nment-4-numpy-and-its-applications

February 12, 2024

Software Engineering Lab Assignment-4: NumPy and its Applications

NAME : Tuhin Mondal ROLL NO: 22CS10087

[31]:
```
# Import necessary libraries
import numpy as np
import time
import csv
import cv2
import matplotlib.pyplot as plt
```

[18]:
```
# ------------------------------------------------
# PROGRAM 1: Read data from CSV, find max and min
# ------------------------------------------------
print("PROGRAM 1 :")

# Read data from CSV file ('book1.csv')
array = np.genfromtxt('book1.csv', delimiter='\t', usecols=1, skip_header=1,␣
 ↪dtype=int)
print(array)

# Find maximum and minimum elements in the array
max_element = np.max(array)
min_element = np.min(array)
print("Max element = ", max_element, "\nMin element = ", min_element)
```

```
PROGRAM 1 :
[ 16  18  24  34  85  17  69  86  83  82  47  43   4   6  34  22   5  47
  32  42  71  26  65  70  63  23  62  99  62  68  14  29  67  21  55  24
  37  96  25  33  53  66  10   9   8  21  54  95   5  96  52  21  46  59
   7  28  57 100  49  50  66  84  39  72   7   5   6  28  75  39  62  39
  75  99  30  61  77  70  34  60  56  58  25  60  55  97  53  83  73  17
  83  42  90  46  66  71  25  94  57  47  75  81  61  94  74  61  42   1
  39  38  19   6  30  27  95   1  91  38  27  60  25  76  77  16  47  19
  91   3  74  51  97  25  96  40  89  45  82  18  78  44  24  28  67   2
  32  27  95  95  99  71  55  56  63  43  41  39  87  87  57  78  37  40
  22  70  54  20  63   6  28  68   1  25  69  41  47  88  12  35  18  88
  12  45  88  13  22  62  75  69  63  94  13  94  96  14  64  90  15  36
  34  96  46   5  54  19  42  81  57  92  14  29  95  23  11  53  62  13
```

```
  94  76  89  38  20  19  95  19  45  23  35  73  17  65  13  93  12  12
  46  96  93  98  73  69  37  67  13  77 100  54  69   5  28  89  45  70
   9  66  37  86  13  52   6  71  10  29   2  25   3  82  60  92  32  36
  22  67  30  78  53  11  21   4  63  46  51  42  23  13  24  92   5  33
  23  86   6  27  34  88  62  75  83  56  76  12  28  23  74  22   8  95
  48  58  84  29  87  79  19  20  22  11  66  49  56  46  68  89  46  16
   9  76   8  19  17  95  35  55  37  77  67  68  93  97  73  27  23   6
  28  28  86  82  69  82  34  52  22  45  16  24  66  12  53  48  18  83
   2  70  16  64  54  37  55  85  85  84  55  34  78  85  92  19   8  15
  70  39  37  81  18  83  43  81  18  88  32  57  75  33  29  81  42   5
  87  26  75  59  47  80  15  61   8  26   7  32  59  17  75  22  32  98
  79  78  83  44  54  30  28  58  36  78  70  99  13  61  59  32  47  77
  28  52  24  34  16  60  53  67  86 100  71  65  89  92  15  65   8  78
  61  83  32  34  17   9  88  33  59  62  34  16  95  41   1  62  93  36
  13  85  14  46  79  18   3  83  82  54   1  75  35  36   6  22  69  48
  12  60  74  24  24  36  72  43  92  10  33  40  89  31  96  74  53  33
  66  28  14  85  98  66  78  36  53  97  10  42  36  22  32  89   2  62
  69  82  99  47  35  67  74  81  64  27  57  20  20  85  42  66  18   9
  79  31  94   6  78  32  50  84  79  34  71  92  24  29   7  70  12   7
  37   8  77  74  24  55  18  97  75  19  14  51  87  90  84   9  61  25
  34  82  82  71  84  52  95  33  70  68  53  70   6  59  53  60  35  45
  14  75  84  52  45  14  75  19  78  45  79  95  36  93  53  77  93  83
  95  80  59   9  94  71  11  99  23  31  22  21  55  34  30  11  29  18
  98  29   1  77  68  86  52  12  35  72  10  69  72  50  68  39  46  31
  27   3  77   1  29  97  44  27  42  96  57   4   2  52  48  54  76  82
  28  61  49  32  65  18  74  94  13  71  29  66  33  37  66  39  70  40
  15  61  73  41  23  19  28  96  59   4  82  83  37  45  14  10  38  21
  81  33  88  56  40  57  78  44  81  23  83  36  87  33  26  70  37  72
  85  79  17  34  22   3  14   7  51  66  86  63   1  11  74  43  74  45
  52   2   6  52  50   6  64  23  87  97   8  42  90  16  40  10  46   8
  62  73  63  40  12  83  53  42  44  98  77   3  60  66   4  77  39  43
  43  10  65  21  95  24  86  27  78  19  23   5  25 100   4  71  52  65
  60  58  38   9  38  56  99  24  40   5   2  48   9  46  63  71   2  81
  32  82  68  88  40  70   4  38  13  70  61  54  41   7  38  78  75  22
  80  89  88  28  45  12  29  20  51  93  13  75  57  58  74  43  66  12
  27  19  30  17   6  63  92  65  77  29  55  87  24  27  88   2  10  27
  56  18  98  11  18  69   4  19  19   1  28  29  52  44  48  15  95  51
  49  40  84  53  73  57  90  41  20  68   8  92  61  23  28  64  19  39
  32  92 100  93  40  66  45  91  32  39  45  73  30  57  73  33  59   8
   8  23  17  86  35  28  54  71   1  43  47  70   8  45  49  16  81  93
  23  68  43   9  48  54  85  66  19  76  56  29   7  25  60  44  47  69
  74  89  75  15  12  86   5  14   6  39  57  52  33  52  33  41  69  16
   1  50  86  41  42  29  12  76  44  98  54  11   6  61  95  22  44  30
  83   6  12  51  27  65  95  29  65   3]
Max element =  100
Min element =  1
```

[19]:
```
# ------------------------------------------------
# PROGRAM 2: Sort the array
# ------------------------------------------------
print("\nPROGRAM 2 :")

# Sort the array obtained from 'book1.csv'
sorted_array = np.sort(array)
print(sorted_array)
```

```
PROGRAM 2 :
[  1   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2
   2   2   3   3   3   3   3   3   3   4   4   4   4   4   4   4   4   5
   5   5   5   5   5   5   5   5   5   6   6   6   6   6   6   6   6   6
   6   6   6   6   6   6   6   7   7   7   7   7   7   7   7   8   8   8
   8   8   8   8   8   8   8   8   8   8   9   9   9   9   9   9   9   9
   9   9  10  10  10  10  10  10  10  10  10  11  11  11  11  11  11  11
  11  12  12  12  12  12  12  12  12  12  12  12  12  12  12  12  13  13
  13  13  13  13  13  13  13  13  13  13  14  14  14  14  14  14  14  14
  14  14  14  15  15  15  15  15  15  15  16  16  16  16  16  16  16  16
  16  16  17  17  17  17  17  17  17  17  17  18  18  18  18  18  18  18
  18  18  18  18  18  18  19  19  19  19  19  19  19  19  19  19  19  19
  19  19  19  19  19  20  20  20  20  20  20  20  21  21  21  21  21  21
  21  22  22  22  22  22  22  22  22  22  22  22  22  22  22  22  23  23  23
  23  23  23  23  23  23  23  23  23  23  23  23  24  24  24  24  24  24
  24  24  24  24  24  24  24  25  25  25  25  25  25  25  25  25  25  26
  26  26  26  27  27  27  27  27  27  27  27  27  27  27  27  27  28  28
  28  28  28  28  28  28  28  28  28  28  28  28  28  28  28  29  29  29
  29  29  29  29  29  29  29  29  29  29  29  29  29  30  30  30  30  30
  30  30  30  31  31  31  31  32  32  32  32  32  32  32  32  32  32  32
  32  32  32  33  33  33  33  33  33  33  33  33  33  33  33  33  34  34
  34  34  34  34  34  34  34  34  34  34  34  34  35  35  35  35  35  35
  35  35  36  36  36  36  36  36  36  36  36  36  37  37  37  37  37  37
  37  37  37  37  37  38  38  38  38  38  38  38  38  39  39  39  39  39
  39  39  39  39  39  39  39  40  40  40  40  40  40  40  40  40  40  40
  41  41  41  41  41  41  41  41  42  42  42  42  42  42  42  42  42  42
  42  42  43  43  43  43  43  43  43  43  43  43  44  44  44  44  44  44
  44  44  44  45  45  45  45  45  45  45  45  45  45  45  45  45  45  46
  46  46  46  46  46  46  46  46  46  46  47  47  47  47  47  47  47  47
  47  47  48  48  48  48  48  48  48  49  49  49  49  49  50  50  50  50
  50  51  51  51  51  51  51  51  52  52  52  52  52  52  52  52  52  52
  52  52  52  52  53  53  53  53  53  53  53  53  53  53  53  53  53  54
  54  54  54  54  54  54  54  54  54  54  54  55  55  55  55  55  55  55
  55  55  56  56  56  56  56  56  56  56  57  57  57  57  57  57  57  57
  57  57  57  57  58  58  58  58  58  59  59  59  59  59  59  59  59  59
  60  60  60  60  60  60  60  60  60  60  61  61  61  61  61  61  61  61
  61  61  61  61  62  62  62  62  62  62  62  62  62  62  62  63  63  63  63
```

```
 63  63  63  63  63  64  64  64  64  64  65  65  65  65  65  65  65  65
 65  65  66  66  66  66  66  66  66  66  66  66  66  66  66  66  66  66
 67  67  67  67  67  67  67  68  68  68  68  68  68  68  68  68  68  69
 69  69  69  69  69  69  69  69  69  69  69  70  70  70  70  70  70  70
 70  70  70  70  70  70  70  70  71  71  71  71  71  71  71  71  71  71
 71  71  72  72  72  72  72  73  73  73  73  73  73  73  73  73  74  74
 74  74  74  74  74  74  74  74  74  74  75  75  75  75  75  75  75  75
 75  75  75  75  75  75  75  76  76  76  76  76  76  76  77  77  77  77
 77  77  77  77  77  77  77  77  78  78  78  78  78  78  78  78  78  78
 78  78  78  79  79  79  79  79  79  79  80  80  80  81  81  81  81  81
 81  81  81  81  81  82  82  82  82  82  82  82  82  82  82  82  82  83
 83  83  83  83  83  83  83  83  83  83  83  83  84  84  84  84  84
 84  84  84  85  85  85  85  85  85  85  85  85  86  86  86  86  86  86
 86  86  86  86  86  87  87  87  87  87  87  87  87  88  88  88  88  88
 88  88  88  88  88  89  89  89  89  89  89  89  89  89  90  90  90  90
 90  91  91  91  92  92  92  92  92  92  92  92  92  92  93  93  93  93
 93  93  93  93  93  94  94  94  94  94  94  94  94  95  95  95  95  95
 95  95  95  95  95  95  95  95  95  95  95  96  96  96  96  96  96  96
 96  96  97  97  97  97  97  97  97  98  98  98  98  98  98  98  99  99
 99  99  99  99  99 100 100 100 100 100]
```

[20]:
```
# ---------------------------------------------------
# PROGRAM 3: Reverse the sorted array
# ---------------------------------------------------
print("\nPROGRAM 3 :")

# Reverse the sorted array
reversed_array = np.flip(sorted_array)
print(reversed_array)
```

```
PROGRAM 3 :
[100 100 100 100 100  99  99  99  99  99  99  99  98  98  98  98  98  98
  98  97  97  97  97  97  97  97  96  96  96  96  96  96  96  96  96  95
  95  95  95  95  95  95  95  95  95  95  95  95  95  95  95  94  94  94
  94  94  94  94  94  93  93  93  93  93  93  93  93  93  92  92  92  92
  92  92  92  92  92  92  91  91  91  90  90  90  90  90  89  89  89  89
  89  89  89  89  89  88  88  88  88  88  88  88  88  88  88  87  87  87
  87  87  87  87  87  86  86  86  86  86  86  86  86  86  86  86  85  85
  85  85  85  85  85  85  85  84  84  84  84  84  84  84  84  84  83  83  83
  83  83  83  83  83  83  83  83  83  83  83  82  82  82  82  82  82  82
  82  82  82  82  82  81  81  81  81  81  81  81  81  81  81  80  80  80
  79  79  79  79  79  79  79  78  78  78  78  78  78  78  78  78  78  78
  78  78  77  77  77  77  77  77  77  77  77  77  77  77  76  76  76  76
  76  76  76  75  75  75  75  75  75  75  75  75  75  75  75  75  75  75
  74  74  74  74  74  74  74  74  74  74  74  74  73  73  73  73  73  73
  73  73  73  72  72  72  72  72  71  71  71  71  71  71  71  71  71  71
  71  71  70  70  70  70  70  70  70  70  70  70  70  70  70  70  70  69
```

```
 69  69  69  69  69  69  69  69  69  69  69  68  68  68  68  68  68  68
 68  68  68  67  67  67  67  67  67  67  66  66  66  66  66  66  66  66
 66  66  66  66  66  66  66  66  65  65  65  65  65  65  65  65  65  65
 64  64  64  64  64  63  63  63  63  63  63  63  63  63  62  62  62  62
 62  62  62  62  62  62  61  61  61  61  61  61  61  61  61  61  61  61
 60  60  60  60  60  60  60  60  60  60  59  59  59  59  59  59  59  59
 59  58  58  58  58  58  57  57  57  57  57  57  57  57  57  57  57  57
 56  56  56  56  56  56  56  56  55  55  55  55  55  55  55  55  55  54
 54  54  54  54  54  54  54  54  54  54  54  53  53  53  53  53  53  53
 53  53  53  53  53  53  52  52  52  52  52  52  52  52  52  52  52  52
 52  52  51  51  51  51  51  51  51  50  50  50  50  50  49  49  49  49
 49  48  48  48  48  48  48  48  47  47  47  47  47  47  47  47  47  47
 46  46  46  46  46  46  46  46  46  46  46  45  45  45  45  45  45  45
 45  45  45  45  45  45  45  44  44  44  44  44  44  44  44  44  43  43
 43  43  43  43  43  43  43  43  42  42  42  42  42  42  42  42  42  42
 42  42  41  41  41  41  41  41  41  41  40  40  40  40  40  40  40  40
 40  40  40  39  39  39  39  39  39  39  39  39  39  39  39  39  38  38  38
 38  38  38  38  38  37  37  37  37  37  37  37  37  37  37  37  36  36
 36  36  36  36  36  36  36  36  35  35  35  35  35  35  35  35  34  34
 34  34  34  34  34  34  34  34  34  34  34  34  33  33  33  33  33  33
 33  33  33  33  33  33  33  32  32  32  32  32  32  32  32  32  32  32
 32  32  32  31  31  31  31  30  30  30  30  30  30  30  30  29  29  29
 29  29  29  29  29  29  29  29  29  29  29  29  29  28  28  28  28  28
 28  28  28  28  28  28  28  28  28  28  28  28  27  27  27  27  27  27
 27  27  27  27  27  27  27  26  26  26  26  25  25  25  25  25  25  25
 25  25  25  24  24  24  24  24  24  24  24  24  24  24  24  24  23  23
 23  23  23  23  23  23  23  23  23  23  23  23  23  22  22  22  22  22
 22  22  22  22  22  22  22  22  22  21  21  21  21  21  21  21  20  20
 20  20  20  20  20  19  19  19  19  19  19  19  19  19  19  19  19  19
 19  19  19  19  18  18  18  18  18  18  18  18  18  18  18  18  18  17
 17  17  17  17  17  17  17  17  16  16  16  16  16  16  16  16  16  16
 15  15  15  15  15  15  15  14  14  14  14  14  14  14  14  14  14  14
 13  13  13  13  13  13  13  13  13  13  13  13  12  12  12  12  12  12
 12  12  12  12  12  12  12  12  12  11  11  11  11  11  11  11  11  10
 10  10  10  10  10  10  10  10   9   9   9   9   9   9   9   9   9   9
  8   8   8   8   8   8   8   8   8   8   8   8   8   7   7   7   7   7
  7   7   7   6   6   6   6   6   6   6   6   6   6   6   6   6   6   6
  6   5   5   5   5   5   5   5   5   5   5   4   4   4   4   4   4   4
  4   3   3   3   3   3   3   3   2   2   2   2   2   2   2   2   2   1
  1   1   1   1   1   1   1   1   1   1]
```

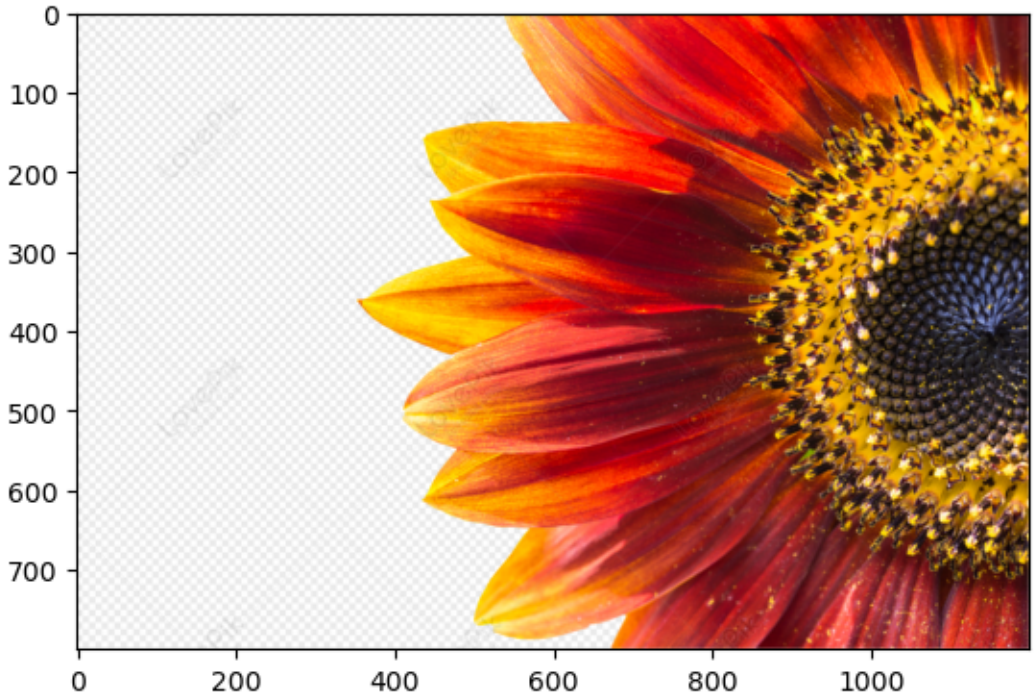
[21]:
```python
# ------------------------------------------------
# PROGRAM 4: Calculate means of arrays from CSV files
# ------------------------------------------------
print("\nPROGRAM 4 :")

# Define CSV filenames
```

```
filenames = ['Book1.csv', 'Book2.csv', 'Book3.csv']
array_list = []

# Read data from each CSV file and calculate mean
for filename in filenames:
    array = np.genfromtxt(filename, delimiter='\t', usecols=1, skip_header=1,␣
 ↪dtype=float)
    array_list.append(array)

means = []
for array in array_list:
    if array is not None:
        mean = np.mean(array)
        means.append(mean)
    else:
        means.append(None)
print("Means of all arrays:", means)
```

```
PROGRAM 4 :
Means of all arrays: [48.566, 51.08844672, 513.326]
```

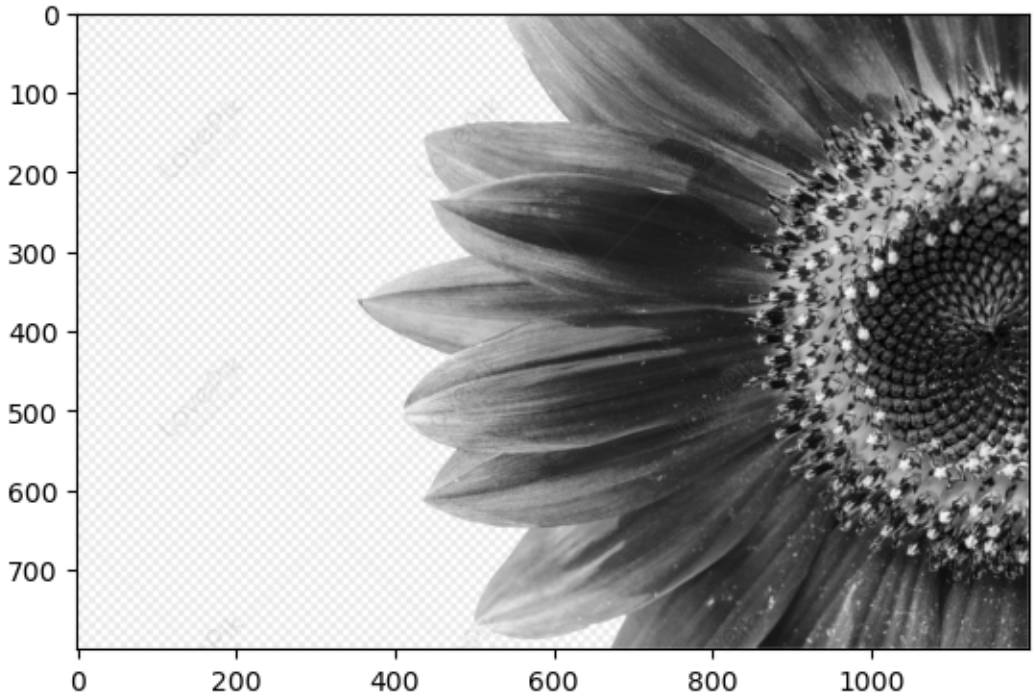
[22]:
```
# ------------------------------------------------
# PROGRAM 5: Load and display an image
# ------------------------------------------------
print("\nPROGRAM 5 :")

# Load and display an image ('a.png') using OpenCV
image = cv2.imread('a.png')
X = np.array(image)
img = cv2.cvtColor(X, cv2.COLOR_BGR2RGB)
plt.imshow(img)

if image is not None:
    cv2.imshow('Image', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("Error loading image")
```

```
PROGRAM 5 :
```



[23]:
```
# ------------------------------------------------
# PROGRAM 6: Convert image to grayscale and display
# ------------------------------------------------
print("\nPROGRAM 6 :")

# Convert the loaded image to grayscale and display
grey_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(grey_image, cmap = 'gray')

if grey_image is not None:
    cv2.imshow('Image', grey_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("Error loading image")
```

```
PROGRAM 6 :
```



[24]:
```
# ------------------------------------------------
# PROGRAM 7: Perform matrix multiplication using NumPy
# ------------------------------------------------
print("\nPROGRAM 7 :")

# Perform matrix multiplication on grayscale image using NumPy
start_time = time.time()
X_image = np.array(image)
X = np.mean(X_image, axis=2)
Y = np.transpose(X).astype(float)
Z = np.dot(X, Y).astype(float)

print("Shape of Z: ", Z.shape)
print(Z[:4, :4])

# Save the resulting image and display it
end_time = time.time()
print("Time taken using Numpy =", end_time - start_time, "seconds")
```

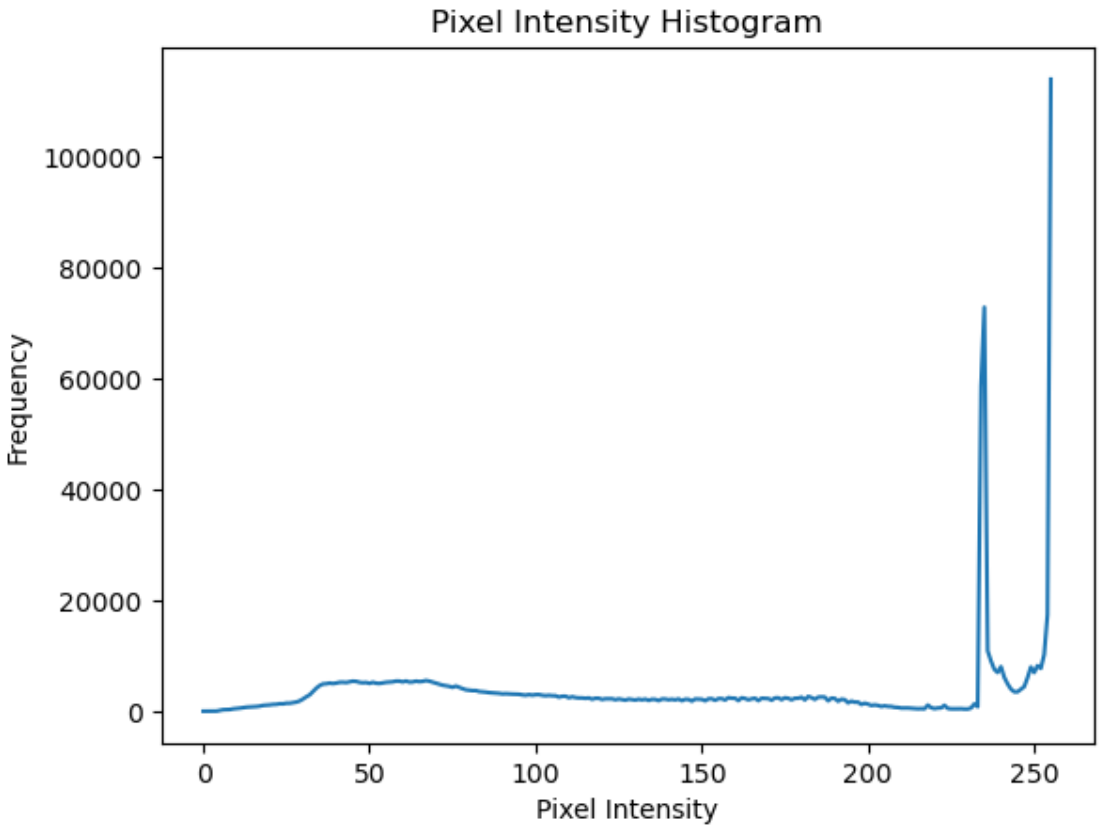
```
PROGRAM 7 :
Shape of Z:  (800, 800)
[[47670836.55555557 42558175.44444444 42656161.66666666 42662520.44444444]
 [42558175.44444444 39469418.8888889  39526680.88888889 39522797.33333334]
```

```
 [42656161.66666666 39526680.88888889 39593172.77777779 39593680.66666666]
 [42662520.44444444 39522797.33333334 39593680.66666666 39602154.8888889 ]]
Time taken using Numpy = 0.08229970932006836 seconds
```

[ ]:
```python
# ---------------------------------------------------
# PROGRAM 8: Perform matrix multiplication without NumPy
# ---------------------------------------------------
print("\nPROGRAM 8 :")

# Define a function to perform matrix multiplication without NumPy
def matrix_multiply(A, B):
    Z = [[0 for _ in range(len(B[0]))] for _ in range(len(A))]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                Z[i][j] += A[i][k] * B[k][j]
    return Z

# Perform matrix multiplication without NumPy and measure time taken
image = cv2.imread('a.png')
X = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
Y = np.transpose(X)

X = X.astype(float)
Y = Y.astype(float)

start = time.time()
Z = matrix_multiply(X, Y)
end = time.time()

print("Time taken without NumPy:", end - start)
print(Z)
```

[26]:
```python
# ---------------------------------------------------
# PROGRAM 9: Plot pixel intensity histogram
# ---------------------------------------------------
print("\nPROGRAM 9 :")

# Compute and plot the histogram of pixel intensities
histogram = cv2.calcHist([grey_image], [0], None, [256], [0, 256])
plt.plot(histogram)
plt.title("Pixel Intensity Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()
```

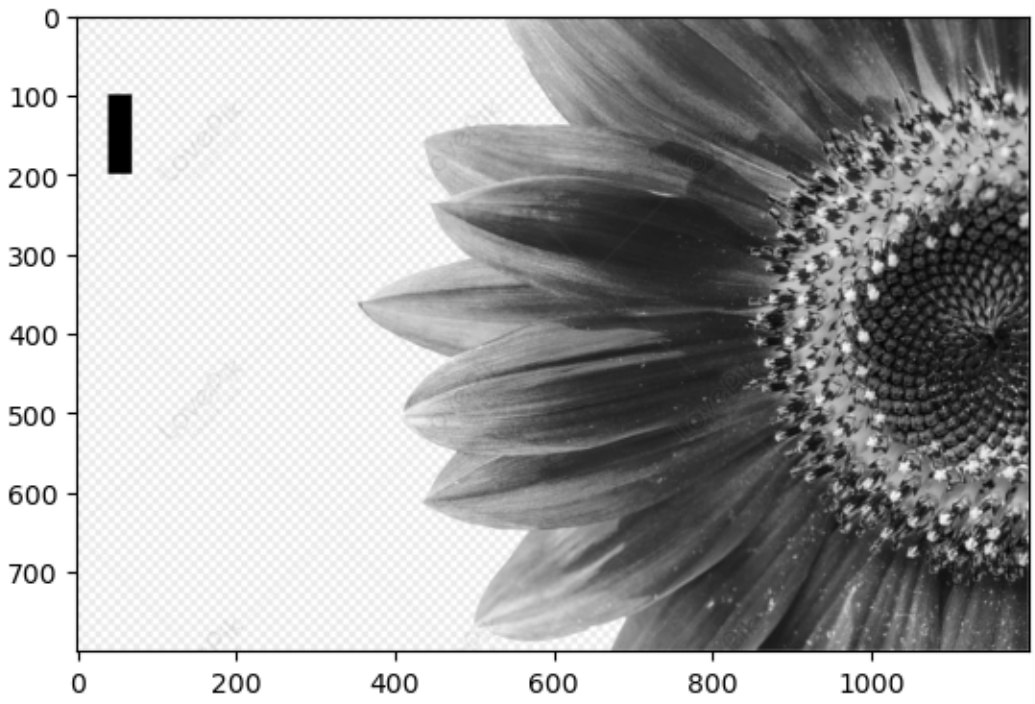
```
PROGRAM 9 :
```



[27]:
```
# ------------------------------------------------
# PROGRAM 10: Draw a black rectangle on the image
# ------------------------------------------------
print("\nPROGRAM 10 :")
grey_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Define coordinates for the rectangle
top_right = (40, 100)
bottom_left = (70, 200)

# Draw a black rectangle on the grayscale image
X_rect = grey_image.copy()
X_rect[100:200, 40:70] = 0

cv2.imshow('Black Rectangled Image', X_rect)
cv2.waitKey(0)
cv2.destroyAllWindows()

plt.imshow(X_rect, cmap = 'gray')
```

PROGRAM 10 :

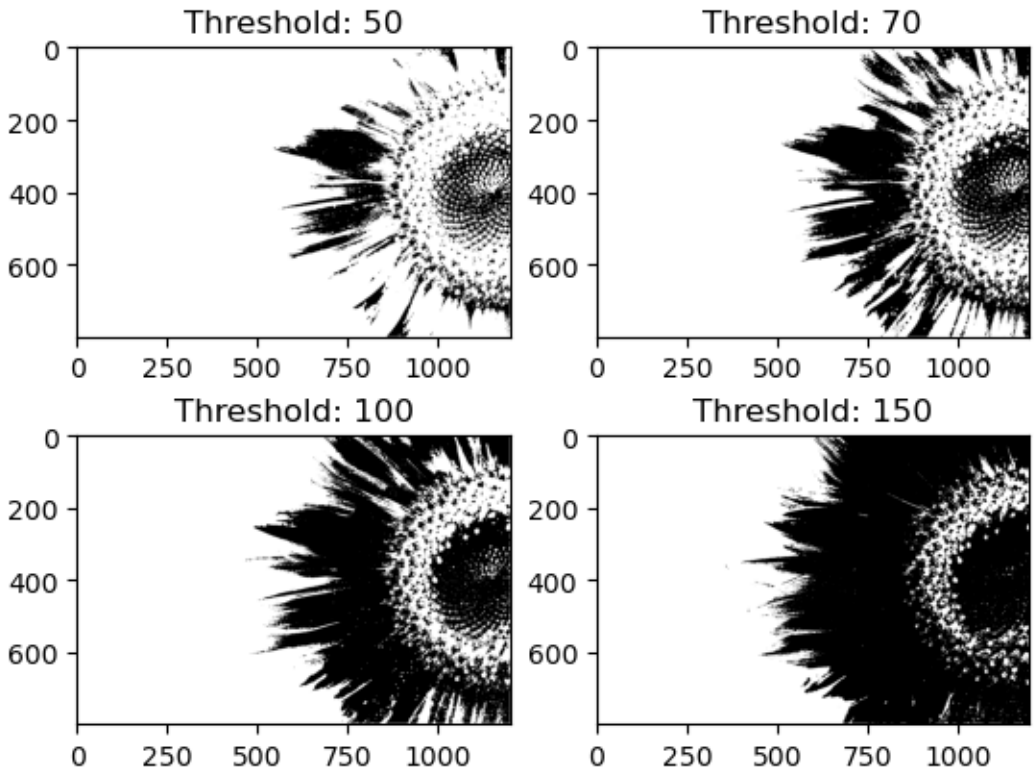
[27]: <matplotlib.image.AxesImage at 0x16f0f65d290>



[28]:
```
# we define the binarize function
def binarize(X, threshold):
    binarized_image = np.where(X > threshold, 255, 0).astype(np.uint8)
    return binarized_image

print("\nPROGRAM 11 :")

# Define threshold values for binarization
threshold_values = [50, 70, 100, 150]

TH50 = binarize(X, 50)
TH70 = binarize(X, 70)
TH100 = binarize(X, 100)
TH150 = binarize(X, 150)

fig, axs = plt.subplots(2, 2)

axs[0, 0].imshow(TH50, cmap='gray')
axs[0, 1].imshow(TH70, cmap='gray')
axs[1, 0].imshow(TH100, cmap='gray')
```

```
axs[1, 1].imshow(TH150, cmap='gray')

axs[0, 0].set_title('Threshold: 50')
axs[0, 1].set_title('Threshold: 70')
axs[1, 0].set_title('Threshold: 100')
axs[1, 1].set_title('Threshold: 150')
plt.show()
```

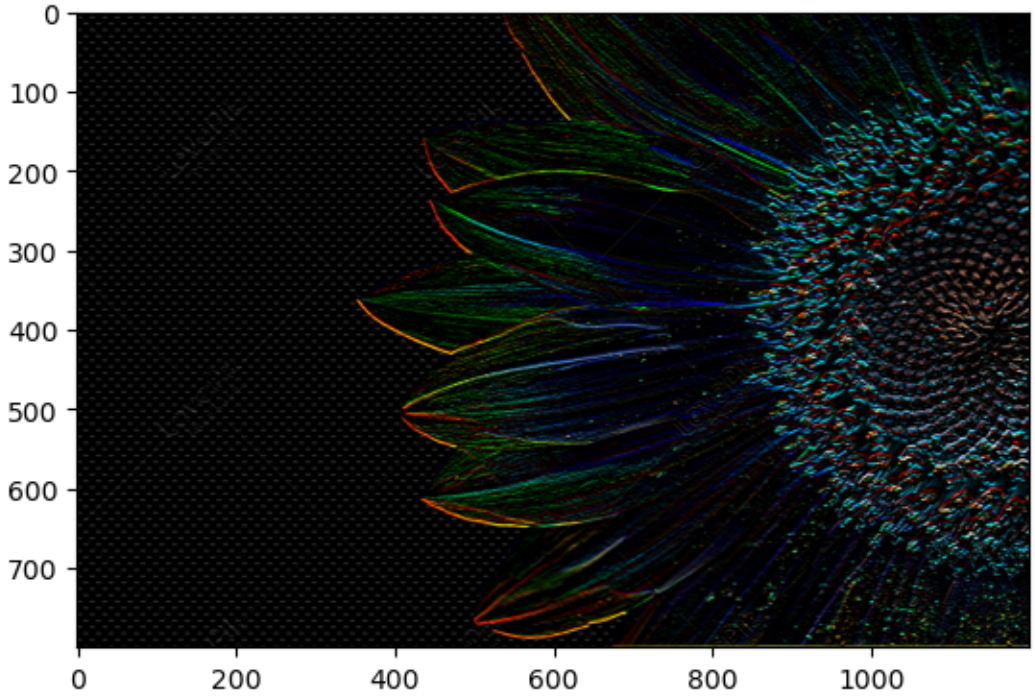
PROGRAM 11 :



[29]:
```
print("\nPROGRAM 12 :")

# Define a filter for convolution
filter_list = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])

# Convolute the original image with the filter and display the result
filtered_image = cv2.filter2D(image, -1, filter_list)
cv2.imshow('Filtered convoluted  image', filtered_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
plt.imshow(filtered_image)
```

PROGRAM 12 :

[29]: <matplotlib.image.AxesImage at 0x16f0ed6df10>



[ ]: