Section 1: Getting familiar with basic OpenCV functions

In [33]:
```
#CO543 LAB 01
#E20100
#importing necessary libraries
import cv2
import  numpy as np
from google.colab.patches import cv2_imshow
```

save image in working directory and load the image

In [34]:
```
img = cv2.imread('/content/sample_data/eb9931c3f39215a193826992d013a17a.jpg')
cv2.imwrite('image.jpg',img)
```

Out[34]:
```
True
```

read and visualize the image

In [ ]:
```
img = cv2.imread('image.jpg')
cv2_imshow(img)
```

view the image in greyscale

In [ ]:
```
# 0 is the flag that use to turn the image to grayscale
img1 = cv2.imread('image.jpg',0)
cv2_imshow(img1)
```

view image details

In [ ]:
```
# number of rows columns and channels(if image is color)
print(img.shape)
# number of pixels
print(img.size)
```

```
(700, 700, 3)
1470000
```

view image data type

In [ ]:
```
print(img.dtype)
```

```
uint8
```

convert the image from RGB to Grayscale

In [ ]:
```
img2 = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
cv2_imshow(img2)
```

image resizing

In [ ]:
```
# reduce image to 100cols and 50 rows
resized_image1 = cv2.resize(img,(100,50))
# reduce both axes by half
resized_image2 = cv2.resize(img,(0,0),fx=0.5,fy=0.5)
# specify interpolation method
resized_image3 = cv2.resize(img,(0,0),fx=0.5,fy=0.5, interpolation=cv2.INTER_NEARES
cv2_imshow(resized_image1)
cv2_imshow(resized_image2)
cv2_imshow(resized_image3)
```

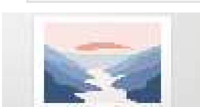


image rotation

In [ ]:
```
# get dimensions of the image and claculate the center of the image
(h,w) = img.shape[:2]
center = (w/2,h/2)
# Computing the matrix (M) that can be used for rotating the image
# center - Point around which, the image is rotated
# 180 - Angle by which image is rotated
# 1.0 - Scaling factor (No scaling in this case)
M = cv2.getRotationMatrix2D(center,180,1.0)
print(M)
# Perform the actual rotation
```

```
rotated = cv2.warpAffine(img,M,(w,h))
cv2_imshow(rotated)
```

```
[[-1.0000000e+00  1.2246468e-16  7.0000000e+02]
 [-1.2246468e-16 -1.0000000e+00  7.0000000e+02]]
```

image cropping

In [ ]:
```
# startY,startX - starting coordinates where cropping should begin.
# endY,endX - ending coordinates
cropped = img[0:200,0:200]
cv2_imshow(cropped)
```

image complementing

In [ ]:
```
inverted = cv2.bitwise_not(img)
cv2_imshow(inverted)
```

image flipping

In [ ]:
```
# flipMode - If 0, flipping around x-axis
# If positive, flipping around y-axis
# If negative, flipping around both axes
flipped_x = cv2.flip(img,0)
flipped_y = cv2.flip(img,5)
cv2_imshow(flipped_x)
cv2_imshow(flipped_y)
```

Section2: CSV to image conversion

In [ ]:
```
# import necessary
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

read the csv file and visualize the the data as a table

In [ ]:
```
img_data = pd.read_csv('Digits_Lab_01.csv')
img_data.head()
```

Out[ ]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774
0	0	0	0	0	0	0	0	0	0	0	...	(
1	0	0	0	0	0	0	0	0	0	0	...	(
2	0	0	0	0	0	0	0	0	0	0	...	(
3	0	0	0	0	0	0	0	0	0	0	...	(
4	0	0	0	0	0	0	0	0	0	0	...	(

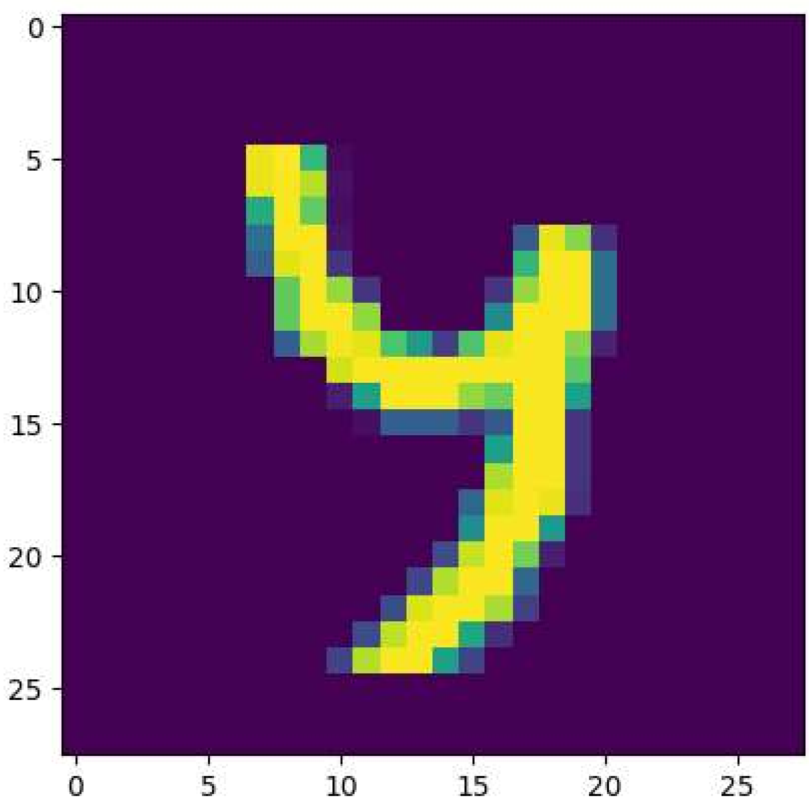
5 rows × 784 columns

reshape the rows of data to images

In [ ]:
```
reshaped_images = img_data.to_numpy().reshape(-1,28,28,1)
```

visualize the iamges

In [ ]:
```
sample_img = np.array(reshaped_images[12]).reshape((28,28))
plt.imshow(sample_img)
plt.show()
```



Section 3 : Geometric Transformations

read the image and check for the shape

In [ ]:
```
img =cv2.imread('image.jpg')
rows,cols,ch = img.shape
```

define the transformation

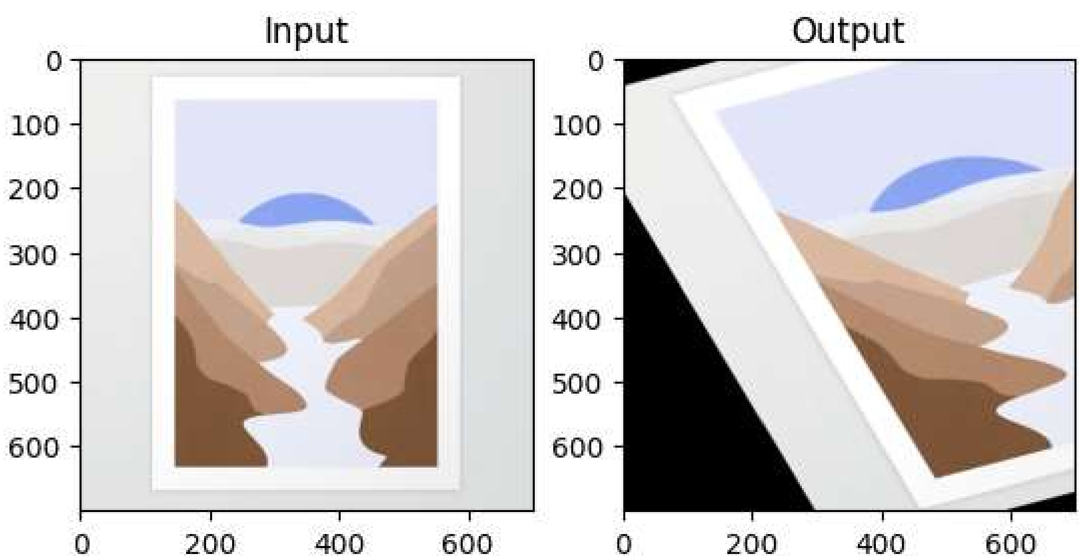
In [ ]:
```
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
M = cv2.getAffineTransform(pts1,pts2)
```

do the transformation

In [ ]:
```
dst = cv2.warpAffine(img,M,(cols,rows))
```

Visualize the original and the transformed images

In [ ]:
```
plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```



Section 4 : Lab tasks

1. Implement the following functions on your own using PythonOpenCV.

In [ ]:
```
# convert image to grayscale
img_grayscale = cv2.imread('image.jpg', 0)
```

In [ ]:
```
# a imcomplement(I) - Inverts I
inverted = cv2.bitwise_not(img_grayscale)
cv2_imshow(inverted)
```

b flipud(I) - Flips image along x-axis

In [ ]:
```
img_x = cv2.flip(img_grayscale, 0)
# display the flip image along x axis
cv2_imshow(img_x)
```

c fliplr(I) - Flips image along y-axis

In [ ]:
```
img_y = cv2.flip(img_grayscale, 1)
# display the flip image along y axis
cv2_imshow(img_y)
```

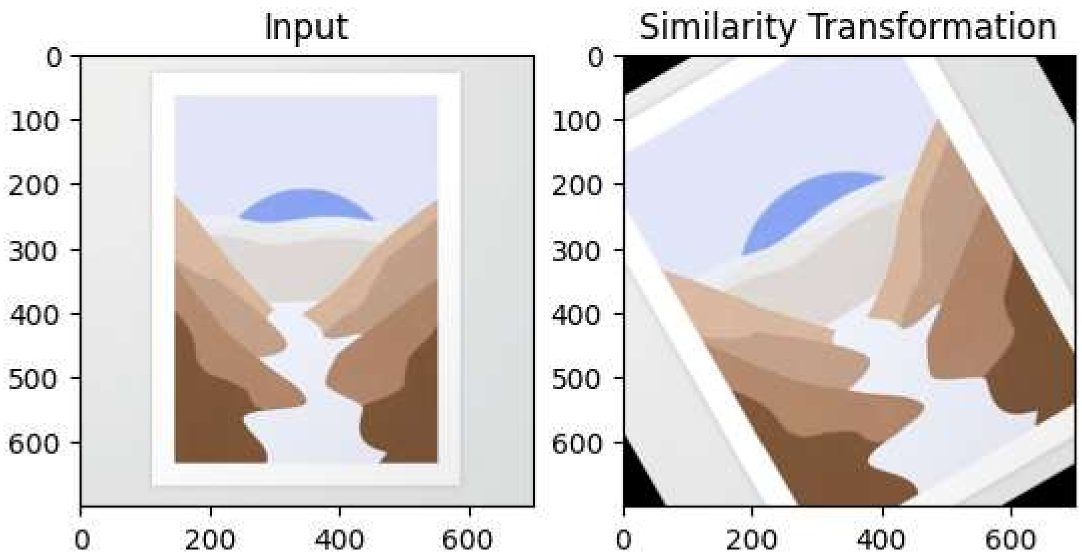
d imresize(I,[x y]) with nearest-neighbour interpolation

In [ ]:
```
resized = cv2.resize(img_grayscale, (0, 0), fx=0.5, fy=0.5, interpolation = cv2.INT
cv2_imshow(resized)
```

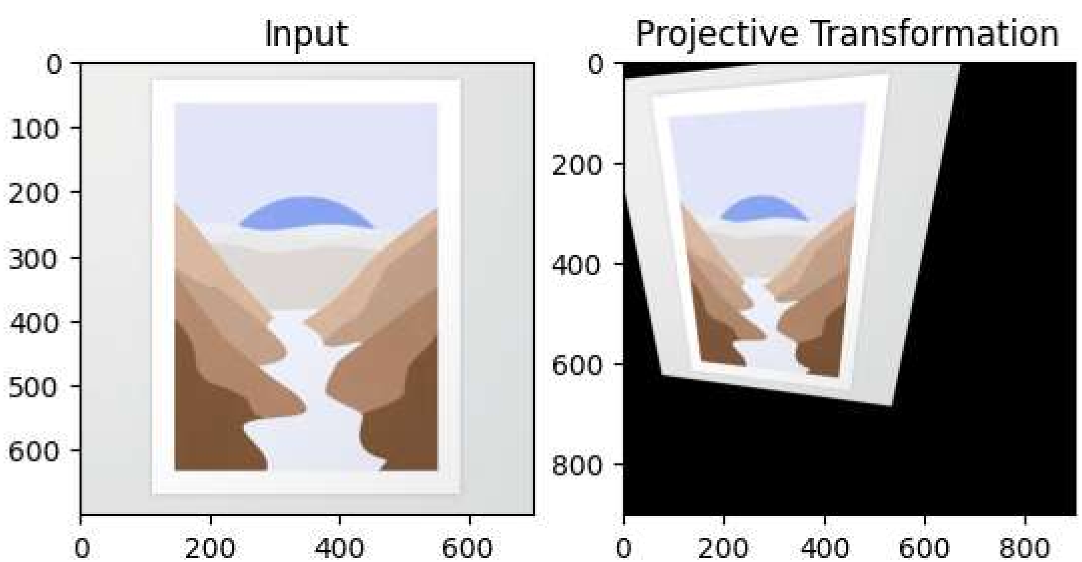
2. Implement the 4 geometric transformation functions using OpenCV in addition to the given example.

In [ ]:
```
#Read the image and check for the shape
img = cv2.imread('image.jpg')
rows,cols,ch = img.shape
```

In [ ]:
```
#similarity

# Define the transformation
scale = 1.2
angle = 30
M_similarity = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle, scale)
# Do the Transformation
dst_similarity = cv2.warpAffine(img, M_similarity, (cols, rows))
# Visualize the original and the transformed images
plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122), plt.imshow(dst_similarity), plt.title('Similarity Transformation'
plt.show()
```

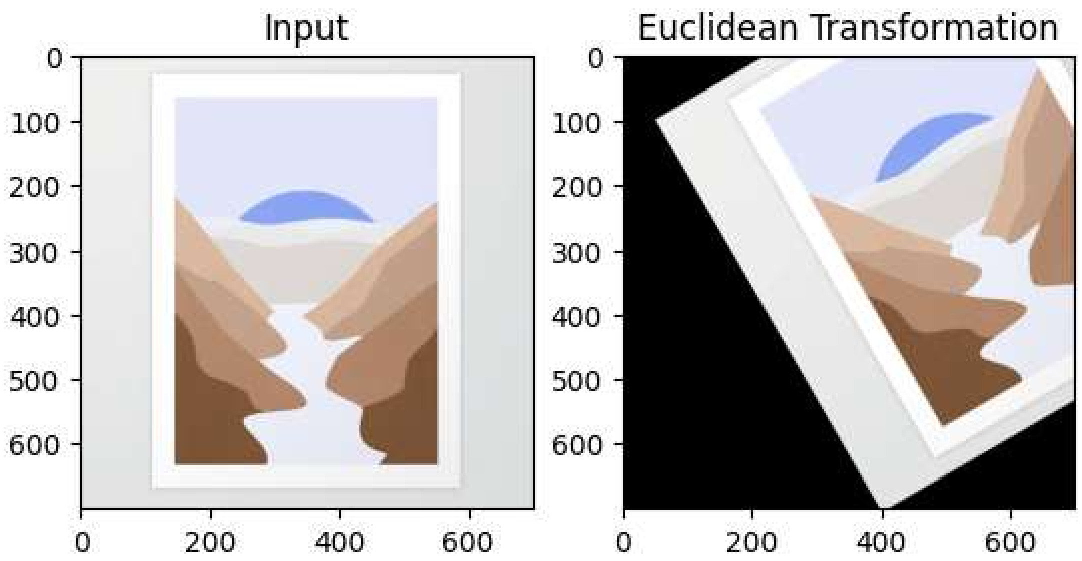


In [ ]:
```
#projective

# Define the transformation
pts1_projective = np.float32([[50, 50], [cols - 50, 50], [50, rows - 50], [cols - 5
pts2_projective = np.float32([[10, 100], [cols - 100, 50], [100, rows - 100], [cols
M_projective = cv2.getPerspectiveTransform(pts1_projective, pts2_projective)
# Adjust canvas size (expand output)
output_size = (cols + 200, rows + 200)  # Add padding for the transformation
# Do the Transformation
dst_projective = cv2.warpPerspective(img, M_projective, output_size)
# Visualize the original and transformed images
plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122), plt.imshow(dst_projective), plt.title('Projective Transformation'
plt.show()
```



In [ ]:
```
# Euclidean

# Define the transformation
angle = 30
tx, ty = 50, 100  # Translation along x and y
```

```
R = cv2.getRotationMatrix2D((0, 0), angle, 1.0)
R[:, 2] += [tx, ty]  # Adding translation
# Do the Transformation
dst_euclidean = cv2.warpAffine(img, R, (cols, rows))
# Visualize the original and transformed images
plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122), plt.imshow(dst_euclidean), plt.title('Euclidean Transformation')
plt.show()
```



In [ ]:

```
# Translation
# Define the transformation
tx, ty = 100, 50  # Translation values
M_translation = np.float32([[1, 0, tx], [0, 1, ty]])
# Do the Transformation
dst_translation = cv2.warpAffine(img, M_translation, (cols, rows))
# Visualize the original and transformed images
plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122), plt.imshow(dst_translation), plt.title('Translation Transformatio
plt.show()
```