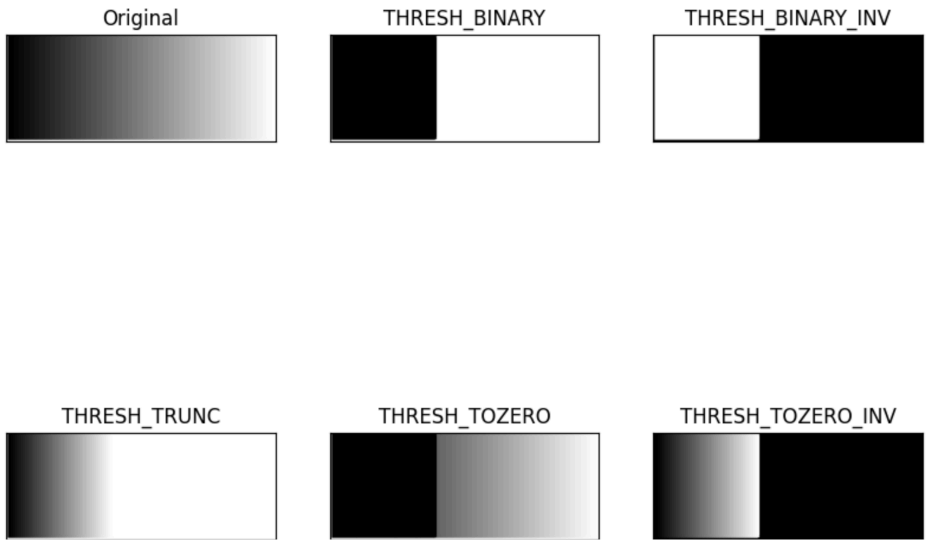
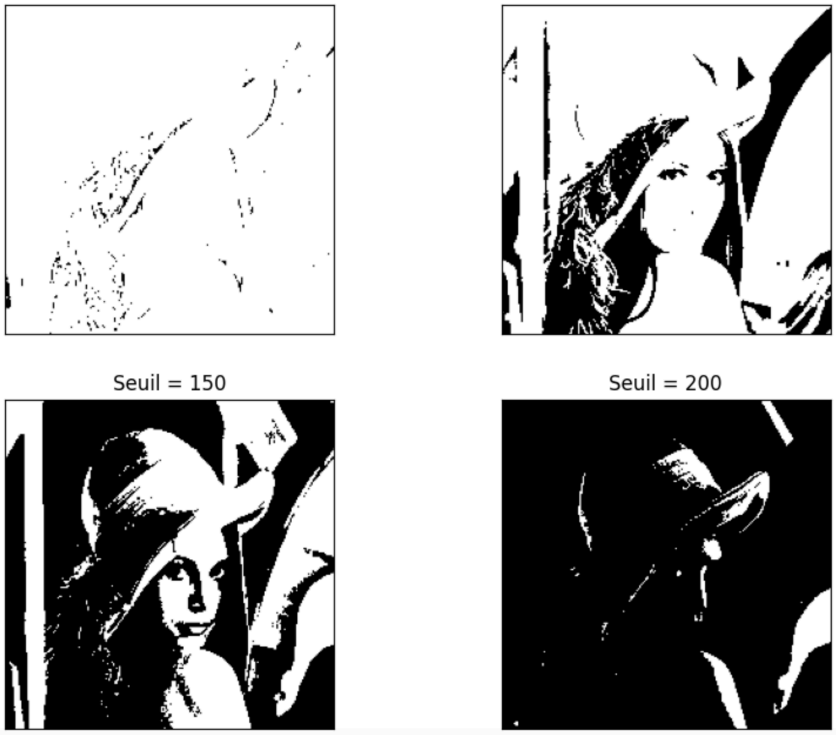
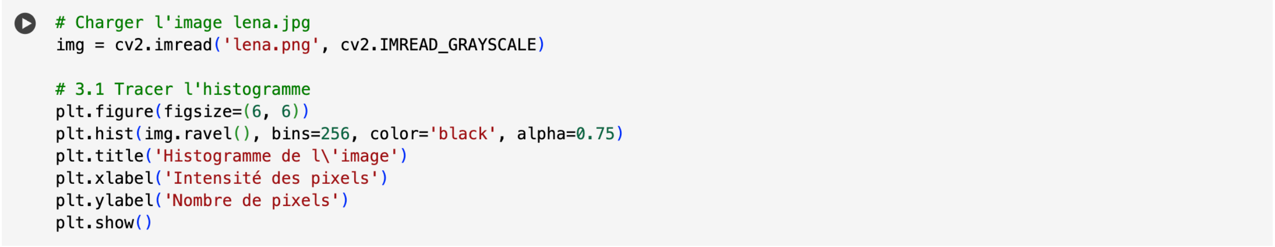
```
[14] import cv2
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[15] # Partie 1 : Mono-seuillage
     # 1.1 Charger l'image en niveaux de gris
     img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

     # Appliquer les différents types de seuillage proposés
     seuil = 100
     _, th1 = cv2.threshold(img, seuil, 255, cv2.THRESH_BINARY)
     _, th2 = cv2.threshold(img, seuil, 255, cv2.THRESH_BINARY_INV)
     _, th3 = cv2.threshold(img, seuil, 255, cv2.THRESH_TRUNC)
     _, th4 = cv2.threshold(img, seuil, 255, cv2.THRESH_TOZERO)
     _, th5 = cv2.threshold(img, seuil, 255, cv2.THRESH_TOZERO_INV)

     # Affichage des résultats
     titles = ['Original', 'THRESH_BINARY', 'THRESH_BINARY_INV', 'THRESH_TRUNC', 'THRESH_TOZERO', 'THRESH_TOZERO_INV']
     images = [img, th1, th2, th3, th4, th5]

     plt.figure(figsize=(10, 8))
     for i in range(6):
         plt.subplot(2, 3, i+1)
         plt.imshow(images[i], cmap='gray')
         plt.title(titles[i])
         plt.xticks([]), plt.yticks([])
     plt.show()
```

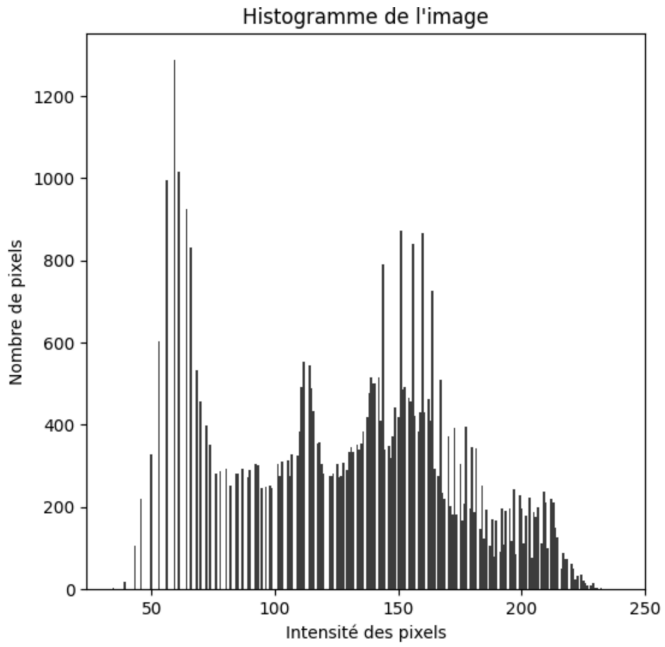


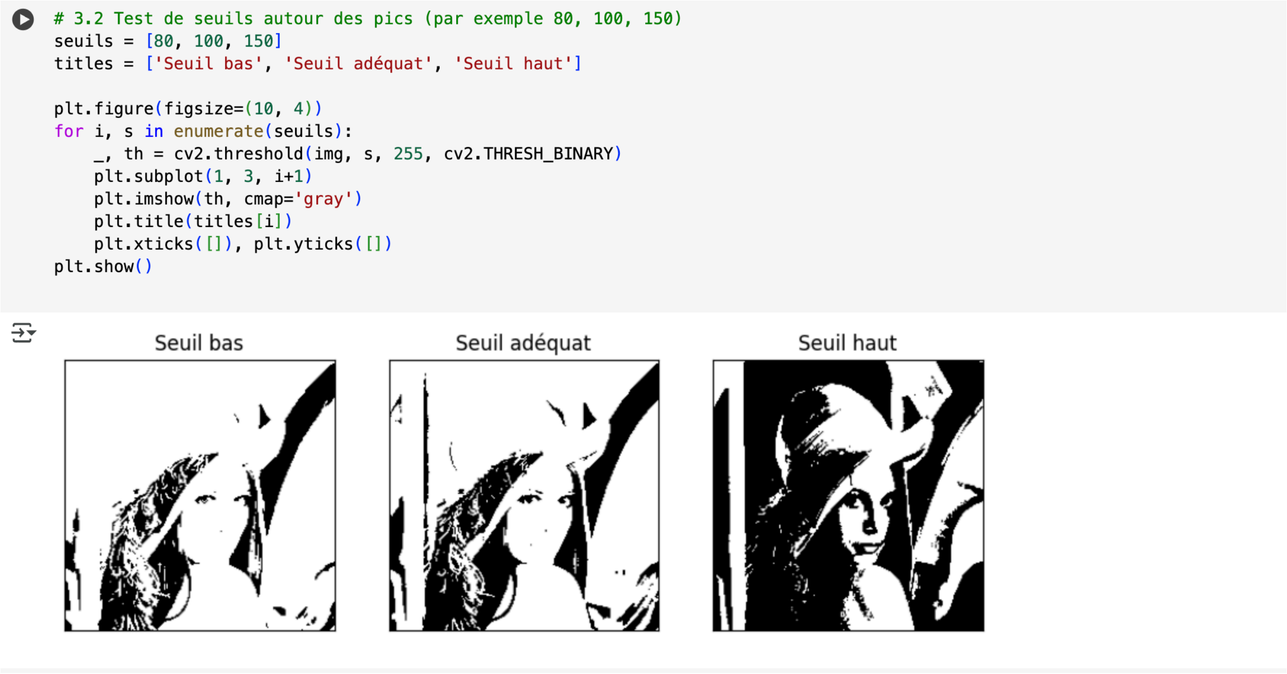
```
# 1.2 Tester plusieurs valeurs de seuil sur lena.jpg
img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)
seuils = [50, 100, 150, 200]

plt.figure(figsize=(10, 8))
for i, s in enumerate(seuils):
    _, th = cv2.threshold(img, s, 255, cv2.THRESH_BINARY)
    plt.subplot(2, 2, i+1)
    plt.imshow(th, cmap='gray')
    plt.title(f"Seuil = {s}")
    plt.xticks([]), plt.yticks([])
plt.show()
```



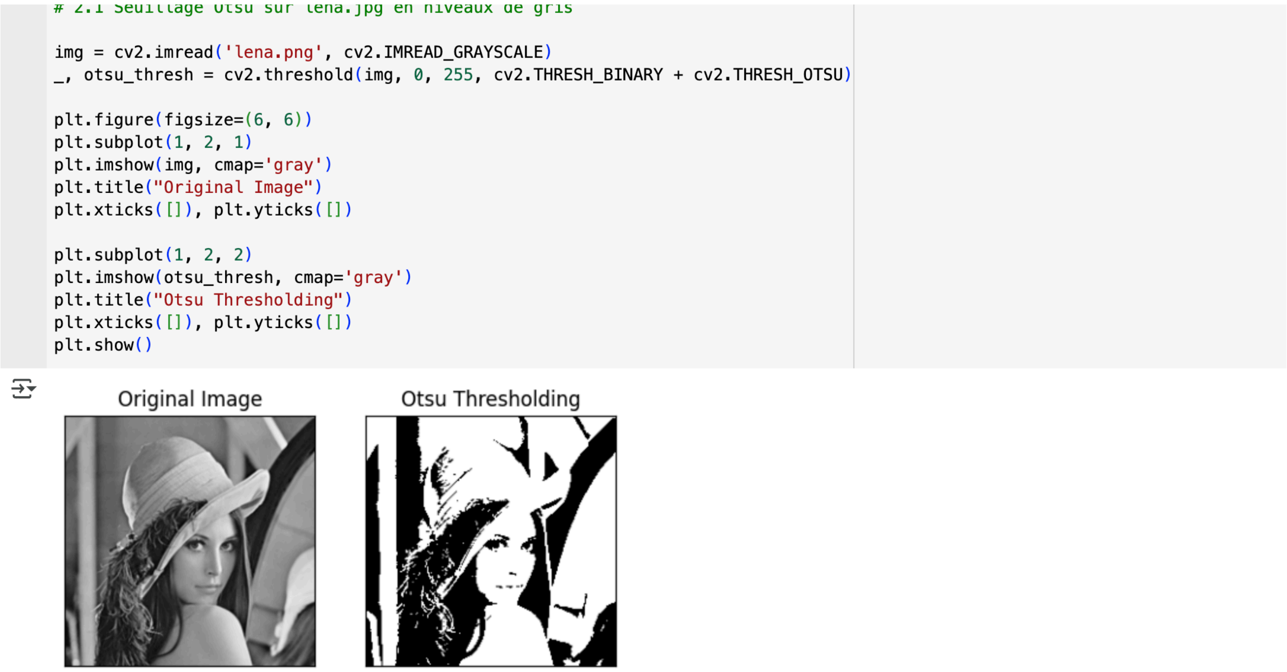


```
# Charger l'image lena.jpg
img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)

# 3.1 Tracer l'histogramme
plt.figure(figsize=(6, 6))
plt.hist(img.ravel(), bins=256, color='black', alpha=0.75)
plt.title('Histogramme de l\'image')
plt.xlabel('Intensité des pixels')
plt.ylabel('Nombre de pixels')
plt.show()
```

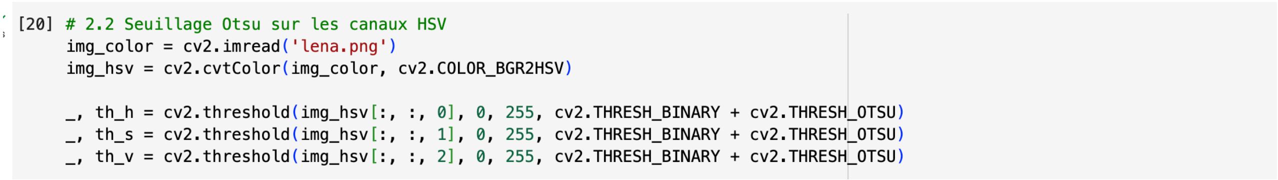




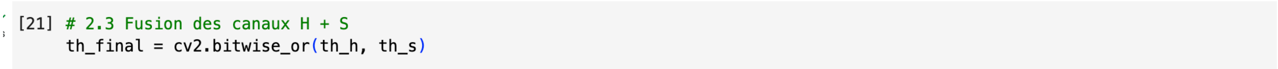
```
# 3.2 Test de seuils autour des pics (par exemple 80, 100, 150)
seuils = [80, 100, 150]
titles = ['Seuil bas', 'Seuil adéquat', 'Seuil haut']

plt.figure(figsize=(10, 4))
for i, s in enumerate(seuils):
    _, th = cv2.threshold(img, s, 255, cv2.THRESH_BINARY)
    plt.subplot(1, 3, i+1)
    plt.imshow(th, cmap='gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```



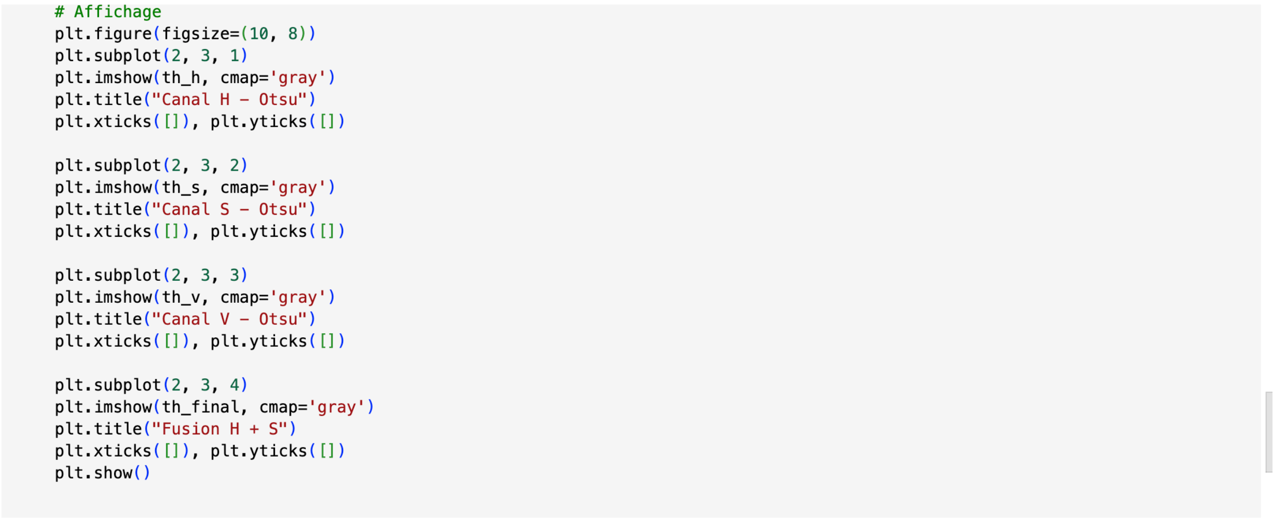
```
# 2.1 Seuillage Otsu sur lena.jpg en niveaux de gris

img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)
_, otsu_thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

plt.figure(figsize=(6, 6))
plt.subplot(1, 2, 1)
plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.xticks([]), plt.yticks([])

plt.subplot(1, 2, 2)
plt.imshow(otsu_thresh, cmap='gray')
plt.title("Otsu Thresholding")
plt.xticks([]), plt.yticks([])
plt.show()
```



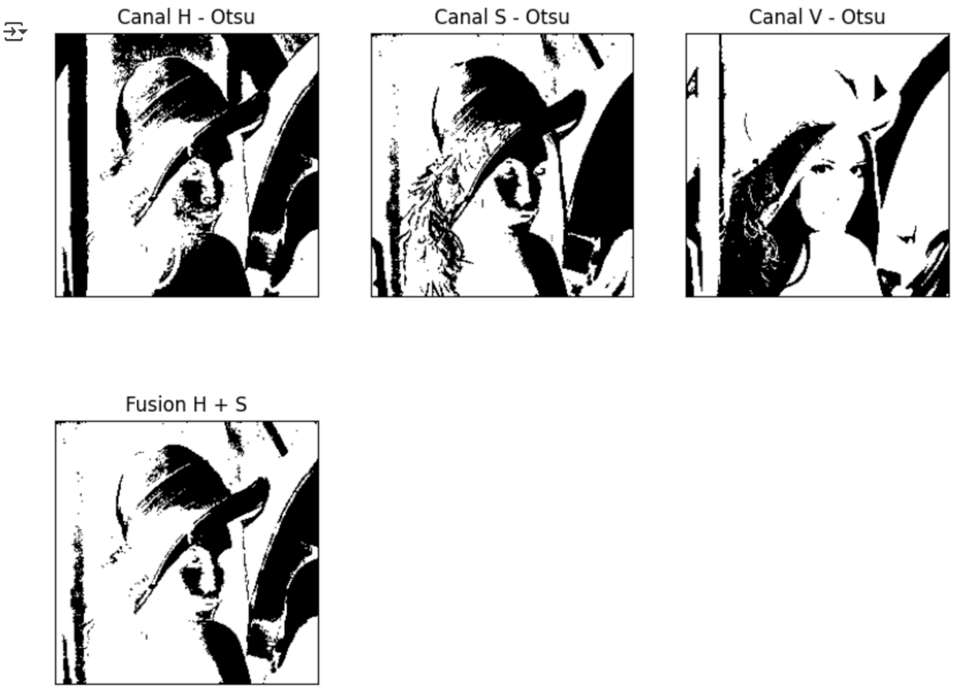
```
[20] # 2.2 Seuillage Otsu sur les canaux HSV
     img_color = cv2.imread('lena.png')
     img_hsv = cv2.cvtColor(img_color, cv2.COLOR_BGR2HSV)

     _, th_h = cv2.threshold(img_hsv[:, :, 0], 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
     _, th_s = cv2.threshold(img_hsv[:, :, 1], 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
     _, th_v = cv2.threshold(img_hsv[:, :, 2], 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```



```
[21] # 2.3 Fusion des canaux H + S
     th_final = cv2.bitwise_or(th_h, th_s)
```



```
# Affichage
plt.figure(figsize=(10, 8))
plt.subplot(2, 3, 1)
plt.imshow(th_h, cmap='gray')
plt.title("Canal H - Otsu")
plt.xticks([]), plt.yticks([])

plt.subplot(2, 3, 2)
plt.imshow(th_s, cmap='gray')
plt.title("Canal S - Otsu")
plt.xticks([]), plt.yticks([])

plt.subplot(2, 3, 3)
plt.imshow(th_v, cmap='gray')
plt.title("Canal V - Otsu")
plt.xticks([]), plt.yticks([])

plt.subplot(2, 3, 4)
plt.imshow(th_final, cmap='gray')
plt.title("Fusion H + S")
plt.xticks([]), plt.yticks([])
plt.show()
```


Canal H - Otsu
Canal S - Otsu
Canal V - Otsu
Fusion H + S