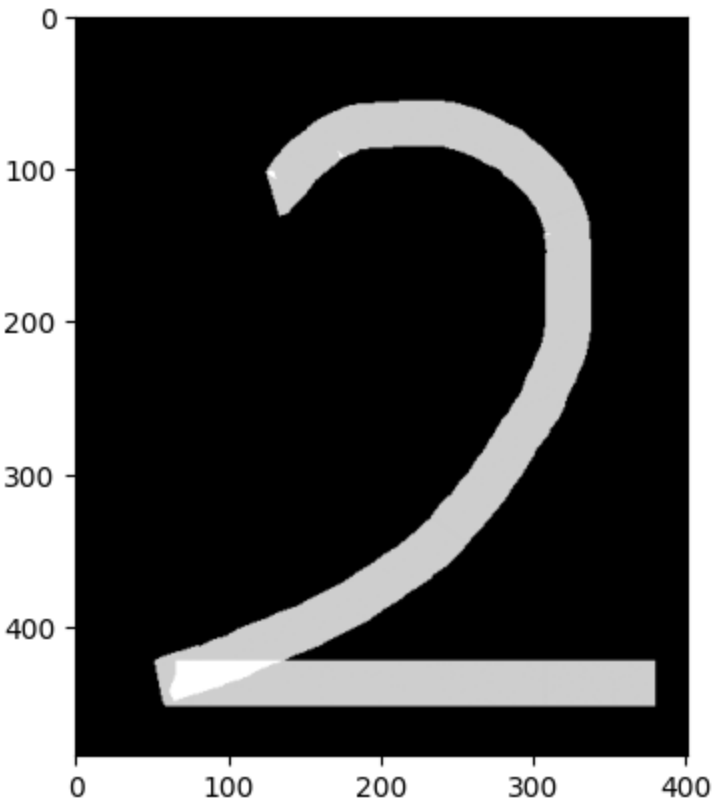
In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

In [3]:
```
kernel = np.ones((5, 5), np.uint8)
print(kernel)
```

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

In [4]:
```
img = cv2.imread('./img/number2_2.png')
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img, cmap='gray')
```

Out[4]: <matplotlib.image.AxesImage at 0x1daae7ba310>



In [6]:
```
def add_padding(image, kernel_size):
    image = image.astype('double')
    height, width = image.shape

    if kernel_size % 2 == 0:
        padding1 = int((kernel_size - 2) / 2)
        padding2 = int(kernel_size / 2)

        top_pad = np.zeros((padding1, width),
                           dtype=np.double)
```

```
        bottom_pad = np.zeros((padding2, width),
                              dtype=np.double)

        vert_pad = np.vstack((top_pad, image, bottom_pad))

        lef_pad = np.zeros((vert_pad.shape[0], padding1),
                           dtype=np.double)
        right_pad = np.zeros((vert_pad.shape[0], padding2),
                             dtype=np.double)

        matrix_pad = np.hstack((lef_pad, vert_pad, right_pad))

    else:
        padding =  int((kernel_size - 1) / 2)

        top_pad = np.zeros((padding, width),
                           dtype=np.double)
        bottom_pad = np.zeros((padding, width),
                              dtype=np.double)

        vert_pad = np.vstack((top_pad, image, bottom_pad))

        lef_pad = np.zeros((vert_pad.shape[0], padding),
                           dtype=np.double)
        right_pad = np.zeros((vert_pad.shape[0], padding),
                             dtype=np.double)

        matrix_pad = np.hstack((lef_pad, vert_pad, right_pad))
    return matrix_pad
```

In [16]:

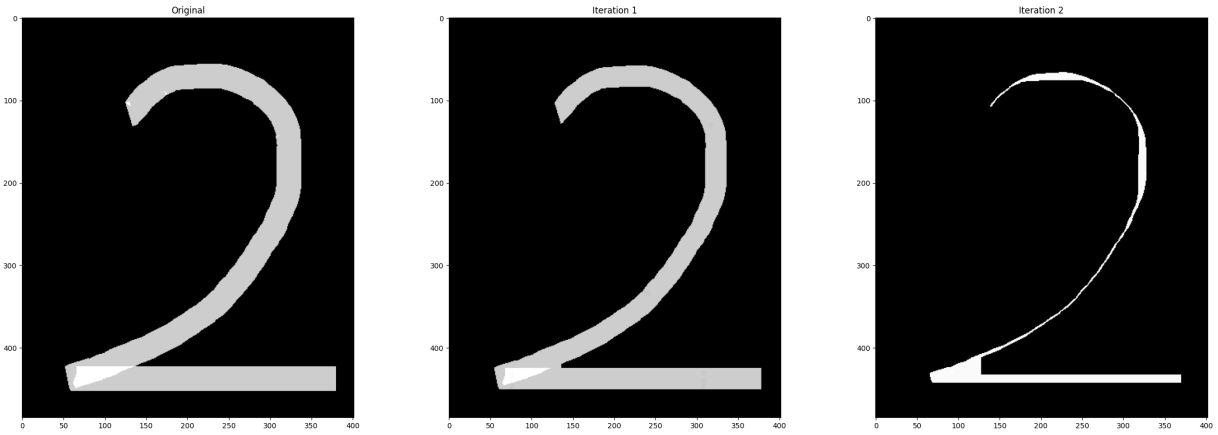
```
def erosion(image, kernel, iteration):
    image = image.astype('double')
    height, width = image.shape
    kernel_size = kernel.shape[0]
    matrix_pad = add_padding(image, kernel_size)
    erosion_img = np.zeros((height, width), dtype=np.double)

    for i in range(iteration):
        for j in range(width):
            for k in range(height):
                erosion_img[k, j] = np.min(matrix_pad[k: k + kernel_size, j: j + ke
        matrix_pad = add_padding(erosion_img, kernel_size)

    # if iteration > 1:
    #     for i in range(iteration - 1):
    #         erosion_img = erosion(erosion_img, kernel, 1)
    erosion_img = erosion_img.astype('uint8')

    return erosion_img
```

In [39]:

```
erosion_img1 = erosion(img, kernel, 1)
erosion_img2 = erosion(img, kernel, 5)


fig = plt.figure(figsize=(30, 10))
```

```
plt.subplot(1, 3, 1)
plt.title('Original')
plt.imshow(img, cmap='gray')

plt.subplot(1, 3, 2)
plt.title('Iteration 1')
plt.imshow(erosion_img1, cmap='gray')

plt.subplot(1, 3, 3)
plt.title('Iteration 5')
plt.imshow(erosion_img2, cmap='gray')
```

Out[39]: <matplotlib.image.AxesImage at 0x1dab6399460>

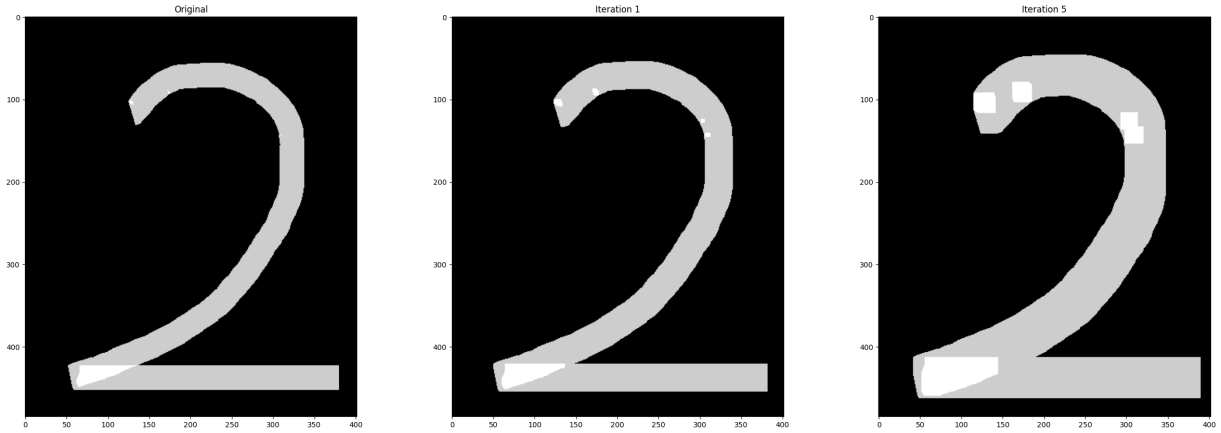


In [18]:
```
def dialate(image, kernel, iteration):
    image = image.astype('double')
    height, width = image.shape
    kernel_size = kernel.shape[0]
    matrix_pad = add_padding(image, kernel_size)
    dialate_img = np.zeros((height, width), dtype=np.double)

    for i in range(iteration):
        for j in range(width):
            for k in range(height):
                dialate_img[k, j] = np.max(matrix_pad[k: k + kernel_size, j: j + ke
        matrix_pad = add_padding(dialate_img, kernel_size)

    dialate_img = dialate_img.astype('uint8')

    return dialate_img
```

In [41]:
```
dialate_img1 = dialate(img, kernel, 1)
dialate_img2 = dialate(img, kernel, 5)


fig = plt.figure(figsize=(30, 10))
plt.subplot(1, 3, 1)
plt.title('Original')
plt.imshow(img, cmap='gray')

plt.subplot(1, 3, 2)
plt.title('Iteration 1')
```

```
plt.imshow(dialate_img1, cmap='gray')

plt.subplot(1, 3, 3)
plt.title('Iteration 5')
plt.imshow(dialate_img2, cmap='gray')
```

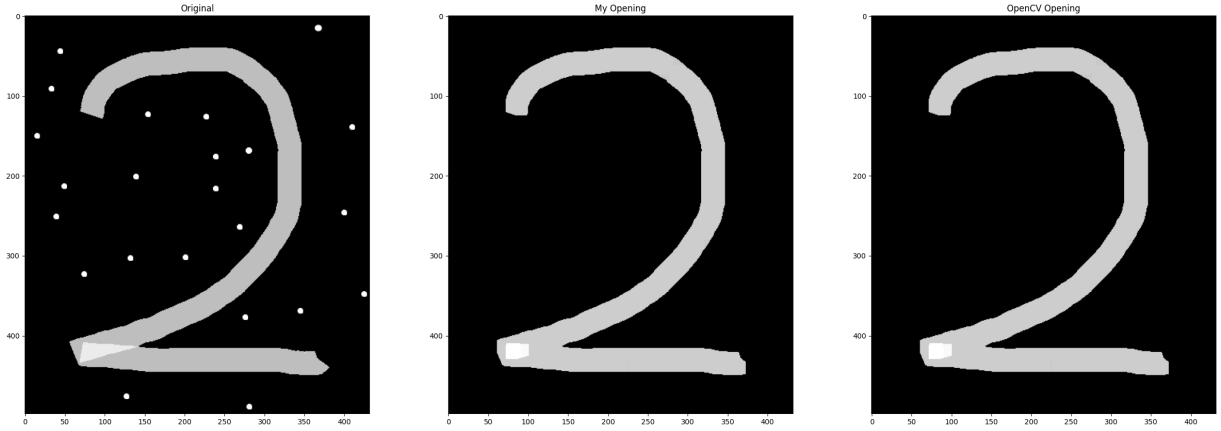
Out[41]: <matplotlib.image.AxesImage at 0x1dabab16580>



In [20]:
```
def opening(image, kernel):
    return dialate(erosion(image, kernel, 1), kernel, 1)
```

In [24]:
```
img2 = cv2.imread('./img/opening.png')
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
kernel = np.ones((15, 15), np.uint8)
opening_img1 = opening(img2, kernel)
opening_img2 = cv2.morphologyEx(img2, cv2.MORPH_OPEN, kernel)

fig = plt.figure(figsize=(30, 10))
plt.subplot(1, 3, 1)
plt.title('Original')
plt.imshow(img2, cmap='gray')

plt.subplot(1, 3, 2)
plt.title('My Opening')
plt.imshow(opening_img1, cmap='gray')

plt.subplot(1, 3, 3)
plt.title('OpenCV Opening')
plt.imshow(opening_img2, cmap='gray')
```

Out[24]: <matplotlib.image.AxesImage at 0x1dab4bd7b80>



In [25]:
```
def closing(image, kernel):
    return erosion(dialate(image, kernel, 1), kernel, 1)
```

In [37]:
```
img3 = cv2.imread('./img/closing.png')
img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2GRAY)
kernel = np.ones((5, 5), np.uint8)
closing_img1 = closing(img3, kernel)
closing_img2 = cv2.morphologyEx(img3, cv2.MORPH_CLOSE, kernel)

fig = plt.figure(figsize=(30, 10))
plt.subplot(1, 3, 1)
plt.title('Original')
plt.imshow(img3, cmap='gray')

plt.subplot(1, 3, 2)
plt.title('My closing')
plt.imshow(closing_img1, cmap='gray')

plt.subplot(1, 3, 3)
plt.title('OpenCV closing')
plt.imshow(closing_img2, cmap='gray')
```

Out[37]: <matplotlib.image.AxesImage at 0x1dab5963910>