Задание.

Загрузить тестовое изображение, выполнить над ним операции:

1. Подготовить тестовое изображение в градациях серого

2. Выполнить пороговую сегментацию изображения

3. Выполнить двухпороговую сегментацию изображения

4. Выполнить сегментацию изображения с глобальным порогом

5. Выполнить сегментацию изображения методом Оцу

Решение.

```
import numpy as np
from hashlib import sha256

def check(image, ref_hash):
    image_hash = sha256(np.ascontiguousarray(image)).hexdigest()
    if image_hash == ref_hash:
        print("All good \U0001F44D")
    else:
        raise RuntimeError("Image hash differ: " + image_hash)
```

Задание 1

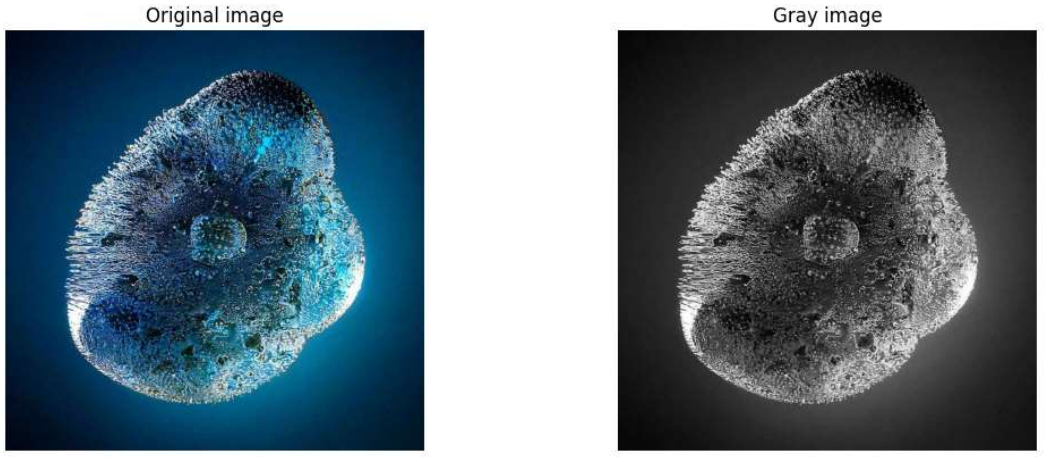
Подготовить тестовое изображение в градациях серого.

```
import cv2
import skimage
import skimage.io
from matplotlib import pyplot as plt
src_image_url =
'https://drive.google.com/u/0/uc?id=1UNjtA2mDcj9GFT2HOa72PHHFeHnfqAfh&export=
download'
src_image = skimage.io.imread(src_image_url)

gray_image = cv2.cvtColor(src_image, cv2.COLOR_RGB2GRAY)

# Отображение оригинального и серого изображения
plt.figure(figsize=[6.4 * 2, 4.8])
plt.subplot(1, 2, 1)
plt.imshow(src_image, cmap='gray')
plt.axis("off")
plt.title("Original image")
plt.subplot(1, 2, 2)
plt.imshow(gray_image, cmap='gray')
```

```
plt.axis("off")
plt.title("Gray image")
plt.show()
```

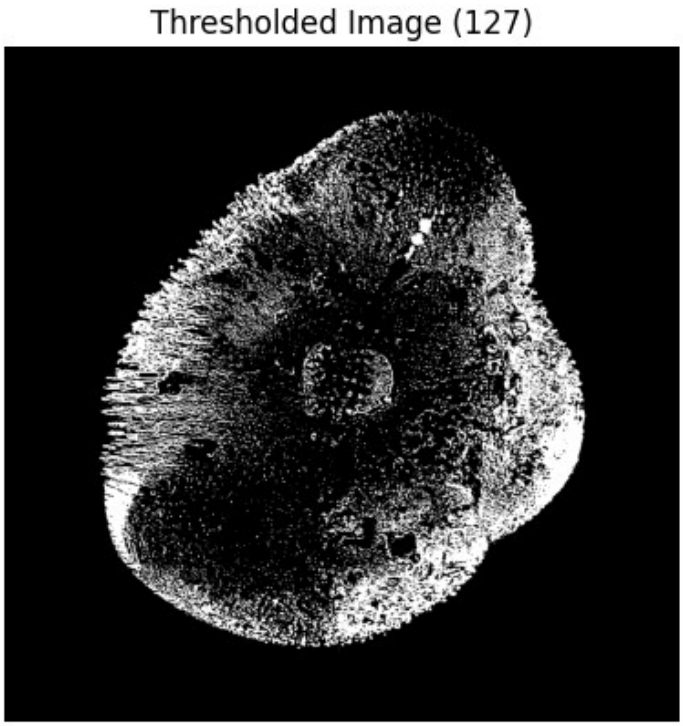


Задание 2

Выполнить пороговую сегментацию изображения.

```
_, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)

plt.figure(figsize=[6.4, 4.8])
plt.imshow(binary_image, cmap='gray')
plt.axis("off")
plt.title("Thresholded Image (127)")
plt.show()
```



Задание 3

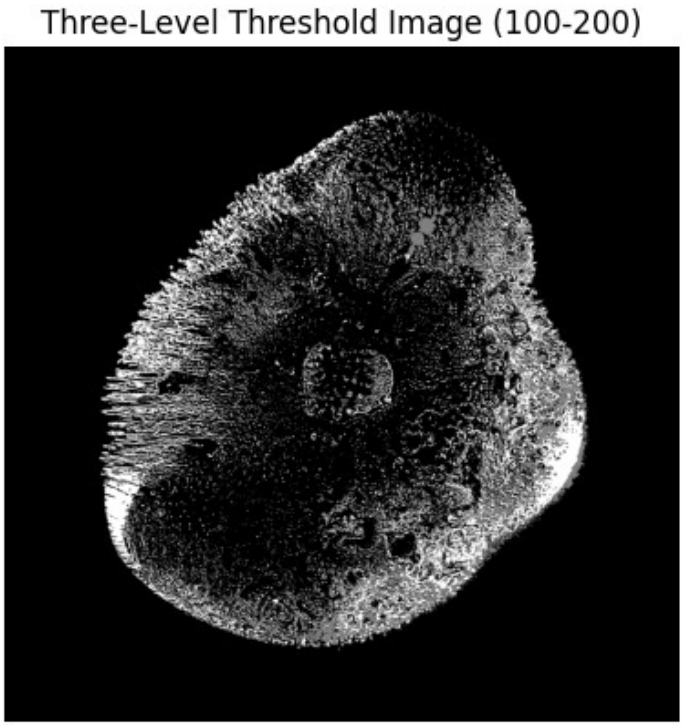
Выполнить двухпороговую сегментацию для получения 3 уровней яркости.

```
low_thresh = 100
high_thresh = 200

# Применение порогов для создания трех уровней
lower_part = (gray_image <= low_thresh) * 85  # Уровень яркости 85
middle_part = ((gray_image > low_thresh) & (gray_image <= high_thresh)) * 170
# Уровень яркости 170
upper_part = (gray_image > high_thresh) * 255  # Уровень яркости 255

# Комбинирование всех трех уровней
three_level_image = lower_part + middle_part + upper_part

plt.figure(figsize=[6.4, 4.8])
plt.imshow(three_level_image, cmap='gray')
plt.axis("off")
plt.title("Three-Level Threshold Image (100-200)")
plt.show()
```

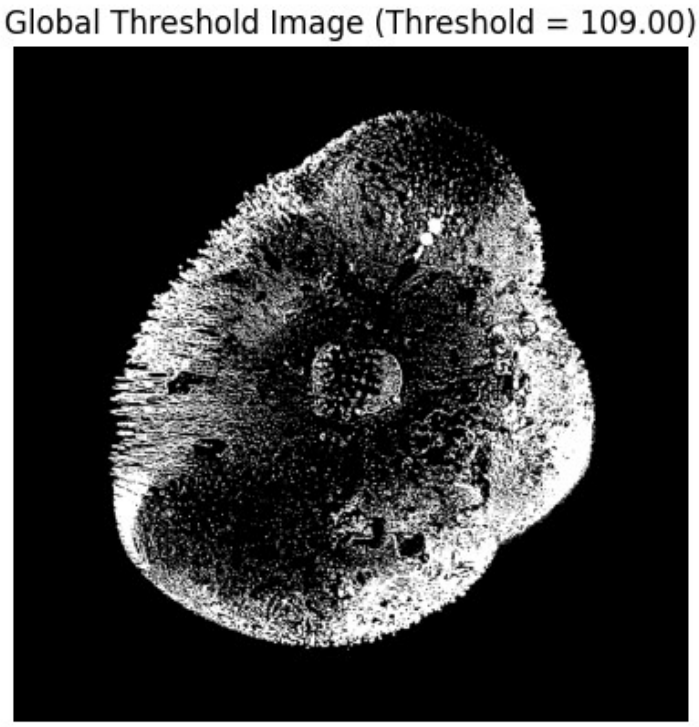


Задание 4

Выполнить сегментацию изображения с использованием глобального порога.

```
# Используем алгоритм свертки для поиска оптимального глобального порога
global_thresh = cv2.threshold(gray_image, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)[0]
_, global_threshold_image = cv2.threshold(gray_image, global_thresh, 255,
cv2.THRESH_BINARY)

plt.figure(figsize=[6.4, 4.8])
plt.imshow(global_threshold_image, cmap='gray')
plt.axis("off")
plt.title(f"Global Threshold Image (Threshold = {global_thresh:.2f})")
plt.show()
```

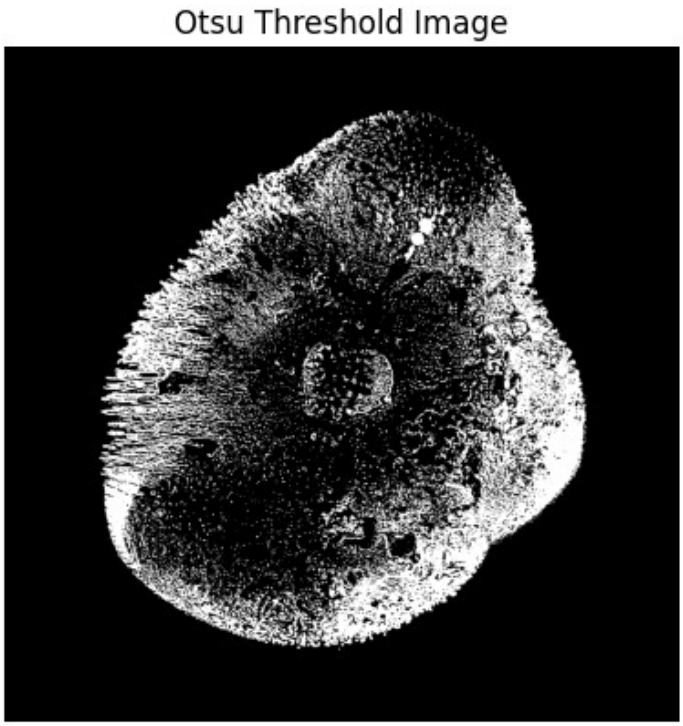


Задание 5

Выполнить сегментацию изображения методом Оцу.

```
_, otsu_image = cv2.threshold(gray_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

plt.figure(figsize=[6.4, 4.8])
plt.imshow(otsu_image, cmap='gray')
plt.axis("off")
plt.title("Otsu Threshold Image")
plt.show()
```



Проверка изображения

```
# Генерация хэша для otsu_image
image_hash = sha256(np.ascontiguousarray(otsu_image)).hexdigest()
print("Otsu Image Hash:", image_hash)
```

```
Otsu Image Hash:
7cd069cf1136dec53b9a6f06de11dc49c72b6b981d2d8618286c4c997856dd00
```

```
ref_hash = image_hash
check(otsu_image, ref_hash)
```

```
All good 
```