ManishaTamang- 2358425

```
[ ] from PIL import Image

[ ] image_colored = Image.open("/content/drive/MyDrive/Artificial Intelligence and Machine Learning/lenna_image.png")

    print("Format:", image_colored.format)
    print("Mode:", image_colored.mode)
    print("Size:", image_colored.size)

    Format: PNG
    Mode: RGBA
    Size: (366, 357)

[ ] image_colored = image_colored.convert("RGB")
    print(image_colored.mode)

    RGB

[ ] image_grayed = image_colored.convert("L")
    display(image_grayed)
```

```
image_grayed = image_colored.convert("L")
display(image_grayed)
```

```
print("Format:", image_grayed.format)
print("Mode:", image_grayed.mode)
print("Size:", image_grayed.size)
```

```
Format: None
Mode: L
Size: (366, 357)
```

```
width,height = image_grayed.size
channels = len(image_grayed.getbands())

print(f"Image shape(gray):({height},{width},{channels})")

image_size_grayed = width*height* 1
print(f"Image size (gray):{image_size_grayed}")
```

```
Image shape(gray):(357,366,1)
Image size (gray):130662
```

```
width,height = image_colored.size
channels = len(image_colored.getbands())

print(f"Image shape(RGB):({height},{width},{channels})")

image_size_colored = width*height* 3
print(f"Image size (RGB):{image_size_colored}")
```

```
Image shape(RGB):(357,366,3)
Image size (RGB):391986
```

```
import numpy as np
image_array_colored = np.array(image_colored)
image_array_colored.shape
```

```
(357, 366, 3)
```

```
image_array_grayed = np.array(image_grayed)
image_array_grayed.shape
```

```
(357, 366)
```

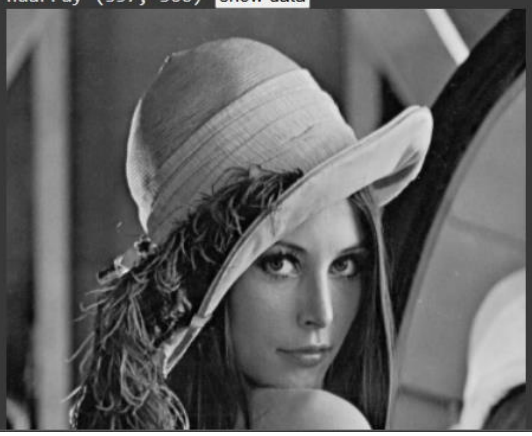
```
red_channels = image_array_colored[:,:,0]
print(red_channels)
display(red_channels)
```

```
[ ]
```

```
[[225 224 226 ...  92  90  91]
 [225 224 224 ...  91  90  91]
 [223 223 224 ...  93  91  91]
 ...
 [ 88  92  95 ... 148 162 175]
 [ 88  92  96 ... 154 168 177]
 [254 254 254 ... 255 255 255]]
ndarray (357, 366) show data
```

```
red_channels = image_array_colored[:,:,1]
print(red_channels)
display(red_channels)
```

```
[[127 130 130 ...  25  23  17]
 [130 128 130 ...  22  22  20]
 [128 133 129 ...  25  22  17]
 ...
 [ 23  22  26 ...  57  68  66]
 [ 26  27  26 ...  63  71  66]
 [254 254 254 ... 255 255 255]]
ndarray (357, 366) show data
```



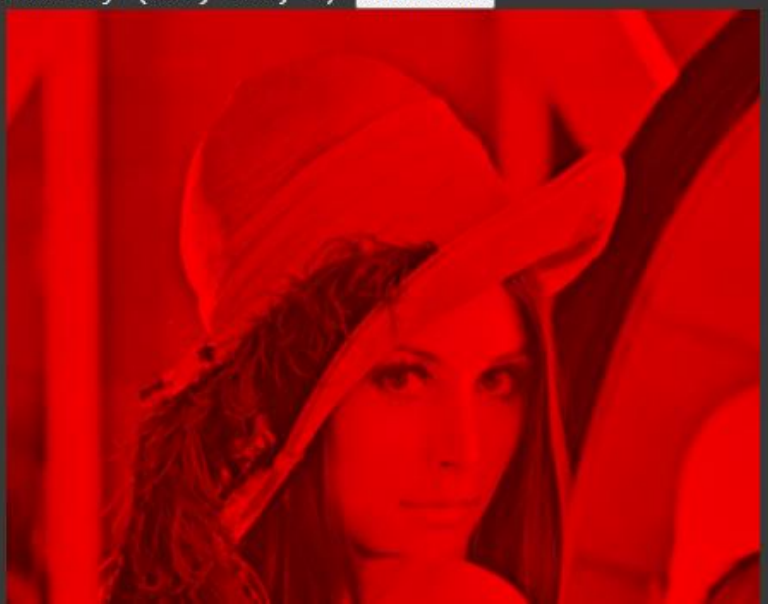
```
red_channels = image_array_colored[:,:,2]
print(red_channels)
display(red_channels)
```

```
[[110 106 105 ...  64  62  56]
 [110 104 114 ...  64  61  63]
 [105 114 111 ...  67  65  61]
 ...
 [ 58  58  60 ...  78  84  78]
 [ 58  61  61 ...  83  85  77]
 [254 254 254 ... 255 255 255]]
ndarray (357, 366) show data
```

```
red_channels = image_array_colored.copy()
red_channels[:,:,1]=0
red_channels[:,:,2]=0
print(red_channels)
display(red_channels)
```

```
[[[225   0   0]
  [224   0   0]
  [226   0   0]
  ...
  [ 92   0   0]
  [ 90   0   0]
  [ 91   0   0]]

 [[225   0   0]
  [224   0   0]
  [224   0   0]
  ...
  [ 91   0   0]
  [ 90   0   0]
  [ 91   0   0]]

 [[223   0   0]
  [223   0   0]
  [224   0   0]
```

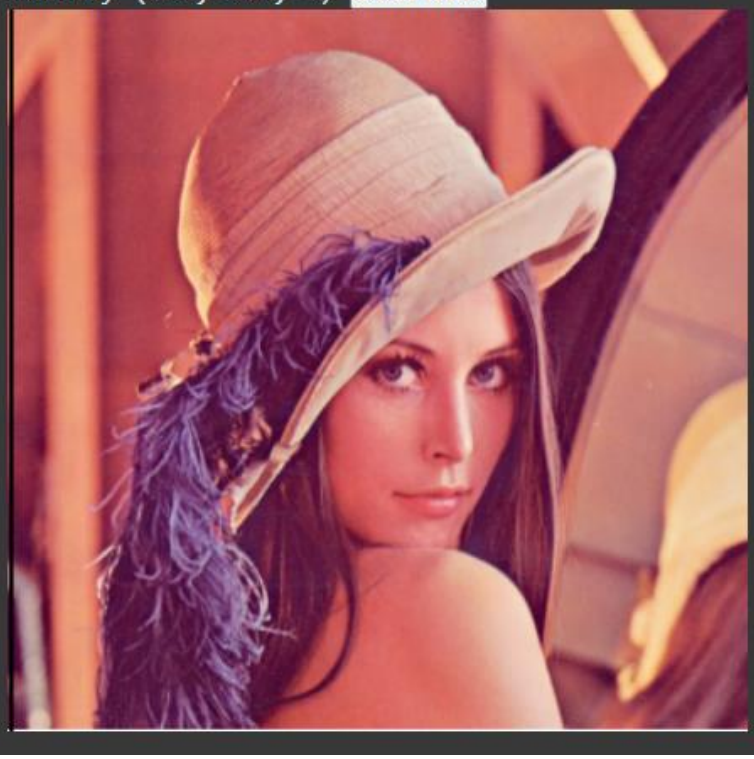
```
 [177   0   0]]

 [[254   0   0]
  [254   0   0]
  [254   0   0]
  ...
  [255   0   0]
  [255   0   0]
  [255   0   0]]]
ndarray (357, 366, 3) show data
```



```
red_channels = image_array_colored.copy()
red_channels[:,1,:]=0
red_channels[:,2,:]=0
print(red_channels)
display(red_channels)
```

```
  [ 92  25  64]
  [ 90  23  62]
  [ 91  17  56]]

 [[225 130 110]
  [  0   0   0]
  [  0   0   0]
  ...
  [ 91  22  64]
  [ 90  22  61]
  [ 91  20  63]]

 [[223 128 105]
  [  0   0   0]
  [  0   0   0]
  ...
  [ 93  25  67]
  [ 91  22  65]
  [ 91  17  61]]
```

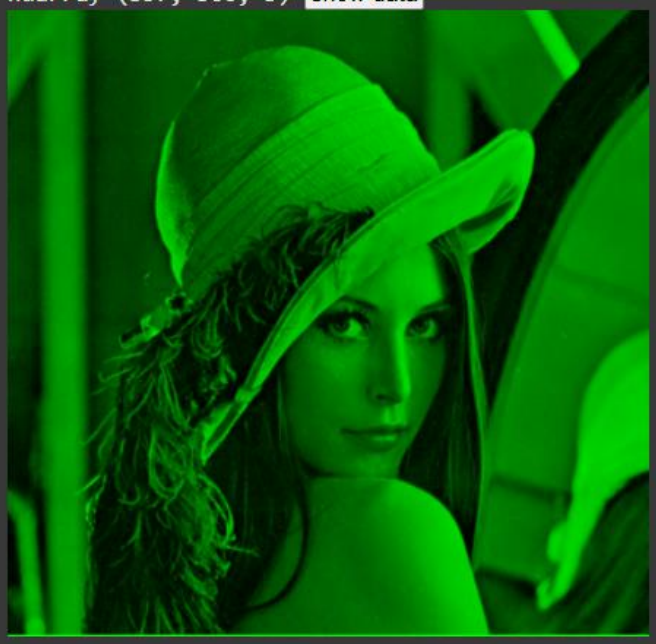
```
   ...
   [255 255 255]
   [255 255 255]
   [255 255 255]]]
ndarray (357, 366, 3) show data
```



```python
green_channels = image_array_colored.copy()
green_channels[:,:,0]=0
green_channels[:,:,2]=0
print(green_channels)
display(green_channels)
```

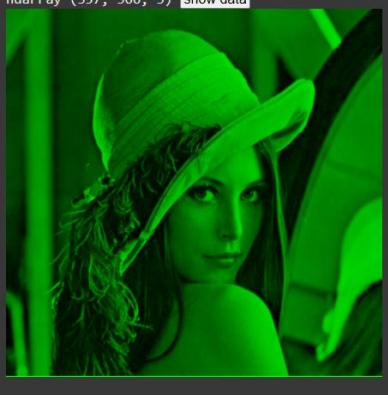
```
  ...
  [  0  22   0]
  [  0  22   0]
  [  0  20   0]]

 [[  0 128   0]
  [  0 133   0]
  [  0 129   0]
  ...
  [  0  25   0]
  [  0  22   0]
  [  0  17   0]]

 ...

 [[  0  23   0]
  [  0  22   0]
  [  0  26   0]
  ...
  [  0  57   0]
  [  0  68   0]
  [  0  66   0]]
```

ndarray (357, 366, 3) show data



```
blue_channels = image_array_colored.copy()
blue_channels[:,:,0]=0
blue_channels[:,:,2]=0
print(blue_channels)
display(blue_channels)
```

```
[[[  0 130   0]
  [  0 128   0]
  [  0 130   0]
  ...
  [  0  22   0]
  [  0  22   0]
  [  0  20   0]]

 [[  0 128   0]
  [  0 133   0]
  [  0 129   0]
  ...
  [  0  25   0]
  [  0  22   0]
  [  0  17   0]]

 ...

 [[  0  23   0]
  [  0  22   0]
  ...
  [  0 255   0]
  [  0 255   0]
  [  0 255   0]]]
ndarray (357, 366, 3) show data
```



```
image_colored = Image.open("/content/drive/MyDrive/Artificial Intelligence and Machine Learning/lenna_image.png").convert("RGB")
r,g,b = image_colored.split()
display(r)
display(g)
display(b)
```

```
row_100 = image_array_colored[100,:,:] #height,width,channels
print(row_100)
```

```
[[169  63  76]
 [166  65  75]
 [170  64  76]
 ...
 [210 132 122]
 [209 133 116]
 [208 134 126]]
```

```
col_100 = image_array_colored[:,50,:] #height,width,channels
print(col_100)
```

```
[[160  61  82]
 [163  63  82]
 [166  61  74]
 ...
 [ 82  19  67]
 [ 79  18  66]
 [254 254 254]]
```

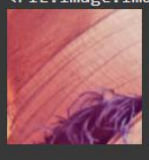
+ Code + Text

```
pixel = image_array_colored[10,50,:]
print(pixel)
```

```
[167  60  72]
```

```
left=100
right=200
upper=50
lower=150
cropped_img = image_colored.crop((left,upper,right,lower))
print(cropped_img)
display(cropped_img)
```

```
<PIL.Image.Image image mode=RGB size=100x100 at 0x7D6EAA411950>
```



```
img_array = np.zeros((100,100,3),dtype = np.uint8) #dtype = np.uint8 beacuse unsigned,integer,8bit
image_from_arr = Image.fromarray(img_array)
display(image_from_arr)
```

```
image_from_arr.save("Output_image.jpg")
```

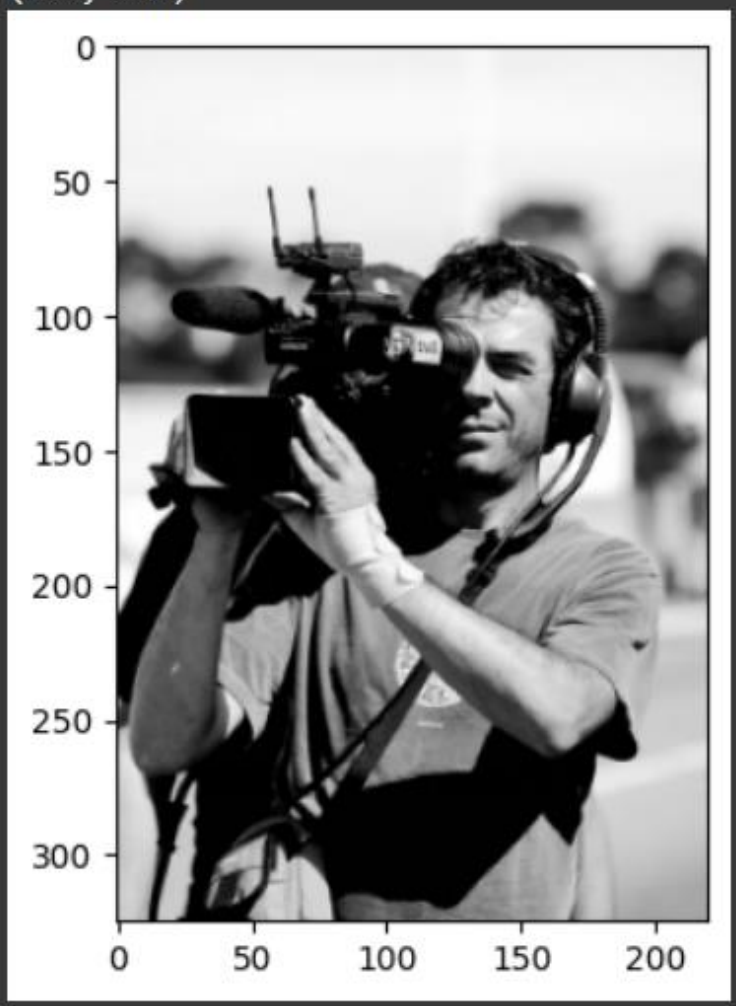
PCA

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

from PIL import Image

image = Image.open("/content/drive/MyDrive/Artificial Intelligence and Machine Learning/camera_man.jpg").convert("L")

image_array = np.array(image)
print(image_array.shape)


height,width = image_array.shape
```

```
(325, 220)
```

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

from PIL import Image

image = Image.open("/content/drive/MyDrive/Artificial Intelligence and Machine Learning/camera_man.jpg").convert("L")

image_array = np.array(image)
print(image_array.shape)


height,width = image_array.shape


data = image_array.copy()

plt.imshow(image_array,cmap='gray')
plt.show()
```

```
(325, 220)
```



```
# Compute the mean of each column ( feature )
mean = np.mean( data , axis =0)
# Subtract mean to center the data
centered_data = data - mean
centered_data = centered_data
```

```
[ ] centered_data.shape
```

```
(325, 220)
```

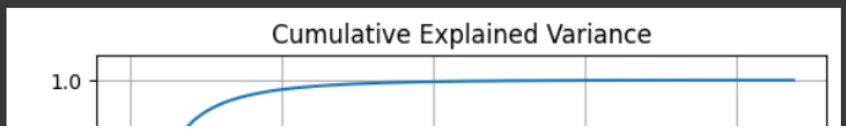
```
[ ] cov_matrix = np.cov(centered_data, rowvar=False)
```

```
[ ] cov_matrix.shape
```

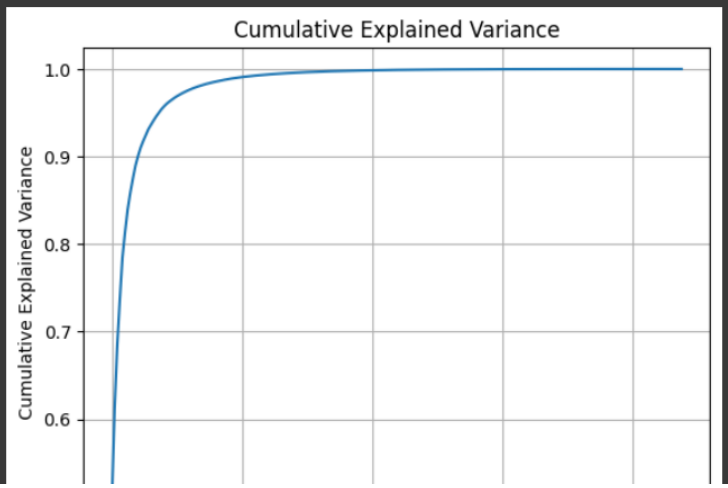
```
(220, 220)
```

```
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)

sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[sorted_indices]
eigenvectors = eigenvectors[:, sorted_indices]
```

```
[ ] explained_variance_ratio = eigenvalues / np .sum ( eigenvalues )
plt.plot ( np . cumsum ( explained_variance_ratio ) )
plt.title (" Cumulative Explained Variance ")
plt.xlabel (" Number of Components ")
plt.ylabel (" Cumulative Explained Variance ")
plt.grid ( True )
plt.show ()
```

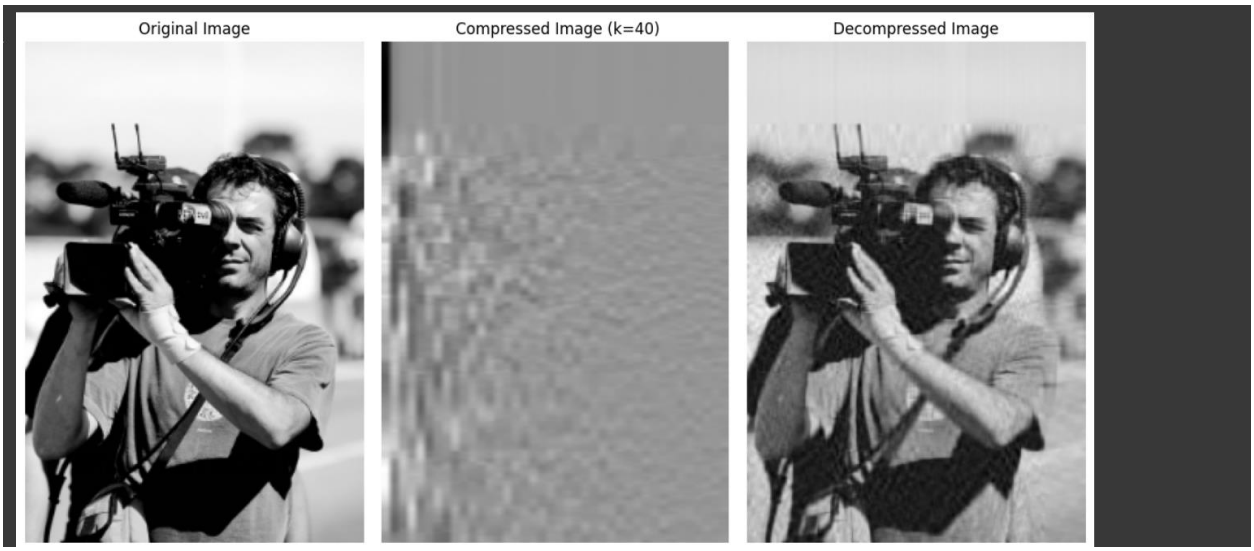


```
explained_variance_ratio = eigenvalues / np .sum ( eigenvalues )
plt.plot ( np . cumsum ( explained_variance_ratio ) )
plt.title (" Cumulative Explained Variance ")
plt.xlabel (" Number of Components ")
plt.ylabel (" Cumulative Explained Variance ")
plt.grid ( True )
plt.show ()
```




```
[ ] k = 40 # Choose k principal components
    components = eigenvectors [: , : k ]

[ ] compressed_data = np . dot ( centered_data , components )

[ ] decompressed_data = np . dot ( compressed_data , components . T ) + mean
```

```
plt . figure ( figsize =(12 , 6) )
# Original Image
plt.subplot (1 , 3 , 1)
plt.imshow ( image_array , cmap ="gray")
plt.title (" Original Image ")
plt.axis ("off")
# Compressed Representation
plt.subplot (1 , 3 , 2)
plt.imshow ( compressed_data , cmap ="gray", aspect ="auto")
plt.title ( f" Compressed Image (k={k})")
plt.axis ("off")
# Decompressed Image
plt.subplot (1 , 3 , 3)
plt.imshow ( decompressed_data , cmap ="gray")
plt.title (" Decompressed Image ")
plt.axis ("off")
plt.tight_layout()
plt.show()
```


Original Image
Compressed Image (k=40)
Decompressed Image