SOFTWARE LAB ASSIGNMENT-4

Name - Jatin Mahawar

Roll.NO. - 22CS30032

1. Write a program to load a .csv file as a NumPy 1-D array. Find the maximum and minimum elements in the array.

```
 # Program 1 -
      import numpy as np
      import time

      # function to read csv and add data to np array
      def read_convert_array(filename):
          data = np.genfromtxt(filename, delimiter="\t", skip_header=1,
      dtype=int)
          new = data[:,1]
          return new

      start = time.time()
      data_array = read_convert_array('book1.csv')
      mx = data_array.max()
      mn = data_array.min()

      print(f"Max: {mx}, Min: {mn}")
      end = time.time()

      print(f"Time: {end - start}")

# Output -

      Max: 100, Min: 1
      Time: 0.028958559036254883
```

---

2. For the Numpy 1-D array as obtained in Q.1, sort the elements in ascending order.

```
# Program 2
      # Sorting the Array from q1 in ascending order
      data_array.sort()
      print(data_array)
```

---

3. For the sorted Numpy 1-D array as obtained in Q.2, reverse the array and print.

```
# Program 3
      # flipping the array got in Q.2
      reverse_array = np.flip(data_array)
      print(reverse_array)
```

---

4. Write a program to load three .csv files (Book1.csv, Book2.csv, and Book3.csv) as a list of Numpy 1-D arrays. Print the means of all arrays as a list.

```
# Program 4
      import numpy as np
      import time
      # function to read and convert array
      start = time.time()
      def read_convert_array(filename):
          data = np.genfromtxt(filename, delimiter="\t", skip_header=1, dtype=int)
          new = data[:,1]
          return new

      # reading and converting arrays from files
      np_a1 = read_convert_array("book1.csv")
      np_a2 = read_convert_array("book2.csv")
      np_a3 = read_convert_array("book3.csv")
```

```
    # list of all arrays
    list_of_all_array = [np_a1, np_a2, np_a3]
    list_of_means = [np.mean(np_a1), np.mean(np_a2), np.mean(np_a3)]
    print(f"Median of Every Array: {list_of_means}")
    end = time.time()

    print(f"Time taken: {end - start} seconds")

# Output
    Median of Every Array: [48.566, 50.57, 513.326]
    Time taken: 0.013991355895996094 seconds
```

---

5. Write a program to read an image, store the image in NumPy 3-D array. For the image, consider a.PNG. Display the image. Let the image stored in the NumPy array be X.
Hint: Use OpenCV to work with image

```
# Program 5
    import cv2
    import numpy as np

    X = cv2.imread('a.png')
    cv2.imshow("Original Image", X)
    cv2.waitKey(0)
    cv2.destroyAllWindows() # Press Any key to close the window
    print(X.ndim) # to check the dimension
```

---

6. Write a program to convert a color image (say a.PNG) into a grescale image. Let the greysacle image stored in the Numpy 2-D array be X. Display the grayscale iamge on the screen.
Hint: Greyscale value of a pixel is the mean of three RGB values of that pixel.

```
# Program
    import numpy as np
```

```
import cv2

original = cv2.imread('a.png')
# changing original coloured image to gray
X = np.mean(original, axis=2, dtype=int).astype(np.uint8)
cv2.imshow("GrayScale Image", X)
cv2.waitKey(0)
cv2.destroyAllWindows()
print(X.ndim)
```

---

7. Let Y be the transpose matrix of X. Write a program to obtain Z = X×Y.

```
#Program 7
import numpy as np
import cv2
import time as t

original = cv2.imread('a.png')
X = np.mean(original, axis=2, dtype=int).astype(np.uint8)

# Tranpose of X
Y = np.transpose(X)

# Multyplication of X and Y, calculating time also
t1 = t.time()
Z = np.matmul(X.astype(np.uint16), Y.astype(np.uint16))
t2 = t.time()
print(Z)
print("Time taken: ", t2 - t1, "Seconds")

# Output
Time taken:  1.9006061553955078 Seconds
```

8. For the problem in Q. 7, write your program without using NumPy library. Compare the computation times doing the same with NumPy and basic programming in Python.

```
# Program 8
      import numpy as np
      import cv2
      import time as t

      original = cv2.imread('a.png')
      X = np.mean(original, axis=2, dtype=int).astype(np.uint8)

      # Transpose of X
      Y = np.zeros((X.shape[1], X.shape[0]), dtype=int)
      for i in range(X.shape[0]):
          for j in range(X.shape[1]):
              Y[j][i] = X[i][j]

      print(X.shape)
      print(Y.shape)
      # Define matrices as lists
      x = X.tolist()
      y = Y.tolist()

      # Multyply both matrices and store in a matrix without using numpy function
      t1 = t.time()
      r1 = X.shape[0]
      c2 = Y.shape[1]
      c1 = X.shape[1]
      result = [[0 for j in range(c2)] for i in range(r1)]

      for i in range(r1):
          for j in range(c2):
              for k in range(c1):
                  result[i][j] += x[i][k] * y[k][j]
      t2 = t.time()
      print("Time taken without using numpy function: ", t2 - t1)
```

```
#output
     Time taken without using numpy function:  215.22431588172913
```

---

9. Plot the pixel intensity histogram of the grescale image stored in X. Hint: Use matplotlib to plot the histogram.

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
# function to plot the INtensity plot
def Plot_Instensity_Diagram(filename):
    original = cv2.imread(filename)
     # checking the file has opened or not
    if original is None:
        print('Could not open or find the image:', filename)
        exit(0)

    X = np.mean(original, axis=2, dtype=int).astype(np.uint8)

    plt.hist(X.flatten(), bins=range(0,256), color='blue', alpha= 0.7)
    plt.title("Pixel Intensity Histogram")
    plt.xlabel("Pixel Intensity")
    plt.ylabel("Frequency")
    plt.tight_layout()
    plt.show()


Plot_Instensity_Diagram('a.png')
```

---

10. Create a black rectangle at the position [(40,100) top right, (70, 200) bottom left] in the grayscale image. Display the image.

```
import cv2
```

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

def draw_black_rectangle(image, top_right, bottom_left):
    new_image = image.copy() # making copy of image array

    cv2.rectangle(new_image, top_right, bottom_left, (0,0,0), -1)
    return new_image

filename = 'a.png'
original = cv2.imread(filename)

if original is None:
    print('Could not open or find the image:', filename)
    exit(0)
else:

    top_right = (40,100)
    bottom_left = (70,200)
    new_image = draw_black_rectangle(original, top_right, bottom_left)
    X = np.mean(new_image, axis=2, dtype=int).astype(np.uint8)

    plt.subplot(1,1,1)
    plt.imshow(X, cmap='gray')
    plt.title("Image with Black Rectangle")

    plt.tight_layout()
    plt.show()
```

---

11. Using the grayscale image stored in X, transform it into the binarized image with thresholds: [50, 70, 100, 150]. Let the binarized images are stored in Z50, Z70, Z100, and Z150, respectively. Hint: Binarizing is thresholding each pixel value, i.e., if pixel>threshold, then 1 else 0.

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

```
def display_binarized_images(X, thresholds):
    # Display binarized images with different thresholds
    for i, threshold in enumerate(thresholds, start=2):
        binarized_image = np.where(X > threshold, 255,
0).astype(np.uint8)
        plt.subplot(2, 2, i-1)
        plt.imshow(binarized_image, cmap='binary')
        plt.title(f"Binarized (Threshold={threshold})")

    plt.tight_layout()
    plt.show()

image_path = 'a.png'

image = cv2.imread(image_path)
X = np.mean(image, axis=2, dtype=int).astype(np.uint8)

# Display the binarized images with different thresholds
thresholds = [50, 70, 100, 150]
display_binarized_images(X, thresholds)
```

---

12. Consider the color image stored in a.png. Create a filter of [[-1,-1,-1][0,0,0][1,1,1]], and multiply this filter to each pixel value in the image. Display the image after filtering.

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


image_path = 'a.png'
# Read the color image using OpenCV
image = cv2.imread(image_path)
```

```
if image is None:
    print(f"Error: Unable to read the image from {image_path}")
else:
    # Define the filter/filtre
    filtre = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])

    # Apply the filter to the color image
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            image[i, j] = np.matmul(image[i][j], filtre)


    # Display the filtered image
    plt.subplot(1, 1, 1)
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title("Resultant Image after Applying Filter")

    plt.tight_layout()
    plt.show()
```
