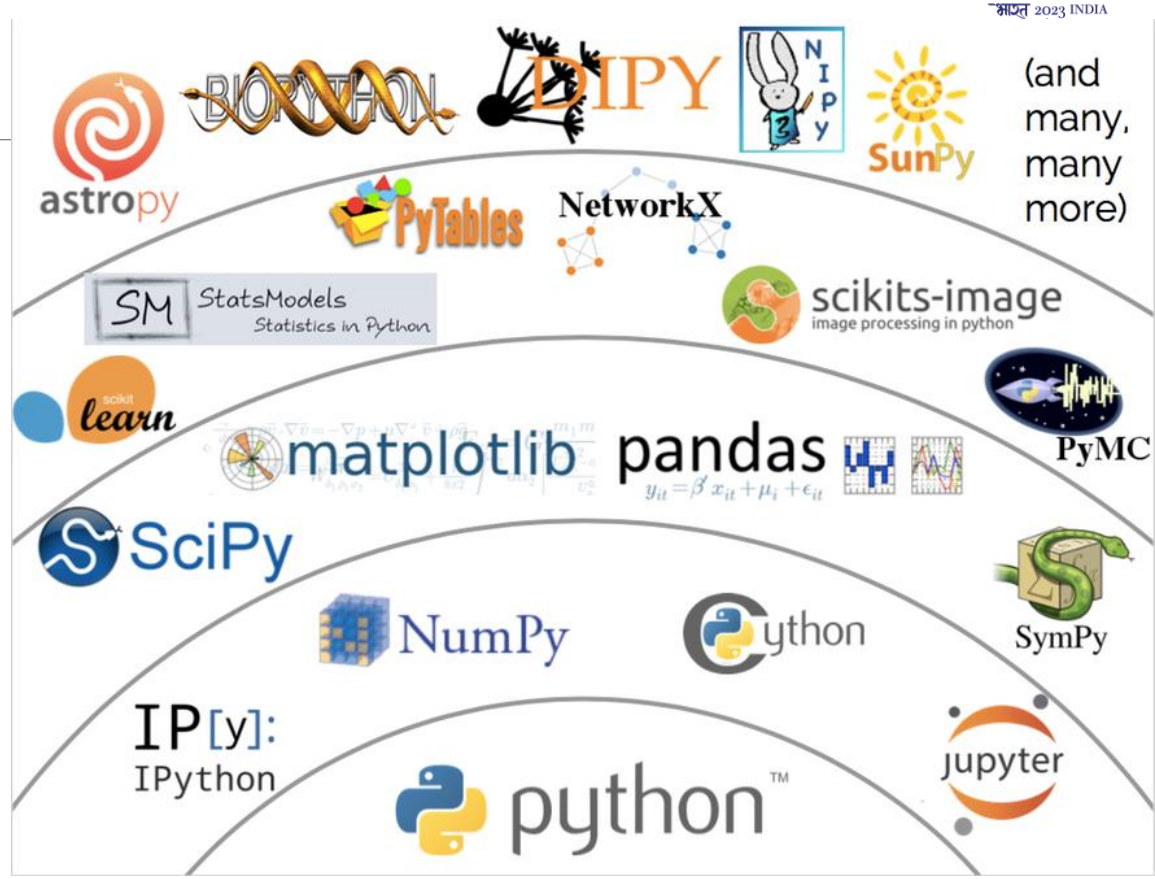
Machine Learning
for
Computer Vision

Dr. Sarwan Singh
Joint Director, NIELIT Chandigarh

Agenda

- Numpy
- Handling Images with Numpy



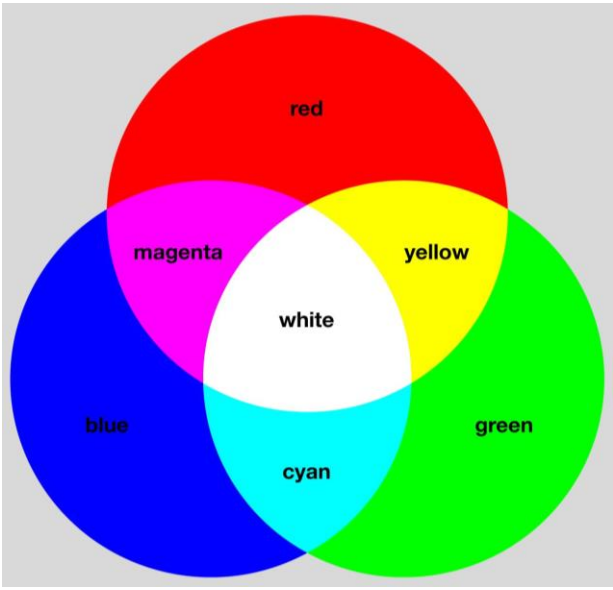
Websites : geeksforgeeks.com, java2blog.com, tutorialpoints.com, docs.python.org, analyticsvidhya.com

Pixel and colors

- Images are stored digitally as an **array of pixels**.
- A pixel (picture element) is the smallest element of an image.
- Every pixel has a color and the way colors are defined is called the **color space**.
- The most used color space is the **RGB color space**.
- In this model, every color is defined by three values, one for **red**, one for **green**, and one for **blue**.
- Usually, these values are 8-bit unsigned integers

  (a range of 0–255), this is called the **color depth**.
- The color **(0, 0, 0)** is **black**, **(0, 255, 0)** is **green**,

  (127, 0, 127) is **purple**.



Save Numpy array as image in Python

- Wide range of libraries are available in python that can be utilized to read and process images efficiently.
- In computer vision tasks, Images are also processed efficiently in the available libraries of python.
- Images are stored and proc3ssed in the form of Numpy array.
- Every image stores a pixel in a specific color code at a given position

Methods for saving Images

- PIL library to save numpy array as image
- matplotlib.image library to save numpy array as image
- opencv library to save numpy array as image
- imageio library to save numpy array as image
- scipy.misc library to save numpy array as image

PIL library to save Numpy array as image

- The Python Imaging Library(PIL) was one of the standard packages for image processing.
- First, we will define an array that represents an image.
- Then, we will use the fromarray() function to read this array as an image object.
- This object will be exported to an external file using the save() function.

Support for this package stopped in 2011, and after that, a fork of the same package as PIL was added as a standard library.

Code

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249]
```

1. import numpy as np
2. from PIL import Image
3. a = np.arange(0, 250, 1, np.uint8)
4. print(a)
5. img = Image.fromarray(a)
6. img.save("image.png")

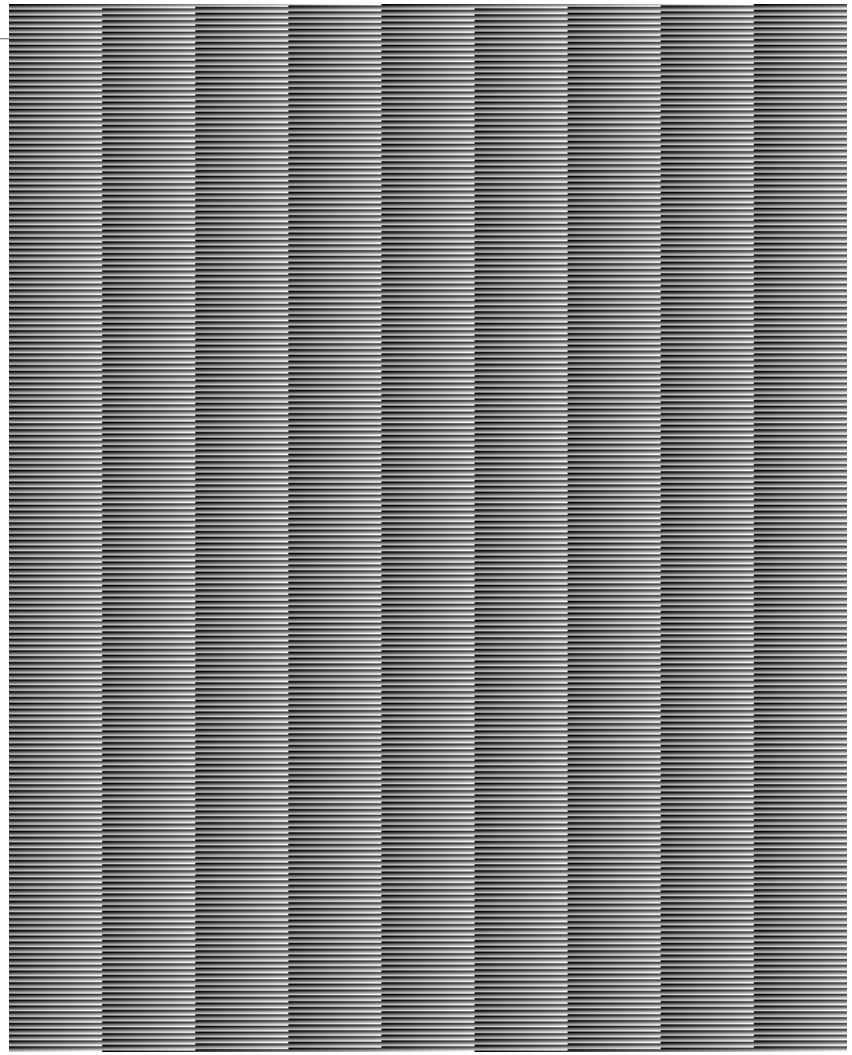
Code

1. import numpy as np
2. from PIL import Image
3. a = np.arange(0, 250, 1, np.uint8)
4. a = np.reshape(a, (25,10))
5. print(a)
6. img = Image.fromarray(a)
7. img.save("image1.png")

```
[[  0   1   2   3   4   5   6   7   8   9]
 [ 10  11  12  13  14  15  16  17  18  19]
 [ 20  21  22  23  24  25  26  27  28  29]
 [ 30  31  32  33  34  35  36  37  38  39]
 [ 40  41  42  43  44  45  46  47  48  49]
 [ 50  51  52  53  54  55  56  57  58  59]
 [ 60  61  62  63  64  65  66  67  68  69]
 [ 70  71  72  73  74  75  76  77  78  79]
 [ 80  81  82  83  84  85  86  87  88  89]
 [ 90  91  92  93  94  95  96  97  98  99]
 [100 101 102 103 104 105 106 107 108 109]
 [110 111 112 113 114 115 116 117 118 119]
 [120 121 122 123 124 125 126 127 128 129]
 [130 131 132 133 134 135 136 137 138 139]
 [140 141 142 143 144 145 146 147 148 149]
 [150 151 152 153 154 155 156 157 158 159]
 [160 161 162 163 164 165 166 167 168 169]
 [170 171 172 173 174 175 176 177 178 179]
 [180 181 182 183 184 185 186 187 188 189]
 [190 191 192 193 194 195 196 197 198 199]
 [200 201 202 203 204 205 206 207 208 209]
 [210 211 212 213 214 215 216 217 218 219]
 [220 221 222 223 224 225 226 227 228 229]
 [230 231 232 233 234 235 236 237 238 239]
 [240 241 242 243 244 245 246 247 248 249]]
```

Code …

```
1. import numpy as np
2. from PIL import Image
3. a = np.arange(0, 414720, 1, np.uint8)
4. a = np.reshape(a, (720, 576))
5. img = Image.fromarray(a)
6. img.save("image1.png")
```

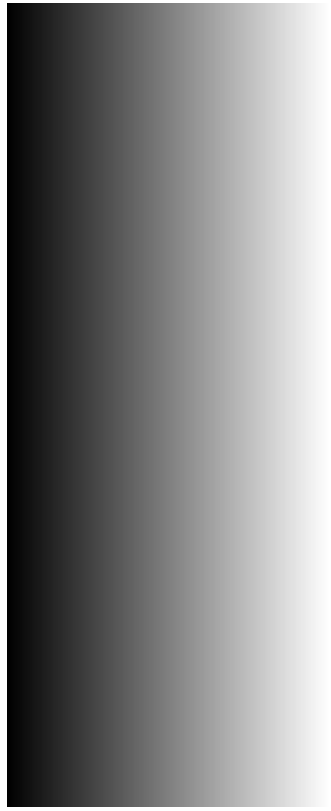


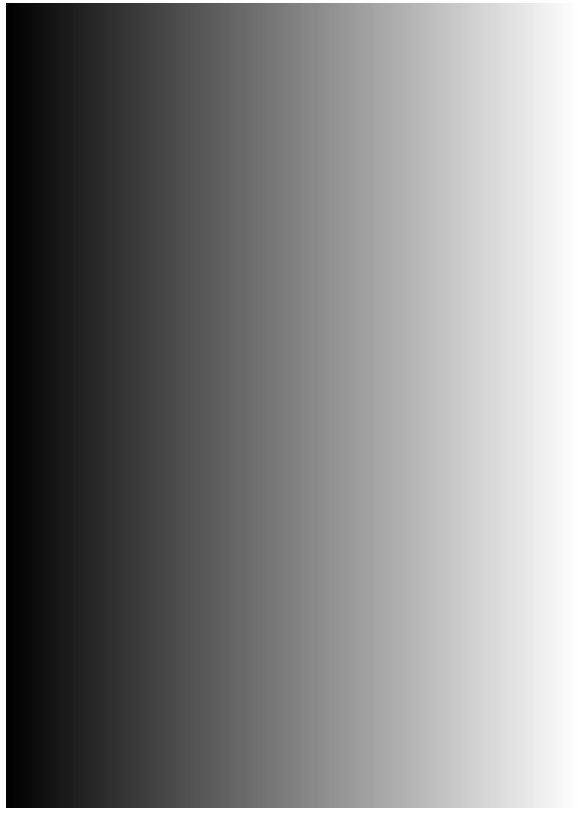
Matplotlib to save Numpy array as image

- matplotlib library is one of the standard libraries that is used for creating graphical plots. It is based on the matlab interface.
- The matplotlib.image class of this library allows us to read and process images.
- imsave() method is used to save image.

```
1. import matplotlib.image
2. a = np.arange(0, 414720, 1, np.uint8)
3. a = np.reshape(a, (720, 576))
4. matplotlib.image.imsave('image.png', a)
```

Required output





Numpy Array

```
[[  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]
 ...
 [  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]
 [  0   1   2 ... 253 254 255]]
```

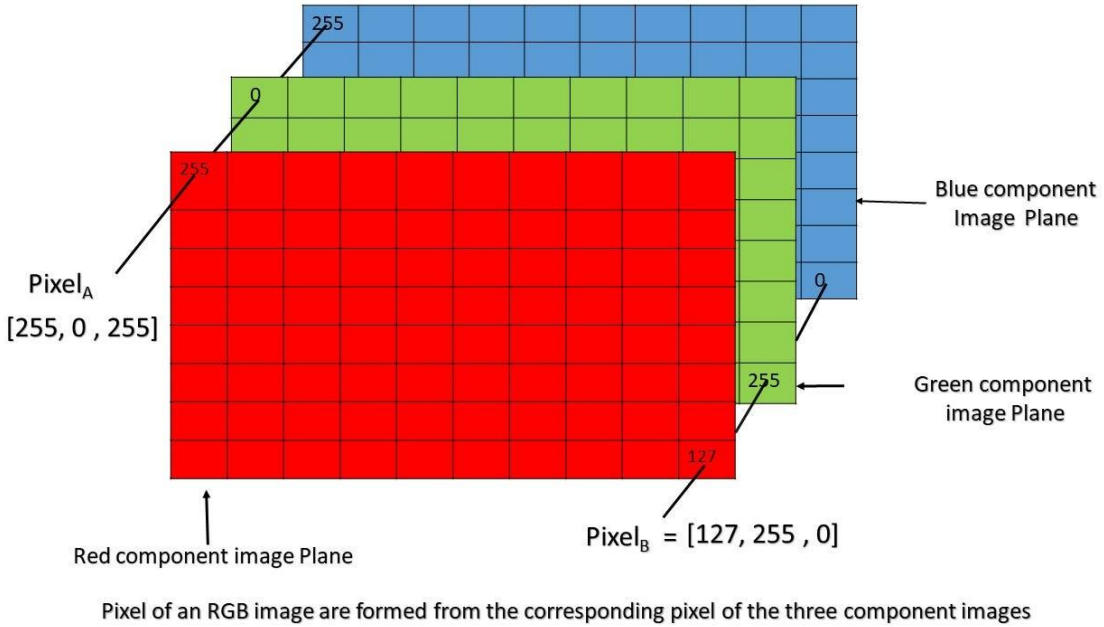
Coding

```
1. arr1 = np.arange(0, 256, 1, np.uint8)
2. for i in range(512):
3.   arr2 = np.arange(0, 256, 1, np.uint8)
4.   arr1=np.append(arr1,arr2)
5. a2 =np.reshape(arr1,(513,256) )
6. print (a2)
7. img = Image.fromarray(a2)
8. img.save("image.png")
```

1. img = Image.fromarray(_____)
2. img.save("image.png")

Reading Images

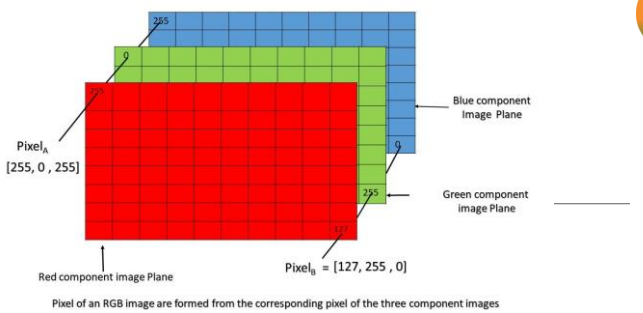
1. import numpy as np
2. import matplotlib.pyplot as plt
3. from PIL import Image, ImageOps



Pixel of an RGB image are formed from the corresponding pixel of the three component images

4. img = np.array(Image.open('apj3.jpg'))
5. plt.imshow(img)

Image in Numpy Array



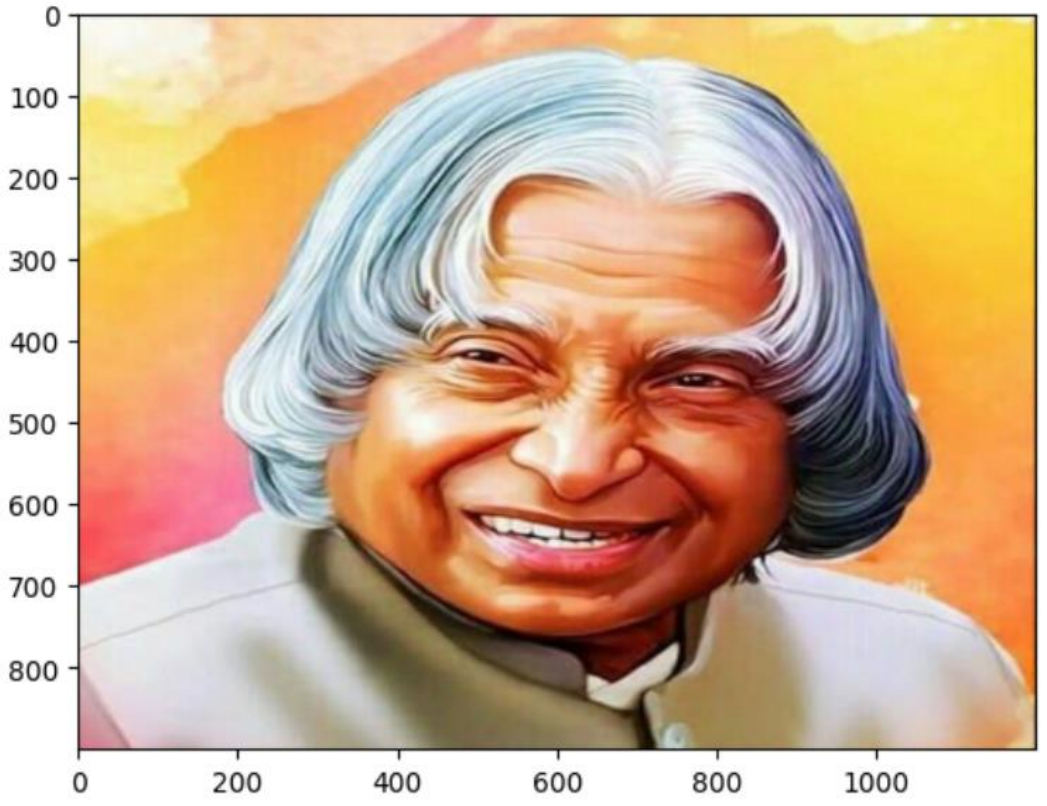
Pixel of an RGB image are formed from the corresponding pixel of the three component images

```
print('# of dims: ',img.ndim)      # dimension of an image
print('Img shape: ',img.shape)     # shape of an image
print('Dtype: ',img.dtype)
print(img[20, 20])
```

```
# of dims:  3
Img shape:  (900, 1200, 3)
Dtype:  uint8
[251 251 241]
```

```
img = np.array(Image.open('apj3.jpg'))
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fb56c5d43d0>
```



Reducing Red Pixels to zero

```
img = np.array(Image.open('apj3.jpg'))
# making red as zero
for r in range (900):
  for c in range(1200):
    img[r,c,0]=0

print(img[0])
```

```
[[  0 250 248]
 [  0 250 248]
 [  0 250 248]
 ...
 [  0 205 160]
 [  0 194 149]
 [  0 193 148]]
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fb5574dafb0>
```

Reducing Green Pixels to zero

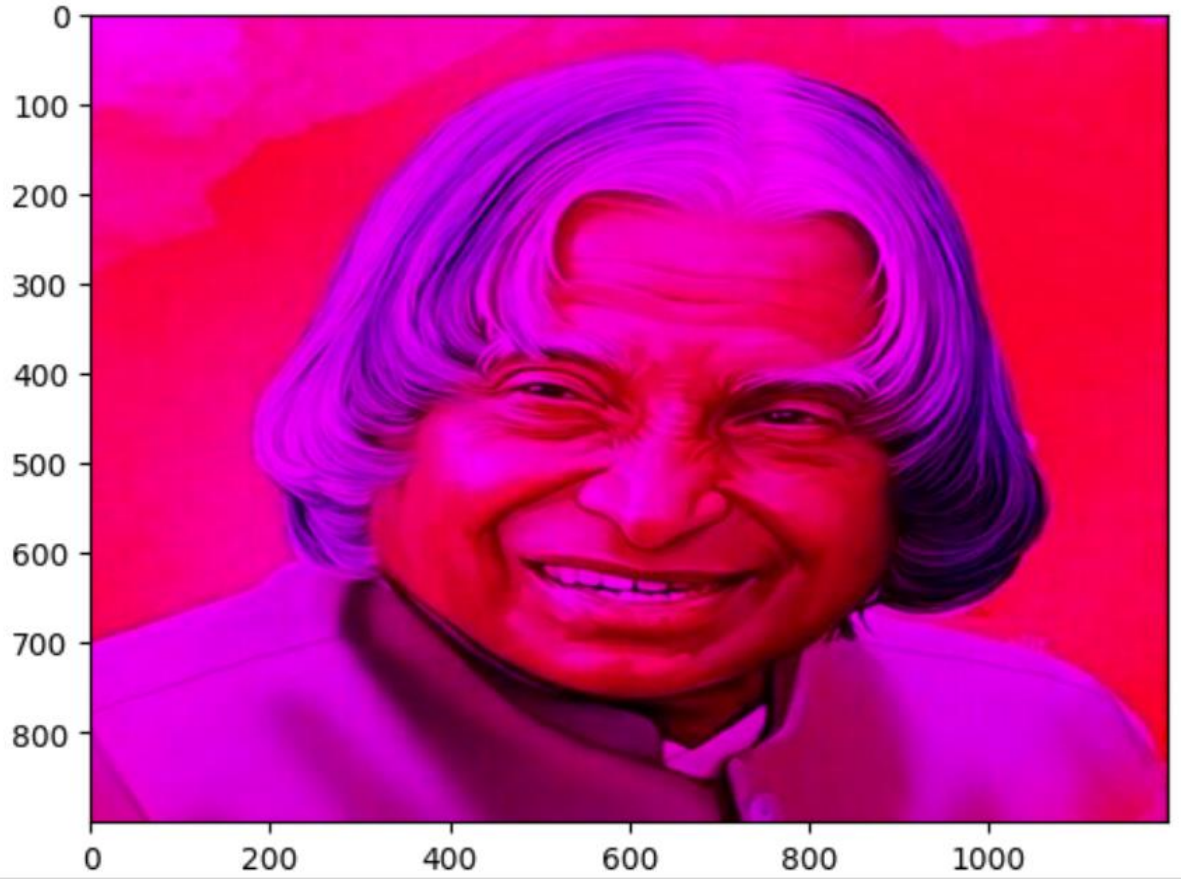
```
img = np.array(Image.open('apj3.jpg'))
# making red as green
for r in range (900):
  for c in range(1200):
    img[r,c,1]=0

print(img[0])
```

```
[[251   0 248]
 [251   0 248]
 [251   0 248]
 ...
 [209   0 160]
 [198   0 149]
 [197   0 148]]
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fb56c244400>
```



Reducing blue Pixels to zero

```
img = np.array(Image.open('apj3.jpg'))
# keeping only Red
for r in range (900):
  for c in range(1200):
    img[r,c,2]=0
    img[r,c,1]=0

print(img[0])
```

```
[[251   0   0]
 [251   0   0]
 [251   0   0]
 ...
 [209   0   0]
 [198   0   0]
 [197   0   0]]
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fb5576867d0>
```

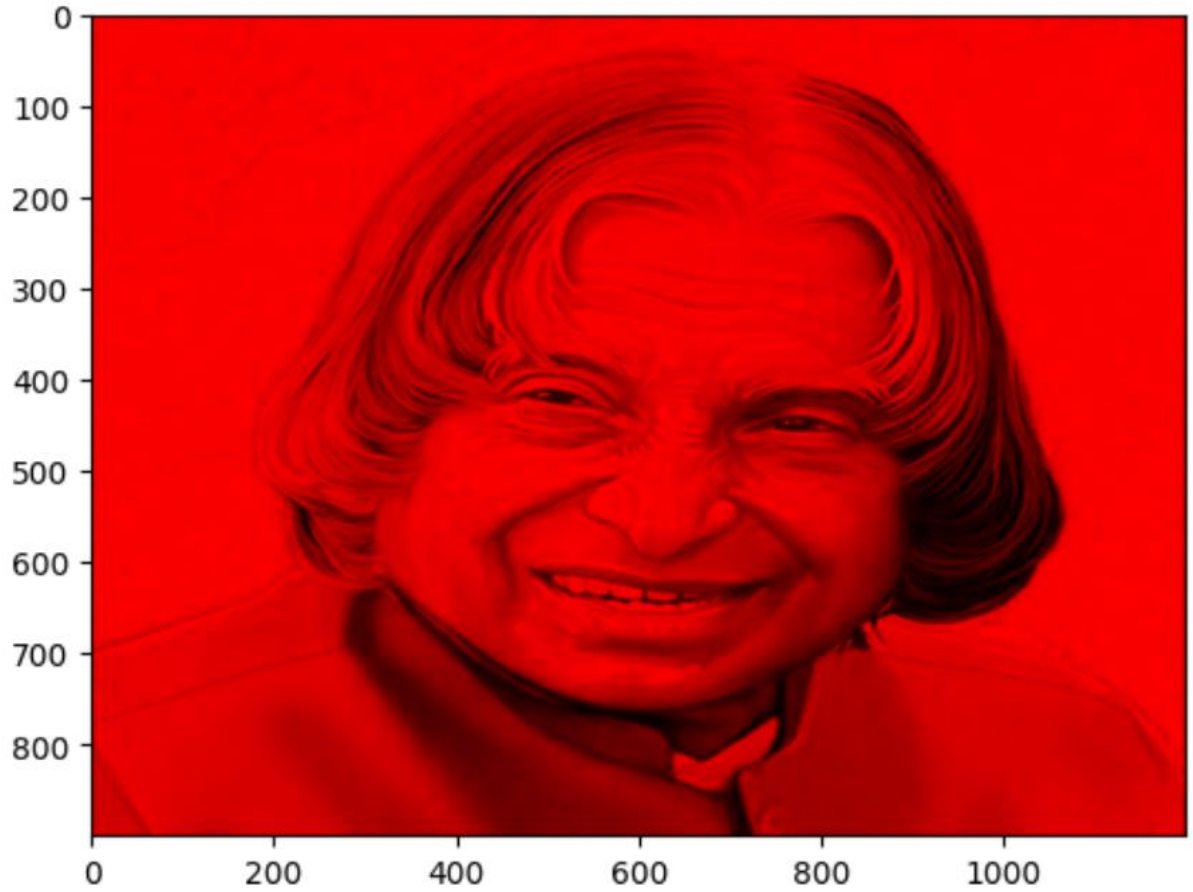


Image Filters

- **Grayscale**
- **Brightness**
- **Contrast**
- **Saturation**
- **Gamma correction**
- **Blur**

Filter – Grayscale

- Converting an image into grayscale is very straight forward.
- We need to convert three color values into a single color value.
- We can do this by taking their average (also known as the brightness).
- We will then put this value in red, green, and blue of the new image.
- Though, we have to make sure the new values are still integers, so we will use the function int() .

$$\text{grayscale} = \frac{r + g + b}{3}$$

Grayscale

```
1.  def filter(image):
2.      width, height = image.size   # Get image dimesions
3.      # Create a white RGB image
4.      new_image = Image.new("RGB", (width, height), "white")
5.      # Grayscale filter
6.      for x in range(width):
7.       for y in range(height):
8.        # Get original pixel colors
9.        r, g, b = image.getpixel((x, y))
10.       r_ = g_ = b_ = (r+g+b)/3   # New pixel colors
11.       # Change new pixel
12.       new_pixel = (int(r_), int(g_), int(b_))
13.       new_image.putpixel((x, y), new_pixel)
14.     return new_image
```

```python
def filter(image):
    width, height = image.size   # Get image dimesions

    # Create a white RGB image
    new_image = Image.new("RGB", (width, height), "white")

    # Grayscale filter
    for x in range(width):
      for y in range(height):
        # Get original pixel colors
        r, g, b = image.getpixel((x, y))

        r_ = g_ = b_ = (r+g+b)/3   # New pixel colors

        # Change new pixel
        new_pixel = (int(r_), int(g_), int(b_))
        new_image.putpixel((x, y), new_pixel)

    return new_image
```

Grayscale

```
1. # Open image
2. image = Image.open('mountain.png')
3. # Show result
4. new_image = filter(image)
5. new_image.show()
```

Weighted average grayscale

Another way to do this is by taking a weighted sum of the red, green, and blue values.

$$\text{grayscale} = \alpha_1 r + \alpha_2 g + \alpha_3 b$$

where $\boldsymbol{\alpha_1}$, $\boldsymbol{\alpha_2}$, and $\boldsymbol{\alpha_3}$ are positive constants, such that $\boldsymbol{\alpha_1 + \alpha_2 + \alpha_3 = 1}$. These values are usually chosen to be (YUV color encoding system):

$$\text{grayscale} = 0.299r + 0.587g + 0.114b$$

1. for x in range(width):
2. for y in range(height):
3. r, g, b = image.getpixel((x, y))
4. r_ = g_ = b_ = 0.299*r + 0.587*g + 0.114*b
5. new_pixel = (int(r_), int(g_), int(b_))
6. new_image.putpixel((x, y), new_pixel)

Brightness

-64 brightness (left), +0 brightness (center) and +64 brightness (left)

Contrast

-100 contrast (left), +0 contrast (middle), and +100 (right)

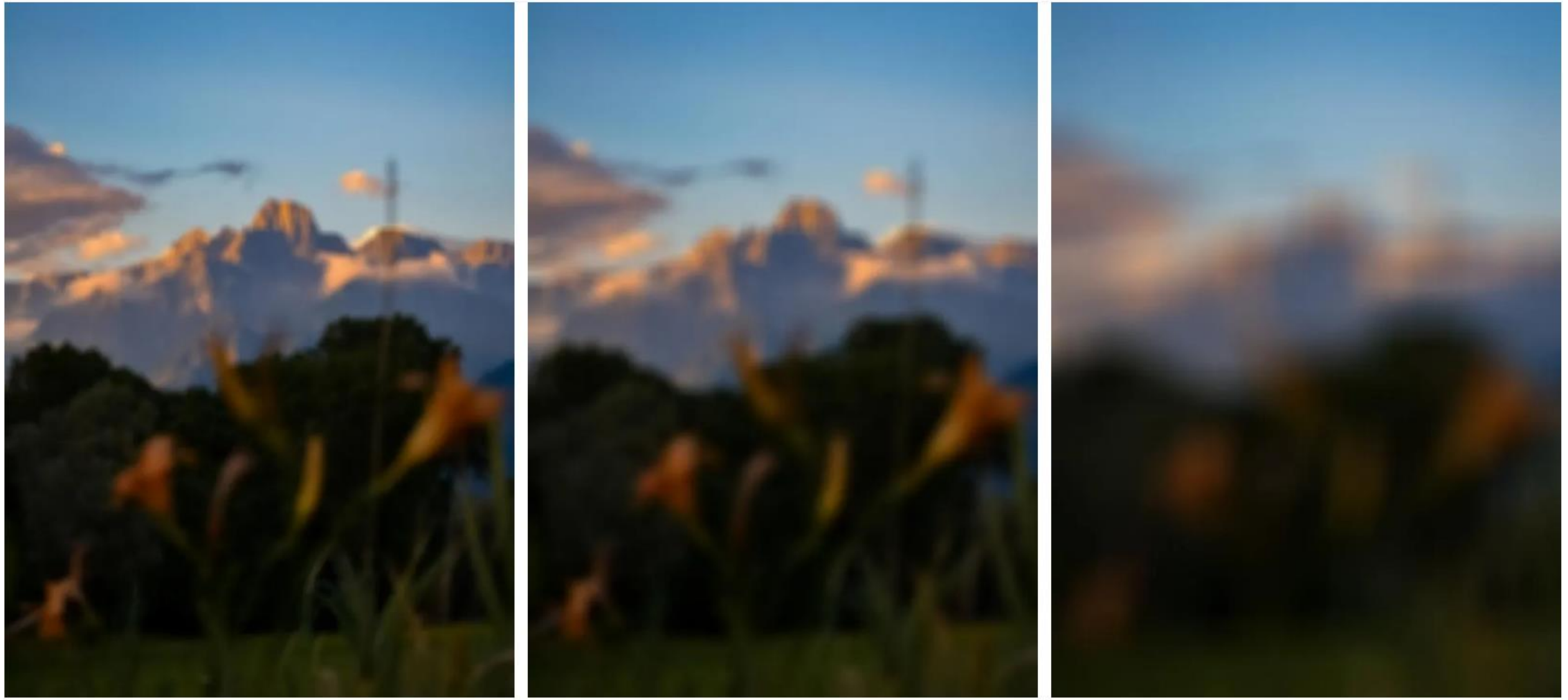
Saturation

-100 saturation (left), +0 saturation (middle), and +100 saturation (right)

Gamma correction

Gamma values (from left to right, top to bottom) 0.33, 0.66, 1.00,

Blur



11 blur (left), 21 blur (middle), 71 blur (right)

Other filter

- Levels
- Curves
- Sepia
- Color balance (including white balance)
- Vibrance
- Exposure
- Sharpening

Color Isolation

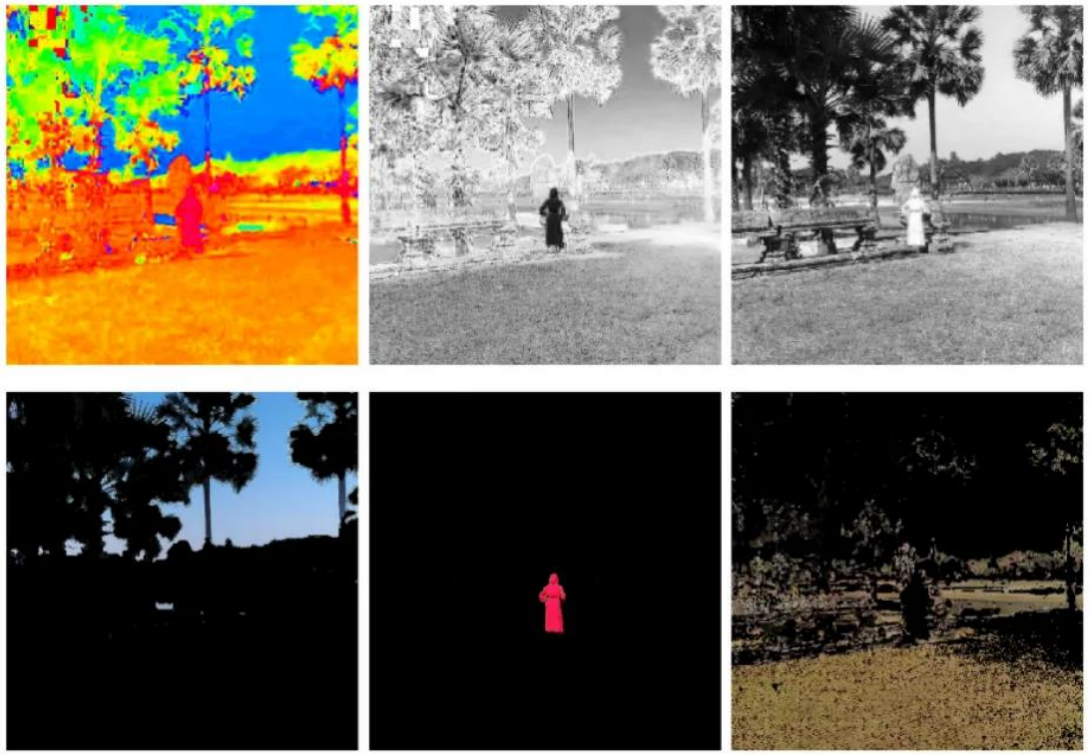
Image Processing with Python — Color Isolation for Beginners

- How to isolate sections of your image by their color

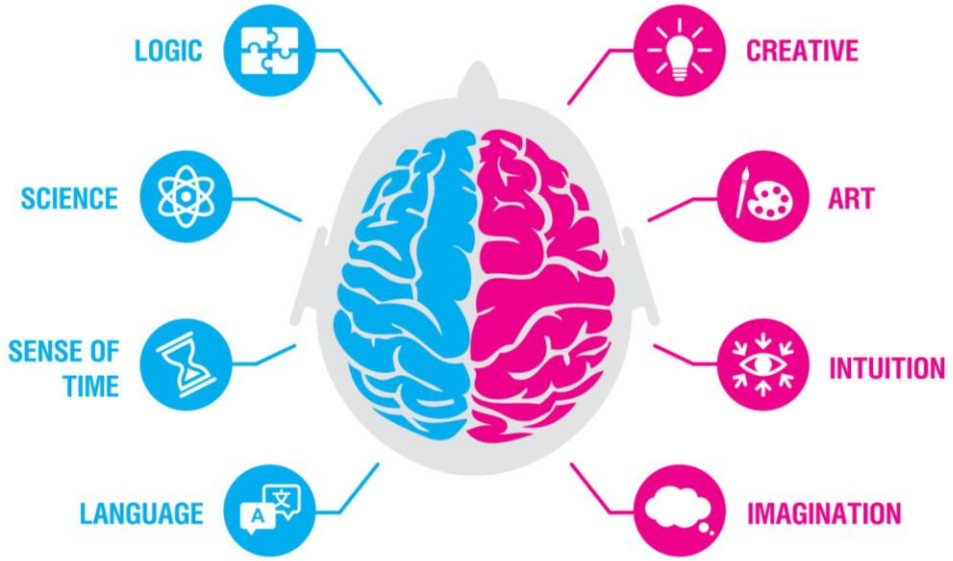
- Image Processing and

Pixel Manipulation: Photo Filters

- Image Processing using Numpy




Q
A


LOGIC
CREATIVE
SCIENCE
ART
SENSE OF TIME
INTUITION
LANGUAGE
IMAGINATION