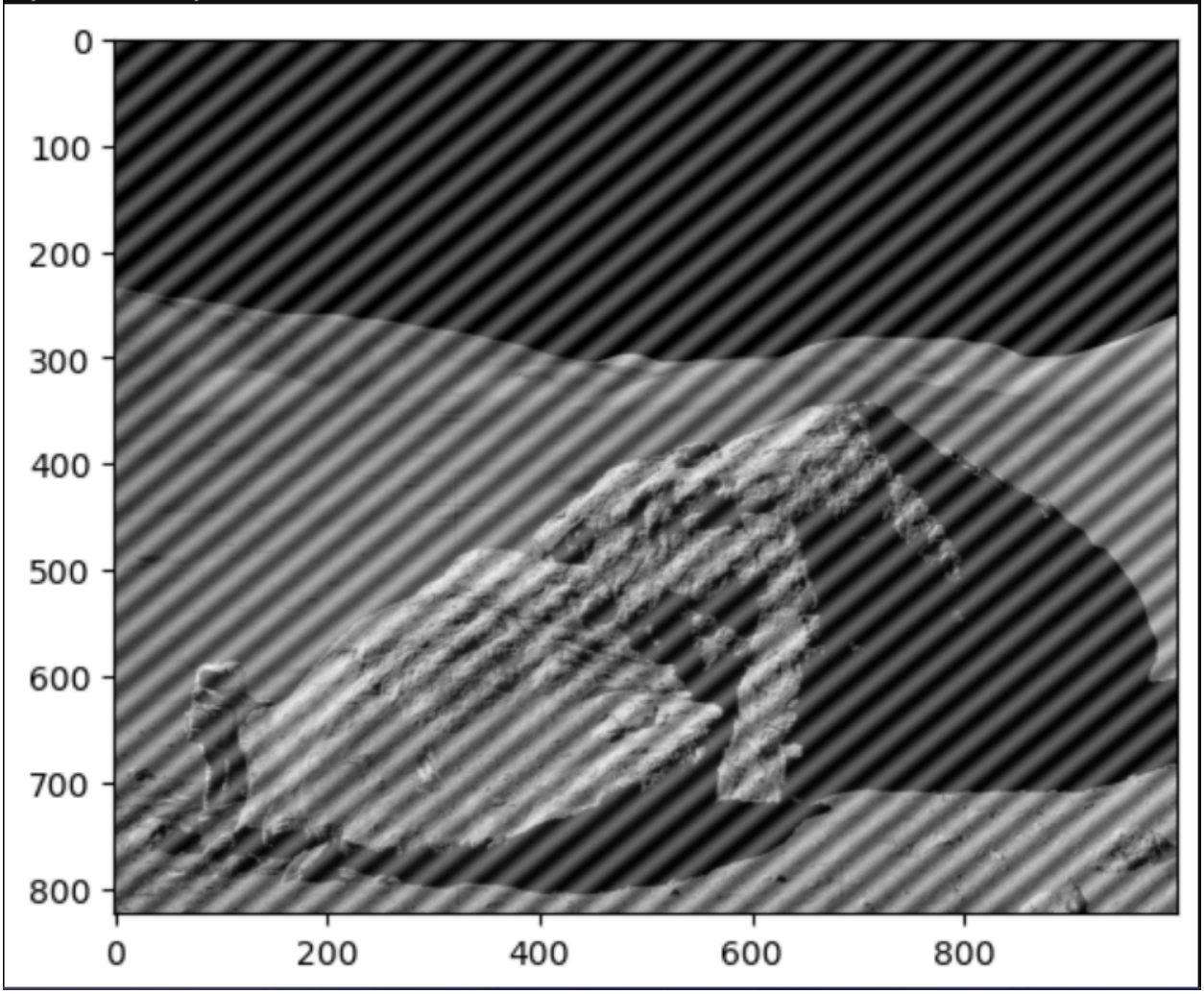
Name: Zacci Oduneye
Student #: 002-48-7453

```
import numpy as np
from scipy.fft import fft2, ifft2, fftshift, ifftshift
import matplotlib.pyplot as plt
from skimage import io, morphology
from skimage.feature import peak_local_max
from skimage.morphology import erosion, dilation
```

- Here are the imports I used in my jupyter

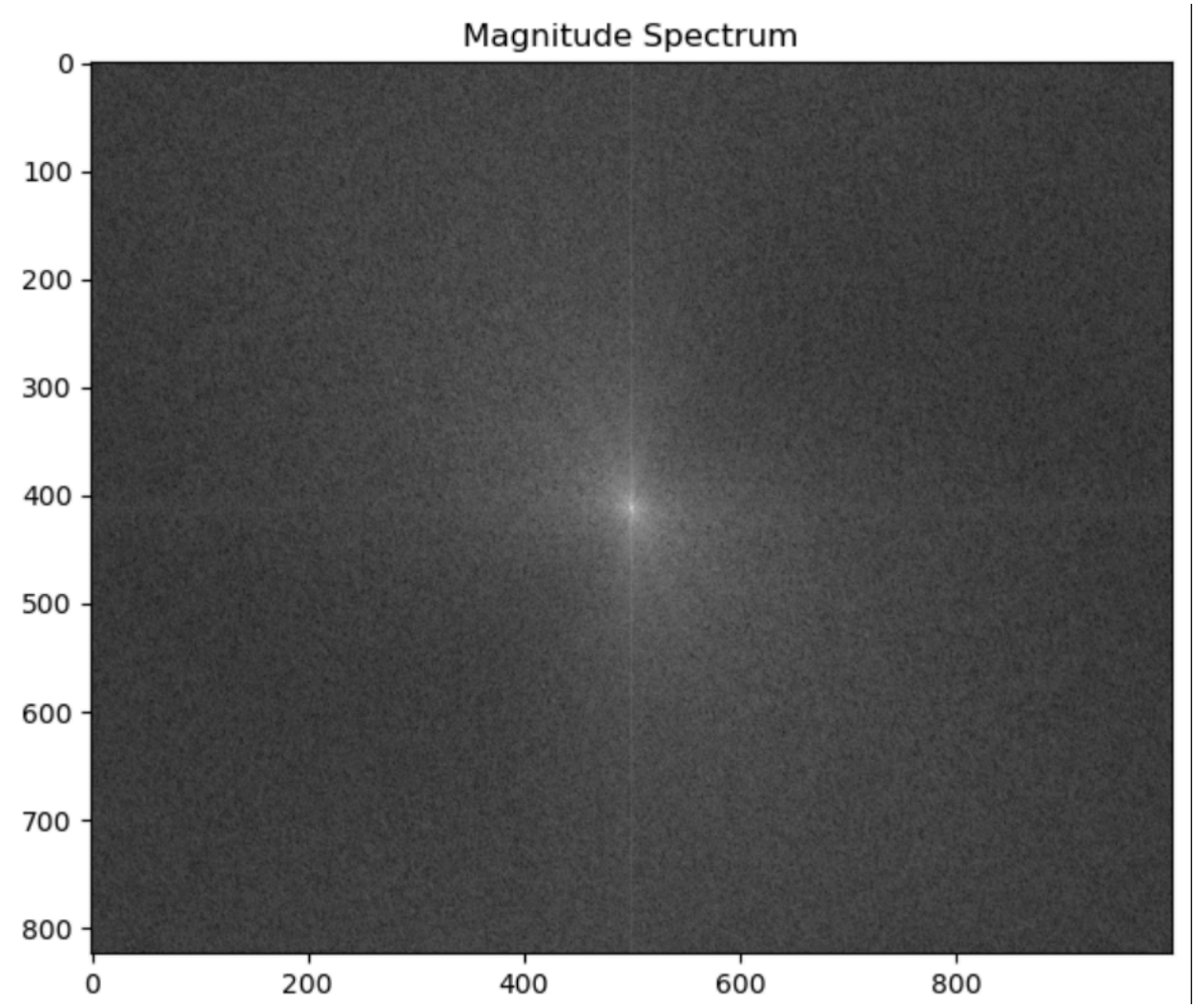
```
# get and display the HW2_DegradedImage.jpeg
img1 = io.imread('./HW2_DegradedImage.jpeg', as_gray=True)
plt.imshow(img1, cmap='gray')
print(img1.shape)
plt.show()
```

```
(824, 1000)
```



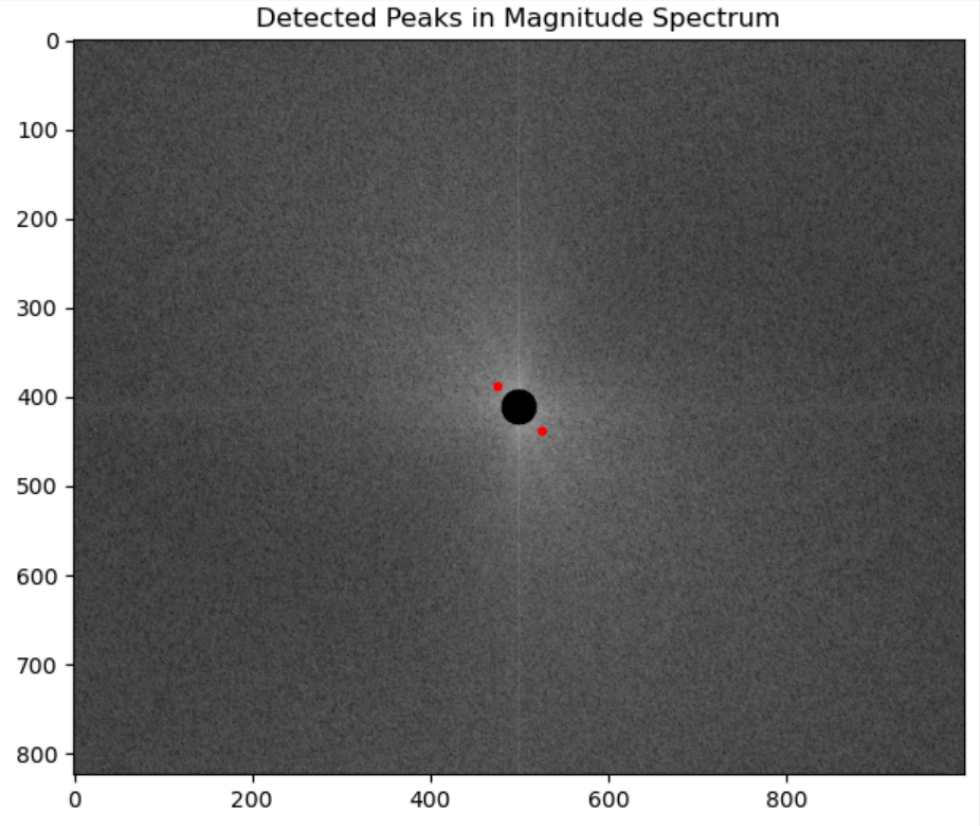
- Once I have got all the in the file I then uploaded the first image

```python
# compute the Fourier Transform and shift it
img1_fft = np.fft.fft2(img1)
img1_ffts = np.fft.fftshift(img1_fft)

plt.figure(figsize = (10,6))
plt.imshow(np.log(1 + np.abs(img1_ffts)), cmap = 'gray')
plt.title('Magnitude Spectrum')
# plt.axis('off')
plt.show()
```

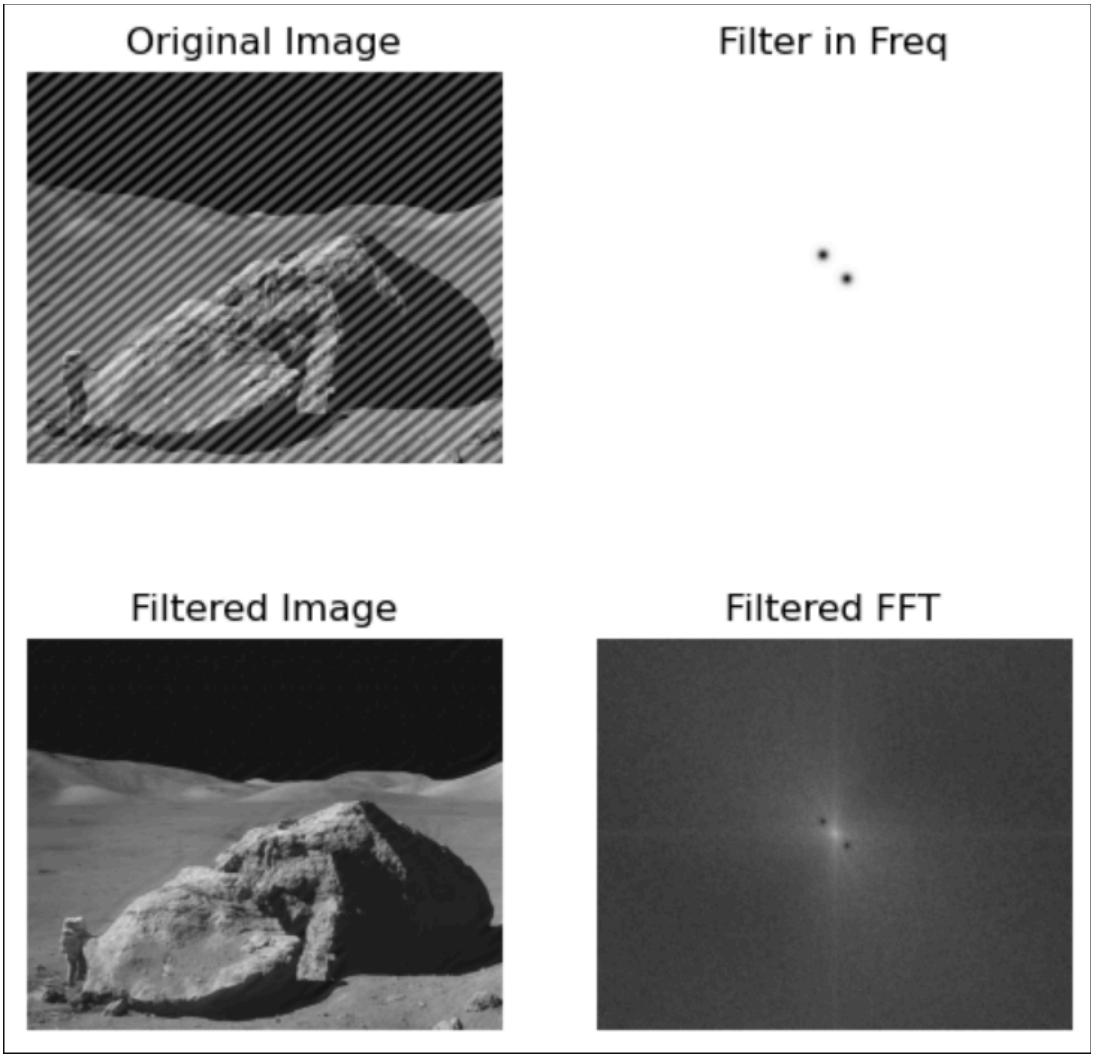


- I then use the code above to get the Fourier Transform of the image and get the magnitude spectrum

```python
# compute the magnitude spectrum
magnitude_spectrum = np.log(1 + np.abs(img1_ffts))

# this is to mask the center of the magnitude spectrum
radius = 20  # Radius of the circle to mask
N, M = magnitude_spectrum.shape
center_x, center_y = N // 2, M // 2
x, y = np.ogrid[:N, :M]
center_mask = (x - center_x) ** 2 + (y - center_y) ** 2 <= radius ** 2

magnitude_spectrum[center_mask] = 0

# detect local maxima (noise frequencies)
coordinates = peak_local_max(magnitude_spectrum, min_distance=20, threshold_abs=10 )

# visualize the detected peaks on the Fourier Transform
plt.figure(figsize=(10, 6))
plt.imshow(magnitude_spectrum, cmap='gray')
plt.scatter(coordinates[:, 1], coordinates[:, 0], color='red', s=10)# Plot detected peaks
# print(coordinates)
plt.title('Detected Peaks in Magnitude Spectrum')
plt.show()
```



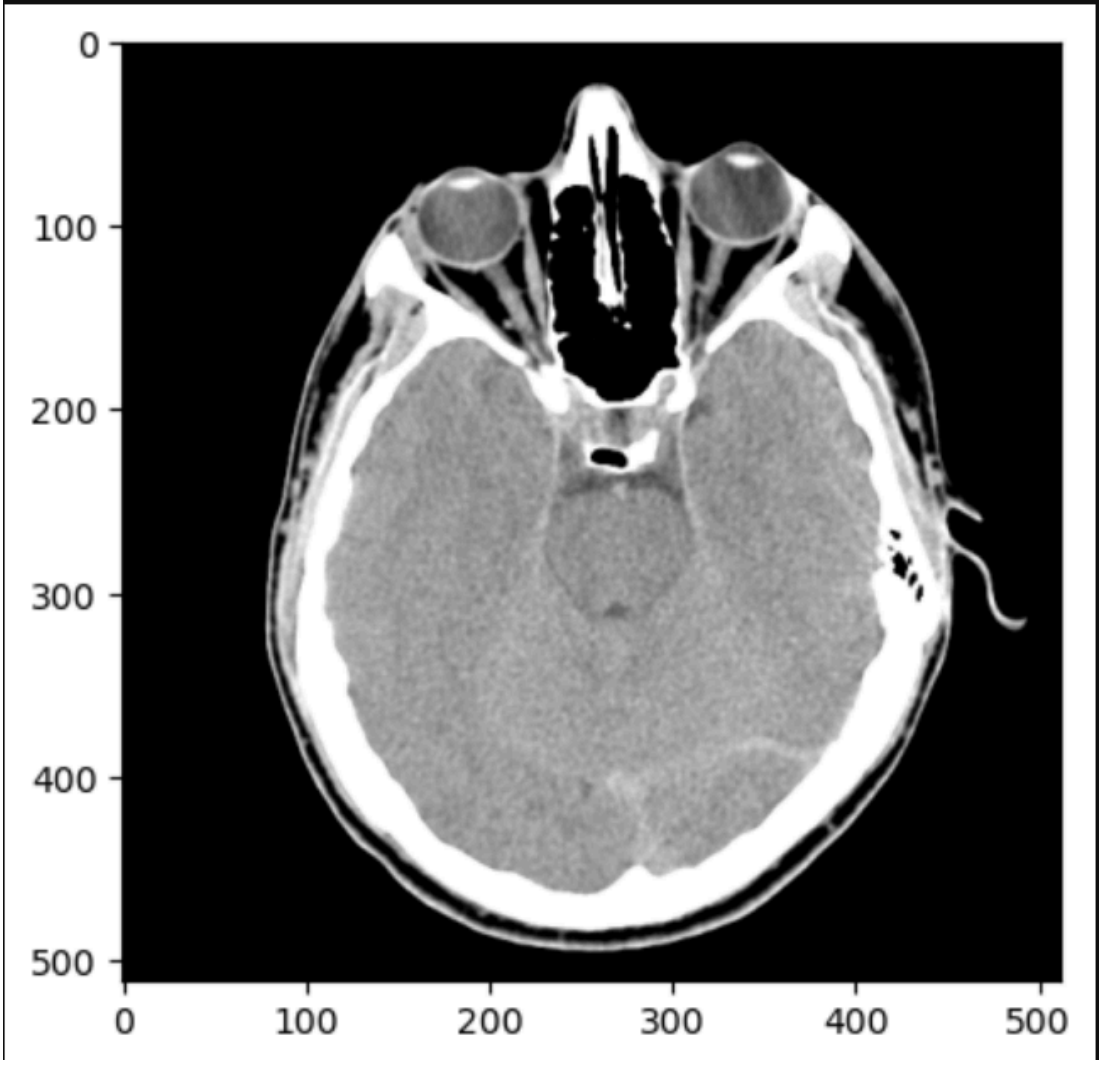
- Here I use this code to find the coordinates of peaks/noise in the image. I use a mask at the center of the magnitude spectrum image so that the peak_local_max function does not see it as a valid peak.



- I then use a notch filter generator to place notch filters at coordinates where peaks were found. After that we apply to our magnitude spectrum image then do Fourier Transform and a shift to get the original image with no noise.

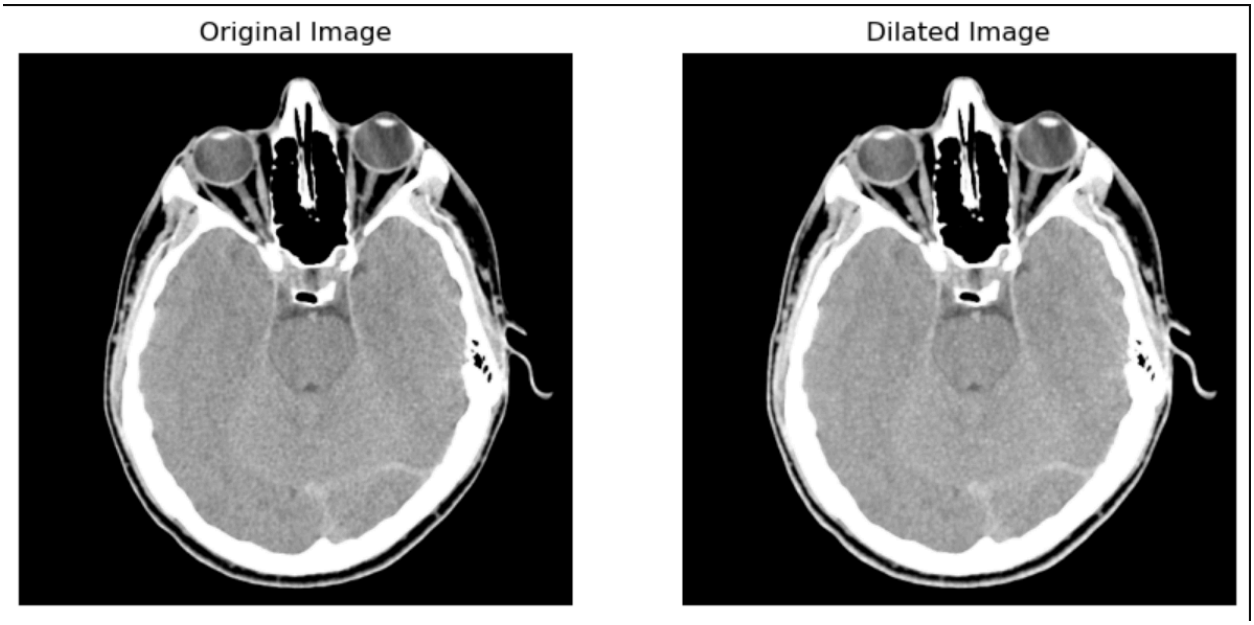
```
img2 = io.imread('./HW2_Head.tif', as_gray=True)
print(img2.shape)
plt.imshow(img2, cmap='gray')
plt.show()
```

```
(512, 512)
```



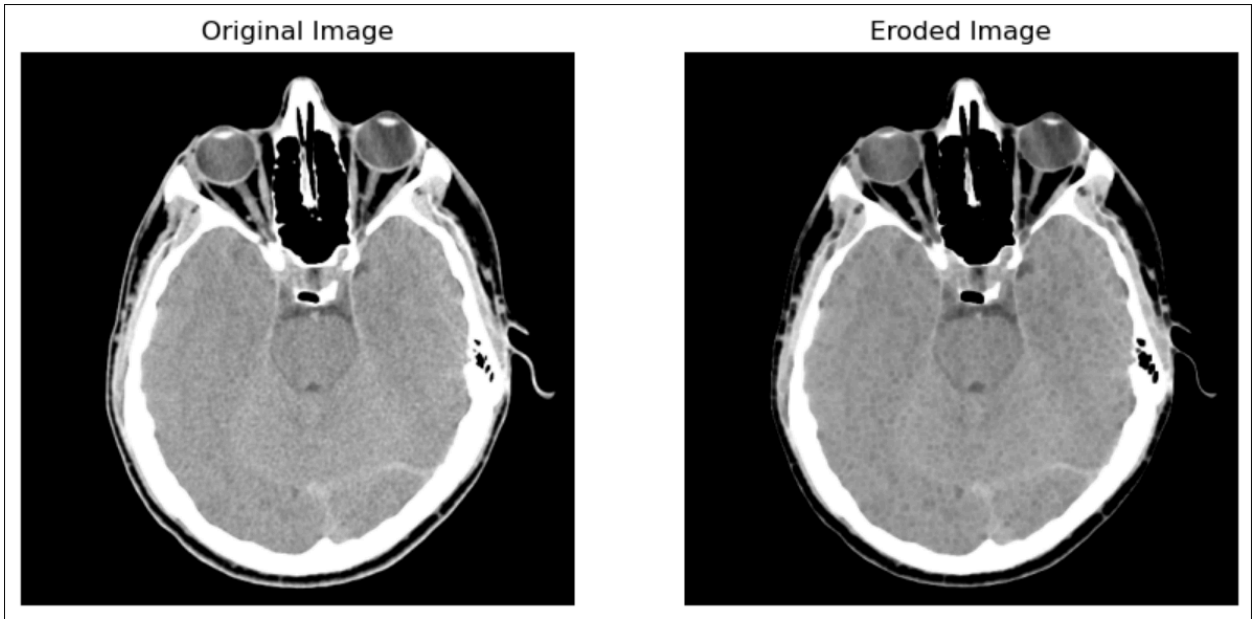
- Here I just started question 2 and uploaded the second image to jupyter.

```python
# Defining the square-shaped sampling element
selem = morphology.square(2)

dilated_img2 = dilation(img2, footprint=selem)

# Display the original and dialated image
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(img2, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(dilated_img2, cmap='gray')
plt.title('Dilated Image')
plt.axis('off')

plt.show()
```



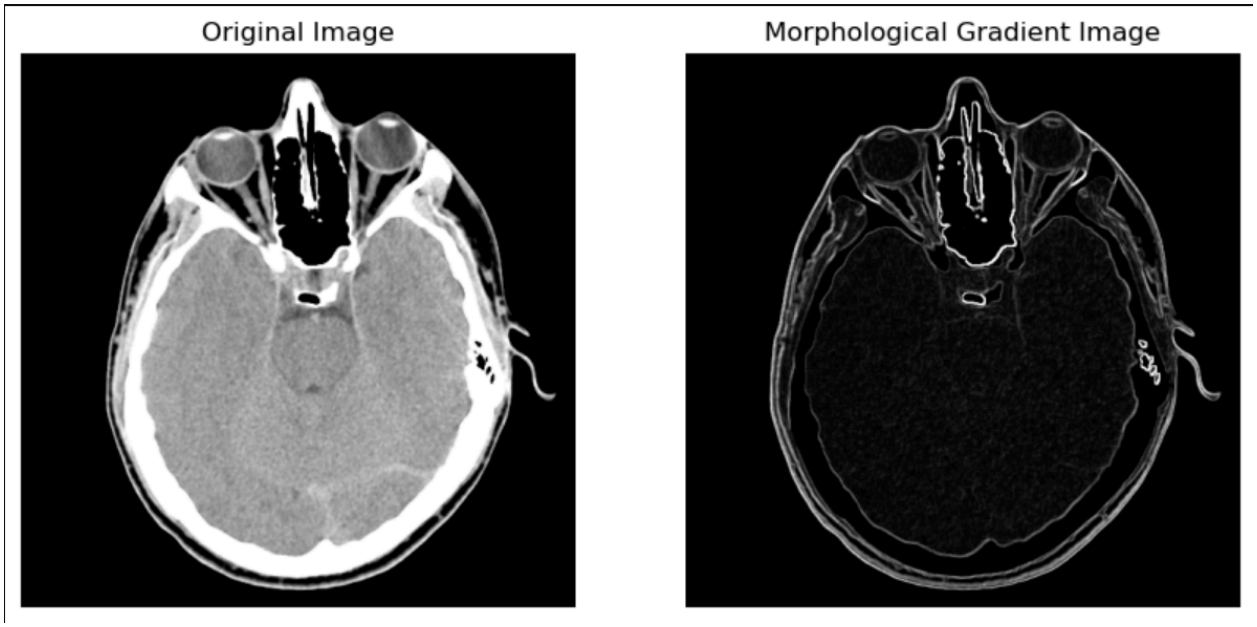
- Here I was trying to dilate the image so I created a structure element using skimage.morphology.square and made the width 2. After I used the dilation function from skimage.morphology and used the second image and the structure element as the parameters. Then I displayed it.

```
selem = morphology.square(3)
eroded_img2 = erosion(img2, footprint=selem)

# Display the original and eroded image
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(img2, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(eroded_img2, cmap='gray')
plt.title('Eroded Image')
plt.axis('off')

plt.show()
```

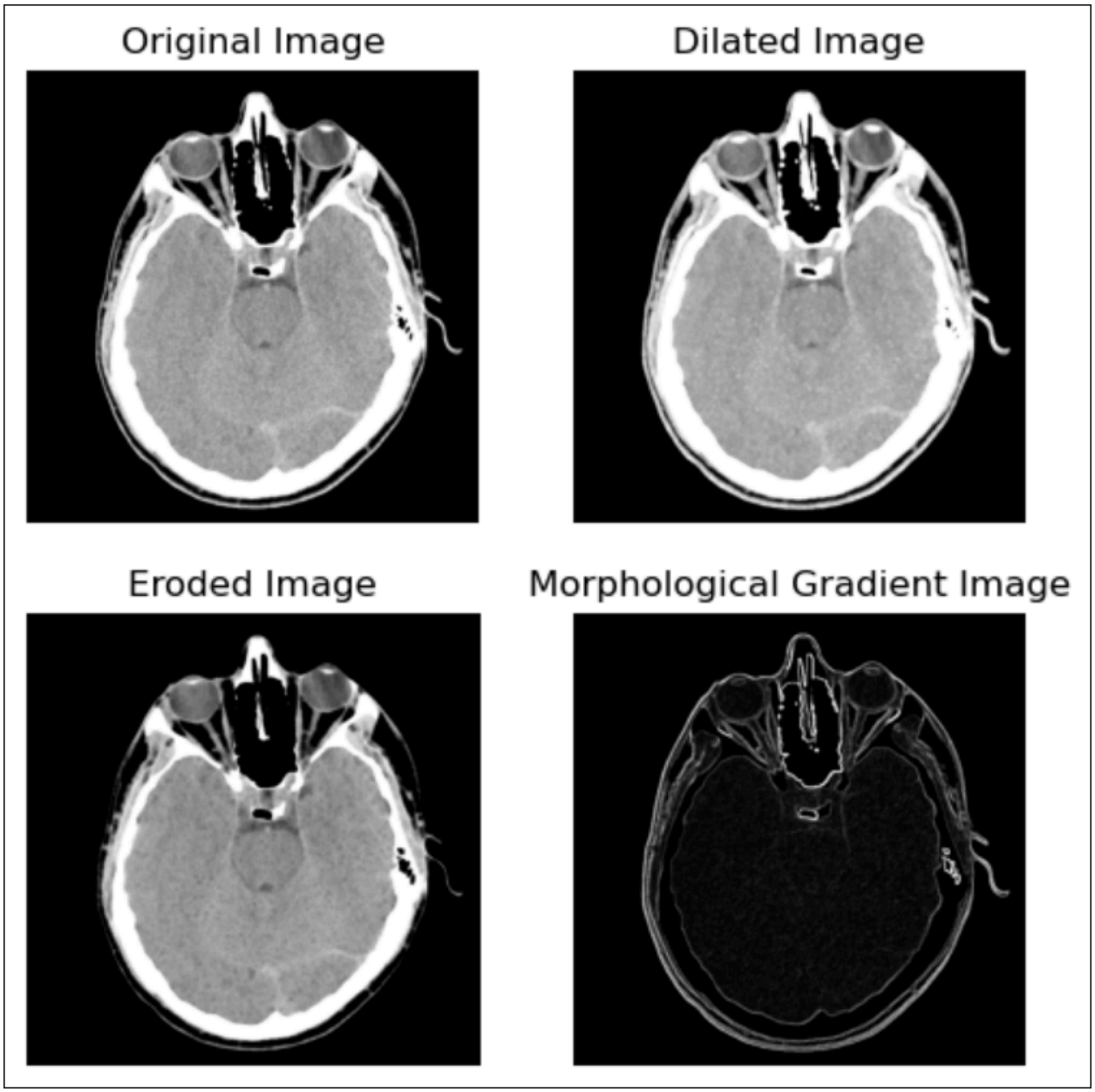


- Here I reassign “selem” so that the morphology.square width parameter is 3. After I used the erosion function from skimage.morphology and used the second image and the structure element as the parameters. Then I displayed it

```
morphological_gradient = dilated_img2 - eroded_img2

dilated_img2 = dilation(img2, footprint=selem)

# Display the original and morphological gradient
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(img2, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(morphological_gradient, cmap='gray')
plt.title('Morphological Gradient Image')
plt.axis('off')

plt.show()
```



- The next thing I did was subtract the eroded image from the dilated image and assigned it to a variable named morphological_gradient. After then I displayed it.



- Here is the Original, Dilated, Eroded, and Morphological Gradient images together.