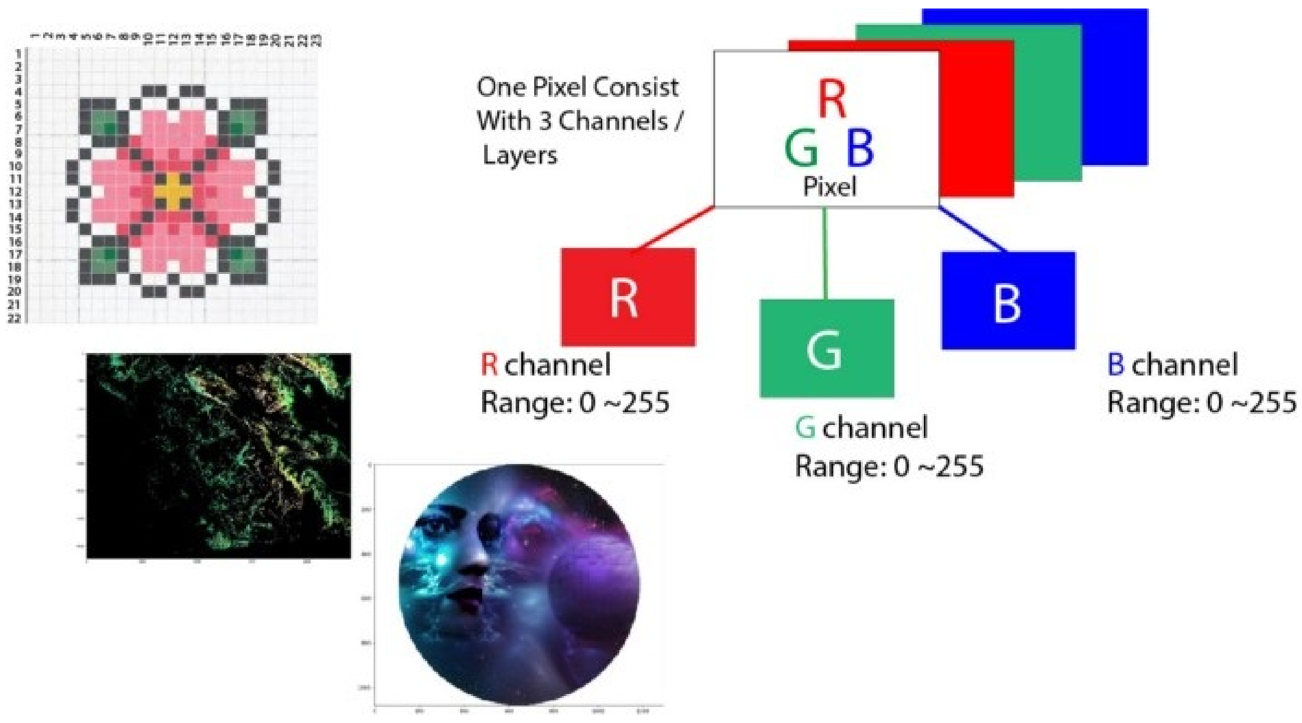
Image Data analysis using python

Contents :



1. Introduction: A Little Bit About Pixel

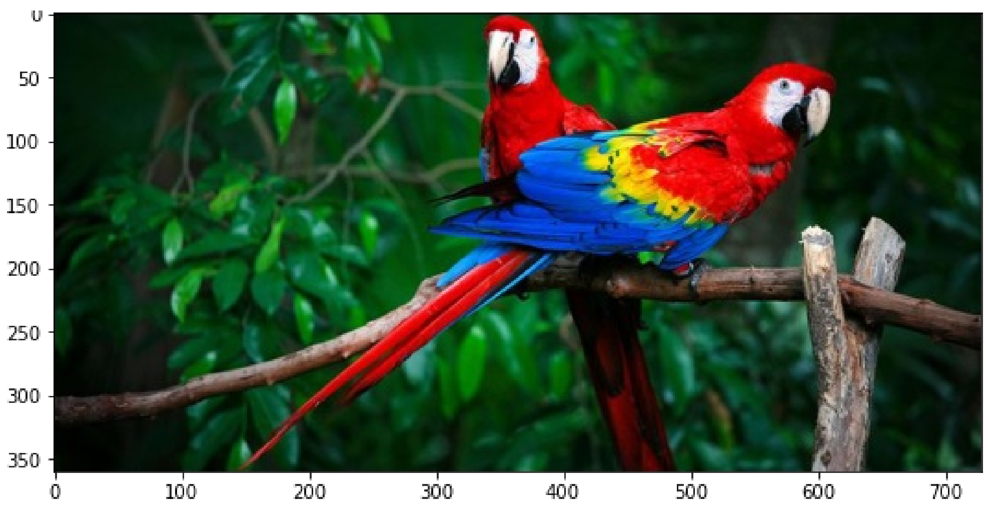
let's load an image and observe its various properties in general.

In [3]:

```
# Importing required modules
import imageio
import matplotlib.pyplot as plt

# The output of plotting commands is displayed inline within frontends.
%matplotlib inline
img = imageio.imread('./parrots.jpg')
plt.figure(figsize = (9,6))
plt.imshow(img)
```

Out[3]:

```
<matplotlib.image.AxesImage at 0x1c5bb42dac0>
```

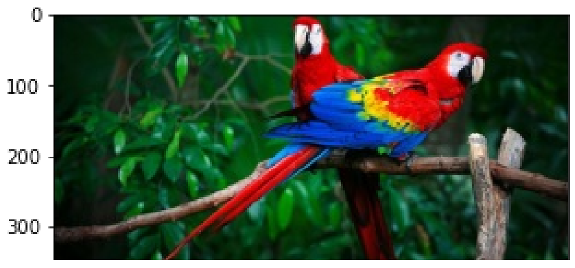


Observe Basic Properties of Image

In [1]:

```
# Importing required modules
import imageio
import matplotlib.pyplot as plt

# The output of plotting commands is displayed inline within frontends.
%matplotlib inline
img = imageio.imread('./parrots.jpg')

# To plot the image with specific size, i took 5,5
plt.figure(figsize = (5,5))
plt.imshow(img)

print('Type of the image : ' , type(img))
# The shape of the ndarray shows that it is a three-layered matrix
print('Shape of the image : {}'.format(img.shape))
print('Image Hight {}'.format(img.shape[0]))
print('Image Width {}'.format(img.shape[1]))
print('Dimension of Image {}'.format(img.ndim))
# ndim() function return the number of dimensions of an array
print('Image size {}'.format(img.size))
print('Maximum RGB value in this image {}'.format(img.max()))
print('Minimum RGB value in this image {}'.format(img.min()))

# A specific pixel located at Row : 110 ; Column : 60
# Each channel's value of it, gradually R , G , B
print('Value of only R channel {}'.format(img[ 110, 60, 0]))
print('Value of only G channel {}'.format(img[ 110, 60, 1]))
print('Value of only B channel {}'.format(img[ 110, 60, 2]))
```

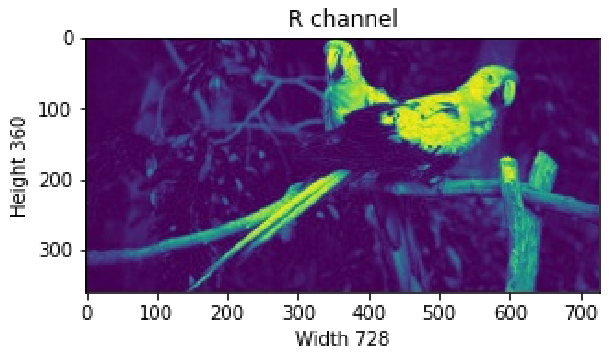
```
Type of the image :  <class 'imageio.core.util.Array'>
Shape of the image : (360, 728, 3)
Image Hight 360
Image Width 728
Dimension of Image 3
Image size 786240
Maximum RGB value in this image 255
Minimum RGB value in this image 0
Value of only R channel 5
Value of only G channel 44
Value of only B channel 17
```



view of each channel in the whole image

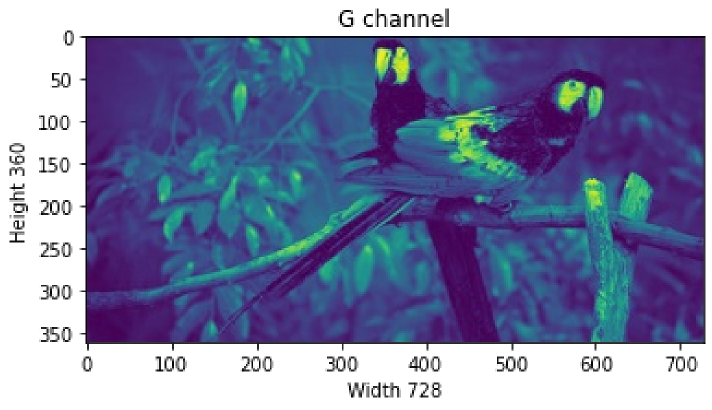
In [13]:

```
# Importing required modules
import imageio
import matplotlib.pyplot as plt

# The output of plotting commands is displayed inline within frontends.
%matplotlib inline
img = imageio.imread('./parrots.jpg')


# To see the specific channel
plt.title('R channel')
plt.ylabel('Height {}'.format(img.shape[0]))
plt.xlabel('Width {}'.format(img.shape[1]))
plt.imshow(img[ : , : , 0])
plt.show()
```



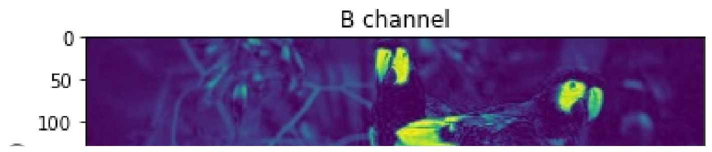
In [14]:

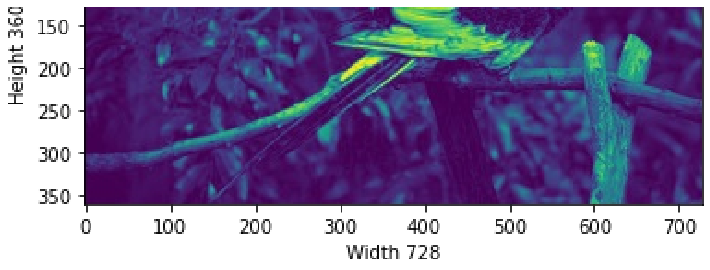
```
plt.title('G channel')
plt.ylabel('Height {}'.format(img.shape[0]))
plt.xlabel('Width {}'.format(img.shape[1]))
plt.imshow(img[ : , : , 1])
plt.show()
```



In [15]:

```
plt.title('B channel')
plt.ylabel('Height {}'.format(img.shape[0]))
plt.xlabel('Width {}'.format(img.shape[1]))
plt.imshow(img[ : , : , 2])
plt.show()
```

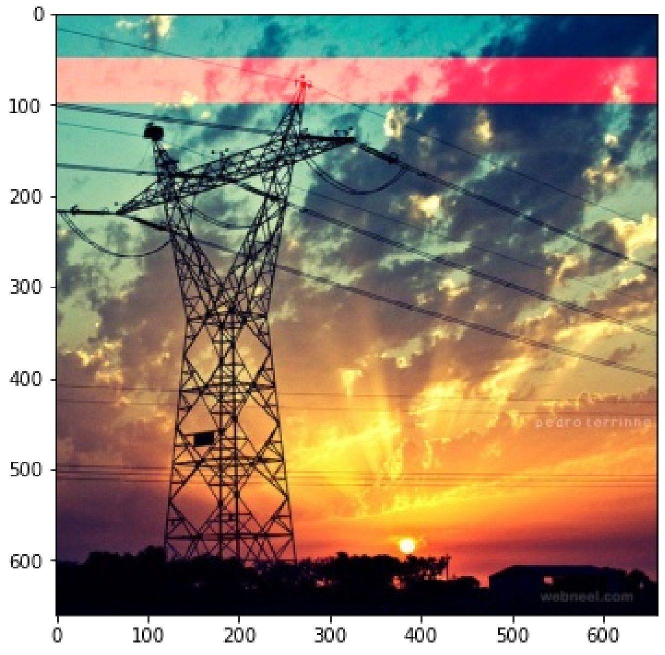




To change the number of RGB values from the existing once

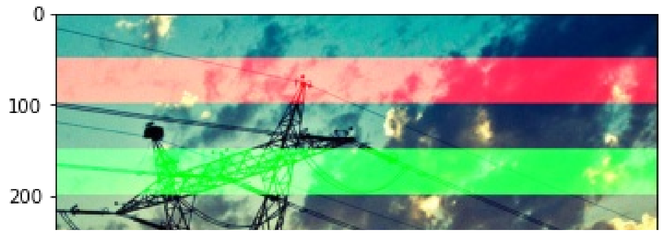
In [34]:

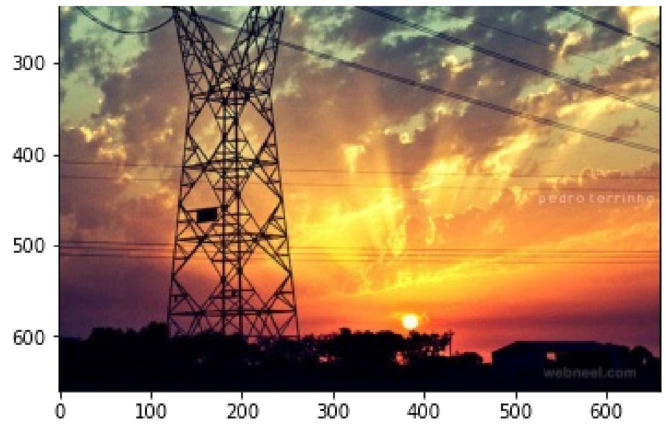
```
# As an example,
# R channel: Row — 100 to 110
# G channel: Row — 200 to 210
# B channel: Row — 300 to 310

# Importing required modules
import imageio
import matplotlib.pyplot as plt

# The output of plotting commands is displayed inline within frontends.
%matplotlib inline
img = imageio.imread('./Tower.jpg')

# full intensity to those pixel's R channel
img[50:100 , : , 0] = 255
plt.figure( figsize = (9,6))
plt.imshow(img)
plt.show()
```



In [35]:

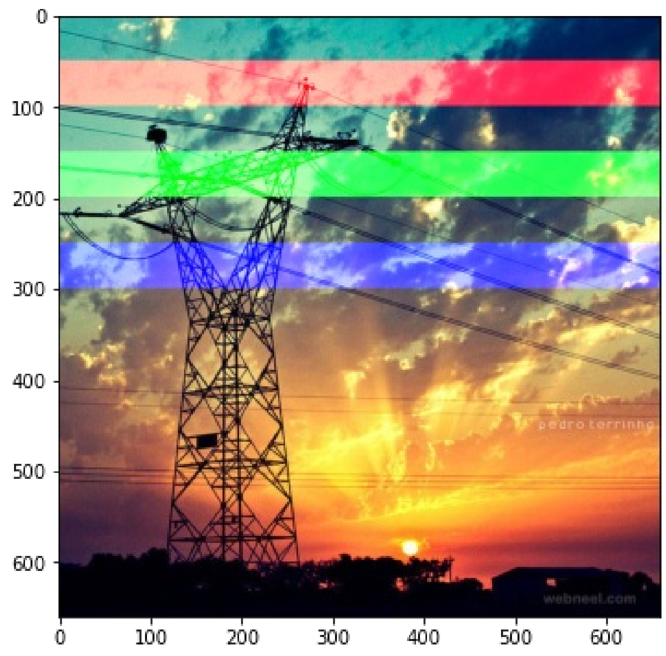
```
# full intensity to those pixel's G channel
img[150:200 , : , 1] = 255
plt.figure( figsize = (10,6))
plt.imshow(img)
plt.show()
```





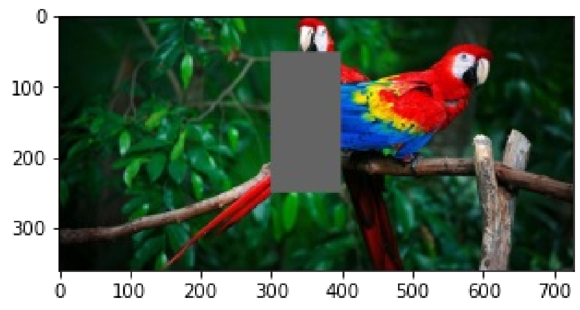
In [36]:

```
# full intensity to those pixel's B channel
img[250:300 , : , 2] = 255
plt.figure( figsize = (9,6))
plt.imshow(img)
plt.show()
```



In [5]:

```
# set value 100 of all channels to those pixels which turns them to white
img[ 50:250 , 300:400 , [0,1,2] ] = 100
plt.figure( figsize = (5,5))
plt.imshow(img)
plt.show()
```

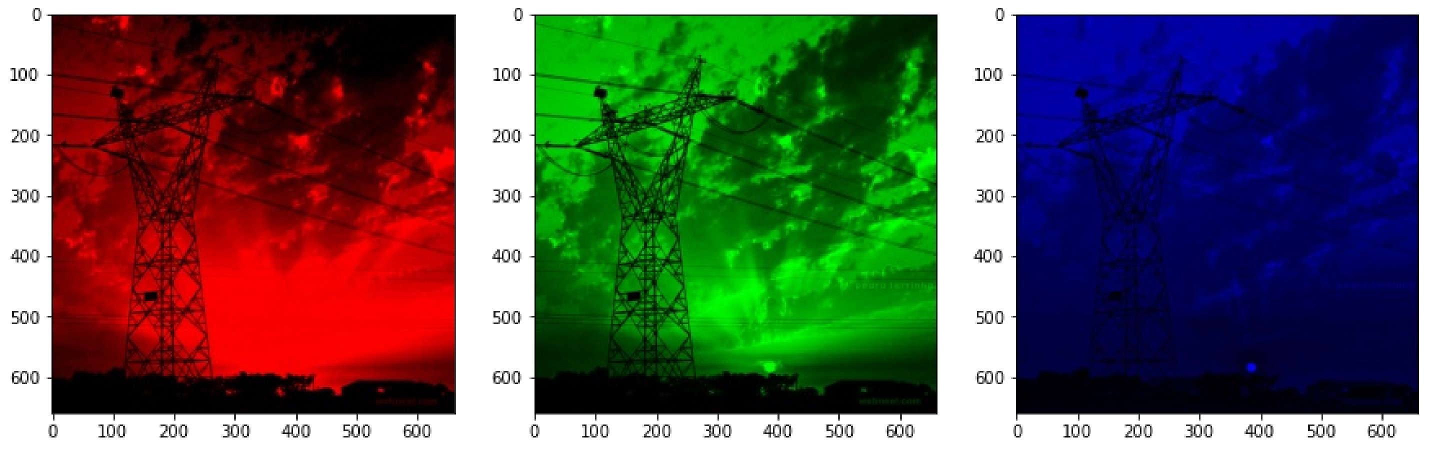


Splitting Layers

In [6]:

```
import numpy as np
```

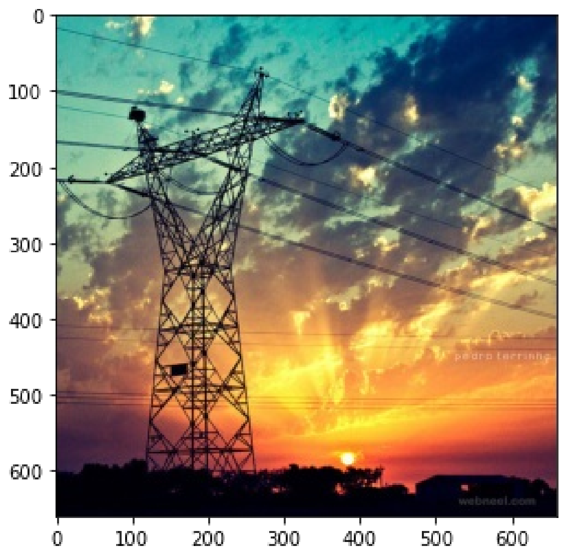
```
pic = imageio.imread('./Tower.jpg')
fig, ax = plt.subplots(nrows = 1, ncols=3, figsize=(15,5))
for c, ax in zip(range(3), ax):
    # create zero matrix
     split_img = np.zeros(pic.shape, dtype="uint8")
     # 'dtype' by default: 'numpy.float64'
    # assing each channel
     split_img[ :, :, c] = pic[ :, :, c]
    # display each channel
     ax.imshow(split_img)
```



Use Logical Operator To Process Pixel Values

In [56]:

```
pic = imageio.imread('./Tower.JPG')
plt.figure(figsize=(9,5))
plt.imshow(pic)
plt.show()
```



In [57]:

```
low_pixel = pic < 20
# to ensure of it let's check if all values in low_pixel are True or not
if low_pixel.any() == True:
    print(low_pixel.shape)
```

```
(660, 660, 3)
```

In [58]:

```
print(pic.shape)
print(low_pixel.shape)
```

```
(660, 660, 3)
(660, 660, 3)
```

In [ ]:

```
```