Image Data Analysis

- Image is a special kind of data where data is in the form of matrices.
- When we have image in the form numbers arranged in a matrix, we can do all matrix operations.

NumPy Matrix

```
In [1]: import numpy as np
```

```
In [2]: mat = np.matrix(
            [[1, 2, 3, 4, 5],
             [3, 4, 5, 6, 1]]
        )
```

```
In [3]: mat.shape

Out[3]: (2, 5)
```

Transpose Operation

```
In [4]: mat.T

Out[4]: matrix([[1, 3],
                [2, 4],
                [3, 5],
                [4, 6],
                [5, 1]])
```

Scalar Multiplication

```
In [5]: 3 * mat

Out[5]: matrix([[ 3,  6,  9, 12, 15],
                [ 9, 12, 15, 18,  3]])
```

Can we convert `matrix` into `image` ?

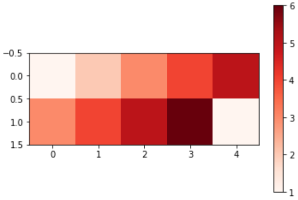
```
In [6]: from matplotlib import pyplot as plt
```

`imshow()` of `plt`

```
In [7]: mat

Out[7]: matrix([[1, 2, 3, 4, 5],
                [3, 4, 5, 6, 1]])
```

```
In [8]: image_mat = plt.imshow(mat, cmap='Reds')
        plt.colorbar(image_mat)
        plt.show()
```



Convert large `matrix` into `image`

In [9]:
```python
import random
```

- There should be 30 rows and 50 columns
- Each row of the matrix should have 50 numbers in the range of 1 and 200

In [10]:
```python
big_mat = [
    [random.randint(1, 200) for i in range(50)]
    for j in range(30)
]
```

In [11]:
```python
big_mat = np.matrix(big_mat)
```

In [12]:
```python
big_mat
```

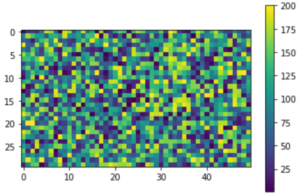
Out[12]:
```
matrix([[  4,  30,  36, ...,  56,  46, 113],
        [ 96, 172, 142, ...,  95,  32,  59],
        [ 71,  54,  14, ..., 148, 106, 165],
        ...,
        [128,  75, 110, ..., 151, 174,  27],
        [123, 193, 143, ...,  57, 112, 102],
        [ 21,  90, 148, ...,  34,  58, 163]])
```

In [13]:
```python
big_mat.shape
```

Out[13]:
```
(30, 50)
```

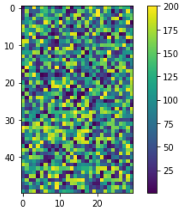
Matrix to Image

In [14]:
```python
image_mat = plt.imshow(big_mat, cmap=None)
plt.colorbar(image_mat)
plt.show()
```



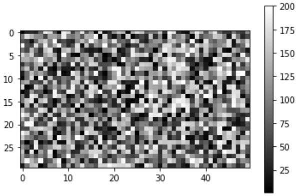
Transpose matrix to Image

In [15]:
```python
trans_mat = big_mat.T
timage_mat = plt.imshow(trans_mat, cmap=None)
plt.colorbar(timage_mat)
plt.show()
```



`grayscale` image

In [16]:
```python
gray_image = plt.imshow(big_mat, cmap='gray')
plt.colorbar(gray_image)
plt.show()
```



How can we convert `image` into `matrix`?

We should use `cv2` (`opencv-python`) package in python to compute matrix operations on images.

```
pip install opencv-python --user
```

A typical **colored image** is comprised of pixels (which are represented as RGB pixels).

- A pixel is simply a number in the range of `0` to `255` for all `R`, `G`, and `B`.
- R → Red → 0 to 255
- G → Green → 0 to 255
- B → Blue → 0 to 255

Image by Author

Some important colors and their RGB values -

- pixel → [R, G, B]
- Red → [255, 0, 0]
- Green → [0, 255, 0]
- Blue → [0, 0, 255]
- White → [255, 255, 255]
- Black → [0, 0, 0]
- Yellow → [255, 255, 0]
- **All colors** → https://www.colorhexa.com/color-names

Let's read the `image` and convert into `matrix`

The image that we will read is -

Read the `image` in the form of `matrix`

In [17]:

```
import cv2
```

In [18]:

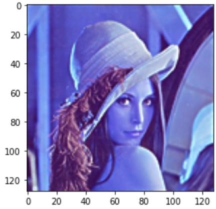
```
image_mat = cv2.imread('lena_image.png')
```

The image matrix would be like -

```
[[[159 183 255] ... [142 202 255]]
 [[140 169 255] ... [ 87  74 159]]
 [[118 164 255] ... [ 62  14  81]]
 ...
 [[ 64  30  96] ... [ 61  33 119]]
 [[ 61  27  92] ... [ 74  65 178]]
 [[ 60  25  89] ... [ 80  72 202]]]
```

In [19]:

```
plt.imshow(image_mat)
plt.show()
```



- By default, the image is read in BGR format.
- We need to convert it into RGB format for our convenience.

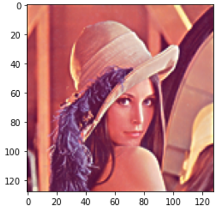
`BGR` → to → `RGB` format

In [20]:
```
image_mat = cv2.cvtColor(image_mat, cv2.COLOR_BGR2RGB)
```

The image matrix would be like -

```
[[[255 183 159] ... [255 202 142]]
 [[255 169 140] ... [159  74  87]]
 [[255 164 118] ... [ 81  14  62]]
 ...
 [[ 96  30  64] ... [119  33  61]]
 [[ 92  27  61] ... [178  65  74]]
 [[ 89  25  60] ... [202  72  80]]]
```

In [21]:
```
plt.imshow(image_mat)
plt.show()
```



Shape of the image matrix - `rows` and `columns`

In [22]:
```
image_mat.shape
```

Out[22]:
```
(128, 128, 3)
```

In [23]:
```
rows, cols, p = image_mat.shape
print(rows)
print(cols)
print(p)
```

```
128
128
3
```

How many `pixels` are there in the above `image` ?

In [24]:
```
pixels = rows * cols
```

In [25]:
```
pixels
```

Out[25]:
```
16384
```

How many `pixel values` are there including `R`, `G`, and `B` values?

In [26]:
```
pixel_values = rows * cols * p
```

In [27]:
```
pixel_values
```

Out[27]:
```
49152
```

Separate `R`, `G`, and `B` from the image

Image by Author

We make use of `cv2.split()` method to separate the `RGB` pixels from the image.

In [28]:
```
rimage_mat, gimage_mat, bimage_mat = cv2.split(image_mat)
```

In [29]:
```
print("R → \n\n", rimage_mat)
```

```
R → 

 [[255 255 255 ... 212 255 255]
 [255 255 247 ... 227 227 159]
 [255 246 232 ... 190 103  81]
 ...
 [ 96 100 100 ... 105 100 119]
 [ 92  97  95 ... 105 124 178]
 [ 89  96  91 ... 115 156 202]]
```

In [30]:
```
print("G → \n\n", gimage_mat)
```

```
G → 

 [[183 174 167 ...  99 203 202]
 [169 156 149 ... 117 126  74]
 [164 142 136 ...  82  25  14]
 ...
 [ 30  33  34 ...  32  26  33]
 [ 27  30  28 ...  33  43  65]
 [ 25  29  24 ...  42  64  72]]
```

In [31]:
```
print("B → \n\n", bimage_mat)
```

```
B → 

 [[159 148 142 ... 101 163 142]
 [140 130 124 ... 107 113  87]
 [118 115 113 ...  83  64  62]
 ...
 [ 64  66  71 ...  63  58  61]
 [ 61  63  67 ...  62  68  74]
 [ 60  62  64 ...  72  82  80]]
```

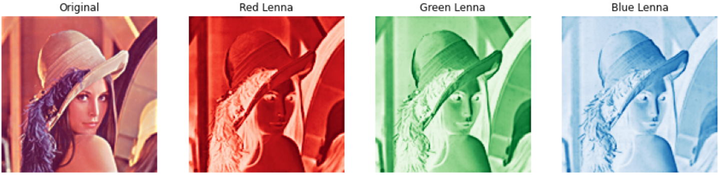
Plot `R`, `G`, and `B` separately

In [32]:
```
cmap_values = [None, 'Reds', 'Greens', 'Blues']
titles = ['Original', 'Red Lenna', 'Green Lenna', 'Blue Lenna']
image_matrices = [image_mat, rimage_mat, gimage_mat, bimage_mat]

fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(15, 10))

for i, ax in zip(range(4), axes):
    ax.axis("off")
    ax.set_title(titles[i])
    ax.imshow(image_matrices[i], cmap=cmap_values[i])

plt.show()
```


Original Red Lenna Green Lenna Blue Lenna

Some matrix operations

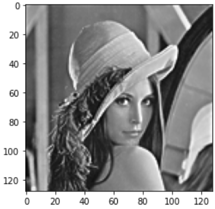
- Let's take grayscale matrix of original image

In [33]:
```
gray_image = cv2.imread('lena_image.png', 0)
```

In [34]:
```
gray_image.shape
```

Out[34]:
```
(128, 128)
```

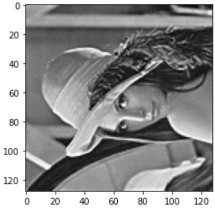
In [35]:
```
plt.imshow(gray_image, cmap='gray')
plt.show()
```



Tranpose gray lenna

In [36]:
```
trans_lenna = gray_image.T
```

In [37]:
```
plt.imshow(trans_lenna, cmap='gray')
plt.show()
```



In [38]:
```
image_mat.shape
```

Out[38]:
```
(128, 128, 3)
```

How can we transpose colored image?

Since each pixel is a combination of 3 values, we have to -

- separate `R`, `G`, and `B` matrices
- apply `transpose` operation to all the 3 matrices

- merge `R`, `G`, and `B` matrices as a one single matrix

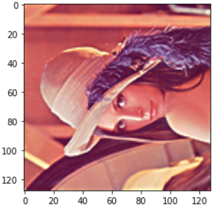
In [39]:

```
# separation of R, G, and B
rimage_mat, gimage_mat, bimage_mat = cv2.split(image_mat)

# transpose operation to all the 3 matrices
trans_r = rimage_mat.T
trans_g = gimage_mat.T
trans_b = bimage_mat.T

# merging R, G, and B matrices into one single matrix
trans_color_lenna = cv2.merge((trans_r, trans_g, trans_b))

# plotting the transposed colored image
plt.imshow(trans_color_lenna)
plt.show()
```



Other operations

- Image Flipping
- Image Mirroring
- Image Equalization (One of the mind blowing operations)
    - Used for enhancing the contrast of an image
- Image Binarization
- Image Inversion
- Image Cropping
- Image Bordering
- Image Convolution with kernels (One of the mind blowing operations)
    - Used for Smoothening, Blurring, Edge detection etc

PS: All you can find in my blog websites.

- Hashnode → https://msameeruddin.hashnode.dev/
- Medium → https://msameeruddin.medium.com/