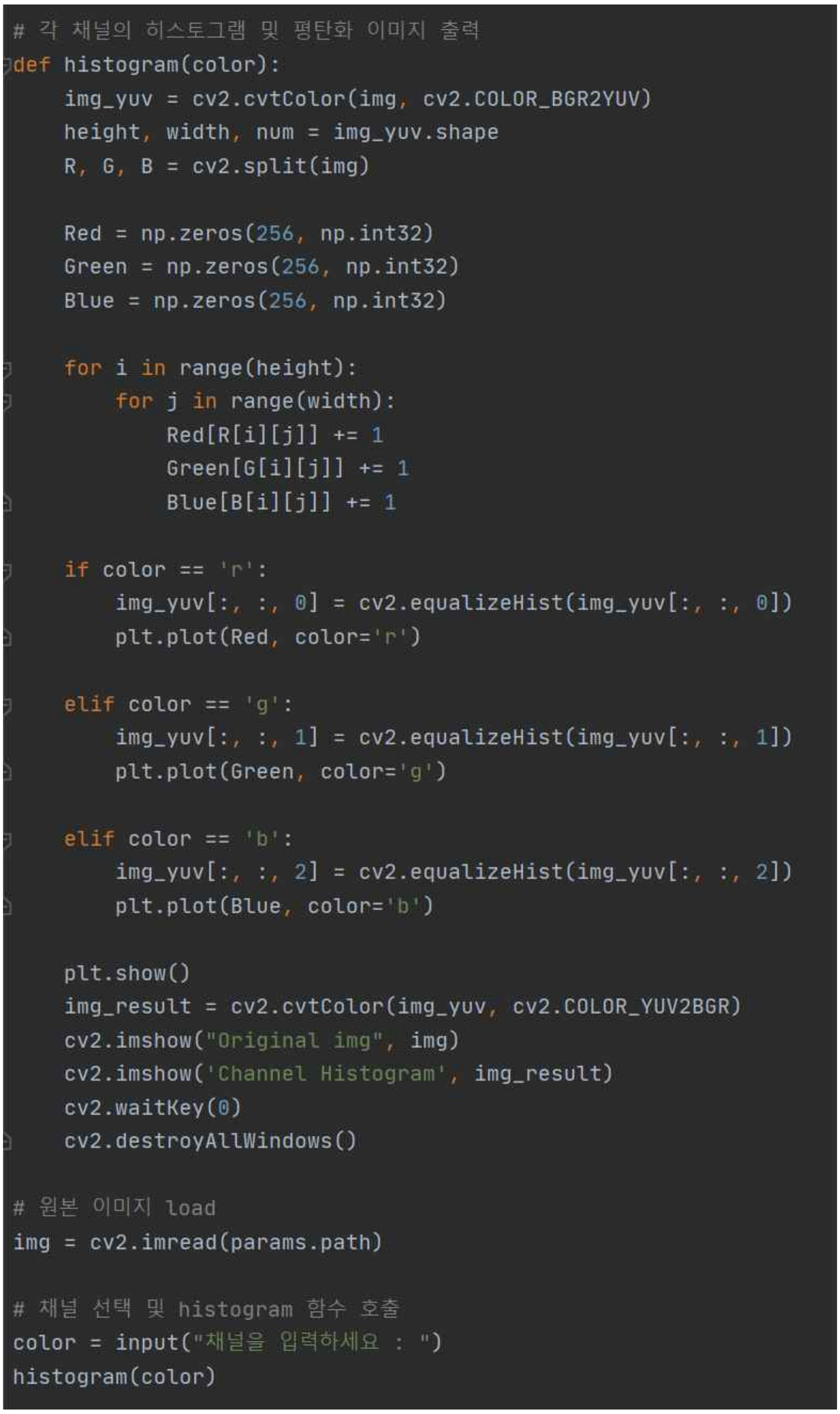
산업 컴퓨터비전 실제

2021254015 봉은정

1. 히스토그램 평탄화
   - 코드



```
# 각 채널의 히스토그램 및 평탄화 이미지 출력
def histogram(color):
    img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
    height, width, num = img_yuv.shape
    R, G, B = cv2.split(img)

    Red = np.zeros(256, np.int32)
    Green = np.zeros(256, np.int32)
    Blue = np.zeros(256, np.int32)

    for i in range(height):
        for j in range(width):
            Red[R[i][j]] += 1
            Green[G[i][j]] += 1
            Blue[B[i][j]] += 1

    if color == 'r':
        img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])
        plt.plot(Red, color='r')

    elif color == 'g':
        img_yuv[:, :, 1] = cv2.equalizeHist(img_yuv[:, :, 1])
        plt.plot(Green, color='g')

    elif color == 'b':
        img_yuv[:, :, 2] = cv2.equalizeHist(img_yuv[:, :, 2])
        plt.plot(Blue, color='b')

    plt.show()
    img_result = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    cv2.imshow("Original img", img)
    cv2.imshow('Channel Histogram', img_result)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# 원본 이미지 load
img = cv2.imread(params.path)

# 채널 선택 및 histogram 함수 호출
color = input("채널을 입력하세요 : ")
histogram(color)
```

(4) BGR에서 YUV로 변환

(5) 각 채널 별 히스토그램 추출

(6) 선택 채널 히스토그램 plot
평탄화 진행

(7) 원본 및 평탄화 이미지 출력

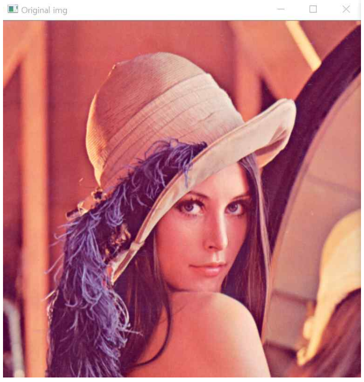
(1) Image Load

(2) r g b 채널 입력

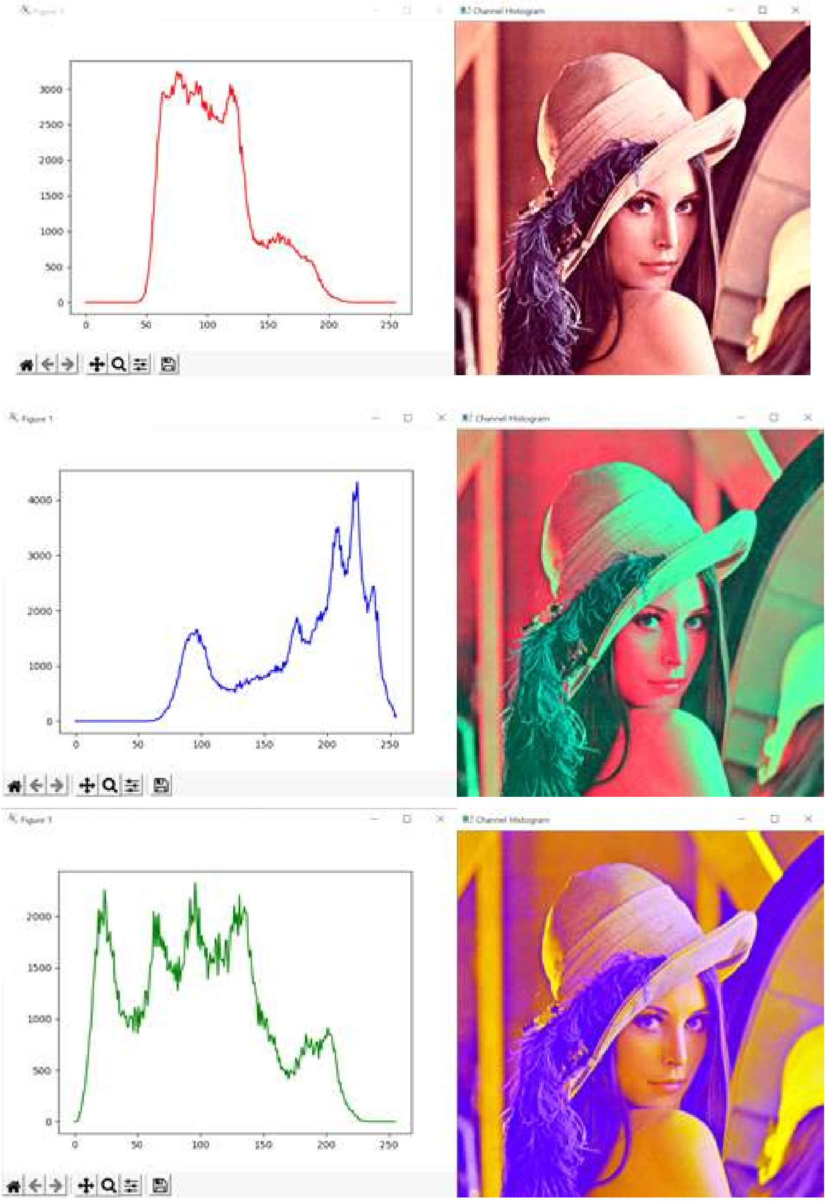
(3) histogram 함수 호출

- 실행 결과

Original Image

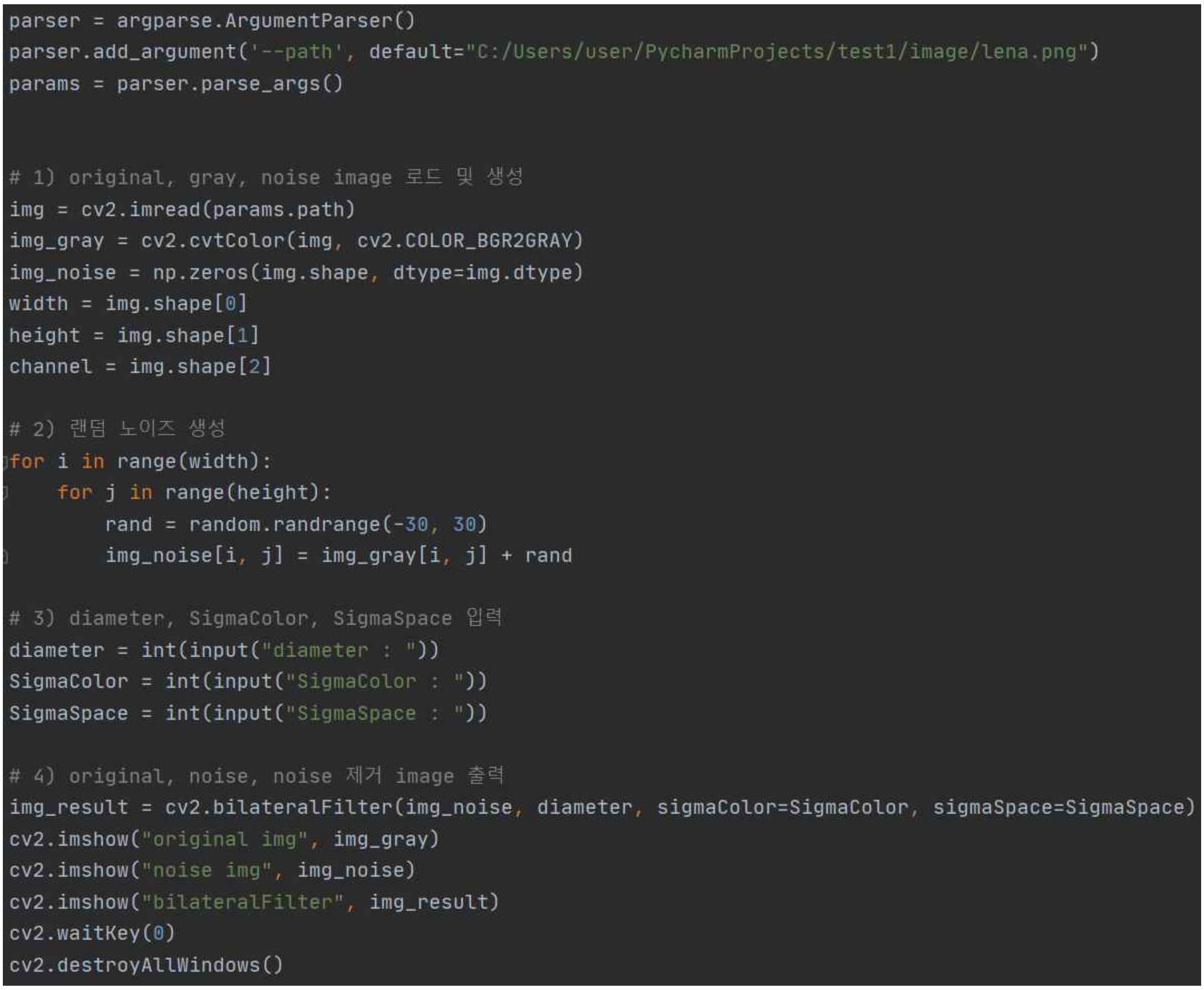


각 채널 히스토그램 및 평탄화 이미지 (R G B 순)



2. 공간 도메인 필터링

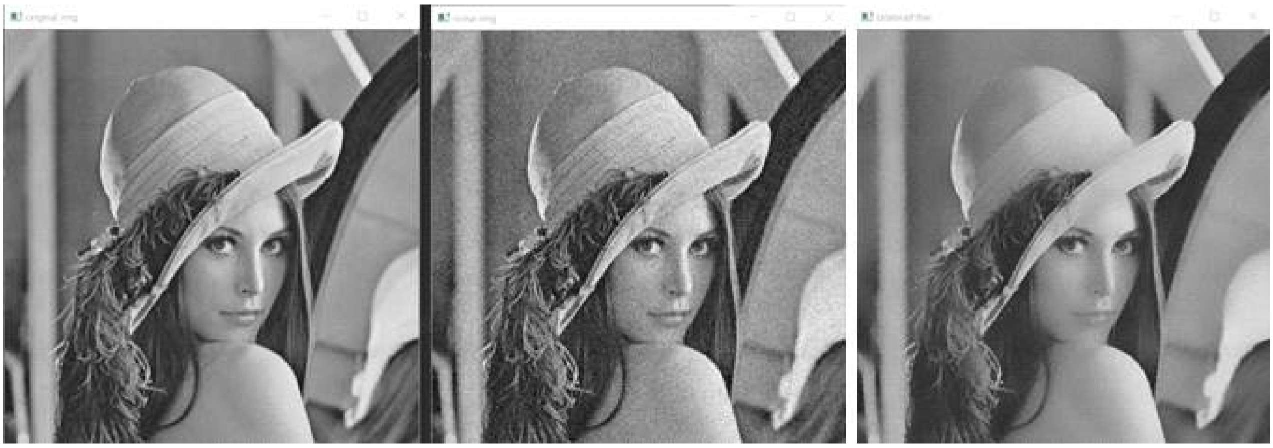
- **코드**



```
parser = argparse.ArgumentParser()
parser.add_argument('--path', default="C:/Users/user/PycharmProjects/test1/image/lena.png")
params = parser.parse_args()


# 1) original, gray, noise image 로드 및 생성
img = cv2.imread(params.path)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_noise = np.zeros(img.shape, dtype=img.dtype)
width = img.shape[0]
height = img.shape[1]
channel = img.shape[2]

# 2) 랜덤 노이즈 생성
for i in range(width):
    for j in range(height):
        rand = random.randrange(-30, 30)
        img_noise[i, j] = img_gray[i, j] + rand

# 3) diameter, SigmaColor, SigmaSpace 입력
diameter = int(input("diameter : "))
SigmaColor = int(input("SigmaColor : "))
SigmaSpace = int(input("SigmaSpace : "))

# 4) original, noise, noise 제거 image 출력
img_result = cv2.bilateralFilter(img_noise, diameter, sigmaColor=SigmaColor, sigmaSpace=SigmaSpace)
cv2.imshow("original img", img_gray)
cv2.imshow("noise img", img_noise)
cv2.imshow("bilateralFilter", img_result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

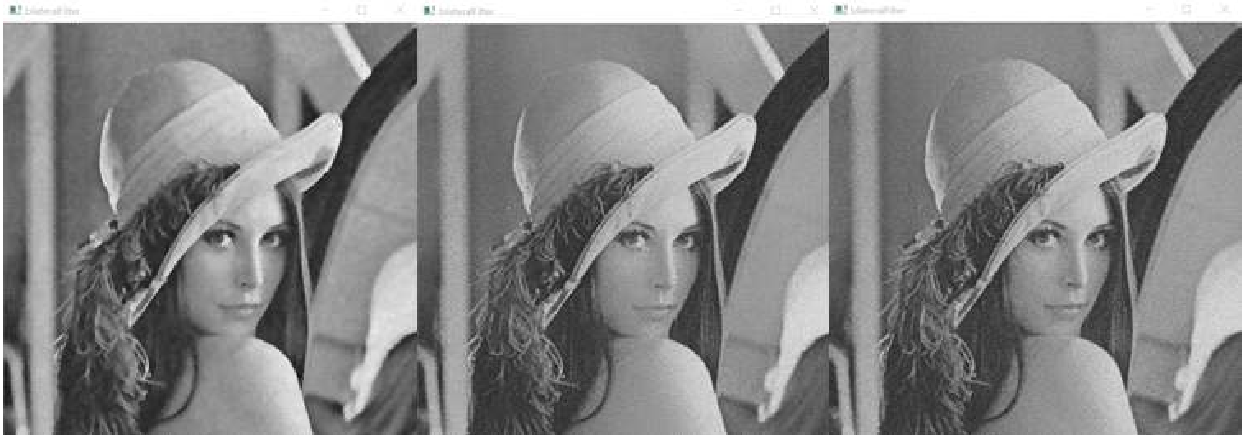
- **실행 결과**

Original, Noise, Result(노이즈 제거, Diameter 30, SigmaColor 100, SigmaSpace 50)

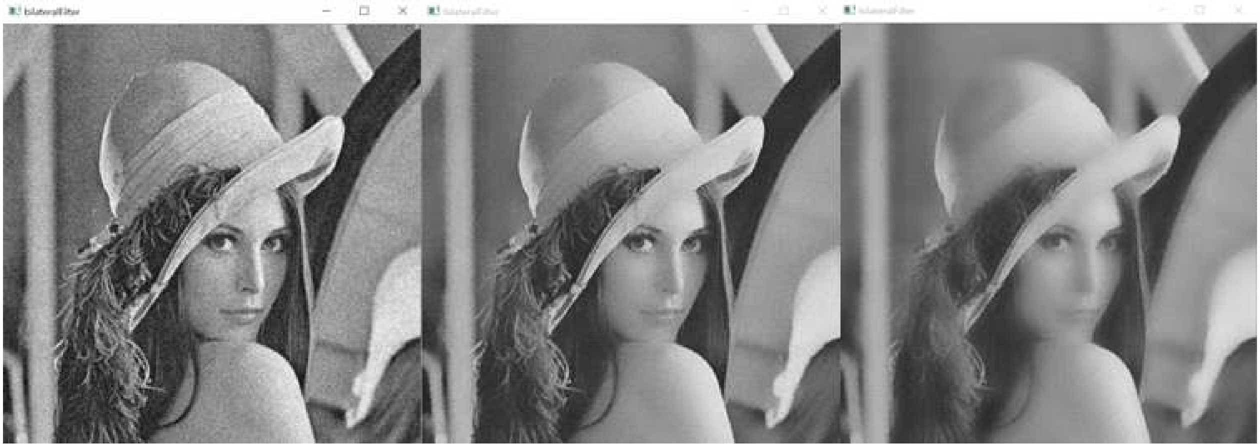


Diameter 30, SigmaColor 100, SigmaSpace 50을 기준으로 Parameter 변경

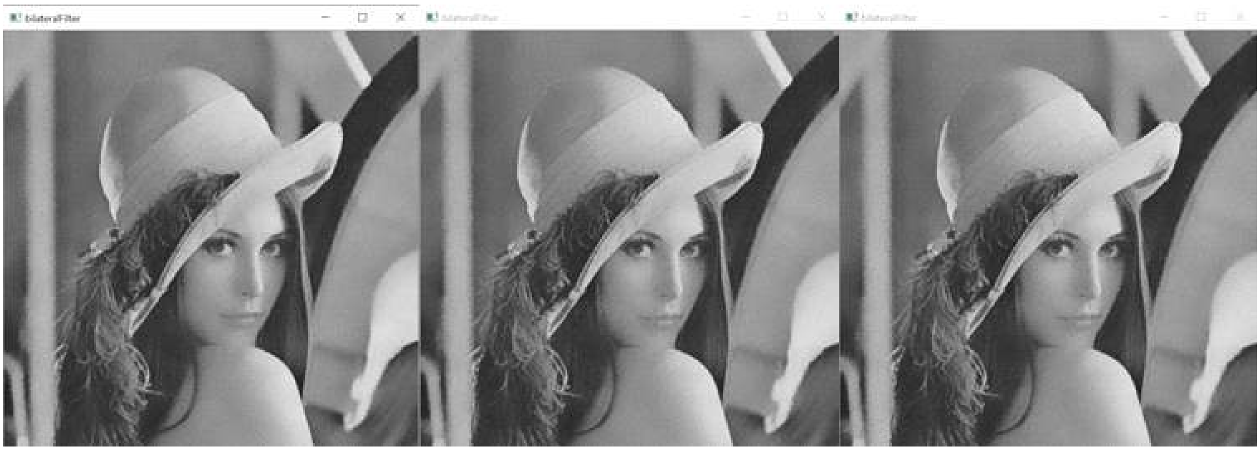
1) Diameter가 증가할수록 선명해짐 (순서대로 10 100 200)



2) SigmaColor가 증가할수록 흐려짐 (순서대로 10 100 200)

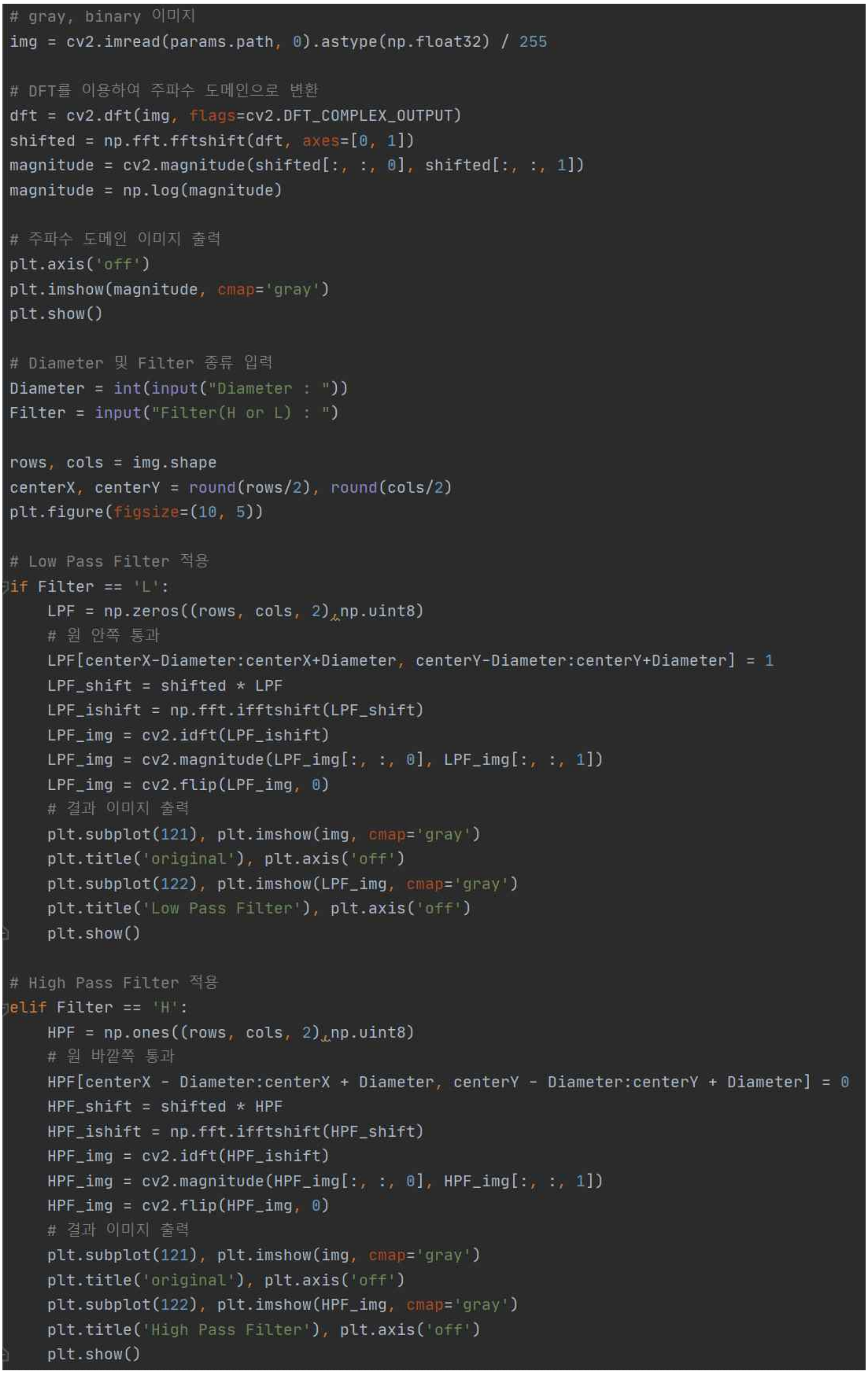


3) SigmaSpace 변화가 거의 없음 (순서대로 10 100 200)



3. 주파수 도메인 필터링

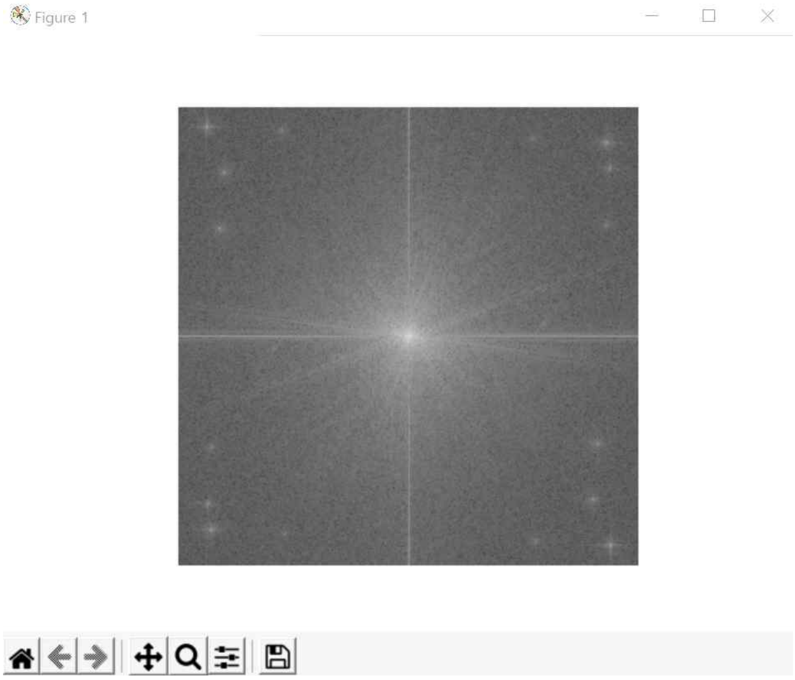
- **코드**



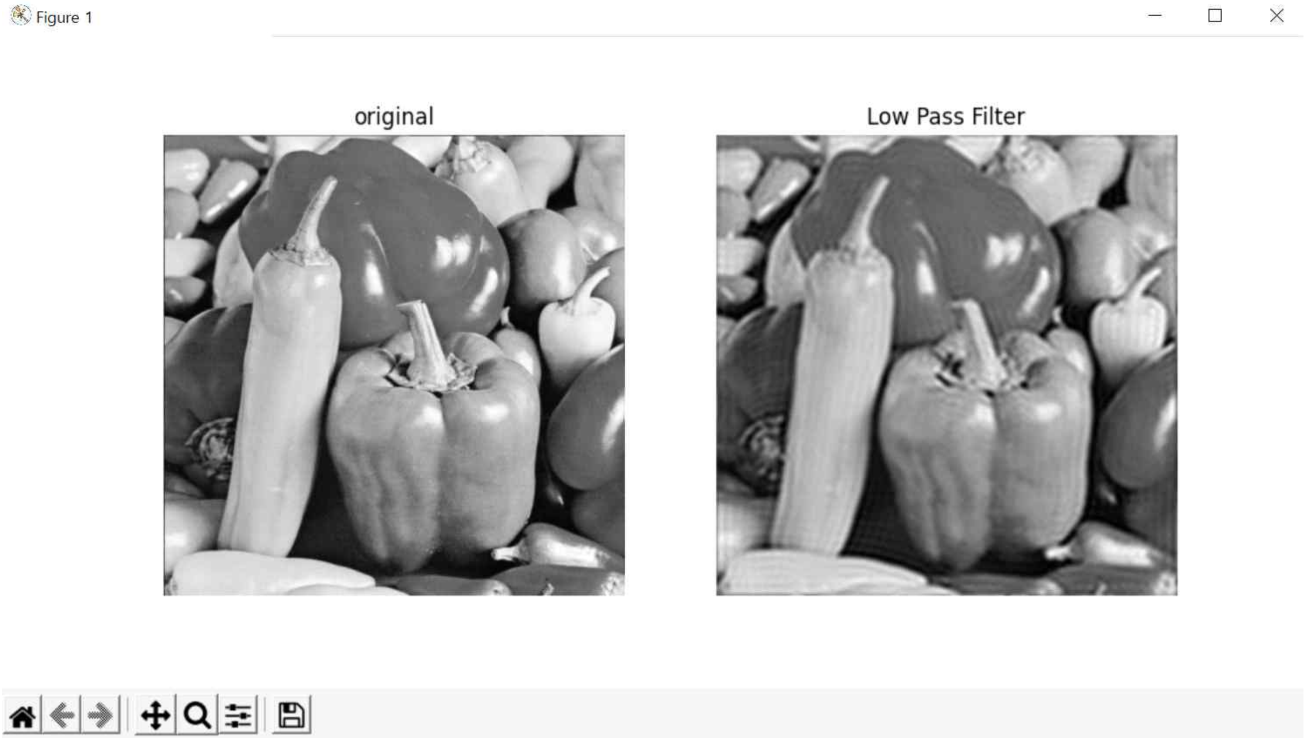
```
# gray, binary 이미지
img = cv2.imread(params.path, 0).astype(np.float32) / 255

# DFT를 이용하여 주파수 도메인으로 변환
dft = cv2.dft(img, flags=cv2.DFT_COMPLEX_OUTPUT)
shifted = np.fft.fftshift(dft, axes=[0, 1])
magnitude = cv2.magnitude(shifted[:, :, 0], shifted[:, :, 1])
magnitude = np.log(magnitude)

# 주파수 도메인 이미지 출력
plt.axis('off')
plt.imshow(magnitude, cmap='gray')
plt.show()

# Diameter 및 Filter 종류 입력
Diameter = int(input("Diameter : "))
Filter = input("Filter(H or L) : ")

rows, cols = img.shape
centerX, centerY = round(rows/2), round(cols/2)
plt.figure(figsize=(10, 5))

# Low Pass Filter 적용
if Filter == 'L':
    LPF = np.zeros((rows, cols, 2),np.uint8)
    # 원 안쪽 통과
    LPF[centerX-Diameter:centerX+Diameter, centerY-Diameter:centerY+Diameter] = 1
    LPF_shift = shifted * LPF
    LPF_ishift = np.fft.ifftshift(LPF_shift)
    LPF_img = cv2.idft(LPF_ishift)
    LPF_img = cv2.magnitude(LPF_img[:, :, 0], LPF_img[:, :, 1])
    LPF_img = cv2.flip(LPF_img, 0)
    # 결과 이미지 출력
    plt.subplot(121), plt.imshow(img, cmap='gray')
    plt.title('original'), plt.axis('off')
    plt.subplot(122), plt.imshow(LPF_img, cmap='gray')
    plt.title('Low Pass Filter'), plt.axis('off')
    plt.show()

# High Pass Filter 적용
elif Filter == 'H':
    HPF = np.ones((rows, cols, 2),np.uint8)
    # 원 바깥쪽 통과
    HPF[centerX - Diameter:centerX + Diameter, centerY - Diameter:centerY + Diameter] = 0
    HPF_shift = shifted * HPF
    HPF_ishift = np.fft.ifftshift(HPF_shift)
    HPF_img = cv2.idft(HPF_ishift)
    HPF_img = cv2.magnitude(HPF_img[:, :, 0], HPF_img[:, :, 1])
    HPF_img = cv2.flip(HPF_img, 0)
    # 결과 이미지 출력
    plt.subplot(121), plt.imshow(img, cmap='gray')
    plt.title('original'), plt.axis('off')
    plt.subplot(122), plt.imshow(HPF_img, cmap='gray')
    plt.title('High Pass Filter'), plt.axis('off')
    plt.show()
```

- 실행 결과

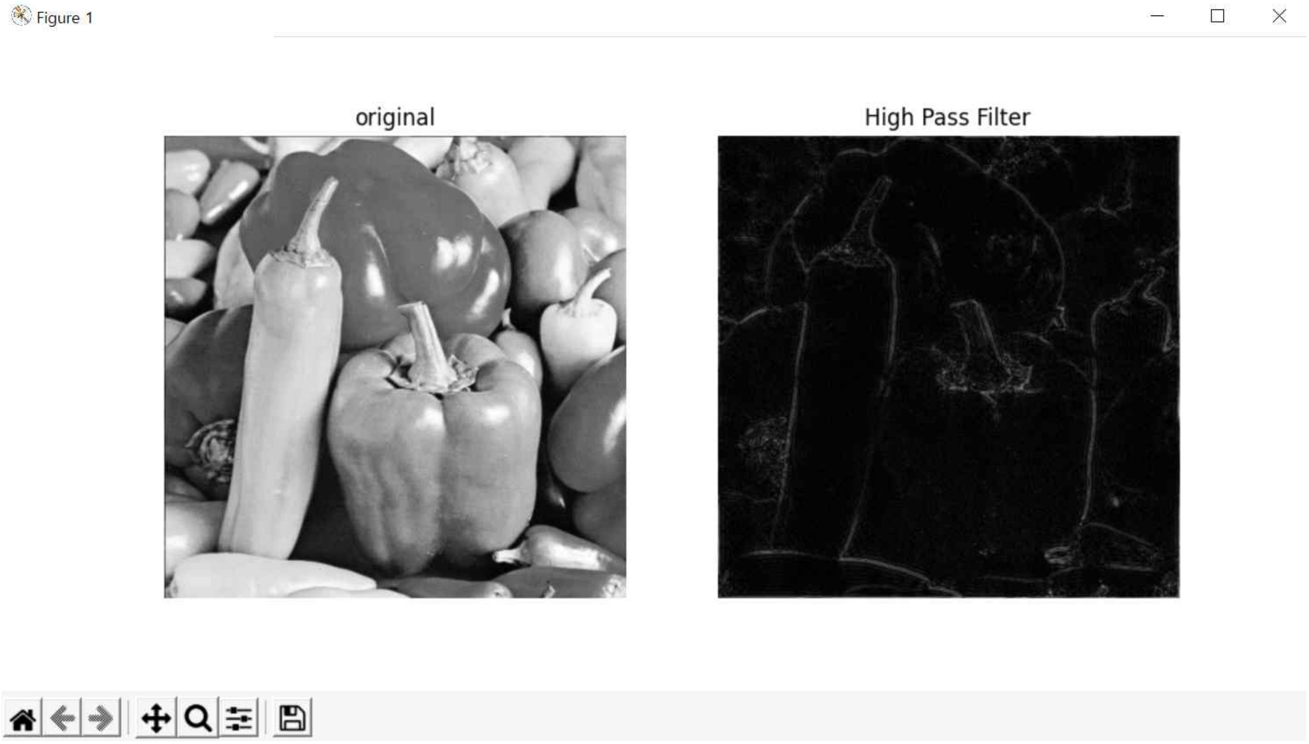
1) 주파수 도메인 이미지



2) LPF 통과 이미지

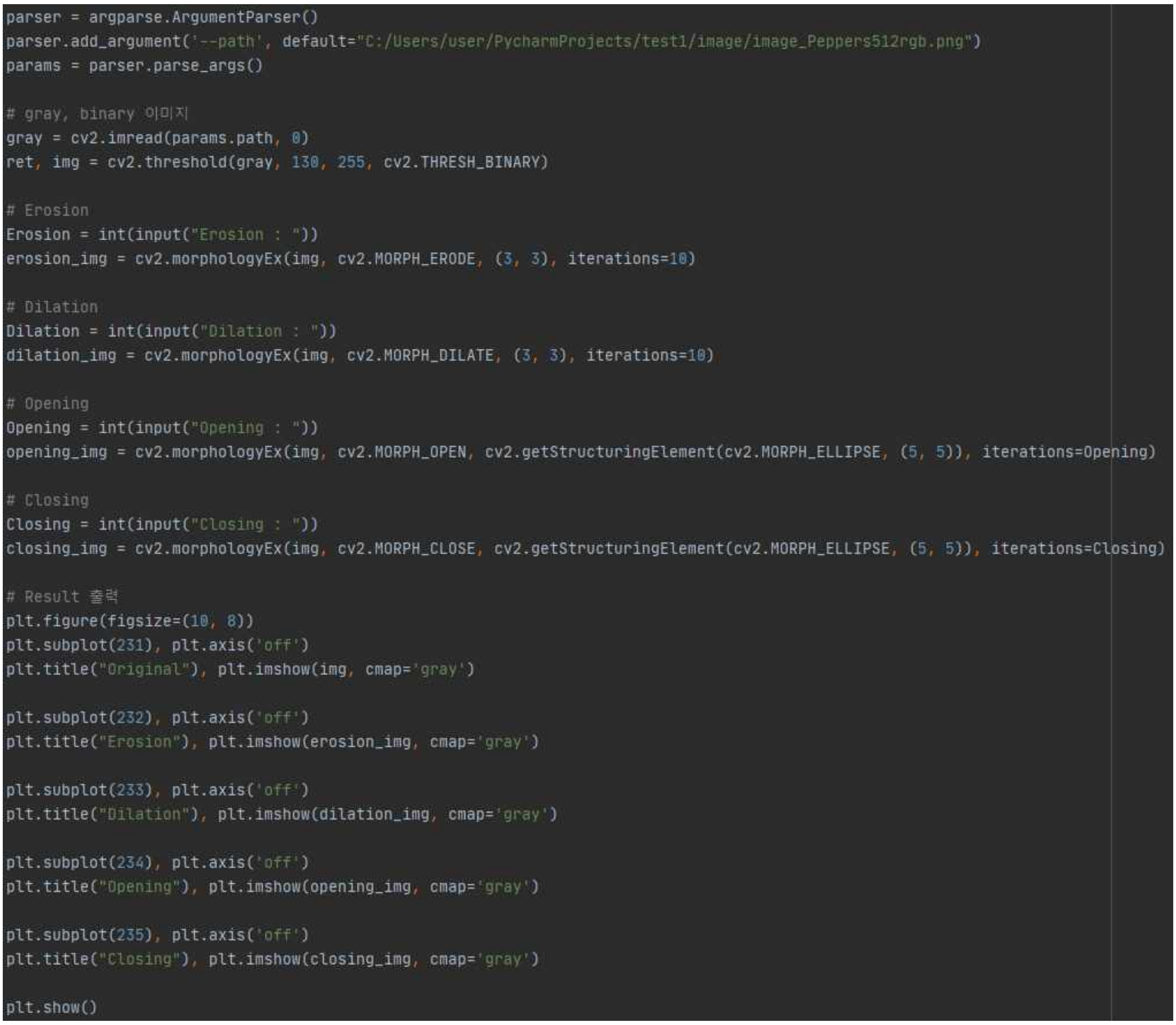


3) HPF 통과 이미지



4. 모폴로지 필터

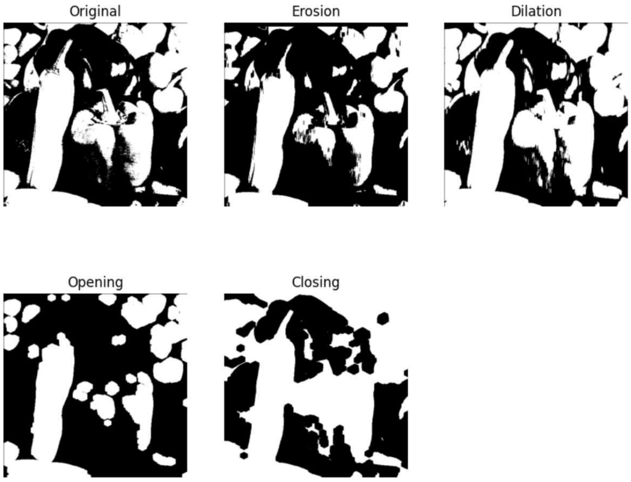
- **코드**



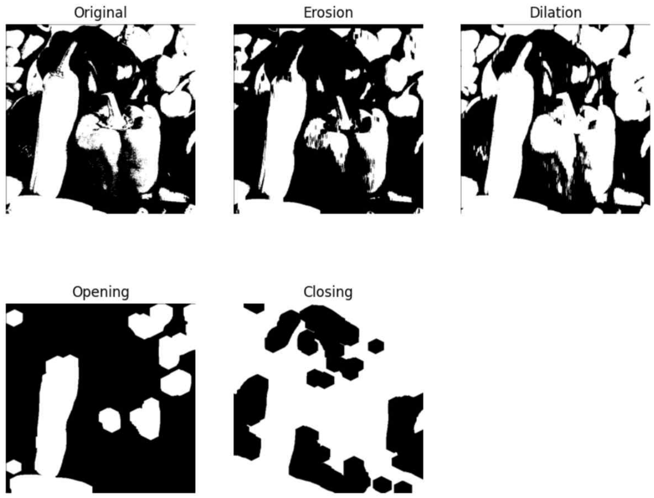
```
parser = argparse.ArgumentParser()
parser.add_argument('--path', default="C:/Users/user/PycharmProjects/test1/image/image_Peppers512rgb.png")
params = parser.parse_args()

# gray, binary 이미지
gray = cv2.imread(params.path, 0)
ret, img = cv2.threshold(gray, 130, 255, cv2.THRESH_BINARY)

# Erosion
Erosion = int(input("Erosion : "))
erosion_img = cv2.morphologyEx(img, cv2.MORPH_ERODE, (3, 3), iterations=10)

# Dilation
Dilation = int(input("Dilation : "))
dilation_img = cv2.morphologyEx(img, cv2.MORPH_DILATE, (3, 3), iterations=10)

# Opening
Opening = int(input("Opening : "))
opening_img = cv2.morphologyEx(img, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5)), iterations=Opening)

# Closing
Closing = int(input("Closing : "))
closing_img = cv2.morphologyEx(img, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5)), iterations=Closing)

# Result 출력
plt.figure(figsize=(10, 8))
plt.subplot(231), plt.axis('off')
plt.title("Original"), plt.imshow(img, cmap='gray')

plt.subplot(232), plt.axis('off')
plt.title("Erosion"), plt.imshow(erosion_img, cmap='gray')

plt.subplot(233), plt.axis('off')
plt.title("Dilation"), plt.imshow(dilation_img, cmap='gray')

plt.subplot(234), plt.axis('off')
plt.title("Opening"), plt.imshow(opening_img, cmap='gray')

plt.subplot(235), plt.axis('off')
plt.title("Closing"), plt.imshow(closing_img, cmap='gray')

plt.show()
```

- **실행 결과**

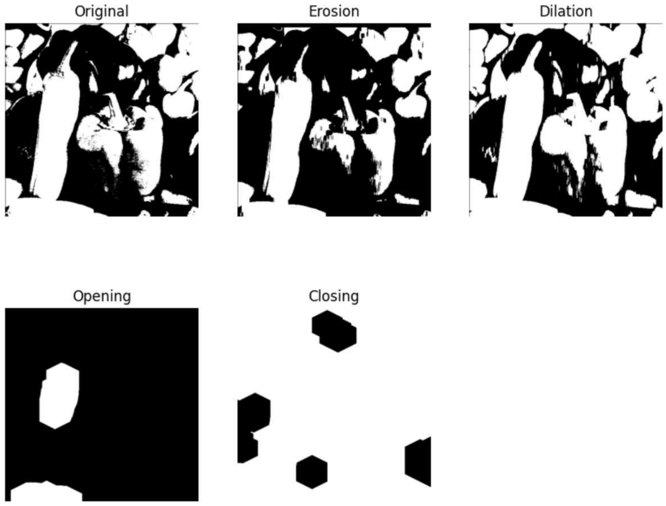
1) Erosion, Dilation, Opening, Closing 각각 5회 진행



2) Erosion, Dilation, Opening, Closing 각각 10회 진행



3) Erosion, Dilation, Opening, Closing 각각 20회 진행


Original
Erosion
Dilation
Opening
Closing