In [ ]:

```
#Автоконтраст черно-белого изображения
from skimage.io import imread, imshow, imsave
img = imread('img.png')
xmax = max(img.ravel())
xmin = min(img.ravel())
k = 255/(xmax-xmin)
ans = ((img-xmin)*k).astype('uint8')
imsave('out_img.png', ans)
```

In [ ]:

```
#Подсчет минимума и максимума устойчивого автоконтраста
from skimage.io import imread, imshow, imsave
img= imread('img.png')
pix = img.shape[0]*img.shape[1]
k= round(pix * 0.05)
v = img.ravel()
v.sort()
xmin = v[k]
xmax = v[pix-k]
print(xmin, xmax)
```

In [ ]:

```
#Устойчивый автоконтраст черно-белого изображения
from skimage.io import imread, imshow, imsave
from numpy import *
img = imread('img.png')
imgf=img.astype('float')
pix = imgf.shape[0]*imgf.shape[1]
k= round(pix * 0.05)
v = imgf.ravel()
v.sort()
xmin = v[k]
xmax = v[pix-k]
koef = 255/(xmax-xmin)
f = ((img-xmin)*koef)
ans=clip(f,0,255)
ans1 = ans.astype('uint8')
imsave('out_img.png', ans1)
```

In [ ]:

```
#Устойчивый цветной автоконтраст
from skimage.io import imread, imsave
from skimage import img_as_float, img_as_ubyte
import numpy as np

img = img_as_float(imread('img.png'))
R, G, B = (img[..., i] for i in range(3))
Y =  0.2126*R+0.7152*G+0.0722*B
U = -0.0999*R-0.3360*G+0.4360*B
V =  0.6150*R-0.5586*G-0.0563*B

vmin, vmax = np.percentile(Y, [5, 95])
Y = np.clip((Y - vmin) / (vmax - vmin), 0, 1)

R = np.clip(Y+1.2803*V, 0, 1)
G = np.clip(Y-0.2148*U-0.3805*V, 0, 1)
B = np.clip(Y+2.1279*U, 0, 1)

img = img_as_ubyte(np.dstack((R, G, B)))
imsave('out_img.png', img)
```

In [ ]:

```
#Преобразование серого мира
from skimage.io import imread, imsave
from skimage import img_as_float, img_as_ubyte
import numpy as np

img = img_as_float(imread('img.png'))
R, G, B = (img[..., i] for i in range(3))

avgR=np.mean(R)
avgG=np.mean(G)
avgB=np.mean(B)

avg=(avgR+avgG+avgB)/3

rw=avgR/avg
rg=avgG/avg
rb=avgB/avg

r=np.clip(R/rw,0,1)
g=np.clip(G/rg,0,1)
b=np.clip(B/rb,0,1)

img = img_as_ubyte(np.dstack((r, g, b)))
imsave('out_img.png', img)
```

In [ ]:

```
#Выравнивание гистограммы
from skimage.io import imread, imsave
from skimage import img_as_float, img_as_ubyte
from numpy import *
from numpy import histogram
from numpy import round
img = imread('img.png')
hist = histogram(img, bins=range(257))[0]
cdf = cumsum(hist)
pix= img.size
minv=cdf[cdf > 0][0]
k=255/(pix-1)
f=round((cdf-minv)*k).astype(uint8)
imsave('out_img.png', f[img])
```