Here are the libraries I used in this lab for applying (Morphology)

```
from skimage.morphology import (erosion, dilation, opening, closing, white_tophat)
from skimage.morphology import (convex_hull_image, skeletonize, black_tophat)
from skimage.color import rgb2gray
from skimage import io
import matplotlib.pyplot as plt
import numpy as np
```

Then I convert my image to grayscale, because morphological operation uses single-channel

```
pic= io.imread('Picture1.jpg')
pic_grayscale= rgb2gray(pic)
io.imshow(pic_grayscale)
plt.axis('off')
```

```
(-0.5, 450.5, 601.5, -0.5)
```

Then I created a comparison function, so that I print the original picture next to the altered one

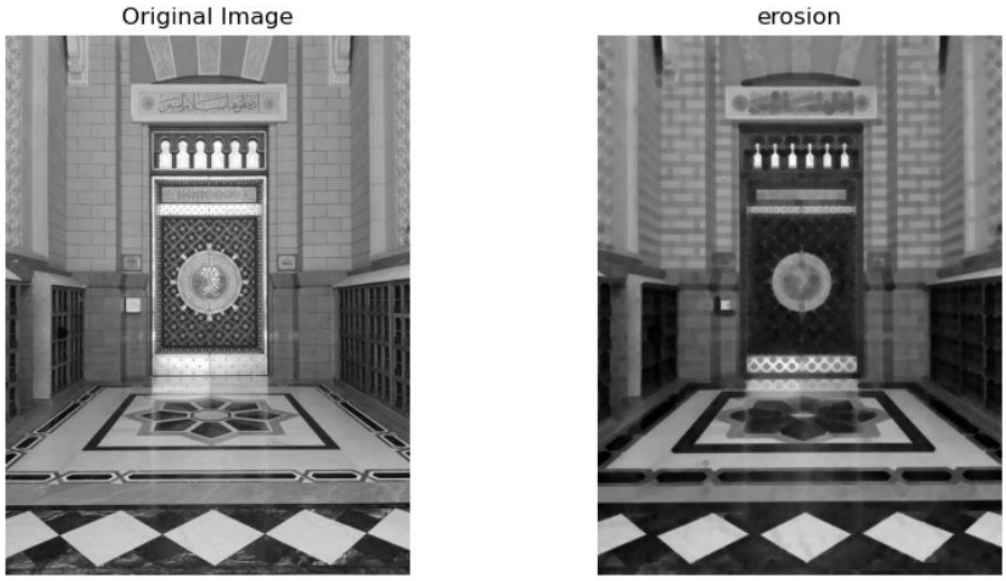
Also, I created a custom shape rather than using the existing shapes ex.(disk, square)

```
def plot_comparison(original, processed, title_processed):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
    ax1.imshow(original, cmap='gray')
    ax1.set_title('Original Image')
    ax1.axis('off')

    ax2.imshow(processed, cmap='gray')
    ax2.set_title(title_processed)
    ax2.axis('off')
    plt.show()


def custom_shape():
    cs = np.array([
        [0, 0, 1, 1, 1, 0, 0],
        [0, 1, 0, 1, 0, 1, 0],
        [1, 0, 0, 1, 0, 0, 1],
        [1, 1, 1, 1, 1, 1, 1],
        [1, 0, 0, 1, 0, 0, 1],
        [0, 1, 0, 1, 0, 1, 0],
        [0, 0, 1, 1, 1, 0, 0]
    ])

    return cs

custom_footprint = custom_shape()
print(custom_footprint)
```

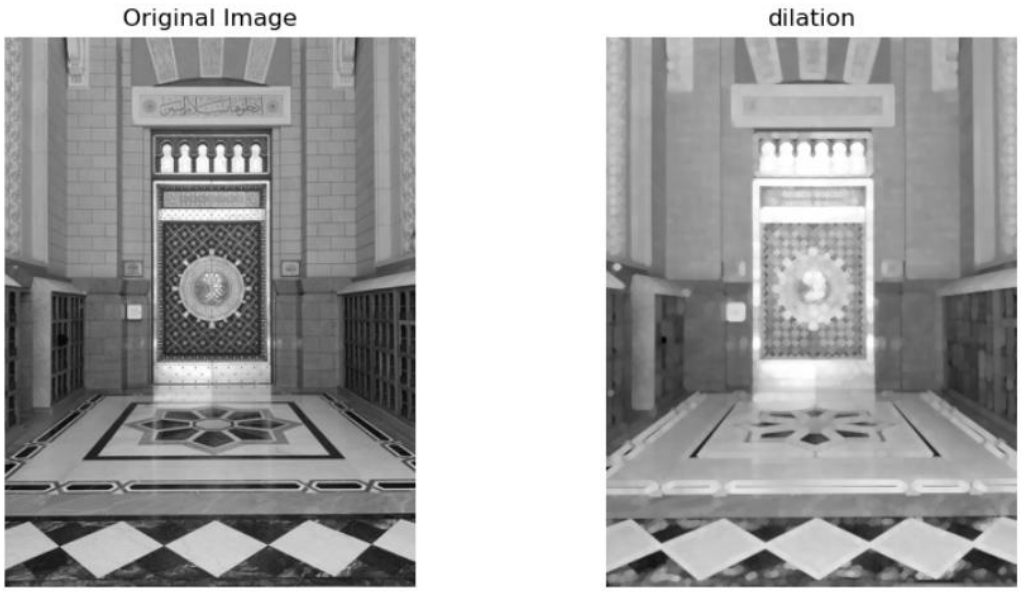
```
[[0 0 1 1 1 0 0]
 [0 1 0 1 0 1 0]
 [1 0 0 1 0 0 1]
 [1 1 1 1 1 1 1]
 [1 0 0 1 0 0 1]
 [0 1 0 1 0 1 0]
 [0 0 1 1 1 0 0]]
```

Here are some of the functions that I took from the Lab Manual and applied on my image:

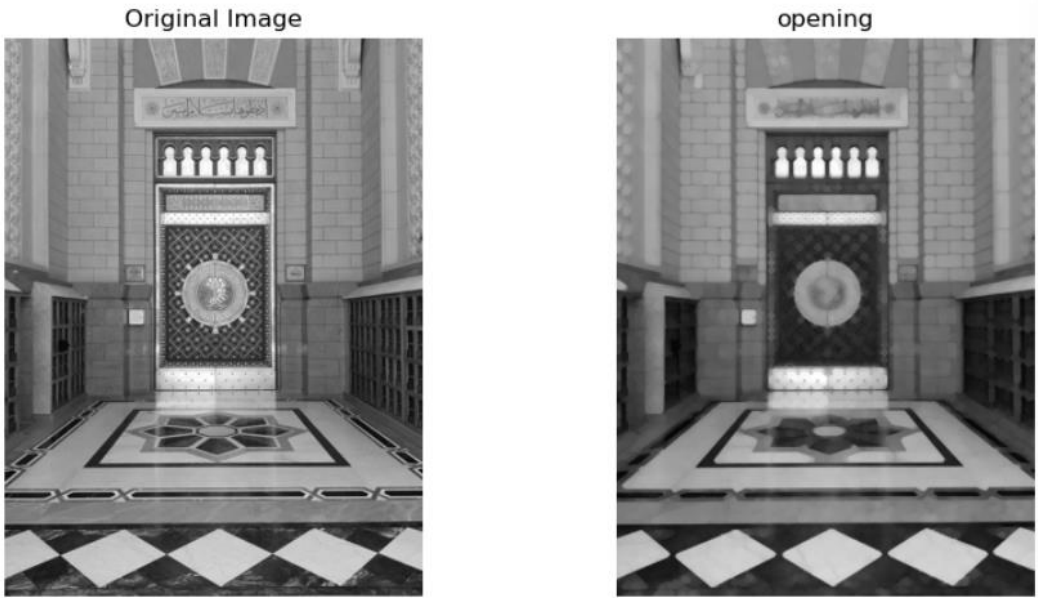
```
eroded= erosion(pic_grayscale, custom_shape())
plot_comparison(pic_grayscale, eroded, 'erosion')
```



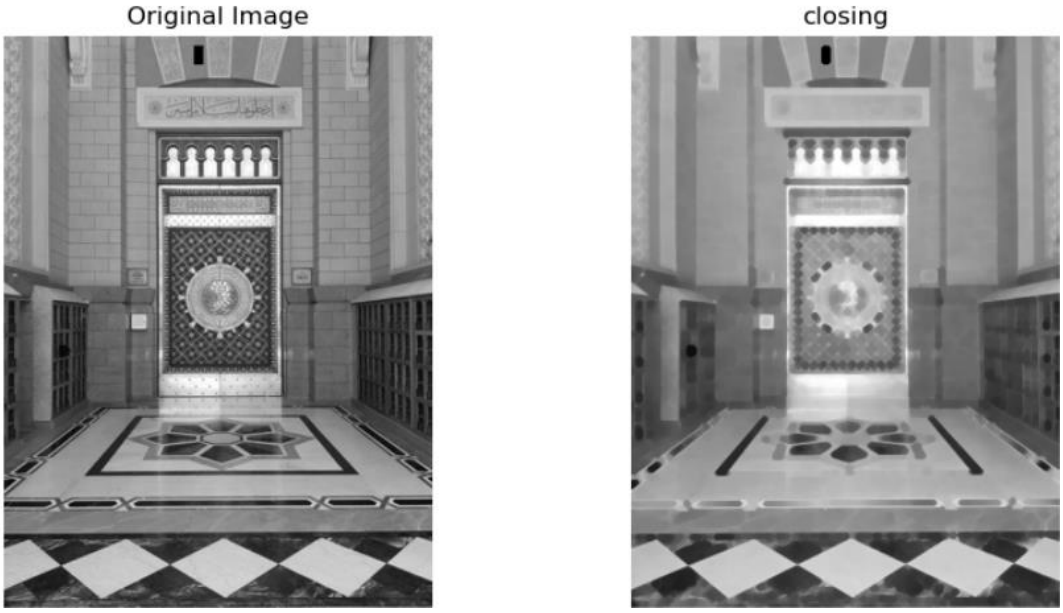

```
dilated= dilation(pic_grayscale, custom_shape())
plot_comparison(pic_grayscale, dilated, 'dilation')
```



```
opened = opening(pic_grayscale, custom_shape())
plot_comparison(pic_grayscale, opened, 'opening')
```

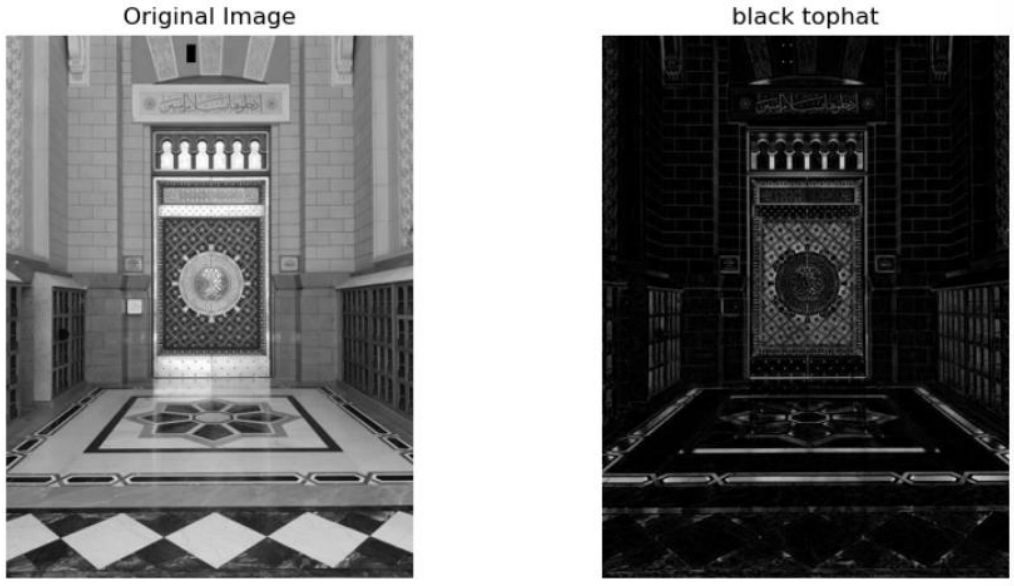



```
phantom = pic_grayscale.copy()
phantom[10:30,200:210]=0

closed = closing(phantom, custom_shape())
plot_comparison(phantom, closed, 'closing')
```



```
w_tophat = white_tophat(phantom, custom_shape())
plot_comparison(phantom, w_tophat, 'white tophat')
```


```
b_tophat = black_tophat(phantom, custom_shape())
plot_comparison(phantom, b_tophat, 'black tophat')
```



Here, I made a copy of the grayscale image to keep the original safe. Then, we check each pixel's brightness:

if it's brighter than 0.5, we make it white (1), otherwise, we make it black (0).

This creates a simple black and white version of the image.

```
sk_pic= pic_grayscale.copy()

for i in range(sk_pic.shape[0]):
    for j in range(sk_pic.shape[1]):
        if sk_pic[i][j] > 0.5 :
            sk_pic[i][j]=1
        else:
            sk_pic[i][j]=0


sk= skeletonize(sk_pic)
plot_comparison(pic_grayscale, sk, 'skeletonize')
```