PCD 9 10

J0303201065 Rahma Fairuz Rania

BAB 9

1. Buat list peta warna

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.colors import ListedColormap, LinearSegmentedColormap

viridis = cm.get_cmap('viridis', 8)
```

In [2]:

```
print(viridis(0.56))
```

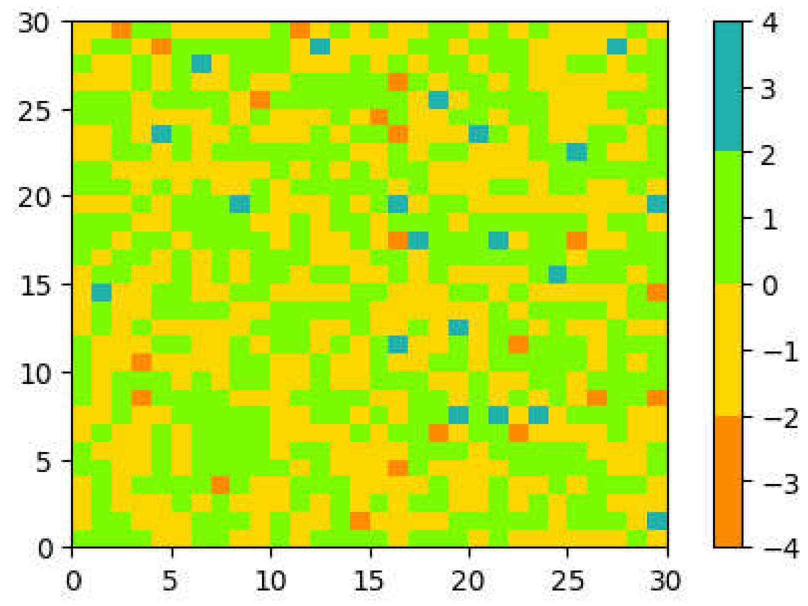
```
(0.122312, 0.633153, 0.530398, 1.0)
```

In [9]:

```
def plot_examples(colormaps):
    '''Helper function to plot data with associated colormap.'''
    np.random.seed(19680801)
    data = np.random.randn(30, 30)
    n = len(colormaps)
    fig, axs = plt.subplots(1, n, figsize=(n * 2 + 2, 3), constrained_layout=True, squeez

    for [ax, cmap] in zip(axs.flat, colormaps):
        psm = ax.pcolormesh(data, cmap=cmap, rasterized=True, vmin=-4, vmax=4)
        fig.colorbar(psm, ax=ax)
    plt.show()
```

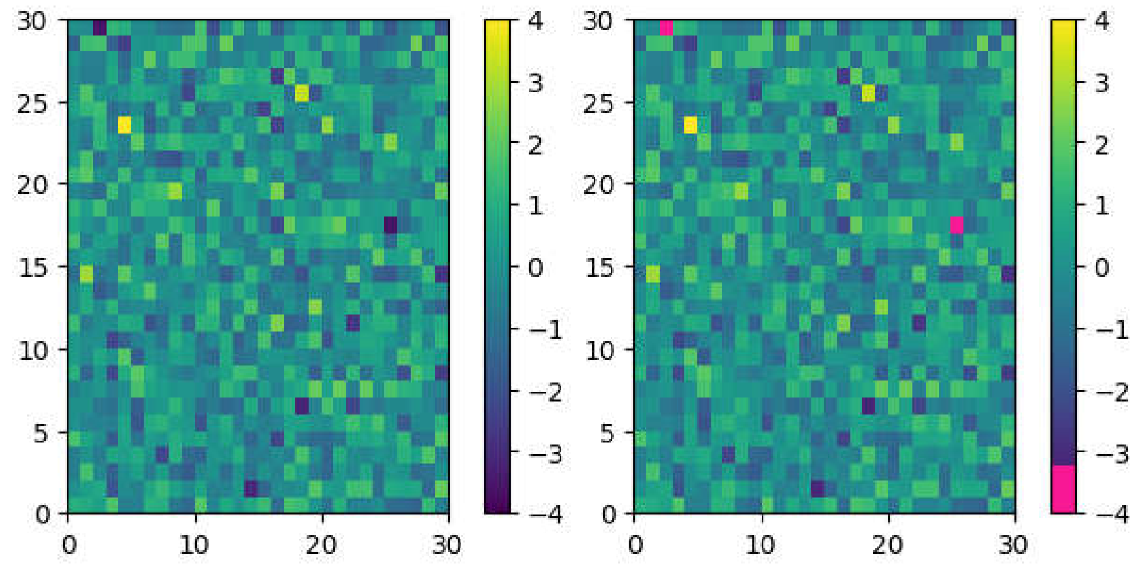
In [6]:

```
cmap = ListedColormap(["darkorange", "gold", "lawngreen", "lightseagreen"])
plot_examples([cmap])
```



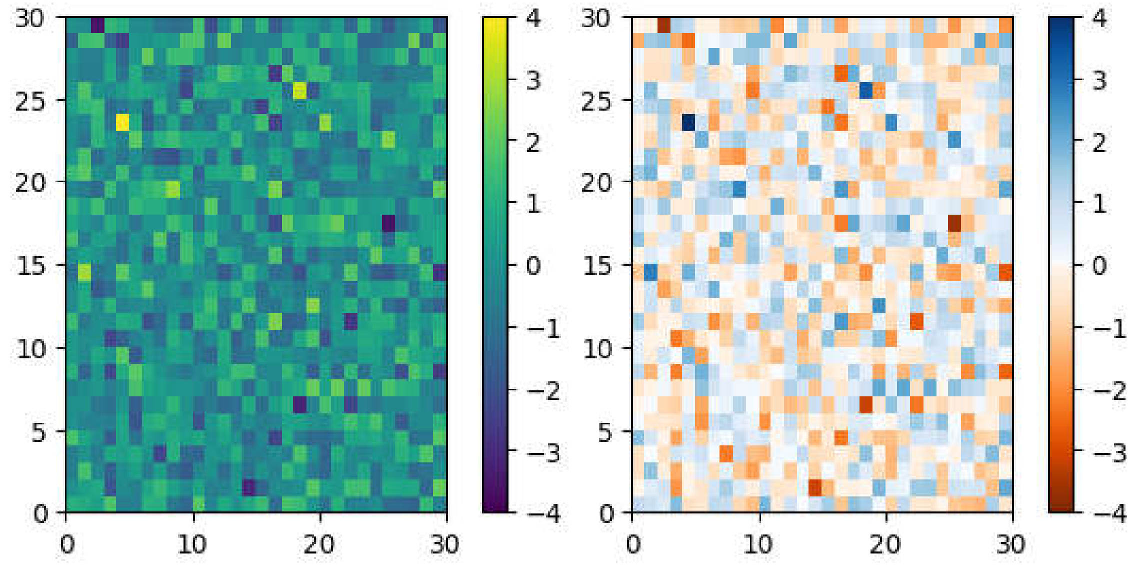
In [10]:

```
viridis = cm.get_cmap('viridis', 256)
newcolors = viridis(np.linspace(0, 1, 256))
pink = np.array([248/256, 24/256, 148/256, 1])
newcolors[:25, :] = pink
newcmp = ListedColormap(newcolors)
plot_examples([viridis, newcmp])
```



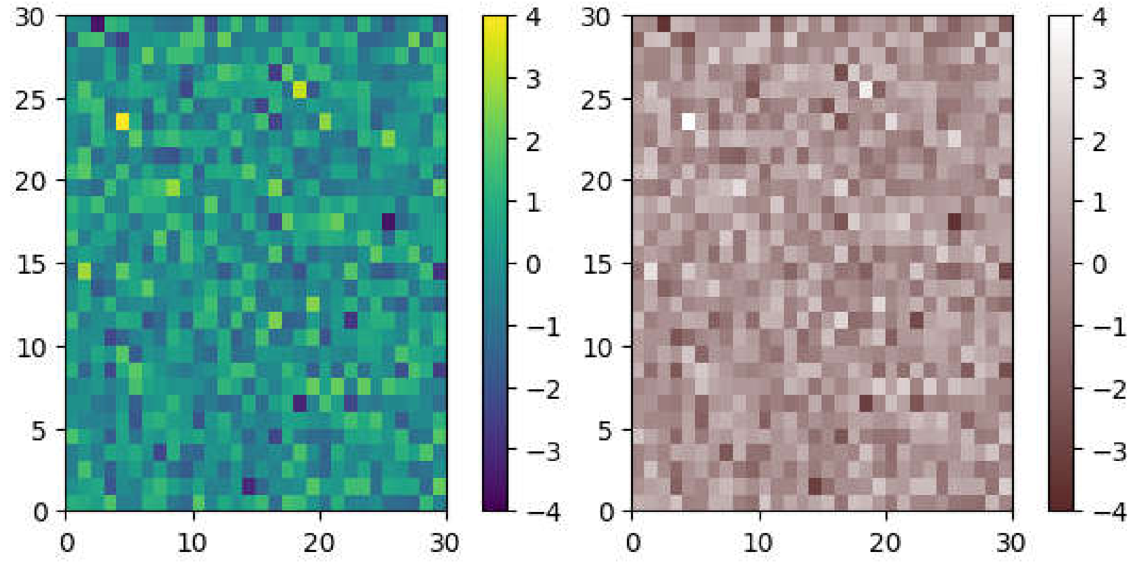
In [11]:

```
top = cm.get_cmap('Oranges_r', 128)
bottom = cm.get_cmap('Blues', 128)
newcolors = np.vstack((top(np.linspace(0, 1,
128)),
 bottom(np.linspace(0,
1, 128))))
newcmp = ListedColormap(newcolors,
name='OrangeBlue')
plot_examples([viridis, newcmp])
```



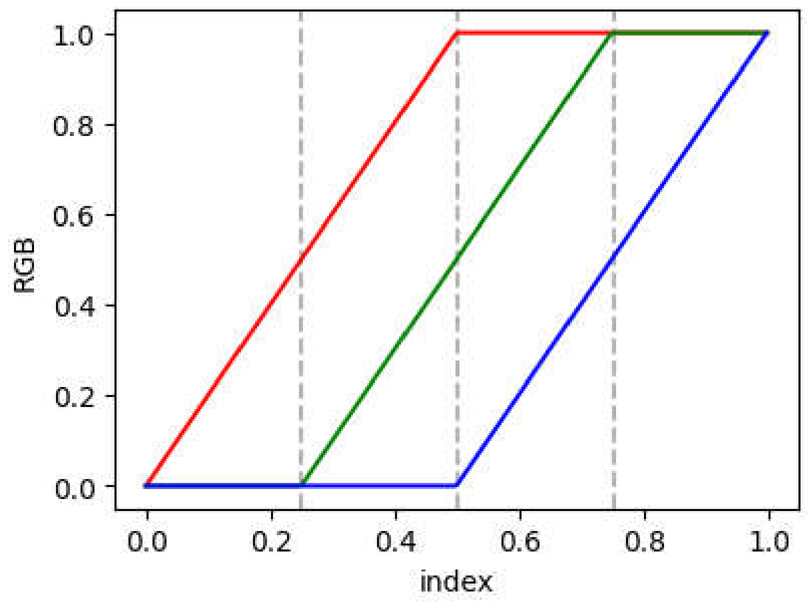
In [12]:

```
N = 256
vals = np.ones((N, 4))
vals[:, 0] = np.linspace(90/256, 1, N)
vals[:, 1] = np.linspace(40/256, 1, N)
vals[:, 2] = np.linspace(40/256, 1, N)
newcmp = ListedColormap(vals)
plot_examples([viridis, newcmp])
```



In [13]:

```
cdict = {'red': [[0.0, 0.0, 0.0], [0.5, 1.0, 1.0], [1.0, 1.0, 1.0]], 'green': [[0.0, 0.0,

def plot_linearmap(cdict):
    newcmp = LinearSegmentedColormap('testCmap', segmentdata=cdict, N=256)
    rgba = newcmp(np.linspace(0, 1, 256))
    fig, ax = plt.subplots(figsize=(4, 3), constrained_layout=True)
    col = ['r', 'g', 'b']
    for xx in [0.25, 0.5, 0.75]:
        ax.axvline(xx, color='0.7', linestyle='--')
    for i in range(3):
        ax.plot(np.arange(256)/256, rgba[:, i], color=col[i])
    ax.set_xlabel('index')
    ax.set_ylabel('RGB')
    plt.show()

plot_linearmap(cdict)
```

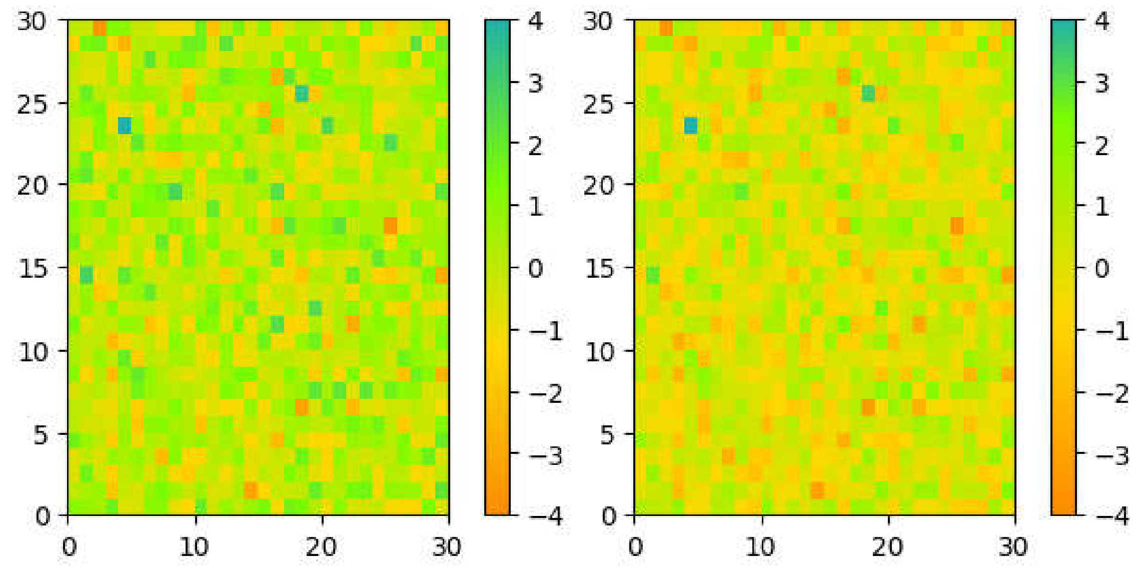


In [14]:

```
colors = ["darkorange", "gold", "lawngreen", "lightseagreen"]
cmap1 = LinearSegmentedColormap.from_list("mycmap", colors)
```

In [15]:

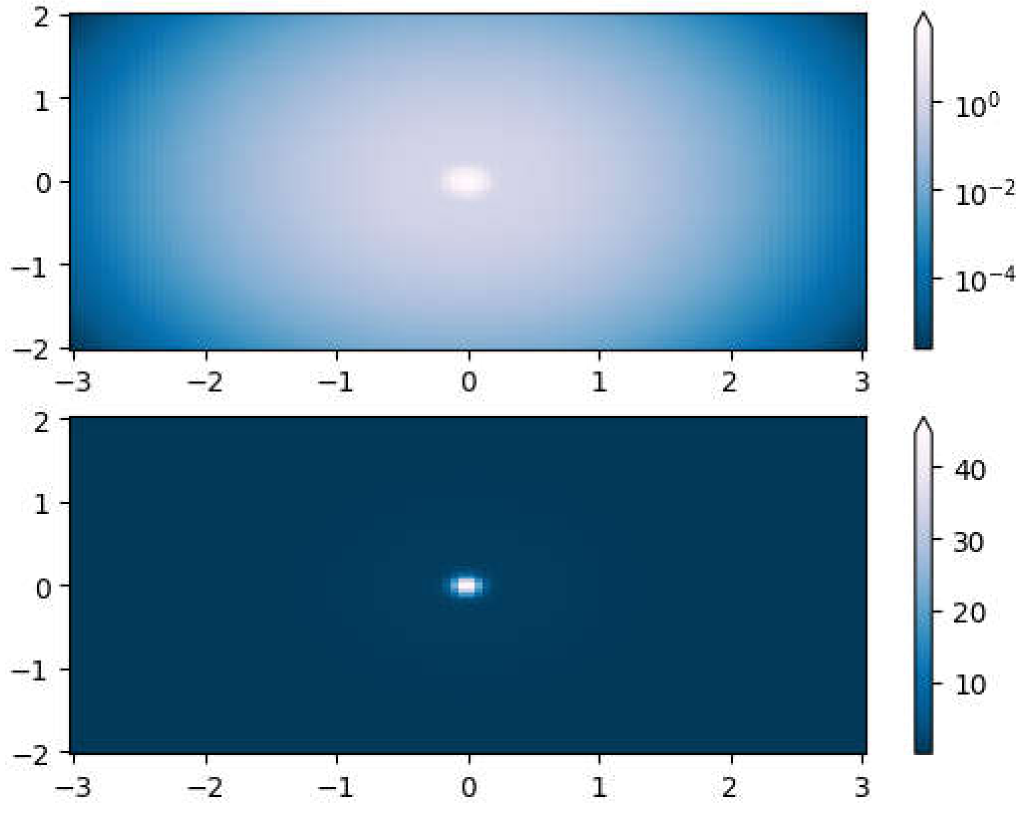
```
nodes = [0.0, 0.4, 0.8, 1.0]
cmap2 = LinearSegmentedColormap.from_list("mycmap", list(zip(nodes, colors)))
plot_examples([cmap1, cmap2])
```



2. Colormap Normalization

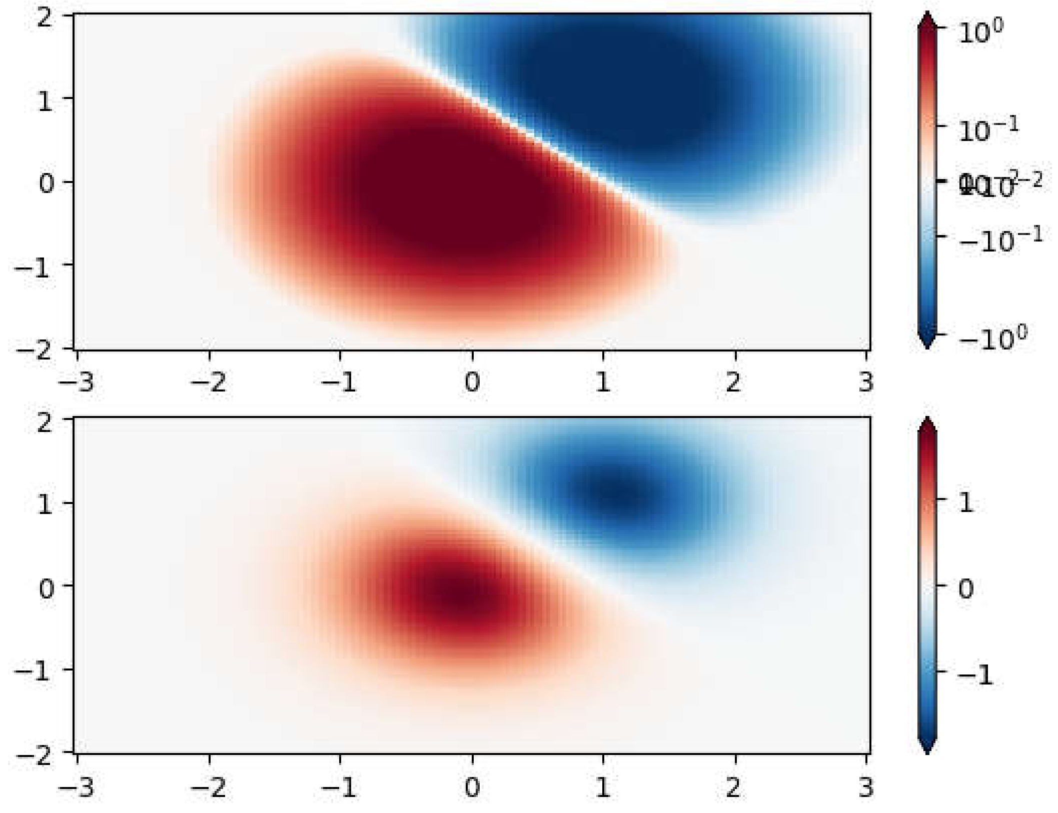
In [16]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import matplotlib.cbook as cbook
from matplotlib import cm

N = 100
X, Y = np.mgrid[-3:3:complex(0, N), -
2:2:complex(0, N)]
# Logarithmic
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X * 10)**2 - (Y * 10)**2)
Z = Z1 + 50 * Z2
fig, ax = plt.subplots(2, 1)
pcm = ax[0].pcolor(X, Y, Z,

norm=colors.LogNorm(vmin=Z.min(), vmax=Z.max()),cmap='PuBu_r', shading='auto')
fig.colorbar(pcm, ax=ax[0], extend='max')
pcm = ax[1].pcolor(X, Y, Z, cmap='PuBu_r', shading='auto')
fig.colorbar(pcm, ax=ax[1], extend='max')
plt.show()
```



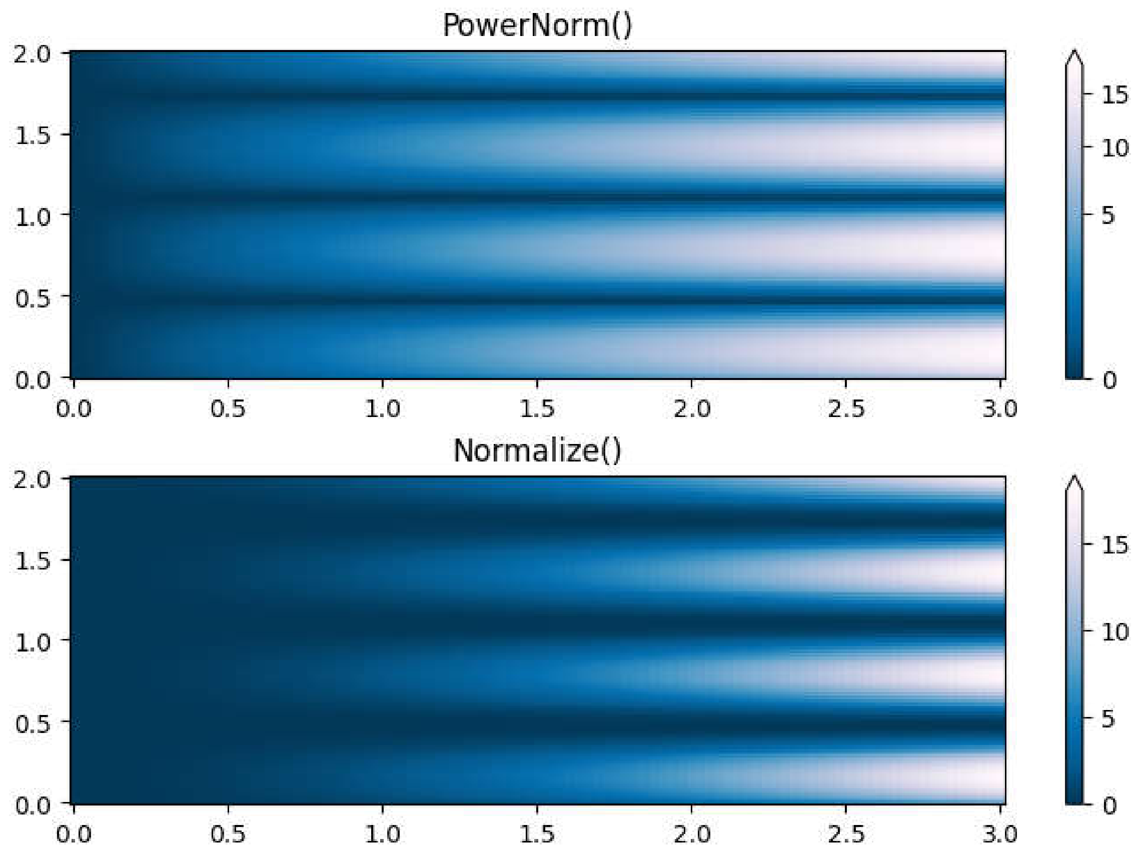
In [17]:

```
# Symmetric logarithmic
N = 100
X, Y = np.mgrid[-3:3:complex(0, N), -
2:2:complex(0, N)]
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X - 1)**2 - (Y - 1)**2)
Z = (Z1 - Z2) * 2
fig, ax = plt.subplots(2, 1)
pcm = ax[0].pcolormesh(X, Y, Z,

norm=colors.SymLogNorm(linthresh=0.03,
linscale=0.03,

vmin=-1.0, vmax=1.0, base=10),
 cmap='RdBu_r',
shading='auto')
fig.colorbar(pcm, ax=ax[0], extend='both')
pcm = ax[1].pcolormesh(X, Y, Z, cmap='RdBu_r',
vmin=-np.max(Z), shading='auto')
fig.colorbar(pcm, ax=ax[1], extend='both')
plt.show()
```



In [18]:

```
N = 100
X, Y = np.mgrid[0:3:complex(0, N),
0:2:complex(0, N)]
Z1 = (1 + np.sin(Y * 10.)) * X**2
fig, ax = plt.subplots(2, 1,
constrained_layout=True)
pcm = ax[0].pcolormesh(X, Y, Z1,
norm=colors.PowerNorm(gamma=0.5),
 cmap='PuBu_r',
shading='auto')
fig.colorbar(pcm, ax=ax[0], extend='max')
ax[0].set_title('PowerNorm()')
pcm = ax[1].pcolormesh(X, Y, Z1,
cmap='PuBu_r', shading='auto')
fig.colorbar(pcm, ax=ax[1], extend='max')
ax[1].set_title('Normalize()')
plt.show()
```



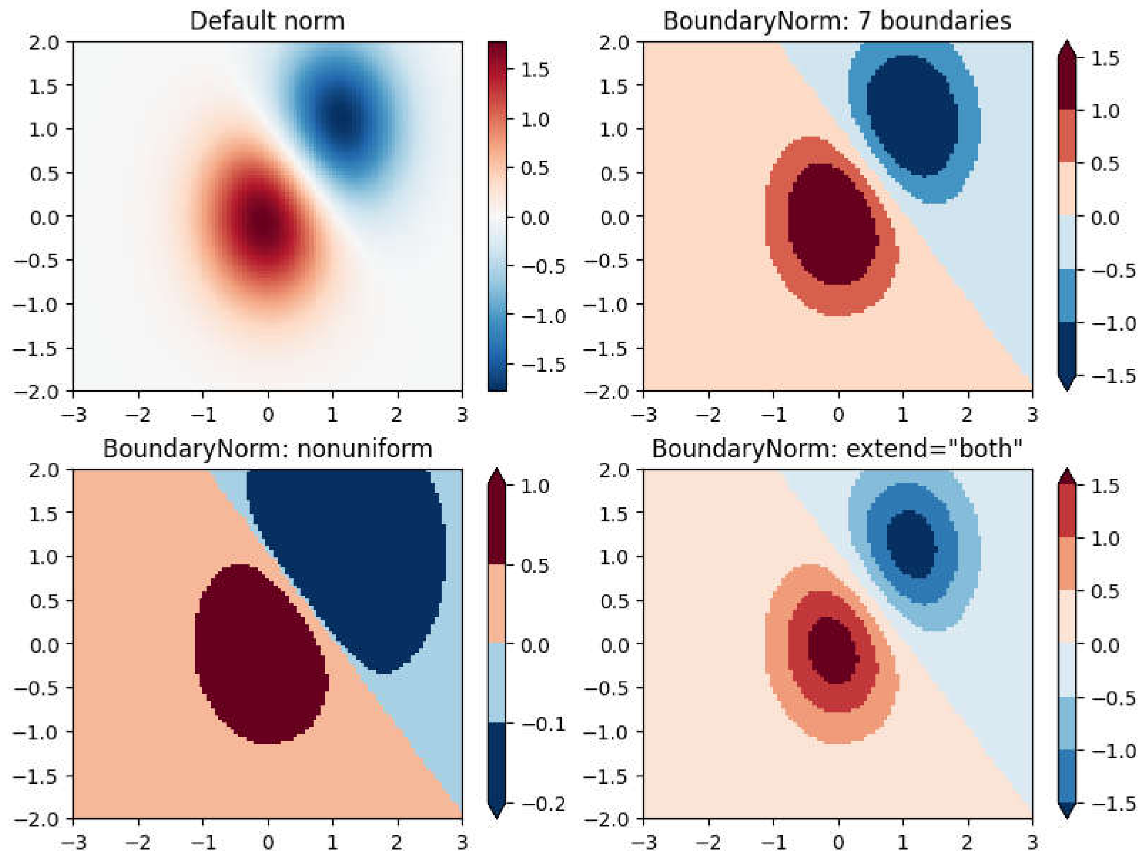
In [19]:

```
import matplotlib.colors as colors
bounds = np.array([-0.25, -0.125, 0, 0.5, 1])
norm = colors.BoundaryNorm(boundaries=bounds, ncolors=4)
print(norm([-0.2, -0.15, -0.02, 0.3, 0.8, 0.99]))
```

```
[0 0 1 2 3 3]
```

In [20]:

```
# Discrete bounds
N = 100
X, Y = np.meshgrid(np.linspace(-3, 3, N), np.linspace(-2, 2, N))
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X - 1)**2 - (Y - 1)**2)
Z = ((Z1 - Z2) * 2)[:-1, :-1]
fig, ax = plt.subplots(2, 2, figsize=(8, 6), constrained_layout=True)

ax = ax.flatten()

# Default norm:
pcm = ax[0].pcolormesh(X, Y, Z, cmap='RdBu_r')
fig.colorbar(pcm, ax=ax[0], orientation='vertical')
ax[0].set_title('Default norm')

# Even bounds give a contour-like effect:
bounds = np.linspace(-1.5, 1.5, 7)
norm = colors.BoundaryNorm(boundaries=bounds, ncolors=256)
pcm = ax[1].pcolormesh(X, Y, Z, norm=norm, cmap='RdBu_r')
fig.colorbar(pcm, ax=ax[1], extend='both', orientation='vertical')
ax[1].set_title('BoundaryNorm: 7 boundaries')

# Bounds may be unevenly spaced:
bounds = np.array([-0.2, -0.1, 0, 0.5, 1])
norm = colors.BoundaryNorm(boundaries=bounds,ncolors=256)
pcm = ax[2].pcolormesh(X, Y, Z, norm=norm, cmap='RdBu_r')
fig.colorbar(pcm, ax=ax[2], extend='both', orientation='vertical')
ax[2].set_title('BoundaryNorm: nonuniform')

# With out-of-bounds colors:
bounds = np.linspace(-1.5, 1.5, 7)
norm = colors.BoundaryNorm(boundaries=bounds, ncolors=256, extend='both')
pcm = ax[3].pcolormesh(X, Y, Z, norm=norm, cmap='RdBu_r')

# The colorbar inherits the "extend" argument from BoundaryNorm.
fig.colorbar(pcm, ax=ax[3], orientation='vertical')
ax[3].set_title('BoundaryNorm: extend="both"')
plt.show()
```



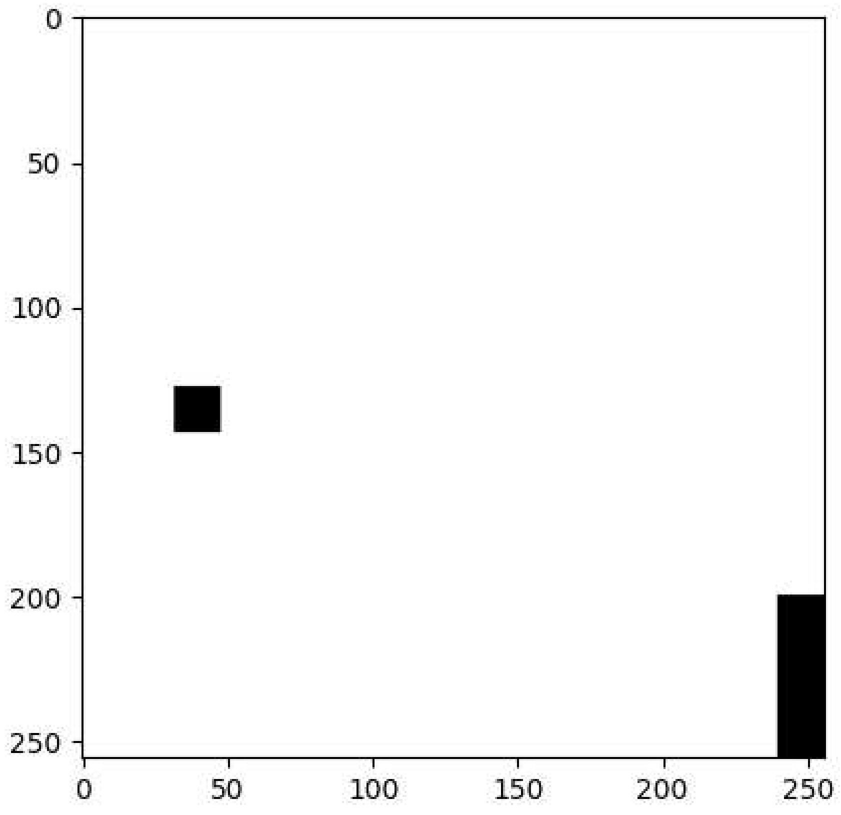
3. Transform Jarak

In [21]:

```
import numpy as np
import mahotas

f = np.ones((256,256), bool)
f[200:,240:] = False
f[128:144,32:48] = False
```

In [22]:

```
from pylab import imshow, gray, show
import numpy as np
f = np.ones((256,256), bool)
f[200:,240:] = False
f[128:144,32:48] = False
gray()
imshow(f)
show()
```

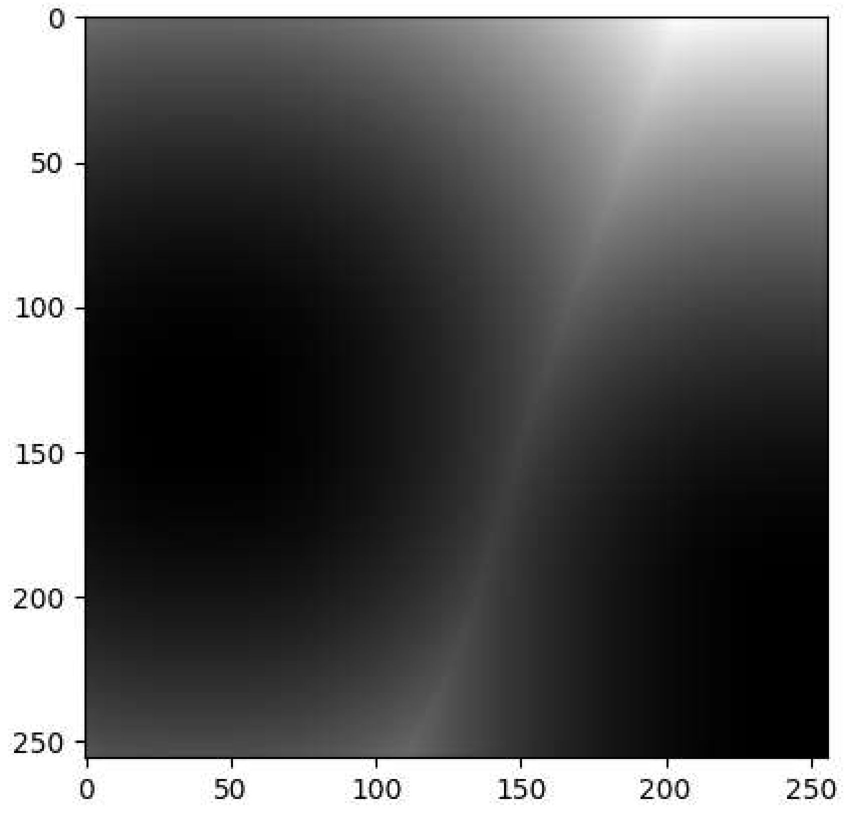


In [23]:

```
import mahotas
dmap = mahotas.distance(f)
```

In [24]:

```
from __future__ import print_function
import pylab as p
import numpy as np
import mahotas

f = np.ones((256,256), bool)
f[200:,240:] = False
f[128:144,32:48] = False
dmap = mahotas.distance(f)
p.imshow(dmap)
p.show()
```



4. Konversi ruang warna

In [25]:

```
import mahotas as mh
lena = mh.demos.load('lena')
print(lena.shape)
```

```
(512, 512, 3)
```

In [26]:

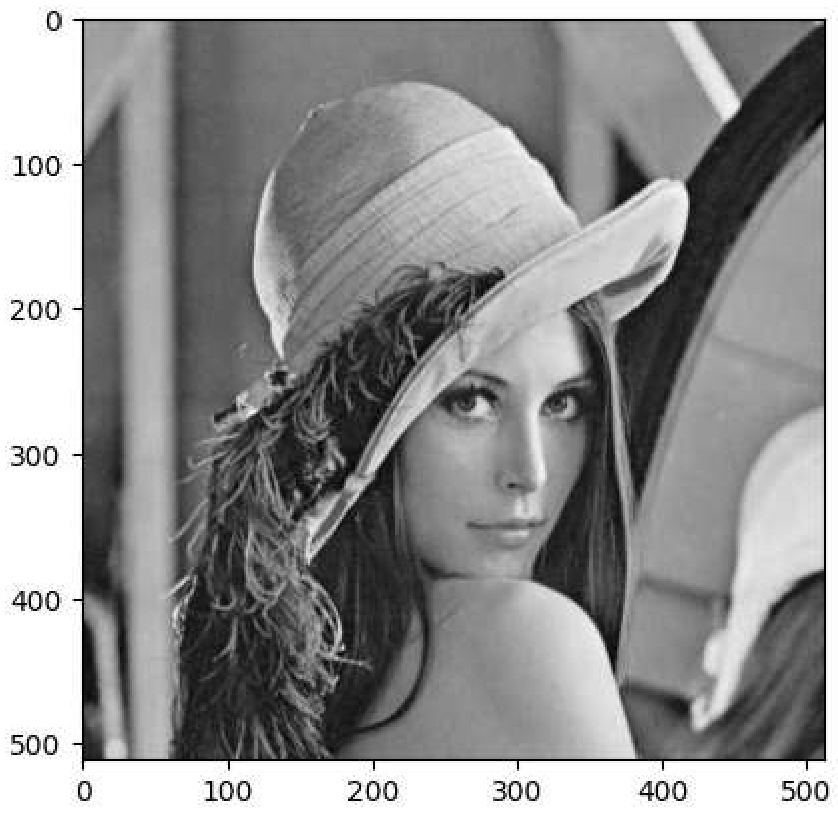
```
import mahotas as mh
lena = mh.demos.load('lena')
lenag = mh.colors.rgb2grey(lena)
```

In [27]:

```
from pylab import imshow
import mahotas as mh
lena = mh.demos.load('lena')
lenag = mh.colors.rgb2grey(lena)
imshow(lenag)
```

Out[27]:

```
<matplotlib.image.AxesImage at 0x1a27129f160>
```

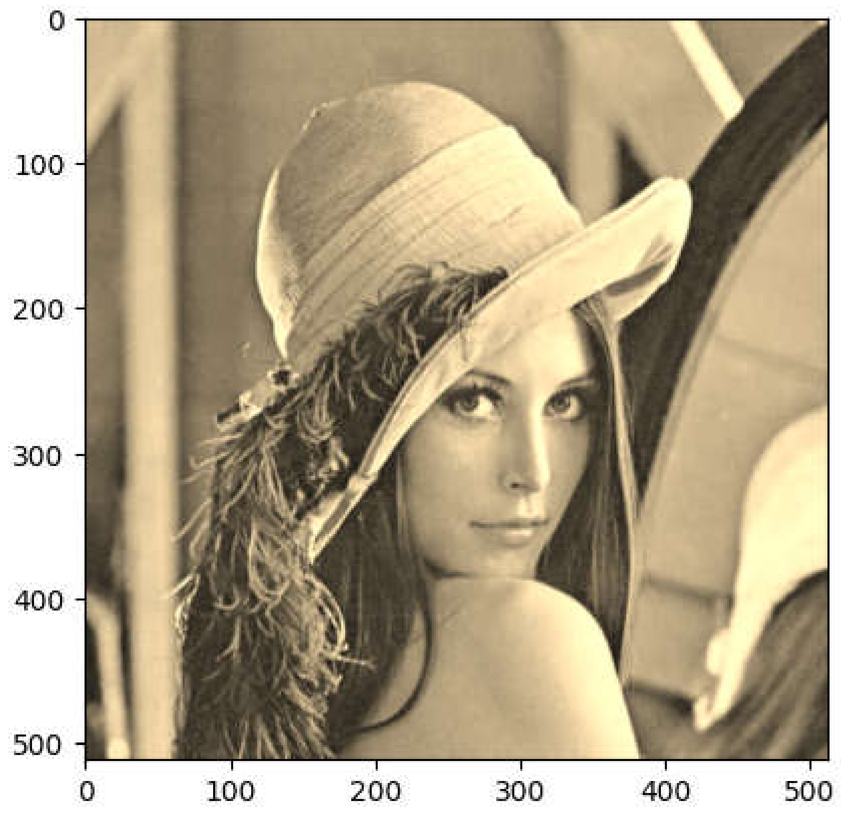


In [28]:

```
lenas = mh.colors.rgb2sepia(lena)
from pylab import imshow
import mahotas as mh
lena = mh.demos.load('lena')
lenas = mh.colors.rgb2sepia(lena)
imshow(lenas)
```

Out[28]:

```
<matplotlib.image.AxesImage at 0x1a271712370>
```



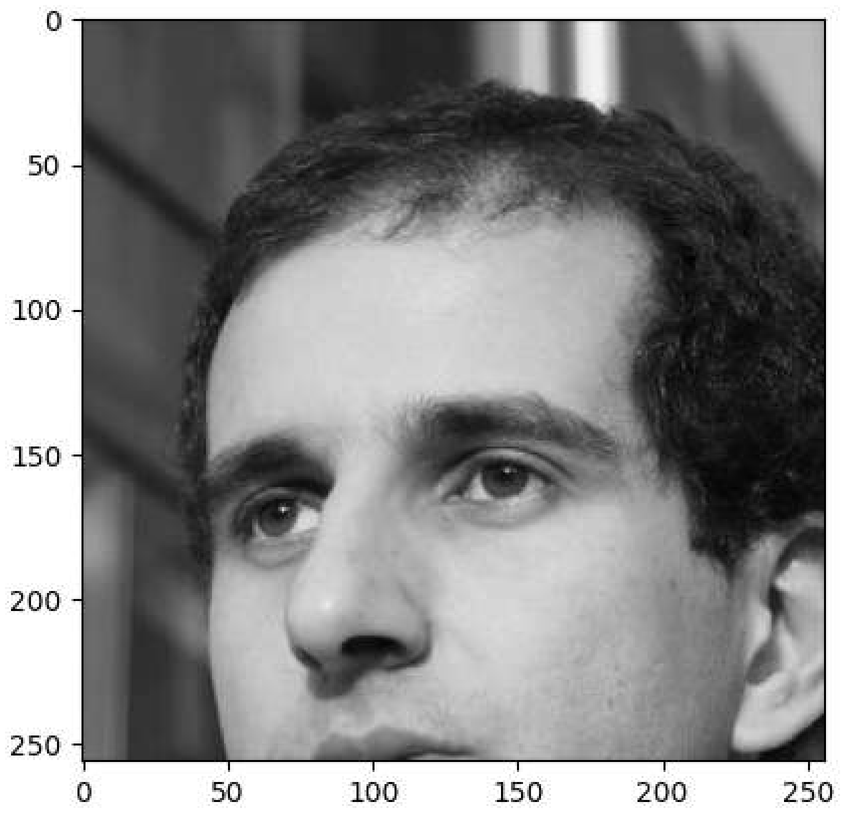
BAB 10

1. penggunaan mahotas.thresholding

In [29]:

```
import numpy as np
import mahotas
import mahotas.demos
from mahotas.thresholding import soft_threshold
from matplotlib import pyplot as plt
from os import path

f = mahotas.demos.load('luispedro',
as_grey=True)
f = f[:256,:256]
plt.gray()

# Show the data:
print("Fraction of zeros in original image: {0}".format(np.mean(f==0)))

plt.imshow(f)
plt.show()
```

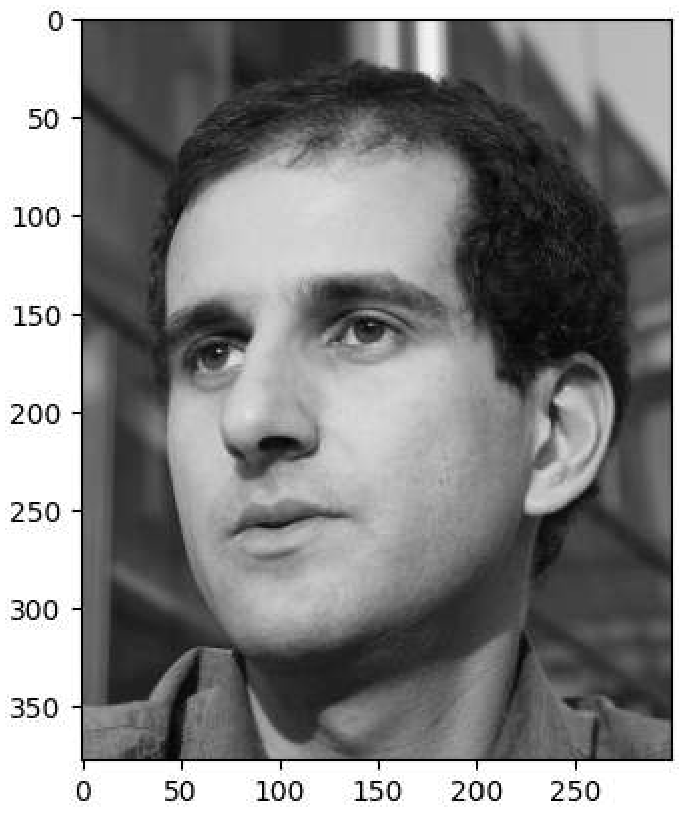
```
Fraction of zeros in original image: 0.0
```



2. mahotas.otsu

In [30]:

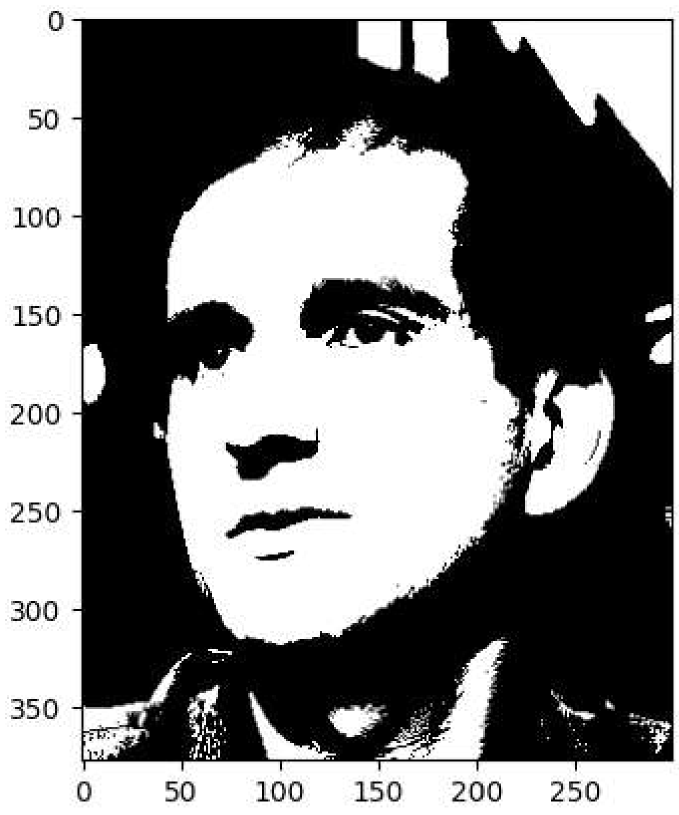
```
import mahotas
import mahotas.demos
import numpy as np
from pylab import imshow, gray, show
from os import path

photo = mahotas.demos.load('luispedro',
as_grey=True)
photo = photo.astype(np.uint8)
gray()
imshow(photo)
show()
```



In [31]:

```
T_otsu = mahotas.otsu(photo)
print(T_otsu)
imshow(photo > T_otsu)
show()
```

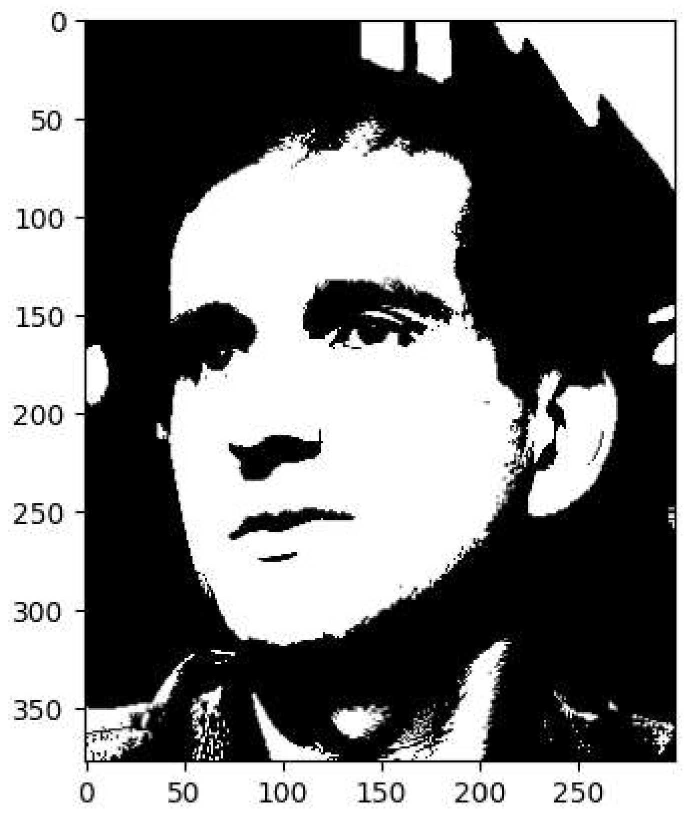
```
116
```



In [32]:

```
import mahotas
import mahotas.demos
import numpy as np
from pylab import imshow, gray, show
from os import path

photo = mahotas.demos.load('luispedro',
as_grey=True)
photo = photo.astype(np.uint8)
T_otsu = mahotas.otsu(photo)
print(T_otsu)
gray()
imshow(photo > T_otsu)
show()
```

```
116
```



In [33]:

```
T_rc = mahotas.rc(photo)
print(T_rc)
imshow(photo > T_rc)
show()
```

```
115.95073523165897
```