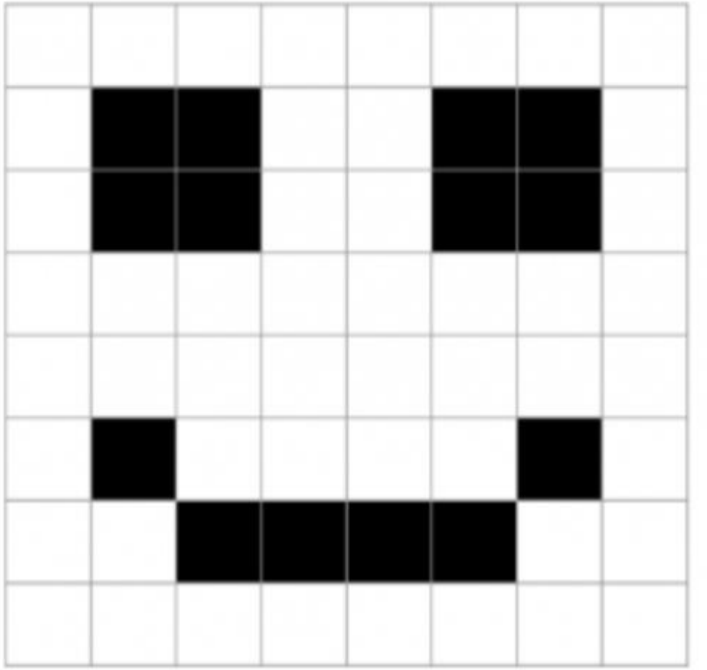
Image subtraction

```
import cv2
im1= cv2.imread('Lena.png', 1)
im2 = cv2.imread('b.png', 1)
img = cv2.subtract(im1,im2)
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of subtraction', img)
cv2.waitKey(0); cv2.destroyAllWindows()
```

Image subtraction

In [86]:

```
import cv2
# Read Image1
im1= cv2.imread('Lena.png', 1)

# Read image2
im2 = cv2.imread('b.png', 1)

# Subtract the images
img = cv2.subtract(im1,im2)
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of subtraction', img)


cv2.waitKey(0); cv2.destroyAllWindows()

```



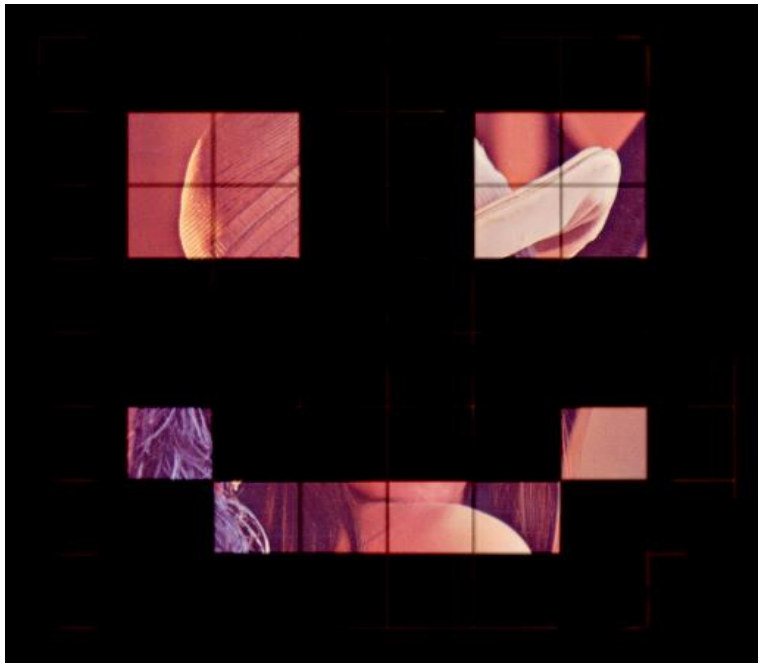


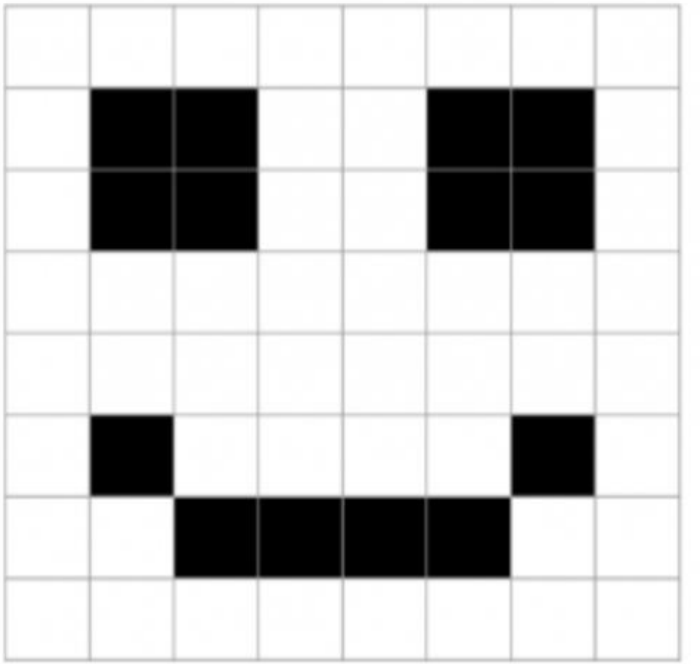
Image multiplication

```
import cv2
im1= cv2.imread('Lena.png', 1)
im2 = cv2.imread('b.png', 1)
img = cv2.multiply(im1,im2)
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of multiplication', img)
cv2.waitKey(0); cv2.destroyAllWindows()
```

Image multiplication

In [87]:

```
import cv2
# Read Image1
im1= cv2.imread('Lena.png', 1)

# Read image2
im2 = cv2.imread('b.png', 1)

# Multiply the images the images
img = cv2.multiply(im1,im2)
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of multiplication', img)


cv2.waitKey(0); cv2.destroyAllWindows()
```



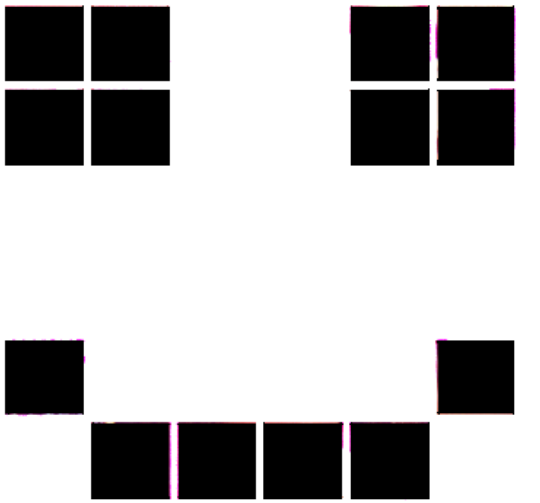


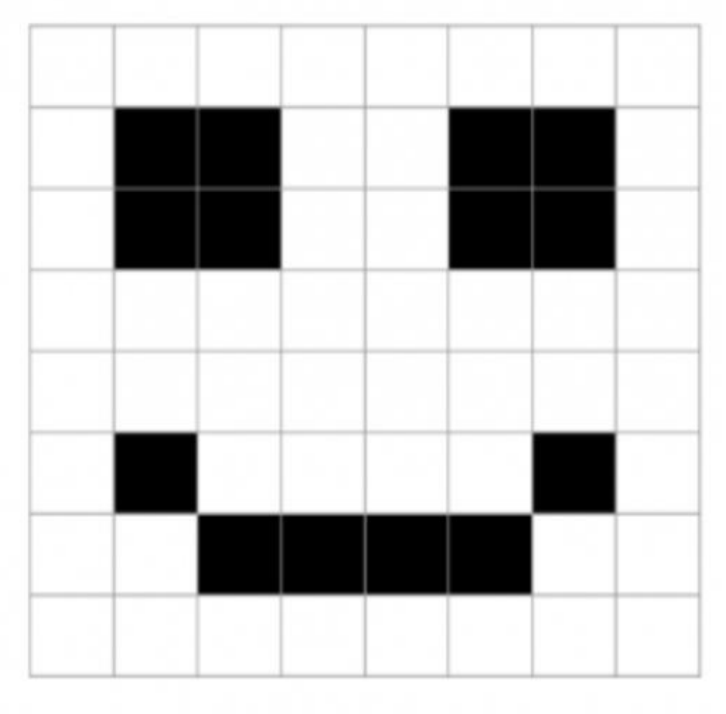
Image division

```
import cv2
im1= cv2.imread('Lena.png', 1)
im2 = cv2.imread('b.png', 1)
img = im1/im2
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of division', img)
cv2.waitKey(0); cv2.destroyAllWindows()
```

Image division

88]:

```
import cv2
# Read Image1
im1= cv2.imread('Lena.png', 1)

# Read image2
im2 = cv2.imread('b.png', 1)

# Divide the images the images
img = im1/im2
cv2.imshow('image one', im1)
cv2.imshow('image two', im2)
cv2.imshow('Result of division', img)


cv2.waitKey(0); cv2.destroyAllWindows()
```



Thank You……..