In [1]:

```
import os
os.getcwd()
os.listdir()
```

```
...
```

In [2]:

```
path = os.getcwd()  ##'''/usr/share/cups/charmaps'
jpg_files = [f for f in os.listdir(path) if f.endswith('.jpg')]
jpg_files
```

```
...
```

In [3]:

```
import numpy as np
```

In [4]:

```
import matplotlib.pyplot as plt
```

In [5]:

```
im_1=plt.imread(jpg_files[2])
```

In [6]:

```
type(im_1)
```

Out[6]:

```
numpy.ndarray
```

In [12]:

```
im_1.ndim
```

Out[12]:

```
3
```

In [7]:

```
im_1.shape
```

Out[7]:

```
(427, 640, 3)
```

In [14]:

```
plt.imshow(im_1)
plt.show()
```

In [8]:

```
im_1.size
```

Out[8]:

```
819840
```

In [12]:

```
im_1[:,:,0] # 0,1,2 ok, 4 5 error
```

Out[12]:

```
array([[ 57,  56,  55, ...,  50,  49,  48],
       [ 58,  56,  54, ...,  50,  49,  49],
       [ 59,  58,  56, ...,  50,  49,  50],
       ...,
       [ 89, 100, 105, ...,  89,  86,  83],
       [ 85,  95, 100, ...,  88,  85,  82],
       [ 80,  88,  93, ...,  87,  85,  82]], dtype=uint8)
```

In [9]:

```
im_2=im_1[:,:,1]
```

In [15]:

```
im_2=im_2-10
```

In [16]:

```
def my_rotate_for_RGB(old_image):
    m,n,p=old_image.shape
    new_image=np.zeros((n,m,3),dtype=int)    # if dtype=int  absent , error
    for i in range(m):
        for j in range(n):
            new_image[j,i,0]=old_image[i,j,0]
            new_image[j,i,1]=old_image[i,j,1]
            new_image[j,i,2]=old_image[i,j,2]
    return new_image
```

In [17]:

```
im_4=my_rotate_for_RGB(im_1)
plt.imshow(im_4)
plt.show()
```

...

In [29]:

```
def convert_RGB_to_Gray(old_image_RGB):
    m,n,p=old_image_RGB.shape
    new_image_gray_level=np.zeros((m,n),)
    for i in range(m):
        for j in range(n):
            s=old_image_RGB[i,j,0]+old_image_RGB[i,j,1]+old_image_RGB[i,j,2]
            s=s/3
            new_image_gray_level[i,j]=int(s)
    return new_image_gray_level
def convert_RGB_to_Binary(old_image_RGB,threshold=40):
    m,n,p=old_image_RGB.shape
    new_image_binary=np.zeros((m,n),)
    for i in range(m):
        for j in range(n):
            s=old_image_RGB[i,j,0]+old_image_RGB[i,j,1]+old_image_RGB[i,j,2]
            s=s/3
            if s>threshold:
                new_image_binary[i,j]=1
            else:
                new_image_binary[i,j]=0
    return new_image_binary
```

In [30]:

```
im_gray=convert_RGB_to_Gray(im_1)
im_binary=convert_RGB_to_Binary(im_1)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: RuntimeW
arning: overflow encountered in ubyte_scalars

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: Runtime
Warning: overflow encountered in ubyte_scalars
  from ipykernel import kernelapp as app
```

In [23]:

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

...

In [32]:

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

...

In [28]:

```
max(im_1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-28-46a535f76c9a> in <module>
----> 1 max(im_1)

ValueError: The truth value of an array with more than one element is ambigu
ous. Use a.any() or a.all()
```

In [57]:

```
im_2=convert_RGB_to_Gray(im_1)
plt.imshow(im_2,cmap='gray')
plt.show()
```

...

In [ ]:

```
plt.imsave("new_image.jpg",im_3)
```

In [16]:

```
im_1.ndim,im_1.shape
```

Out[16]:

```
(3, (427, 640, 3))
```

In [ ]:

```

```

In [17]:

```
im_1.ndim,im_1.shape
```

Out[17]:

```
(3, (427, 640, 3))
```

In [16]:

```
my_histogram_R_G_B={}    # R,G,B her biri için ayrı ayrı  histogram
m,n,p=im_1.shape
for i in range(m):
    for j in range(n):
        s=(im_1[i,j,0])   # ,im_1[i,j,1],im_1[i,j,2])   #  s=im_1[i,j,:], s cannot be Key
        if (0,s) in my_histogram_R_G_B.keys():              # because its type is np.ndar
            my_histogram_R_G_B[(0,s)]=my_histogram_R_G_B[(0,s)]+1
        else:
            my_histogram_R_G_B[(0,s)]=1
my_histogram_R_G_B
```

...

In [19]:

```
# my_histogram_R_G_B={}    # R,G,B her biri için ayrı ayrı  histogram
m,n,p=im_1.shape
for i in range(m):
    for j in range(n):
        s=(im_1[i,j,1])   # ,im_1[i,j,1],im_1[i,j,2])   #  s=im_1[i,j,:], s cannot be Key
        if (1,s) in my_histogram_R_G_B.keys():              # because its type is np.ndar
            my_histogram_R_G_B[(1,s)]=my_histogram_R_G_B[(1,s)]+1
        else:
            my_histogram_R_G_B[(1,s)]=1
my_histogram_R_G_B
```

...

In [20]:

```
# my_histogram_R_G_B={}    # R,G,B her biri için ayrı ayrı  histogram
m,n,p=im_1.shape
for i in range(m):
    for j in range(n):
        s=(im_1[i,j,2])   # ,im_1[i,j,1],im_1[i,j,2])   #  s=im_1[i,j,:], s cannot be Key
        if (0,s) in my_histogram_R_G_B.keys():              # because its type is np.ndar
            my_histogram_R_G_B[(2,s)]=my_histogram_R_G_B[(2,s)]+1
        else:
            my_histogram_R_G_B[(2,s)]=1
my_histogram_R_G_B
```

...

In [18]:

```
t=0
for key in my_histogram_R_G_B.keys():
    t=t+my_histogram_R_G_B[(0,s)]
t,m*n
```

Out[18]:

```
(283136, 273280)
```

In [14]:

...

In [18]:

```
my_histogram={}   # (R,G,B) üçlü histogram
```

In [20]:

```
m,n,p=im_1.shape
for i in range(m):
    for j in range(n):
        s=(im_1[i,j,0],im_1[i,j,1],im_1[i,j,2])   #  s=im_1[i,j,:], s cannot be Key in di
        if s in my_histogram.keys():              # because its type is np.ndarray
            my_histogram[s]=my_histogram[s]+1
        else:
            my_histogram[s]=1
```

In [21]:

```
my_histogram
```

...

In [ ]: