**Random Walker segmentation**

**(1) open one image on histogram filter :**

from skimage import io , img\_as\_float

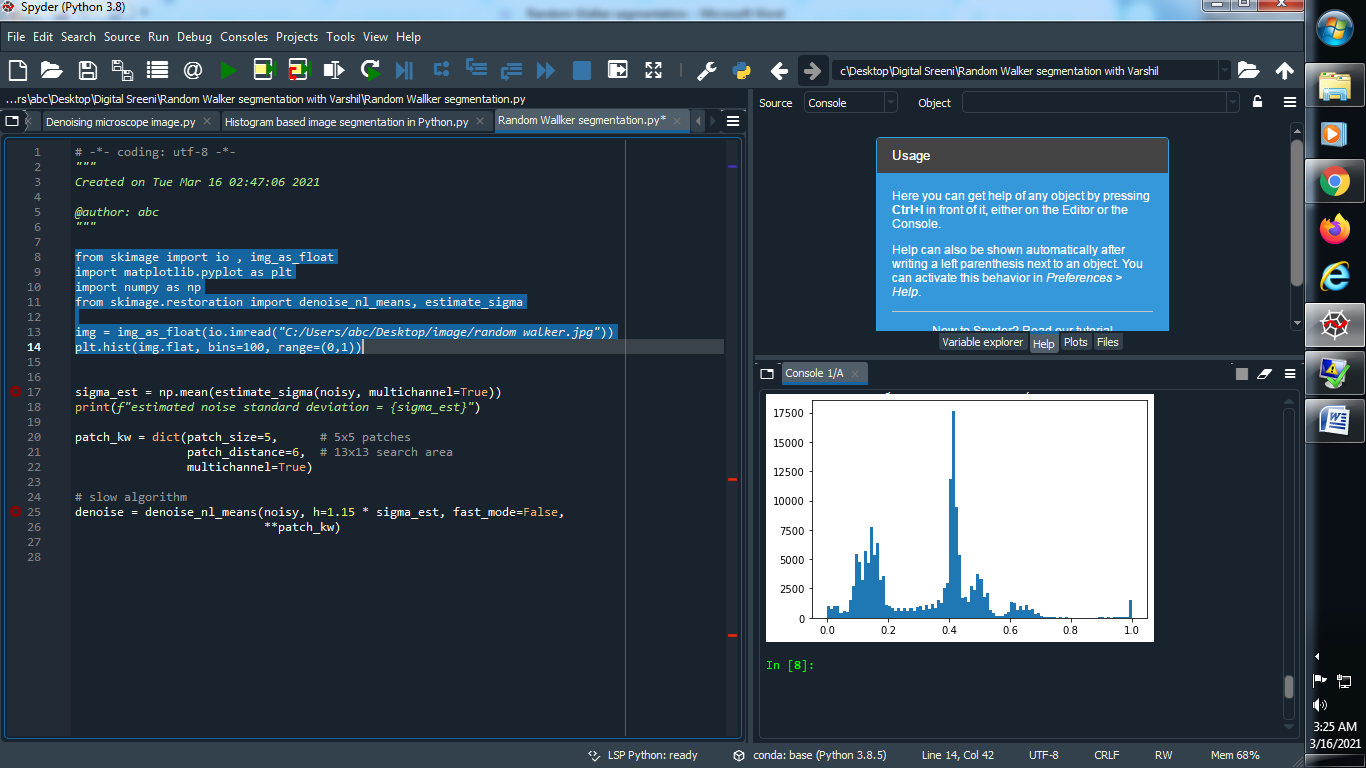
import matplotlib.pyplot as plt

import numpy as np

img = img\_as\_float(io.imread("C:/Users/abc/Desktop/image/random walker.jpg"))

plt.hist(img.flat, bins=100, range=(0,1))

→ Output :

****

**(2) Now open this image using non-local filter of denoise image :**

from skimage import io , img\_as\_float

import matplotlib.pyplot as plt

import numpy as np

from skimage.restoration import denoise\_nl\_means, estimate\_sigma

img = img\_as\_float(io.imread("C:/Users/abc/Desktop/image/random walker.jpg"))

sigma\_est = np.mean(estimate\_sigma(img, multichannel=True))

print(f"estimated noise standard deviation = {sigma\_est}")

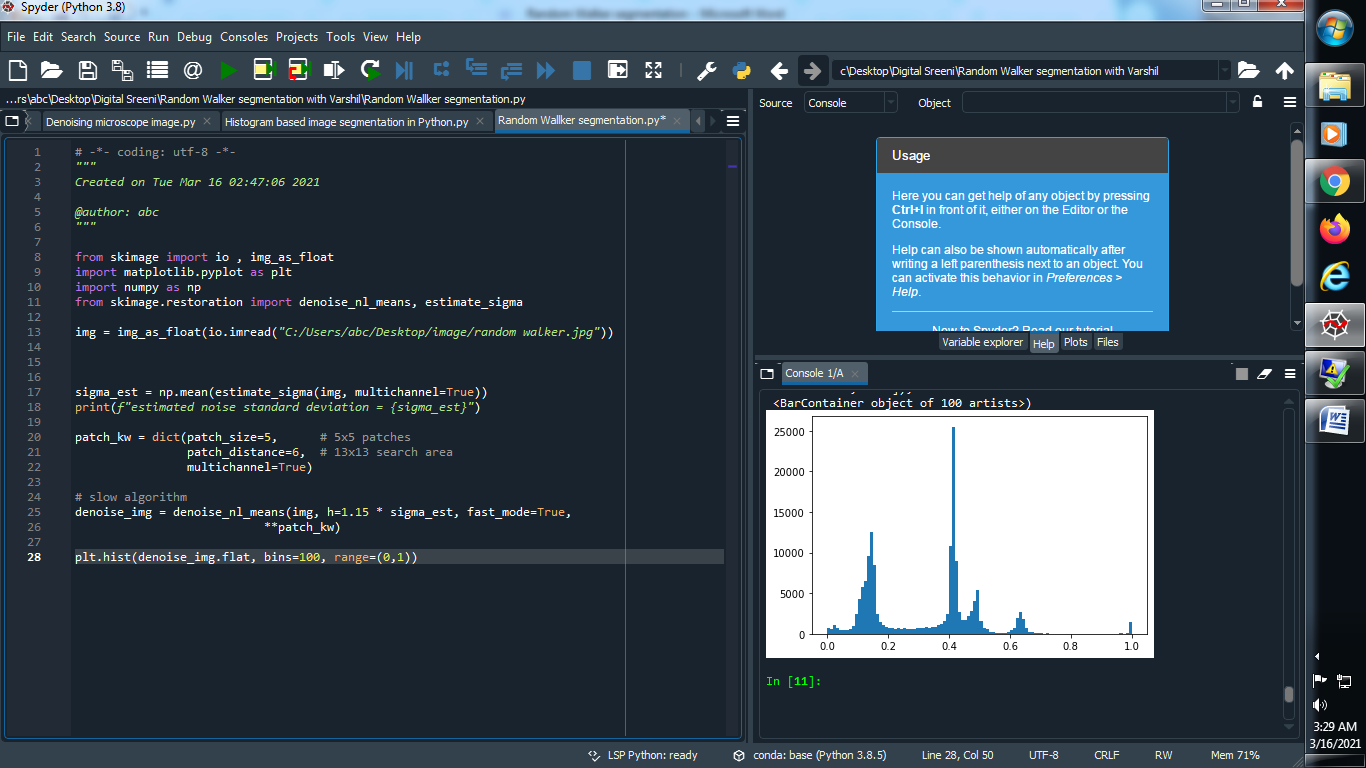
patch\_kw = dict(patch\_size=5, patch\_distance=6, multichannel=True)

denoise\_img = denoise\_nl\_means(img, h=1.15 \* sigma\_est, fast\_mode=True,

\*\*patch\_kw)

plt.hist(denoise\_img.flat, bins=100, range=(0,1))

→ Output :



**(3) Equal Filter of histogram :**

from skimage import io , img\_as\_float

import matplotlib.pyplot as plt

import numpy as np

from skimage.restoration import denoise\_nl\_means, estimate\_sigma

img = img\_as\_float(io.imread("C:/Users/abc/Desktop/image/random walker.jpg"))

sigma\_est = np.mean(estimate\_sigma(img, multichannel=True))

print(f"estimated noise standard deviation = {sigma\_est}")

patch\_kw = dict(patch\_size=5, patch\_distance=6, multichannel=True)

denoise\_img = denoise\_nl\_means(img, h=1.15 \* sigma\_est, fast\_mode=True,

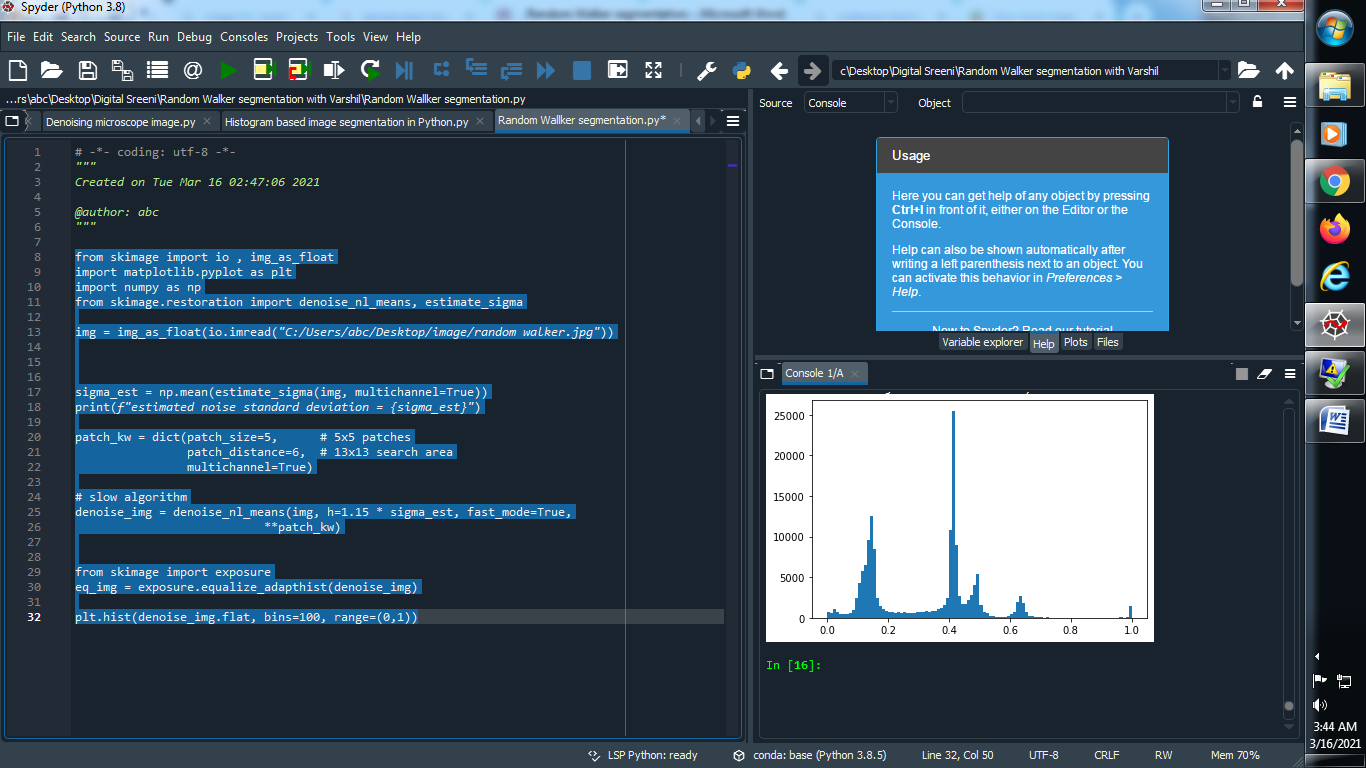
\*\*patch\_kw)

from skimage import exposure

eq\_img = exposure.equalize\_adapthist(denoise\_img)

plt.hist(denoise\_img.flat, bins=100, range=(0,1))

→ Output :



(4) show the image :

from skimage import io , img\_as\_float

import matplotlib.pyplot as plt

import numpy as np

from skimage.restoration import denoise\_nl\_means, estimate\_sigma

img = img\_as\_float(io.imread("C:/Users/abc/Desktop/image/random walker.jpg"))

sigma\_est = np.mean(estimate\_sigma(img, multichannel=True))

print(f"estimated noise standard deviation = {sigma\_est}")

patch\_kw = dict(patch\_size=5,

patch\_distance=6,

multichannel=True)

denoise\_img = denoise\_nl\_means(img, h=1.15 \* sigma\_est, fast\_mode=True,

\*\*patch\_kw)

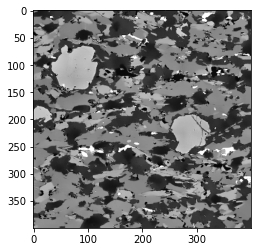
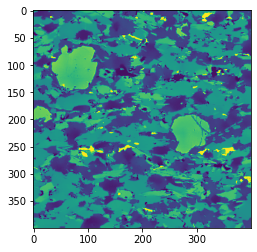
from skimage import exposure

eq\_img = exposure.equalize\_adapthist(denoise\_img)

plt.imshow(eq\_img,cmap='gray')

→ Output :

**Simple image : Gray image :**



**(5) Markers in images :**

from skimage import io , img\_as\_float

import matplotlib.pyplot as plt

import numpy as np

from skimage.restoration import denoise\_nl\_means, estimate\_sigma

img = img\_as\_float(io.imread("C:/Users/abc/Desktop/image/random walker.jpg"))

sigma\_est = np.mean(estimate\_sigma(img, multichannel=True))

print(f"estimated noise standard deviation = {sigma\_est}")

patch\_kw = dict(patch\_size=5, # 5x5 patches

patch\_distance=6, # 13x13 search area

multichannel=True)

denoise\_img = denoise\_nl\_means(img, h=1.15 \* sigma\_est, fast\_mode=True,

\*\*patch\_kw)

from skimage import exposure

eq\_img = exposure.equalize\_adapthist(denoise\_img)

plt.imshow(eq\_img,cmap='gray')

#plt.hist(denoise\_img.flat, bins=100, range=(0,1))

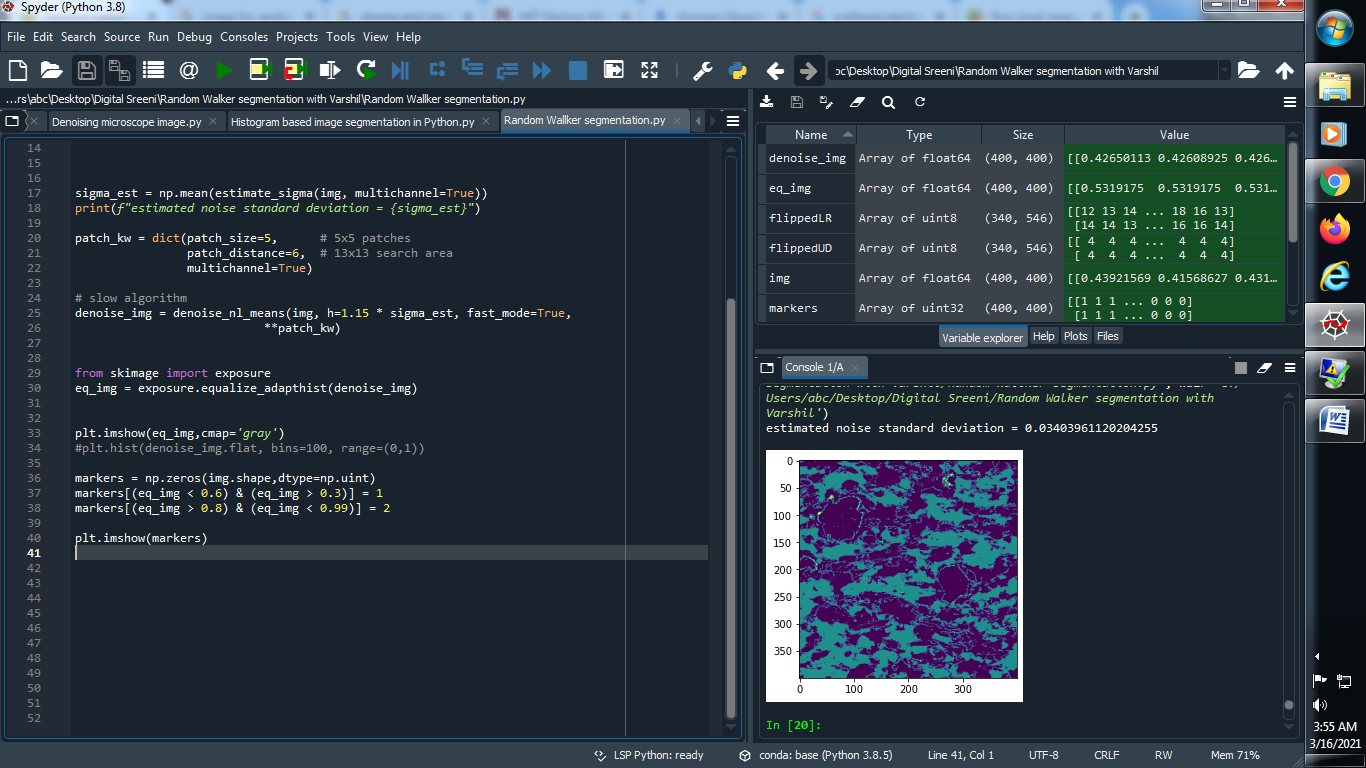
**markers = np.zeros(img.shape,dtype=np.uint)**

**markers[(eq\_img < 0.6) & (eq\_img > 0.3)] = 1**

**markers[(eq\_img > 0.8) & (eq\_img < 0.99)] = 2**

**plt.imshow(markers)**

→ Output :



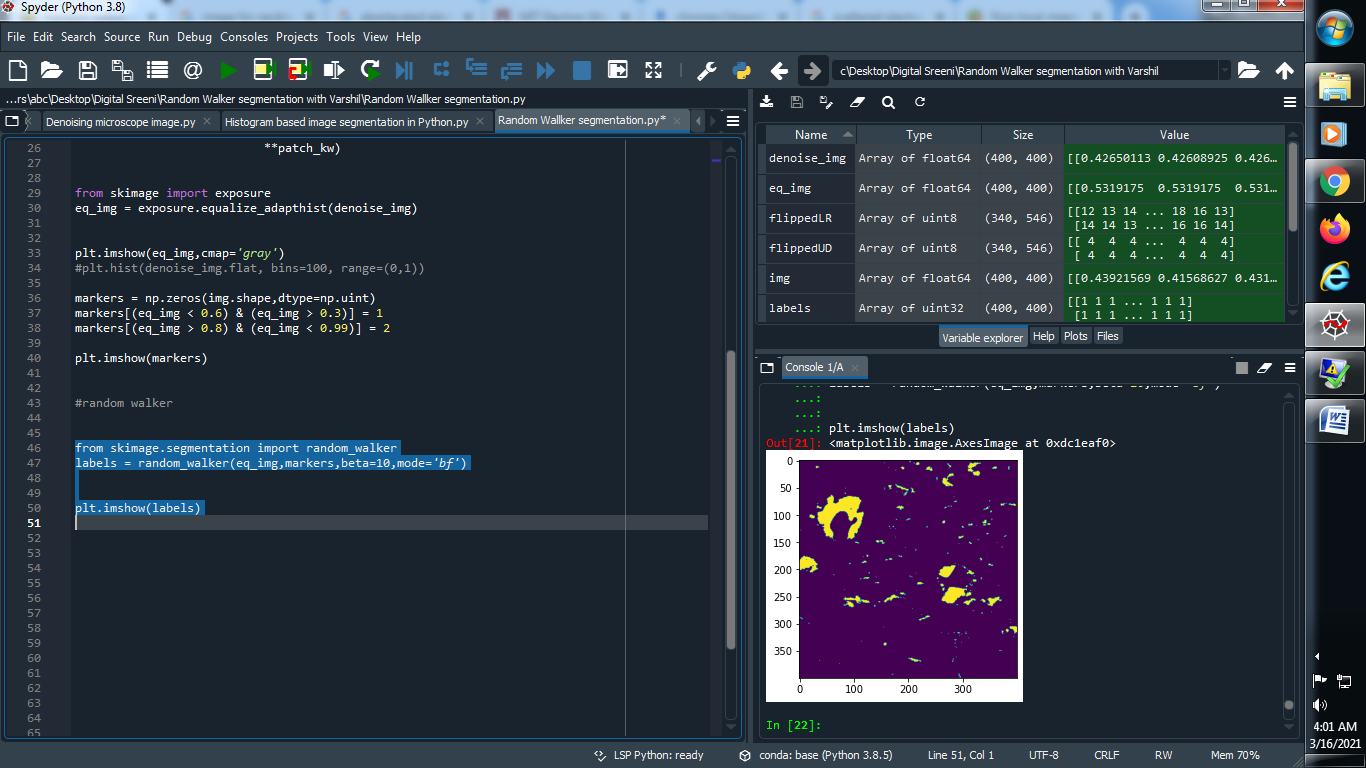
**(6) Random walker image :**

from skimage.segmentation import random\_walker

labels = random\_walker(eq\_img,markers,beta=10,mode='bf')

plt.imshow(labels)

**→ Output :**



**(7) Segmentation in image :**

seg1 = (labels == 1)

seg2 = (labels == 2)

all\_segments = np.zeros((eq\_img.shape[0],eq\_img.shape[1],3))

all\_segments[seg1] = (1,0,0)

all\_segments[seg2] = (0,1,0)

plt.imshow(all\_segments)

→ Output :

