NumPy and its Applications

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

```
import numpy as np

# Load CSV File (skipping header and using 2nd column) into a numpy array
arr = np.loadtxt("book1.csv", skiprows=1, delimiter="\t", usecols=1, dtype=int)

# Maximum and Minimum of elements in the numpy array
print(f"Max: {arr.max()} | Min: {arr.min()}")
# Alt: np.max(arr), np.min(arr)

print(arr)
```

```
> python3 1.py
Max: 100 | Min: 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
```

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

```
import numpy as np

# Load CSV File (skipping header and using 2nd column) into a numpy array
arr = np.loadtxt("book1.csv", skiprows=1, delimiter="\t", usecols=1, dtype=int)

# Sort the numpy array
arr.sort()  # Changes the array
print(arr)
# Alt: print(np.sort(arr)) Doesn't change the array
```

```
> python3 2.py
[  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   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   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  24  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  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
```

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

```
import numpy as np

# Load CSV File (skipping header and using 2nd column) into a numpy array
arr = np.loadtxt("book1.csv", skiprows=1, delimiter="\t", usecols=1, dtype=int)

# Sort the numpy array
arr.sort()
# Reverse the numpy array
arr = np.flip(arr)
print(arr)
```

```
> python3 3.py
[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  94  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  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
```

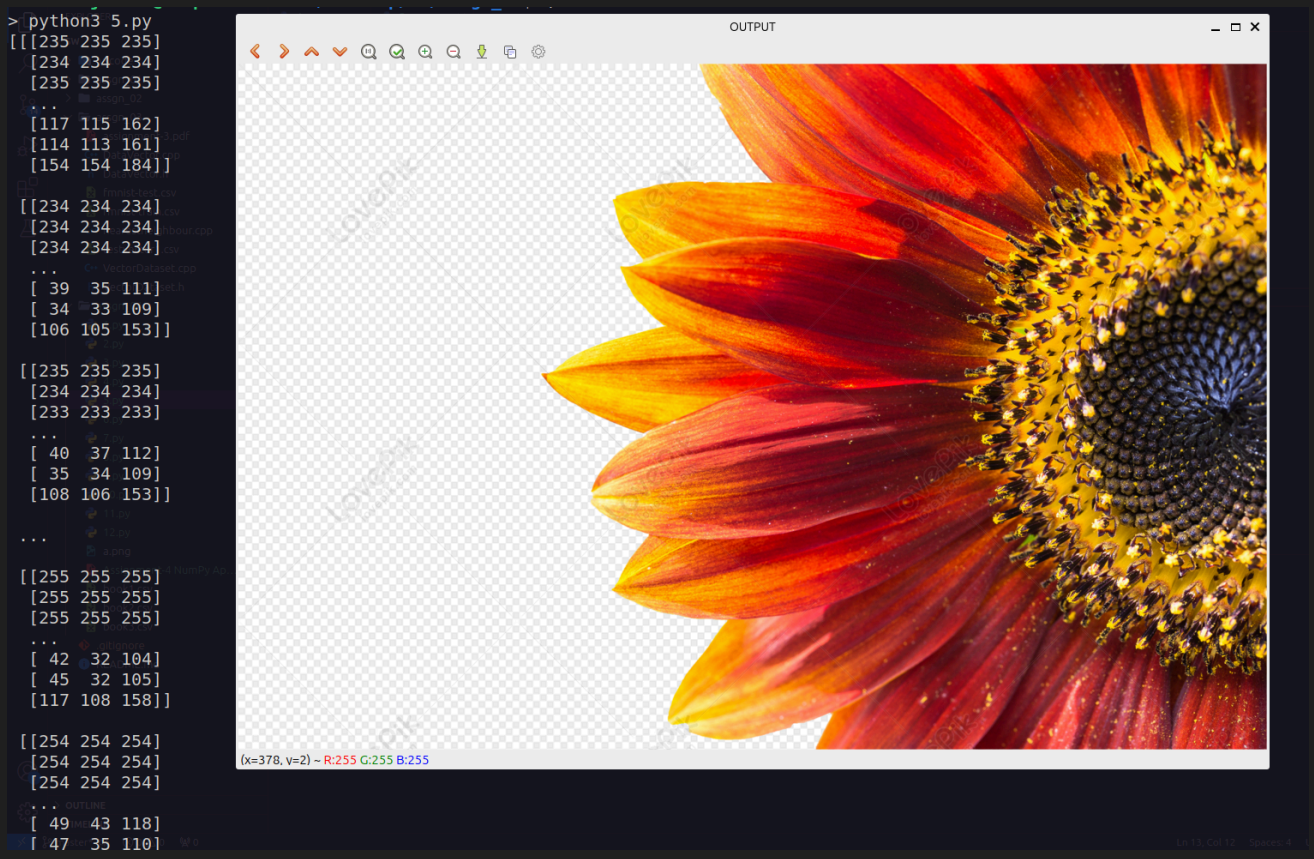
4. Write a program to load three .csv files (`Book1.csv`, `Book2.csv`, and `Book3.csv`) as a list of Numpy 1-D arrays. Print the means of all arrays as a list.

```
import numpy as np

# Load 3 CSV files into separate numpy 1D arrays, calculate
their respective means and store using list comprehension
means = [
    np.loadtxt(
        f"book{i}.csv", skiprows=1, delimiter="\t",
usecols=1, dtype=float
    ).mean()
    for i in range(1, 4)
]
print(means)
```

```
> python3 4.py
[48.566, 51.08844672, 513.326]
```

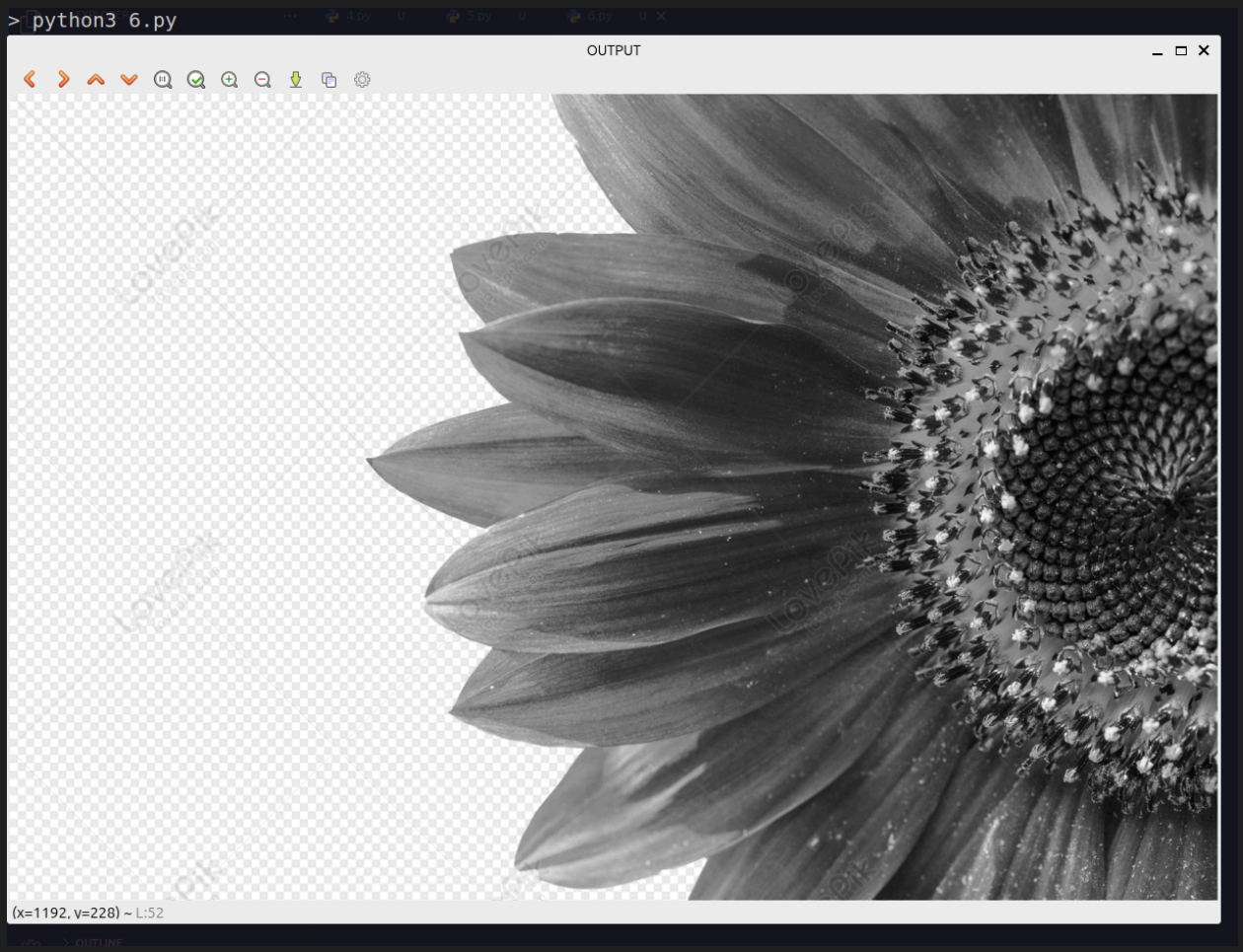
5. Write a program to read an image, store the image in a NumPy 3-D array. For the image, consider a `.PNG`. Display the image. Let the image stored in the NumPy array be X.

```
import numpy as np
import cv2 as cv

# Read color image using OpenCV
img = cv.imread("a.png", cv.IMREAD_COLOR)

# Store the image in numpy 3D array
X = np.asarray(img)
print(X)

# Display Image
cv.imshow("OUTPUT", img)
cv.waitKey(2000)  # Wait 2 seconds
```



6. Write a program to convert a color image (say a .PNG) into a grayscale image. Let the grayscale image stored in the Numpy 2-D array be X. Display the grayscale image on the screen.

```
import numpy as np
import cv2 as cv

# Read color image using OpenCV
img = cv.imread("a.png", cv.IMREAD_COLOR)

# Convert color image to grayscale image by calculating
mean of (R,G,B) values
X = np.asarray([[col.mean() for col in row] for row in
img]).astype("uint8")
print(X)

# Display Image
cv.imshow("OUTPUT", X)
cv.waitKey(2000)  # Wait 2 seconds
```



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

```
import numpy as np
import cv2 as cv
import time

# Read grayscale image using OpenCV
img = cv.imread("a.png", cv.IMREAD_GRAYSCALE)

# Store the image in numpy 2D array
X = np.asarray(img).astype("int64")

# Transpose of matrix X
Y = np.transpose(X)

# Matrix Multiplication using '@' operator of numpy library
s = time.time()
Z = X @ Y
print("Using numpy @:", time.time() - s, "sec")

# Matrix Multiplication using matmul() function of numpy
library
s = time.time()
Z = np.matmul(X, Y)
print("Using numpy matmul:", time.time() - s, "sec")
```

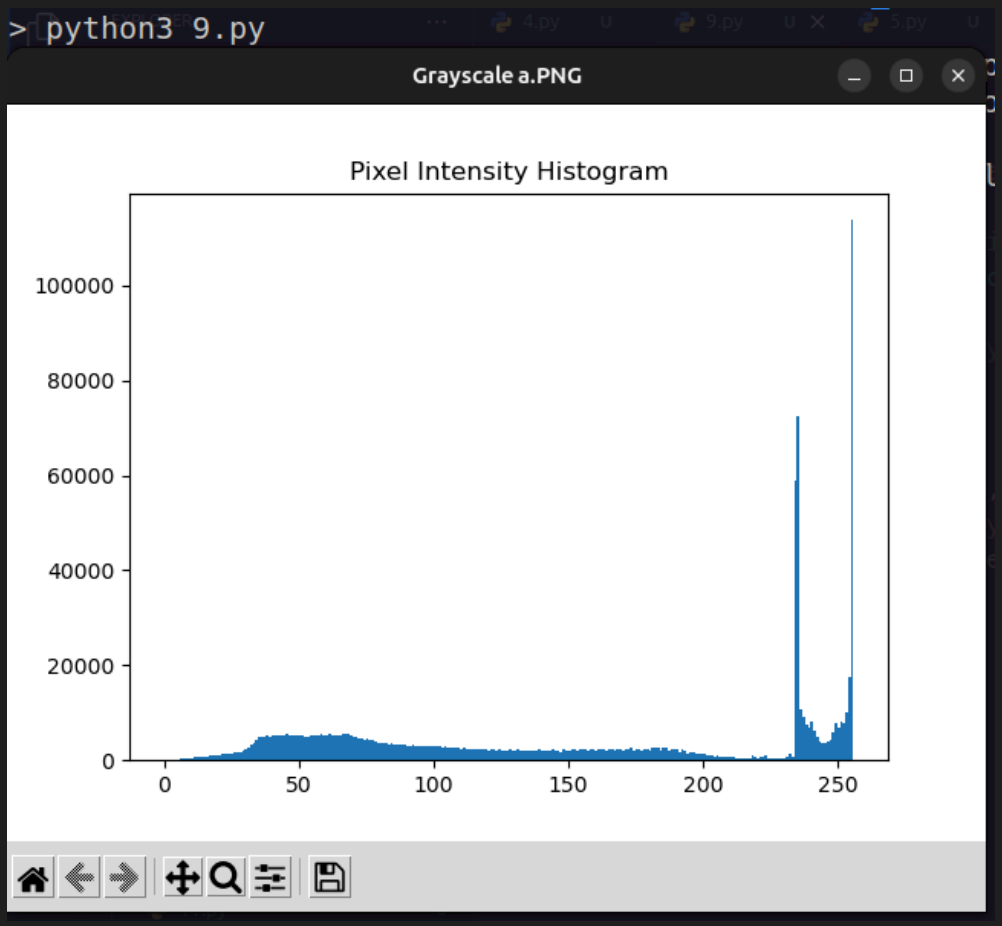
8. For the problem in Q.7, write your program without using the NumPy library. Compare the computation times doing the same with NumPy and basic programming in Python.

```
import numpy as np
import cv2 as cv
import time

# Read grayscale image using OpenCV
img = cv.imread("a.png", cv.IMREAD_GRAYSCALE)

# Store the image in numpy 2D array
X = np.asarray(img).astype("int64")

# Transpose of matrix X
Y = np.transpose(X)

# Convert numpy arrays to python lists
A = X.tolist()
B = Y.tolist()

# Matrix Multiplication without using numpy library
s = time.time()
C = [[sum(a * b for a, b in zip(A_row, B_col)) for B_col in
zip(*B)] for A_row in A]
print("Without using numpy:", time.time() - s, "sec")
```

```
harshit-jain52@inspiron-3593:~/Desktop/SW/assgn_04$ \
> python3 7.py
Using numpy @: 0.554659366607666 sec
Using numpy matmul: 0.5573883056640625 sec
harshit-jain52@inspiron-3593:~/Desktop/SW/assgn_04$ \
> python3 8.py
Without using numpy: 35.689759492874146 sec
```

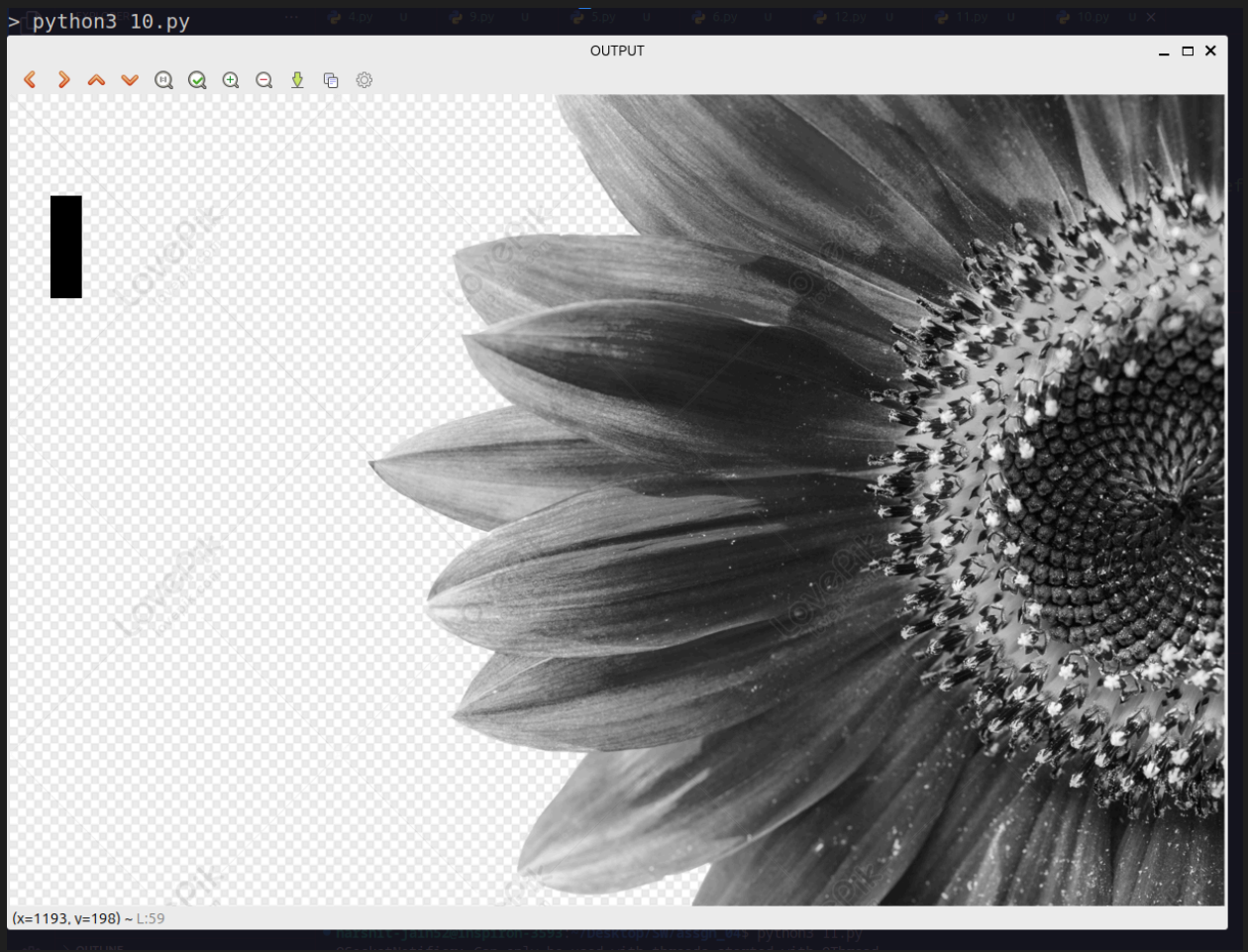
9. Plot the pixel intensity histogram of the greyscale image stored in X.

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

# Read grayscale image using OpenCV
img = cv.imread("a.png", cv.IMREAD_GRAYSCALE)

# Store the image in numpy 2D array
X = np.asarray(img)

# Flatten the numpy array and plot the pixel intensity
histogram
plt.hist(np.ravel(X), 256, [0, 256])
plt.title("Pixel Intensity Histogram")
plt.get_current_fig_manager().canvas.set_window_title("Gray
scale a.PNG")
plt.show()
```



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

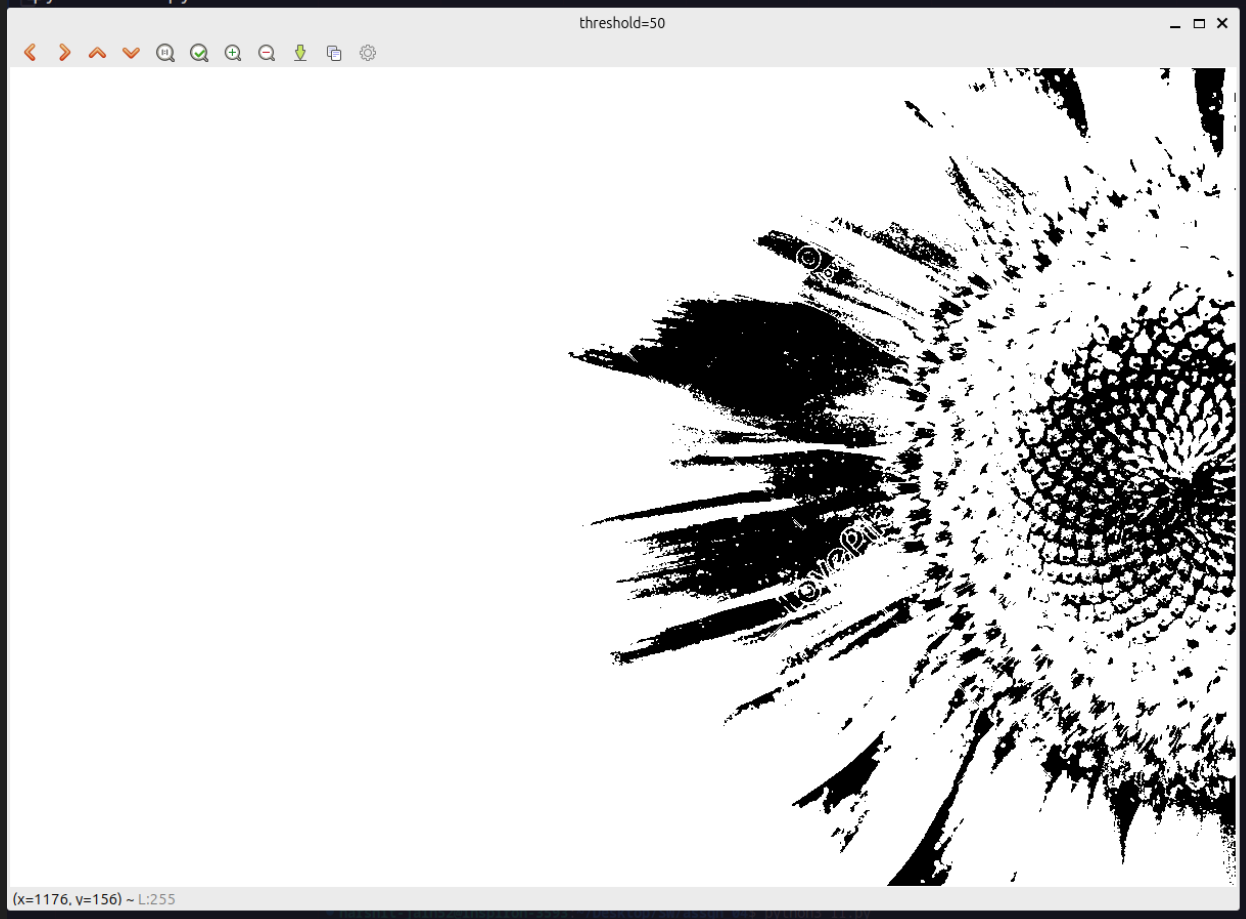
```
import cv2 as cv

# Read grayscale image using OpenCV
img = cv.imread("a.png", cv.IMREAD_GRAYSCALE)

# Draw a black rectangle at the position [(40,100) top
right, (70, 200) bottom left] on the image
cv.rectangle(img, (40, 100), (70, 200), color=(0, 0, 0),
thickness=cv.FILLED)

# Display Image
cv.imshow("OUTPUT", img)
cv.waitKey(2000)  # Wait 2 seconds
```



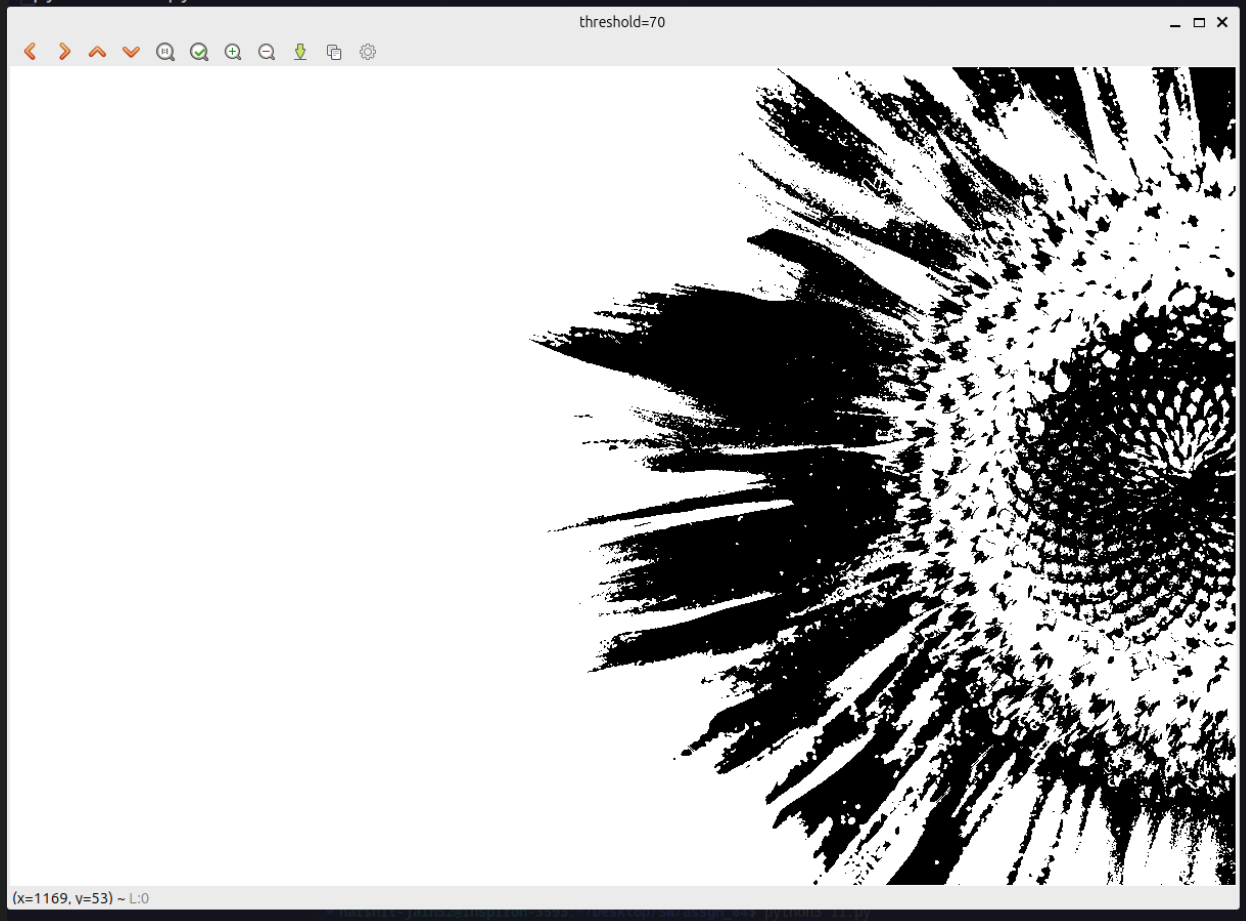
11. Using the grayscale image stored in X, transform it into the binarized image with thresholds: [50, 70, 100, 150]. Let the binarized images are stored in Z50, Z70, Z100, and Z150, respectively.

```
import numpy as np
import cv2 as cv

# Read grayscale image using OpenCV
img = cv.imread("a.png", cv.IMREAD_GRAYSCALE)

# Store the image in numpy 2D array
X = np.asarray(img)

# Two ways to Binarize the grayscale image with different
thresholds (*255 to get black-white distinction)
Z50 = np.array(X > 50).astype("uint8") * 255
Z70 = np.array(X > 70).astype("uint8") * 255
_, Z100 = cv.threshold(X, 100, 255, cv.THRESH_BINARY)
_, Z150 = cv.threshold(X, 150, 255, cv.THRESH_BINARY)

# Display binarized images
for z, thresh in zip([Z50, Z70, Z100, Z150], [50, 70, 100,
150]):
   cv.imshow(f"threshold={thresh}", z)
   cv.waitKey()  # Wait for key press
   cv.destroyAllWindows()
```

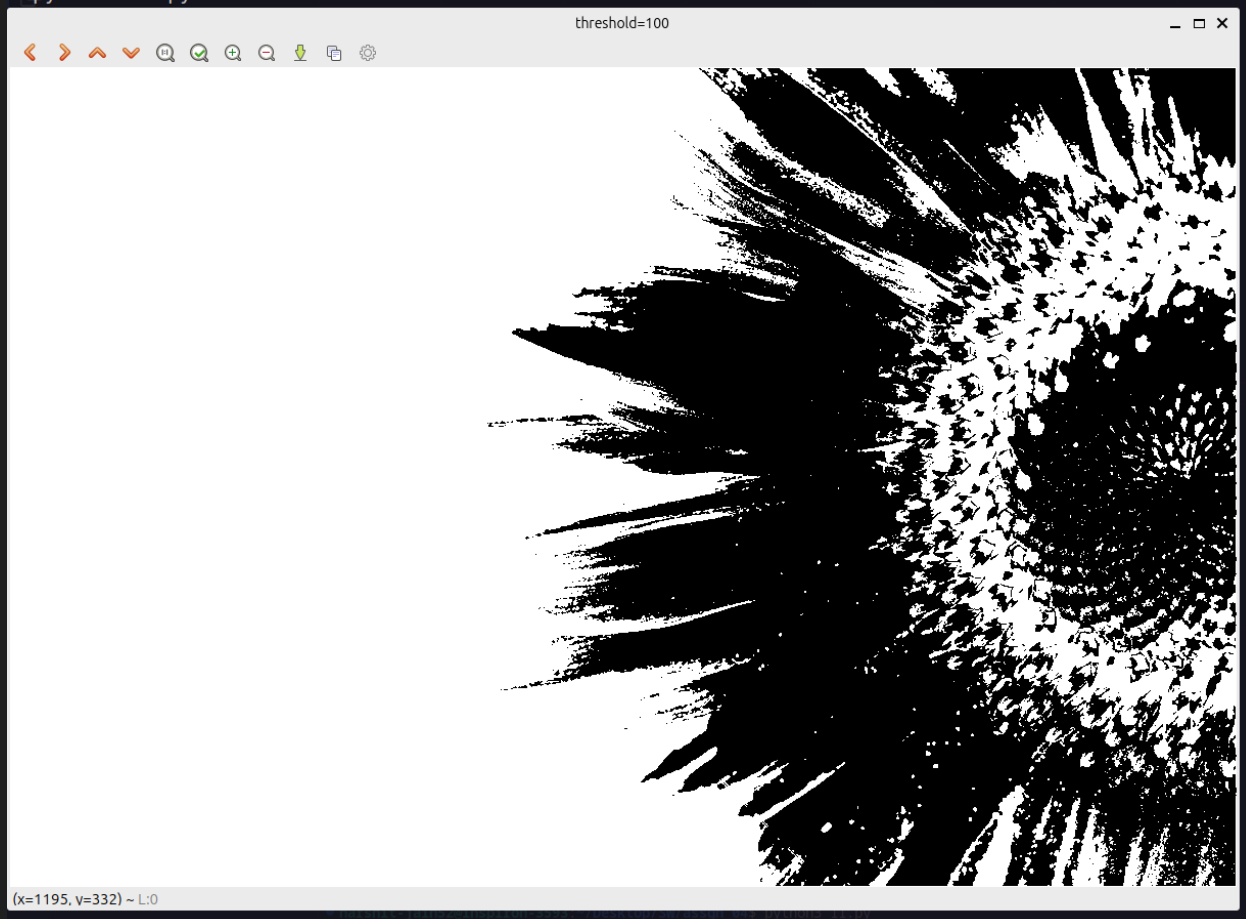
```
> python3 11.py
```



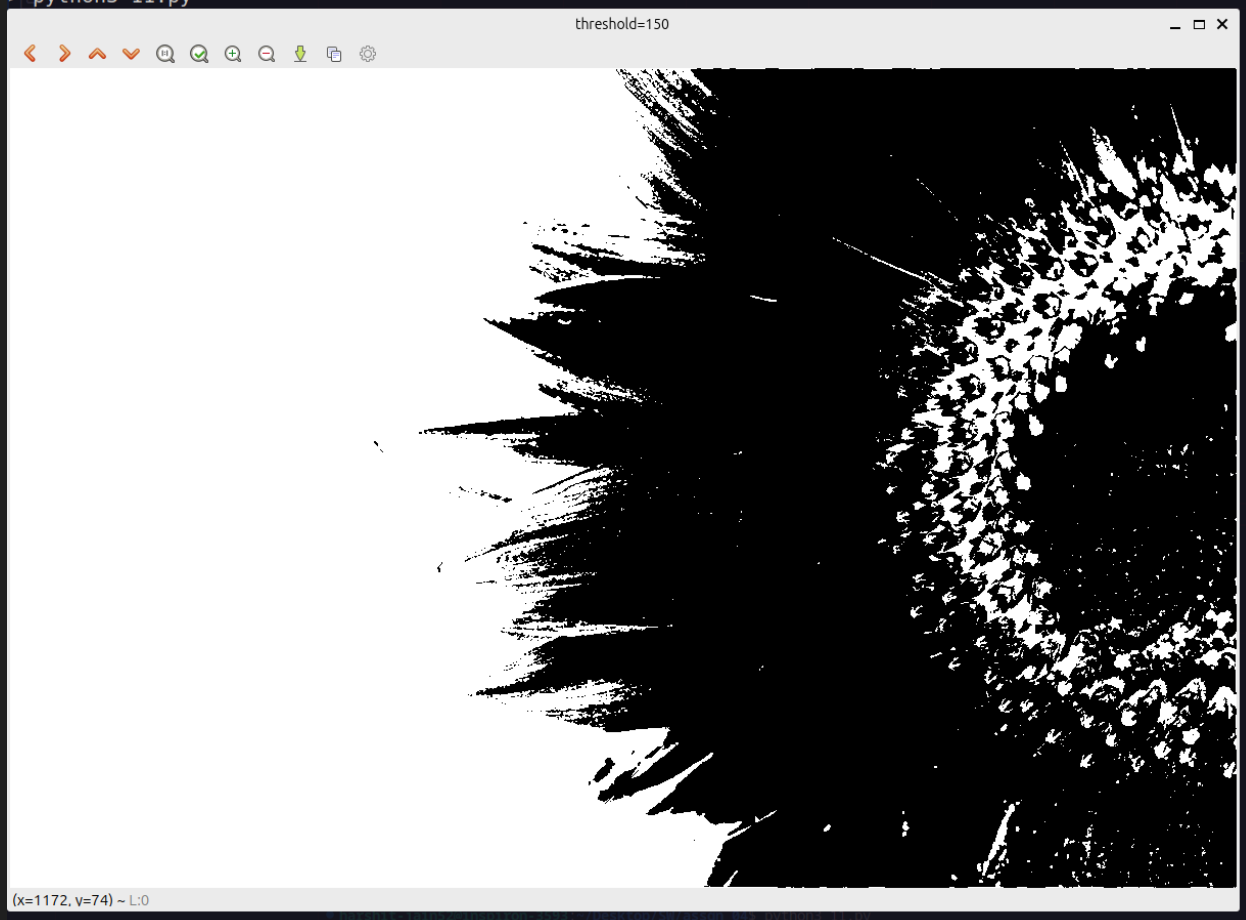

```
> python3 11.py
```



```
> python3 11.py
```




```
> python3 11.py
```



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 numpy as np
import cv2 as cv

# Read color image using OpenCV
img = cv.imread("a.png", cv.IMREAD_COLOR)

# Store the image in numpy 3D array
X = np.asarray(img)

# Store the given filter as numpy 2D array
filter = np.array(
    [[-1, -1, -1], [0, 0, 0], [1, 1, 1]],
)

# Apply filter on the image using convolution
flt_img = cv.filter2D(src=img, ddepth=-1, kernel=filter)

# Display image
cv.imshow("Convolved with filter", flt_img)
cv.waitKey(2000)  # Wait 2 seconds
```

```
> python3 12.py
```