```
import matplotlib.pyplot as plt #import


flower = plt.imread('flower.jpg')


print(flower.shape)
```

```
(549, 732, 3)
```

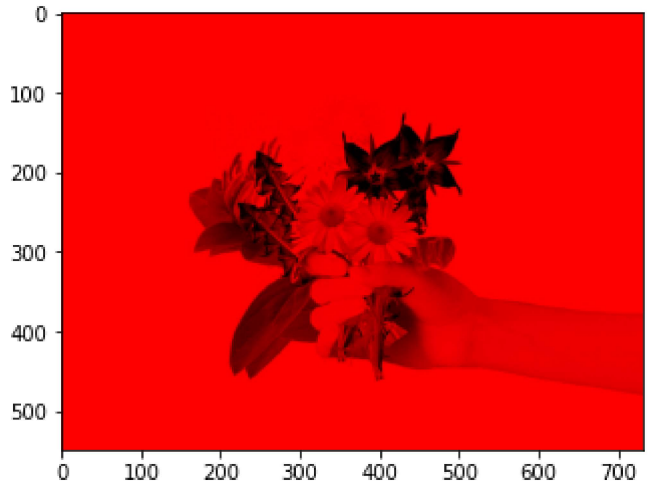
```
# print(flower)
print(flower[:,:,[1]])
```

```
[[[220]
  [220]
  [220]
  ...
  [220]
  [220]
  [220]]

 [[220]
  [220]
  [220]
  ...
  [220]
  [220]
  [220]]

 [[220]
  [220]
  [220]
  ...
  [220]
  [220]
  [220]]

 ...

 [[220]
  [220]
  [220]
  ...
  [220]
  [220]
  [220]]

 [[220]
  [220]
  [220]
  ...
  [220]
  [220]
```

```
        [220]]

       [[220]
        [220]
        [220]
        ...
        [220]
        [220]
        [220]]]
```

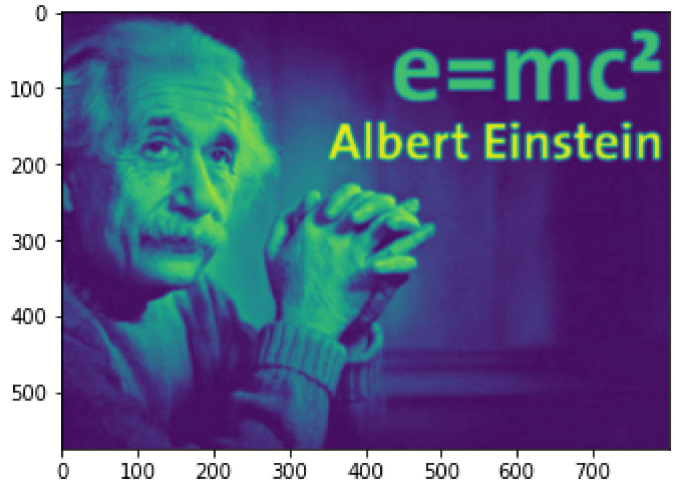
```
plt.imshow(flower)
plt.show()
```


```
red_flower = flower.copy()
# red_flower[:,:,[1,2]] = 0 #red color[1,2], pink color[1,1], and many more color combination
red_flower[:,:,[1,2]] = 0
plt.imshow(red_flower)
plt.show()
```




```
import cv2
import numpy as np
```

```
import matplotlib.pyplot as plt

Image = 'einstein.jpg'
img = cv2.imread(Image, 0)
plt.imshow(img)
plt.show()
img.shape
```




```
(575, 800)
```

```
cv2.imwrite('en.png',flower)
```

```
True
```

```
# Function to obtain histogram of an image
def hist_plot(img):

    # empty list to store the count of each intensity value
    count = []

    # empty list to traverse each intensity value
    r = []

    # loop to traverse each intensity value
    for k in range(0, 256):
        r.append(k)
        count1 = 0


        # loops to traverse each pixel in the image
        for i in range(m):
            for j in range(n):
                if img[i, j] == k:
                    count1 += 1
            count.append(count1)
```

```
            return (r, count)

    # import matplotlib.pyplot as plt

    print(img.shape)
    # To ascertain total numbers of rows and columns of the image, size of the image
    m, n = img.shape
    r1, count1 = hist_plot(img)
    # print(r1, count1)

    # plotting the histogram
    # Syntax: stem([x, ] y, linefmt=None, markerfmt=None, basefmt=None)


    plt.stem(r1, count1, use_line_collection = True)
    plt.xlabel('intensity value')
    plt.ylabel('number of pixels')
    plt.title('Histogram of the original image')
    plt.show()
    print(img.max(), img.min())
```

```
(575, 800)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4376/3848480799.py in <module>
     11
     12
---> 13 plt.stem(r1, count1, use_line_collection = True)
     14 plt.xlabel('intensity value')
     15 plt.ylabel('number of pixels')

C:\Python39\lib\site-packages\matplotlib\pyplot.py in stem(linefmt, markerfmt, basefmt,
bottom, label, use_line_collection, orientation, data, *args)
   2871         label=None, use_line_collection=True, orientation='vertical',
   2872         data=None):
-> 2873     return gca().stem(
   2874         *args, linefmt=linefmt, markerfmt=markerfmt, basefmt=basefmt,
   2875         bottom=bottom, label=label,

C:\Python39\lib\site-packages\matplotlib\__init__.py in inner(ax, data, *args,
**kwargs)
   1410     def inner(ax, *args, data=None, **kwargs):
   1411         if data is None:
-> 1412             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1413
   1414         bound = new_sig.bind(ax, *args, **kwargs)

C:\Python39\lib\site-packages\matplotlib\axes\_axes.py in stem(self, linefmt,
markerfmt, basefmt, bottom, label, use_line_collection, orientation, *args)
   2888                 linestyle = rcParams['lines.linestyle']
   2889             xlines = self.vlines if orientation == "vertical" else self.hlines
-> 2890             stemlines = xlines(
   2891                 locs, bottom, heads,
   2892                 colors=linecolor, linestyles=linestyle, label="_nolegend_")

C:\Python39\lib\site-packages\matplotlib\__init__.py in inner(ax, data, *args,
**kwargs)
   1410     def inner(ax, *args, data=None, **kwargs):
   1411         if data is None:
```

```
# Histogram Stretching using user defined function
```

```
C:\Python39\lib\site-packages\matplotlib\axes\_axes.py in vlines(self, x, ymin, ymax,
```

```
# Histogram Equalization using user defined function
```

```
-> 1134         masked_verts[:, 1, 1] = vmax
```

```
# Histogram Stretching using builtin function

# import cv2, numpy, matplotlib
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
Image = 'Sophia Robot.jpg'
# Image = 'einstein.jpg'
img = cv2.imread(Image, 0)
equ = cv2.equalizeHist(img)
res = np.hstack((img, equ))  # stacking image side-by-side
cv2.imwrite('res.png', res)
plt.imshow(res)
```

```
<matplotlib.image.AxesImage at 0x19c50017d90>
```