AIM: Familiarization of the tool used for computer vision.

CODE:

1. **pip list**

```
Package                       Version
------------------------------------
absl-py                       1.4.0
accelerate                    1.3.0
aiohappyeyeballs              2.4.6
aiohttp                       3.11.12
aiosignal                     1.3.2
alabaster                     1.0.0
albucore                      0.0.23
albumentations                2.0.4
ale-py                        0.10.2
altair                        5.5.0
annotated-types               0.7.0
anyio                         3.7.1
argon2-cffi                   23.1.0
argon2-cffi-bindings          21.2.0
array_record                  0.6.0
arviz                         0.20.0
astropy                       7.0.1
astropy-iers-data             0.2025.2.17.0.34.13
astunparse                    1.6.3
atpublic                      4.1.0
attrs                         25.1.0
audioread                     3.0.1
autograd                      1.7.0
...
yellowbrick                   1.5
yfinance                      0.2.54
zipp                          3.21.0
zstandard                     0.23.0
```

!pip install opencv-python

Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-packages (from opencv-python) (1.26.4)

2. **!pip install tensorflow**

Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages
...
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

3. **!pip install matplotlib**

Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)

Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)

Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)

Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.26.4)

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)

Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)

Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)

Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)

4. **!pip install numpy**

Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)

1.AIM: Write a Program to read and display an image.

CODE:

```
import cv2

# Load an image from file

image = cv2.imread("image.jpeg")  # Replace 'image.jpg' with your image file #

Display the image in a window

cv2.imshow("Loaded Image", image)

# Wait for a key press and close the window

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:

AIM: Write a Program to read and write video files.

CODE:

```
import cv2

# Path to the input video file

video_path = "nature.mp4"

# Create a VideoCapture object

cap = cv2.VideoCapture(video_path)

# Check if the video was successfully opened

if not cap.isOpened():

   print("Error: Could not open video.")

else:

   while True:

      # Read a frame from the video

      ret, frame = cap.read()

      if not ret:

         print("End of video or cannot read frame.")

         break


      # Display the frame

      cv2.imshow('Video Frame', frame)
```

```
break

    # Release the VideoCapture object and close any OpenCV windows

    cap.release()

    cv2.destroyAllWindows()
```

OUTPUT:

AIM: Write a Program to enhance an image using basic functions.

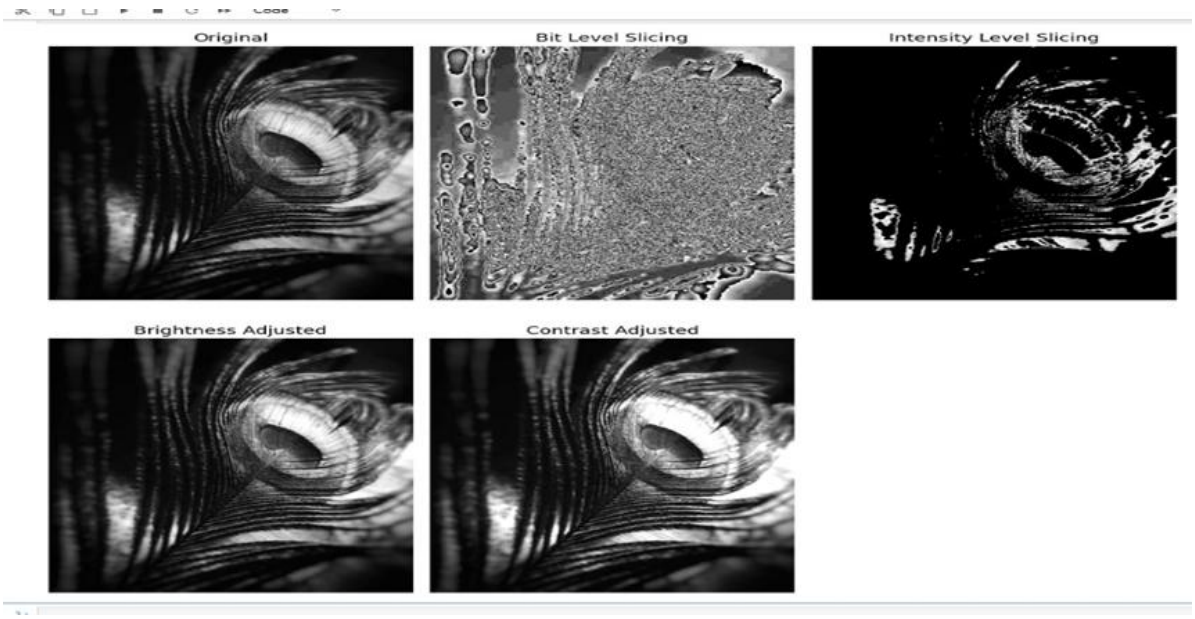
CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


image = cv2.imread('image.jpeg', cv2.IMREAD_GRAYSCALE)


# Function for Bit Level Slicing
def bit_level_slicing(image, bit_position):
    max_bit = 2**bit_position - 1
    sliced_image = np.bitwise_and(image, max_bit)
    return sliced_image


# Function for Intensity Level Slicing
def intensity_level_slicing(image, low, high):
    # Set all pixels outside the range [low, high] to 0
    sliced_image = np.zeros_like(image)
    sliced_image[(image >= low) & (image <= high)] = image[(image >= low) & (image <= high)]
    return sliced_image
```

```
# Function for Brightness Adjustment
def adjust_brightness(image, brightness_value):
    brightness_adjusted = cv2.add(image, brightness_value)
    return brightness_adjusted

# Function for Contrast Adjustment
def adjust_contrast(image, contrast_value):
    contrast_adjusted = cv2.convertScaleAbs(image, alpha=contrast_value, beta=0)
    return contrast_adjusted

# Display the images for comparison
def display_images(images, titles):
    plt.figure(figsize=(10, 10))
    for i, (img, title) in enumerate(zip(images, titles)):
        plt.subplot(2, 3, i+1)
        plt.imshow(img, cmap='gray')
        plt.title(title)
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

```
intensity_sliced = intensity_level_slicing(image, 100, 150)

brightness_adjusted = adjust_brightness(image, 50)

contrast_adjusted = adjust_contrast(image, 1.5)

images = [image, bit_sliced, intensity_sliced, brightness_adjusted, contrast_adjusted]

titles = ['Original', 'Bit Level Slicing', 'Intensity Level Slicing', 'Brightness Adjusted', 'Contrast Adjusted']

display_images(images, titles)
```

OUTPUT:



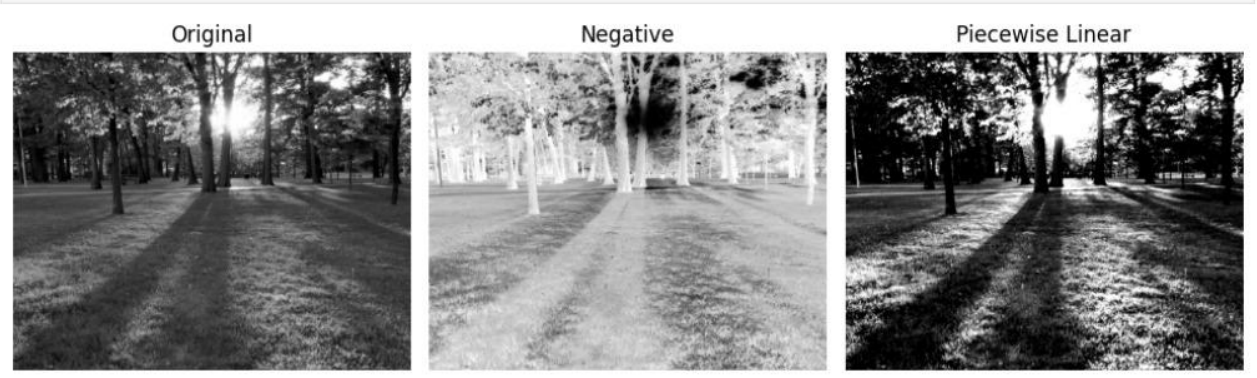
AIM: Write a Program to enhance an image using transformational functions image negative and piecewise linear functions.

CODE:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load the image in grayscale

image = cv2.imread("refs.jpg", cv2.IMREAD_GRAYSCALE)

# 1. Image Negative Transformation

negative = 255 - image

# 4. Piecewise Linear Transformation

def piecewise_linear(img):

  r1, s1 = 50, 0

  r2, s2 = 150, 255

  img = img.astype(np.float32)

  # Apply transformation

  img_transformed = np.piecewise(img,

                    [img <= r1, (img > r1) & (img <= r2), img > r2],

                   [lambda r: (s1 / r1) * r,

                    lambda r: ((s2 - s1) / (r2 - r1)) * (r - r1) + s1,

                    lambda r: ((255 - s2) / (255 - r2)) * (r - r2) + s2])

  return np.uint8(img_transformed)
```

```
piecewise_transformed = piecewise_linear(image)

# Display all images

titles = ['Original', 'Negative', 'Piecewise Linear']

images = [image, negative, piecewise_transformed]

plt.figure(figsize=(10, 6))

for i in range(3):

    plt.subplot(2, 3, i+1)

    plt.imshow(images[i], cmap='gray')

    plt.title(titles[i])

    plt.axis('off')

plt.tight_layout()

plt.show()
```

OUTPUT:



AIM: Write a Program to enhance an image using log transformation and power law transformation.

CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Load the image in grayscale
image = cv2.imread("refs.jpg", cv2.IMREAD_GRAYSCALE)
# 2. Log Transformation (c * log(1 + r))
c = 255 / np.log(1 + np.max(image))  # Compute constant
log_transformed = c * np.log(1 + image)  # Apply log function
log_transformed = np.uint8(log_transformed)  # Convert back to uint8
# 3. Power Law (Gamma) Transformation (s = c * r^γ)
gamma = 2.2  # Change this value to adjust brightness
gamma_transformed = np.array(255 * (image / 255) ** gamma, dtype=np.uint8)
titles = ['Original',  'Log Transform', 'Power Law (Gamma)']
images = [image, log_transformed, gamma_transformed]
plt.figure(figsize=(10, 6))
for i in range(3):
   plt.subplot(2, 3, i+1)
   plt.imshow(images[i], cmap='gray')
```

```
plt.tight_layout()

plt.show()
```

OUTPUT:

AIM: Write a program to implement histogram calculation and equalization for the given image.

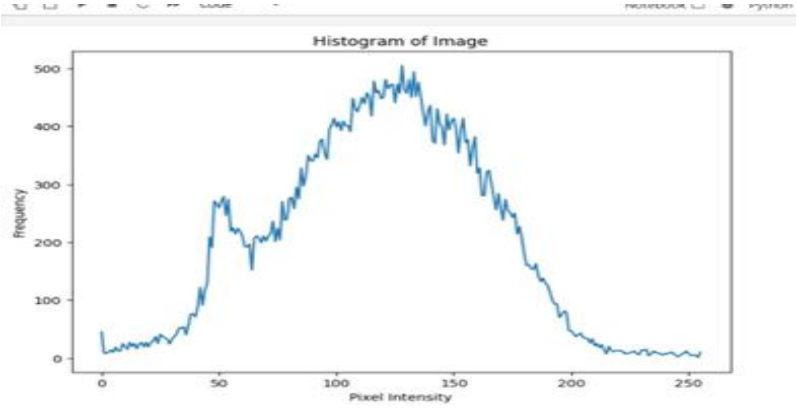
CODE:

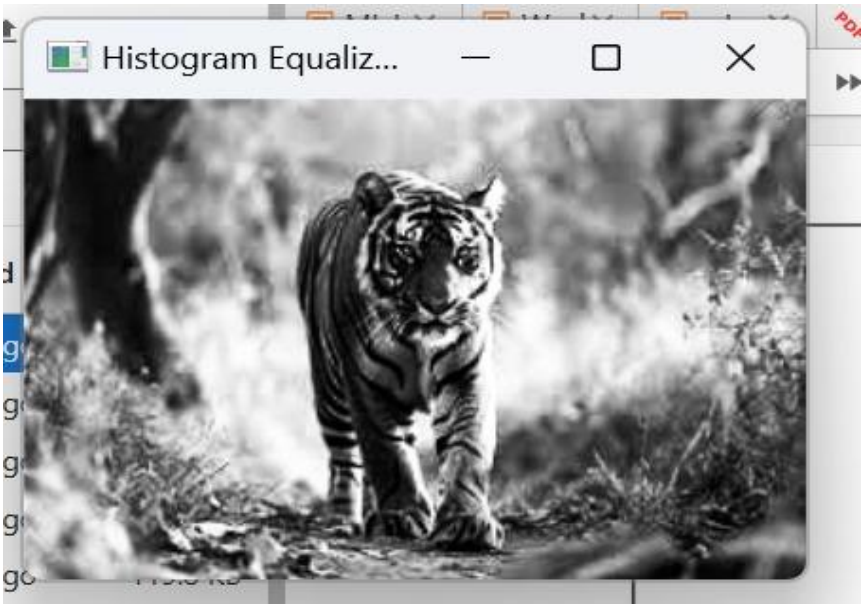
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def calculate_histogram(image):
    """
    Calculate and plot histogram of an image.
    """
    histogram = cv2.calcHist([image], [0], None, [256], [0, 256])
    plt.figure(figsize=(8, 6))
    plt.plot(histogram)
    plt.title("Histogram of Image")
    plt.xlabel("Pixel Intensity")
    plt.ylabel("Frequency")
    plt.show()
def histogram_equalization(image):
    """
    Perform histogram equalization on the image.
    """
    equalized_image = cv2.equalizeHist(image)
```

```
def clahe_equalization(image):
    """
    Perform CLAHE (Contrast Limited Adaptive Histogram Equalization).
    """
    # Create a CLAHE object with default parameters
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    # Apply CLAHE to the image
    clahe_image = clahe.apply(image)
    return clahe_image
# Load the image in grayscale
image_path = 'lion.jpg'  # Replace with the path to your image

 image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
# Step 1: Calculate and plot histogram of original image
calculate_histogram(image)
# Step 2: Perform Histogram Equalization
equalized_image = histogram_equalization(image)
# Step 3: Perform CLAHE
clahe_image = clahe_equalization(image)
# Display results
cv2.imshow("Original Image", image)
cv2.imshow("Histogram Equalized Image", equalized_image)
```

```
cv2.imshow("CLAHE Image", clahe_image)

# Wait for user input to close the images

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:




Histogram Equaliz...


CLAHE Image

AIM: Write a program to implement histogram specification.

CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


def hist_match(source, reference):


    src_hist, bins = np.histogram(source.flatten(), 256, [0, 256])
    ref_hist, _ = np.histogram(reference.flatten(), 256, [0, 256])


    src_cdf = np.cumsum(src_hist) / src_hist.sum()
    ref_cdf = np.cumsum(ref_hist) / ref_hist.sum()
    mapping = np.interp(src_cdf, ref_cdf, np.arange(256))


    matched_image = np.interp(source.flatten(), np.arange(256), mapping).reshape(source.shape)
    return np.uint8(matched_image)



source_image = cv2.imread('source.jpg')
reference_image = cv2.imread('refernece.jpg')
```

```
# Convert both images to grayscale
#source_image_gray = cv2.cvtColor(source_image, cv2.COLOR_BGR2GRAY)
#reference_image_gray = cv2.cvtColor(reference_image, cv2.COLOR_BGR2GRAY)
# Perform histogram matching
result_image = hist_match(source_image, reference_image)
plt.figure(figsize=(10, 5))
plt.subplot(1, 3, 1)
plt.title('Source Image (Grayscale)')
plt.imshow(source_image, cmap='gray')
plt.subplot(1, 3, 2)
plt.title('Reference Image (Grayscale)')
plt.imshow(reference_image, cmap='gray')
plt.subplot(1, 3, 3)
plt.title('Matched Image')
plt.imshow(result_image, cmap='gray')
plt.show()
```

OUTPUT:

AIM: Write a program to perform Arithmetic operators.

CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def arithmetic_operations(image1_path, image2_path):
    # Load images
    image1 = cv2.imread(image1_path)
    image2 = cv2.imread(image2_path)
    image2 = cv2.resize(image2, (image1.shape[1], image1.shape[0]))
    # Perform arithmetic operations
    add_result = cv2.add(image1, image2)
    subtract_result = cv2.subtract(image1, image2)
    multiply_result = cv2.multiply(image1, image2)
    divide_result = cv2.divide(image1.astype(np.float32), image2.astype(np.float32))
    # Resize for display
    display_size = (200,200)

image1_resized = cv2.resize(image1, display_size)
    image2_resized = cv2.resize(image2, display_size)
```

```
multiply_resized = cv2.resize(multiply_result, display_size)
    divide_resized = cv2.resize(divide_result, display_size)
    # Show results
    cv2.imshow('Image 1', image1_resized)
    cv2.imshow('Image 2', image2_resized)
    cv2.imshow('Addition', add_resized)
    cv2.imshow('Subtraction', subtract_resized)
    cv2.imshow('Multiplication', multiply_resized)
    cv2.imshow('Division', divide_resized)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
image1_path = 'india.jpg'  # Replace with your first image file
image2_path = 'ocean.jpg'  # Replace with your second image file
arithmetic_operations(image1_path, image2_path)
```

OUTPUT:

AIM: Write a program to perform Logical operators.

CODE:

```
import cv2
import numpy as np


def logical_operations(image1_path, image2_path):
    # Load images
    image1 = cv2.imread(image1_path, cv2.IMREAD_GRAYSCALE)
    image2 = cv2.imread(image2_path, cv2.IMREAD_GRAYSCALE)
        image1 = cv2.resize(image1, (200, 200))
    image2 = cv2.resize(image2, (200, 200))
    # Perform logical operations
    and_result = cv2.bitwise_and(image1, image2)
 or_result = cv2.bitwise_or(image1, image2)
    xor_result = cv2.bitwise_xor(image1, image2)
 not_result1 = cv2.bitwise_not(image1)
    not_result2 = cv2.bitwise_not(image2)
    # Show results
    cv2.imshow('Image 1 (Resized)', image1)
    cv2.imshow('Image 2 (Resized)', image2)
```

```
 cv2.imshow('AND Operation', and_result)
    cv2.imshow('OR Operation', or_result)
    cv2.imshow('XOR Operation', xor_result)
    cv2.imshow('NOT Image 1', not_result1)
    cv2.imshow('NOT Image 2', not_result2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
# Example usage
image1_path = 'ocean.jpg'  # Replace with your first image file
image2_path = 'doraemon.jpg'  # Replace with your second image file
logical_operations(image1_path, image2_path)
```

OUTPUT: