Computer Vision

실습 2주차

김 현 섭

Department of Computer Science and Engineering

Chungnam National University, Korea

실습 소개

- **과목 홈페이지**
  - DSC 사이버 캠퍼스 (http://ecampus. dscu.ac.kr)

- **TA 연락처**
  - 메일 보내실 때 [CV]를 제목에 붙여주세요
    - 김현섭
    - hyunseop95@gmail.com

목차

- 공지
- 실습
  - Image filtering
    - Filtering
    - Average filter
    - Sharpening filter
  - Padding
  - Gaussian filter
- 과제

Image filtering

- **Filtering**
  - filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])
    - src: 입력 이미지
    - ddepth: 이미지 깊이(자료형), -1일 경우 입력과 동일
    - kernel: 커널 행렬

0	0	0	0	0
15	16	0	0	0
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

*

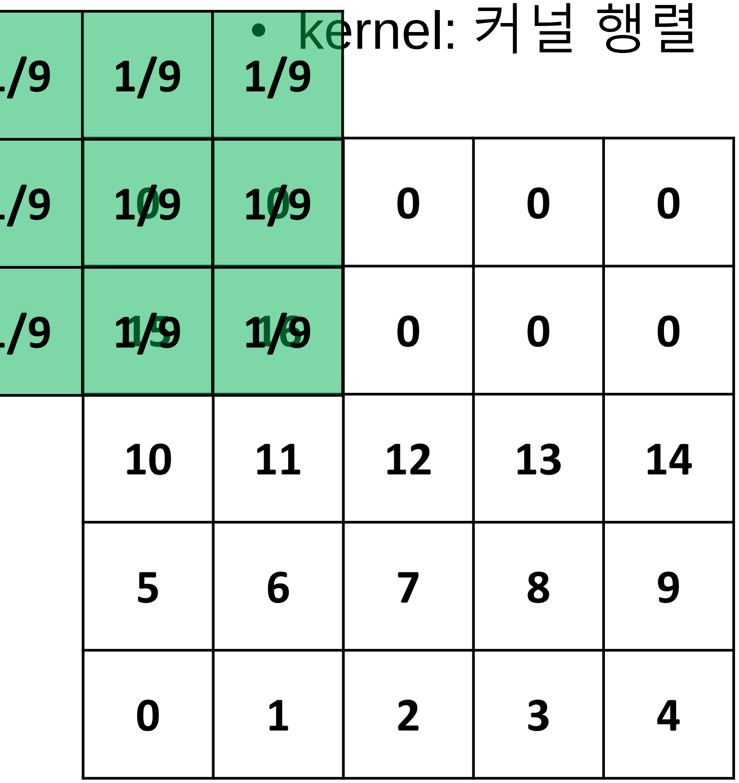
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

=

3	3	2	0	0
6	7	6	4	3
7	9	8	7	5
4	6	7	8	6
1	2	3	4	3

Image filtering

- **Filtering**
  - filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])
    - src: 입력 이미지
    - ddepth: 이미지 깊이(자료형), -1일 경우 입력과 동일
    - kernel: 커널 행렬



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

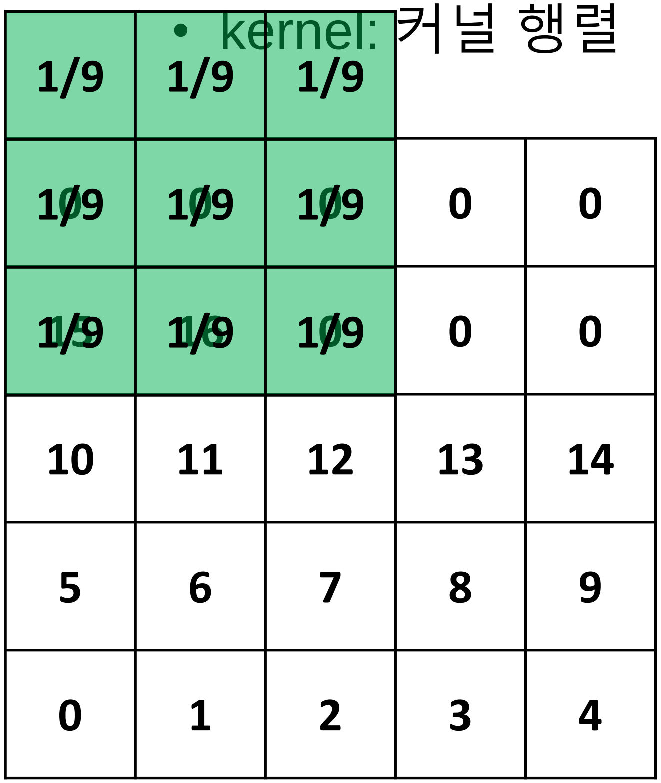
0	0	0	0	0
15	16	0	0	0
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

0×1/9 +
0×1/9 +
… +
15×1/9 +
16×1/9 = 3

3	3	2	0	0
6	7	6	4	3
7	9	8	7	5
4	6	7	8	6
1	2	3	4	3

Image filtering

- **Filtering**
  - filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])
    - src: 입력 이미지
    - ddepth: 이미지 깊이(자료형), -1일 경우 입력과 동일
    - kernel: 커널 행렬



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

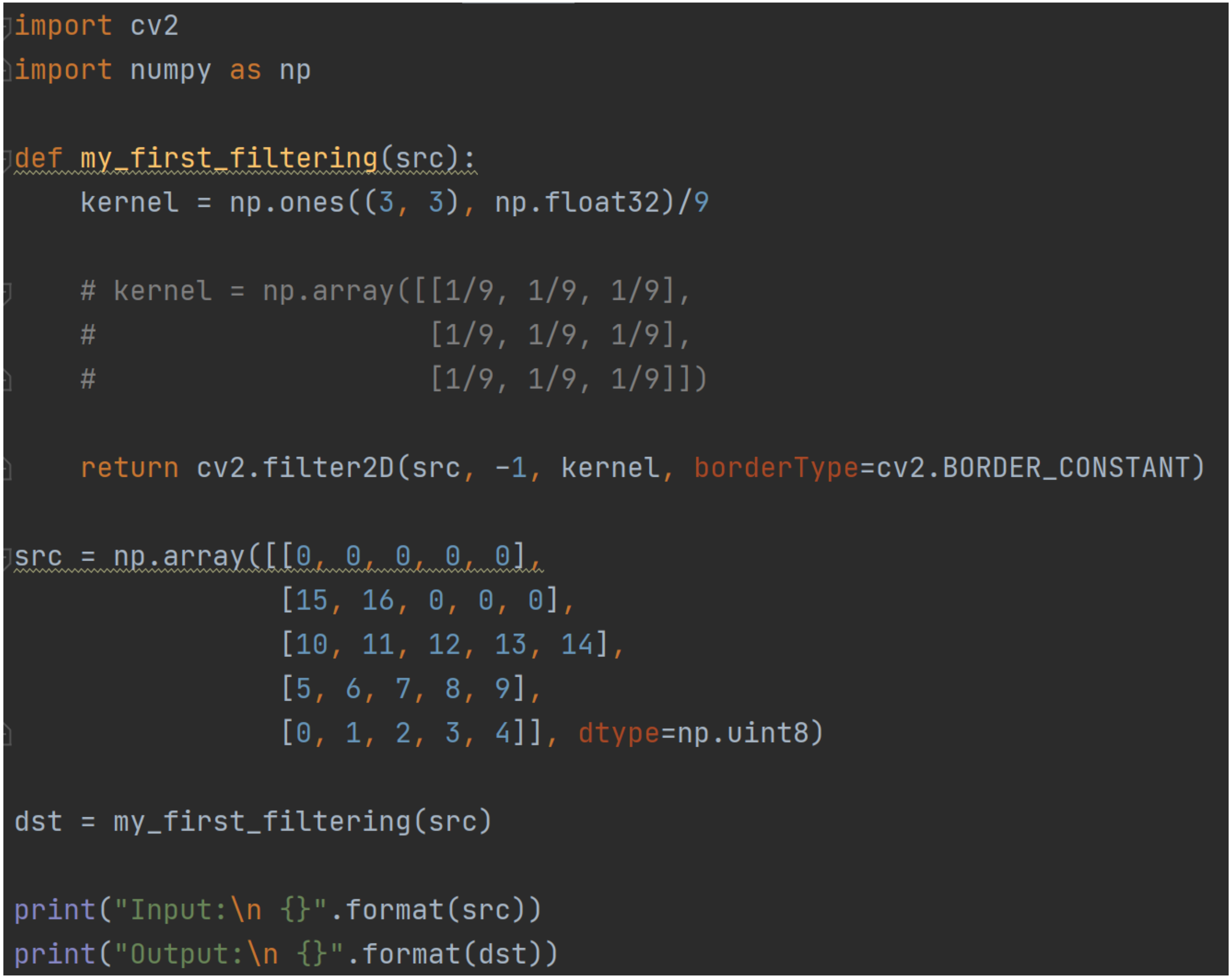
0	0	0	0	0
15	16	0	0	0
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

0×1/9 +
0×1/9 +
… +
16×1/9 +
0×1/9 = 3

3	3	2	0	0
6	7	6	4	3
7	9	8	7	5
4	6	7	8	6
1	2	3	4	3

Image filtering

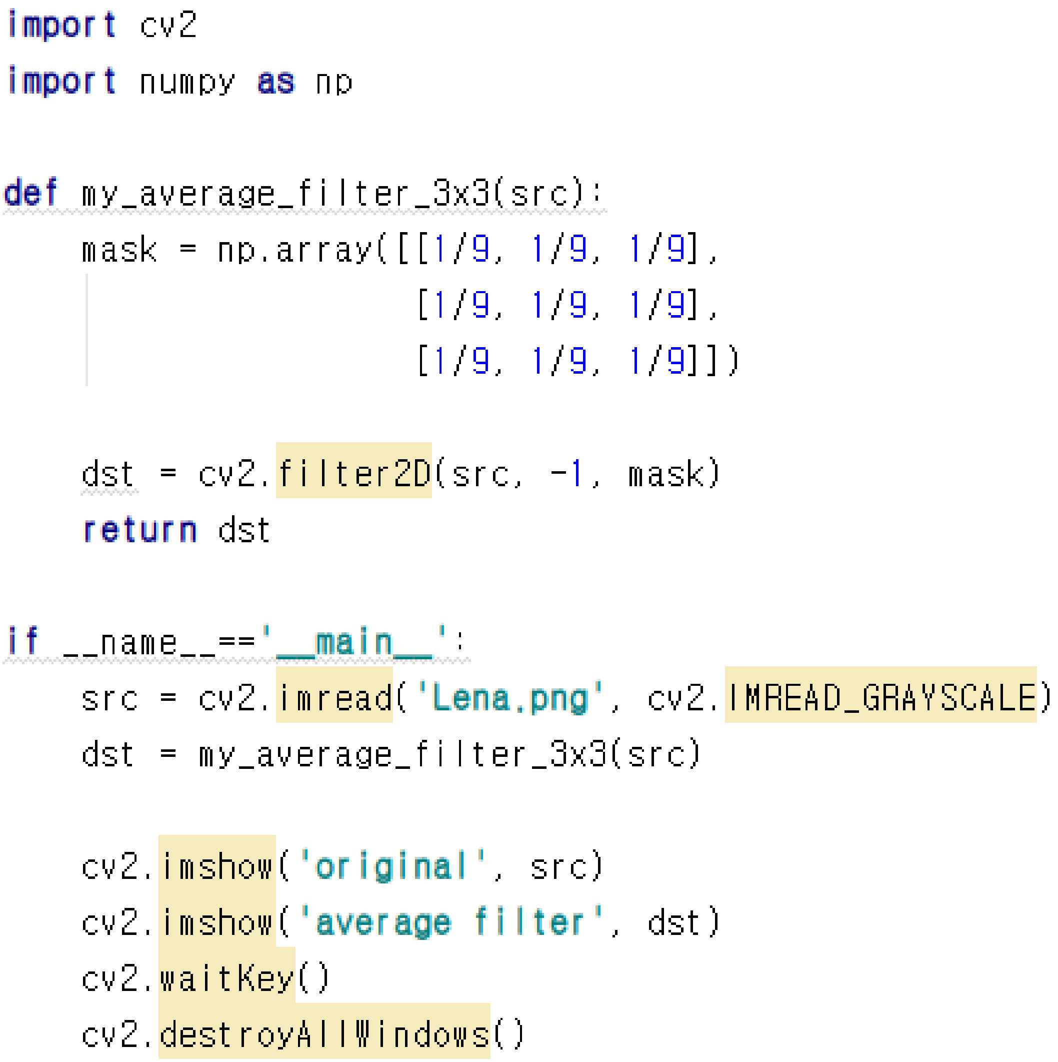
- **Filtering**



```
import cv2
import numpy as np

def my_first_filtering(src):
    kernel = np.ones((3, 3), np.float32)/9

    # kernel = np.array([[1/9, 1/9, 1/9],
    #                    [1/9, 1/9, 1/9],
    #                    [1/9, 1/9, 1/9]])

    return cv2.filter2D(src, -1, kernel, borderType=cv2.BORDER_CONSTANT)

src = np.array([[0, 0, 0, 0, 0],
                [15, 16, 0, 0, 0],
                [10, 11, 12, 13, 14],
                [5, 6, 7, 8, 9],
                [0, 1, 2, 3, 4]], dtype=np.uint8)

dst = my_first_filtering(src)

print("Input:\n {}".format(src))
print("Output:\n {}".format(dst))
```

Image filtering

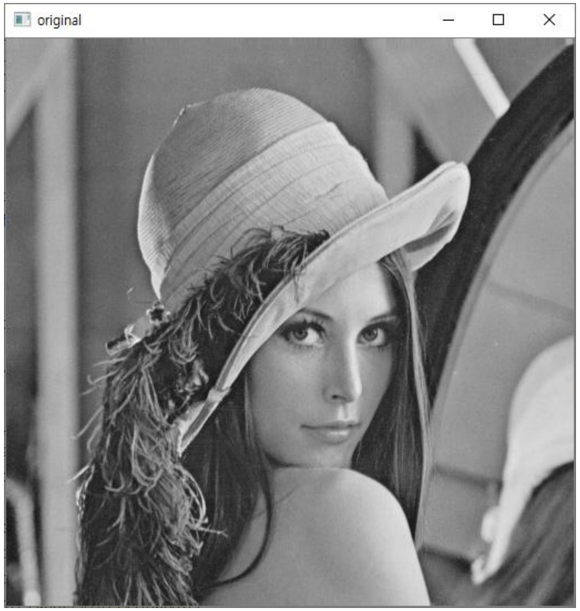
- **Average filtering**
  - 평균값 필터를 이미지에 적용하고 비교해보기



```python
import cv2
import numpy as np

def my_average_filter_3x3(src):
    mask = np.array([[1/9, 1/9, 1/9],
                     [1/9, 1/9, 1/9],
                     [1/9, 1/9, 1/9]])

    dst = cv2.filter2D(src, -1, mask)
    return dst

if __name__=='__main__':
    src = cv2.imread('Lena.png', cv2.IMREAD_GRAYSCALE)
    dst = my_average_filter_3x3(src)

    cv2.imshow('original', src)
    cv2.imshow('average filter', dst)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

Image filtering

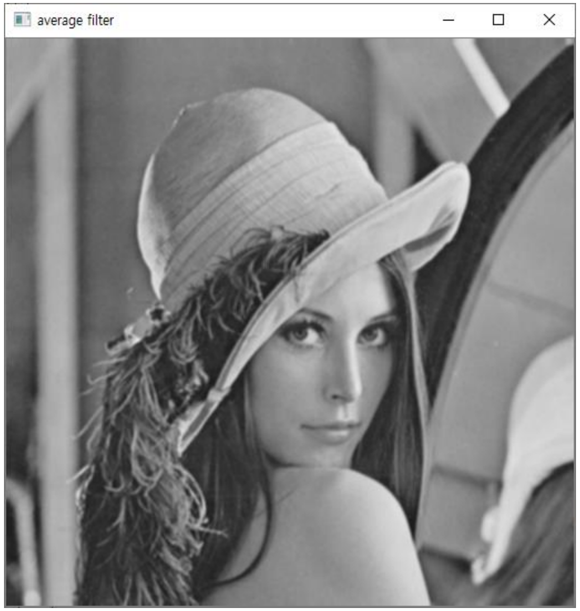
- **Average filtering**
  - Average filter를 이미지에 적용하고 비교해보기



original

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

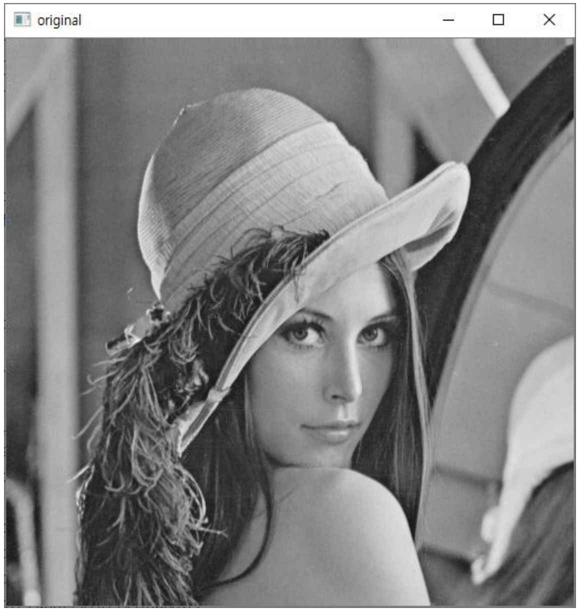
mask



average filter

Image filtering

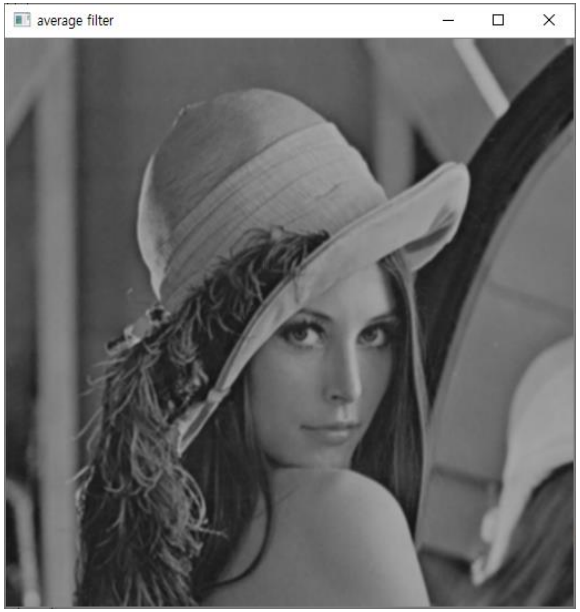
- **Average filtering**



original

1/12	1/12	1/12
1/12	1/12	1/12
1/12	1/12	1/12

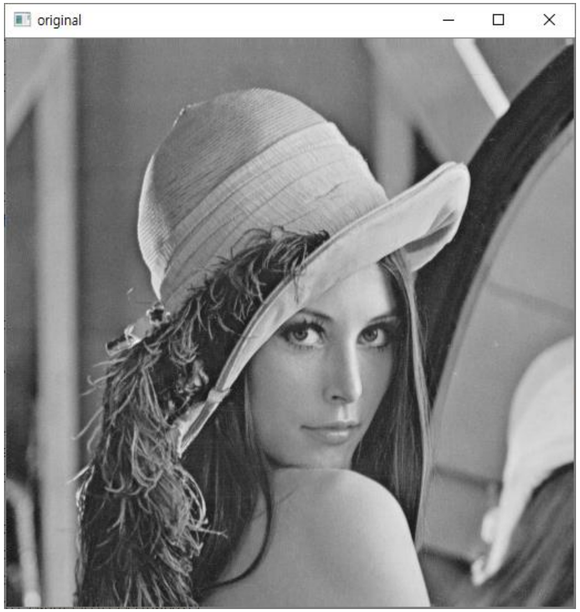
mask



average filter

Image filtering

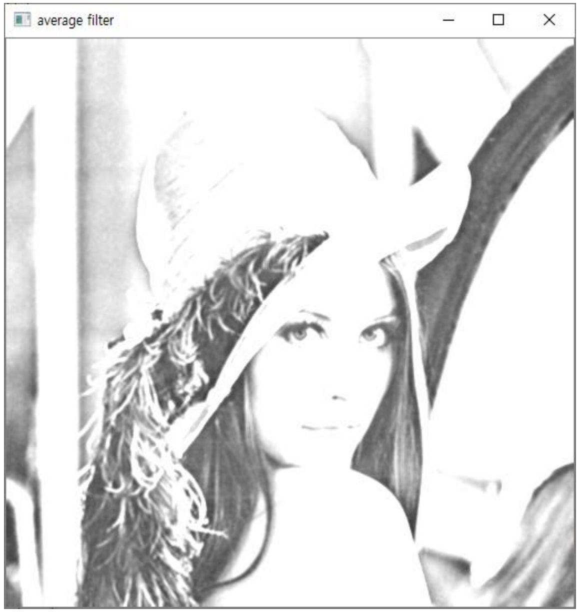
- **Average filtering**



original

1/4	1/4	1/4
1/4	1/4	1/4
1/4	1/4	1/4

mask



average filter

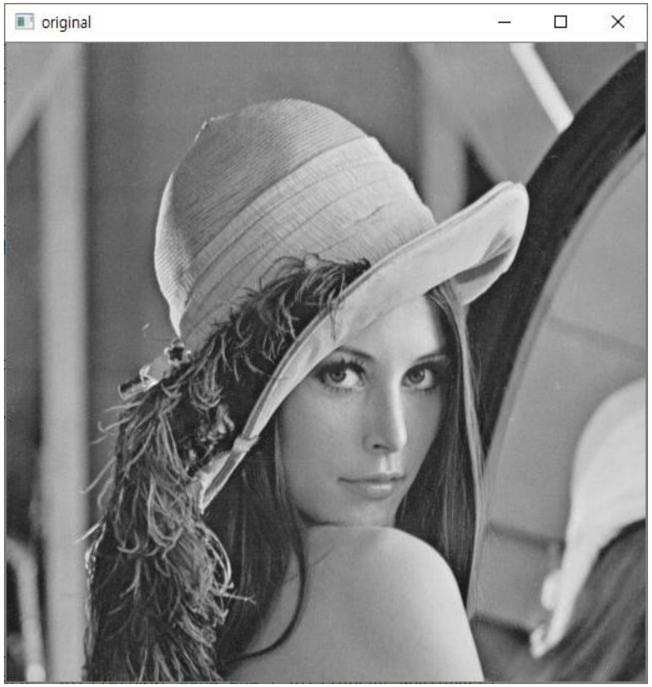
Image filtering

- **Sharpening filtering**
  - Sharpening filter를 적용하고 결과 확인하기
  - 이미지를 선명하게 해주는 효과

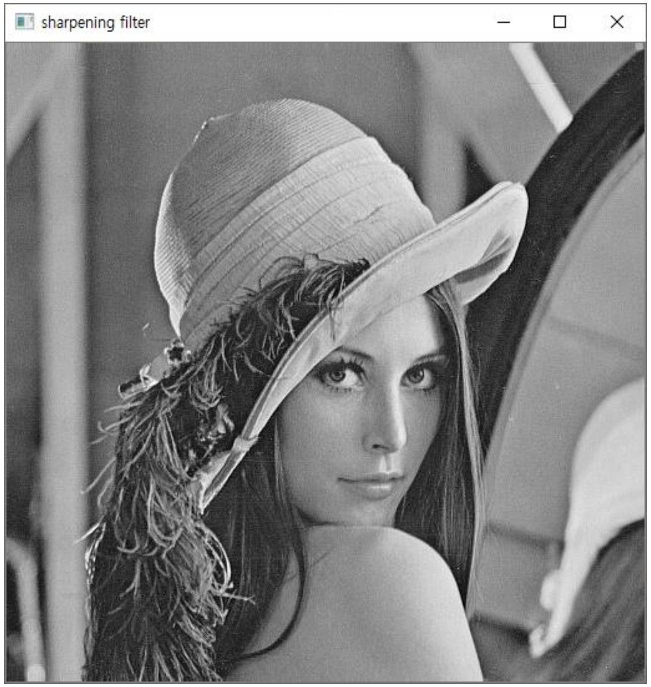
$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \begin{array}{|c|c|c|}\hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline \end{array} = \underset{\text{mask}}{\begin{array}{|c|c|c|}\hline -1/9 & -1/9 & -1/9 \\ \hline -1/9 & 17/9 & -1/9 \\ \hline -1/9 & -1/9 & -1/9 \\ \hline \end{array}}$$

Image filtering

- **Sharpening filtering**
  - Sharpening filter를 적용하고 결과 확인하기
  - 이미지를 선명하게 해주는 효과



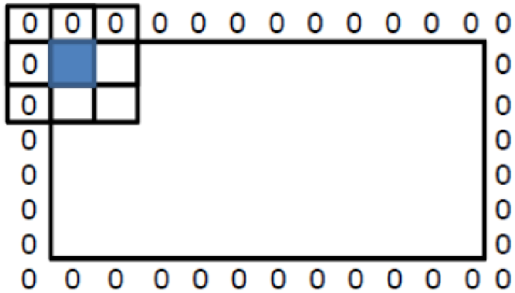
original



3x3 sharpening filter

Padding

- **Padding**
  - 실제 이미지에서 없는 가장자리 부분을 채우는 역할
  - Zero padding
    - 주변 값을 0으로 채움



0 0 0 0 0 0 0 0 0 0 0 0 0

0 0

0 0

0 0

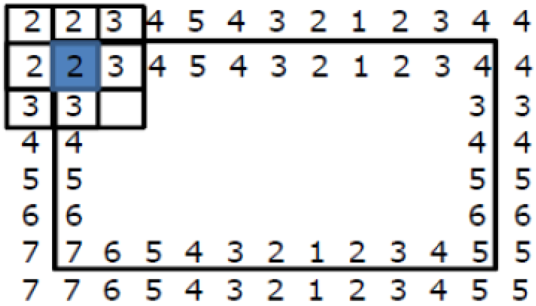
0 0

0 0

0 0

0 0 0 0 0 0 0 0 0 0 0 0 0

  - Repetition padding
    - 주변 값을 가장자리의 값을 복사하여 채움



2 2 3 4 5 4 3 2 1 2 3 4 4

2 2 3 4 5 4 3 2 1 2 3 4 4

3 3 3 3

4 4 4 4

5 5 5 5

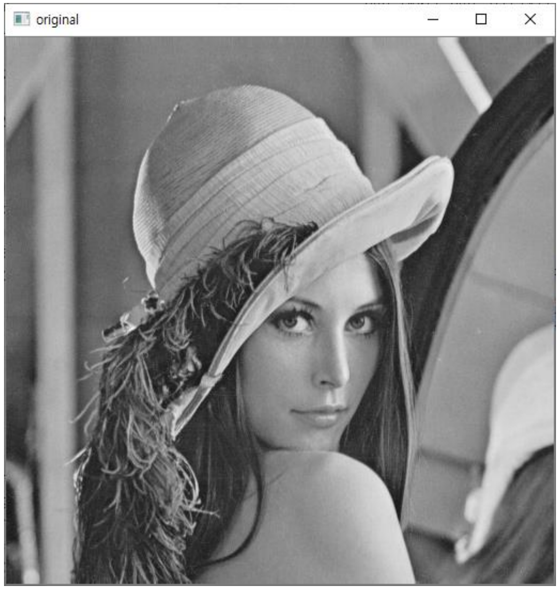
6 6 6 6

7 7 6 5 4 3 2 1 2 3 4 5 5

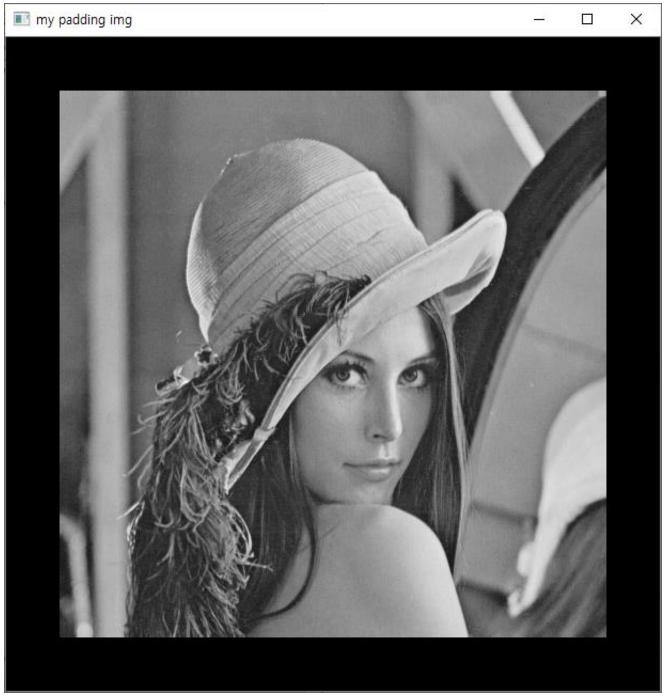
7 7 6 5 4 3 2 1 2 3 4 5 5

Padding

- **Zero padding**
  - 주변 값을 0으로 채움



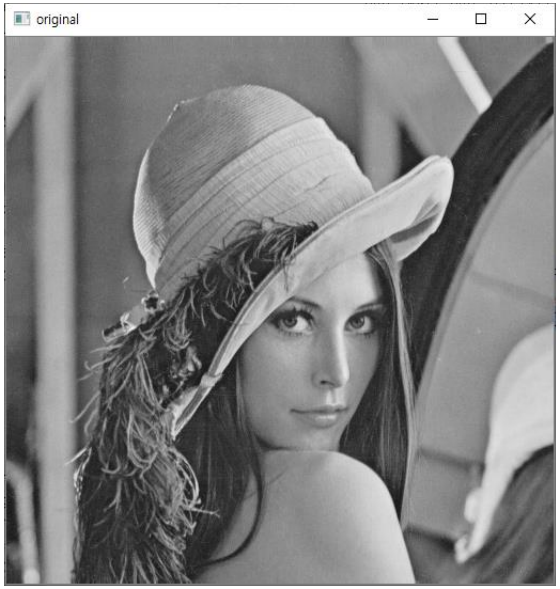
original



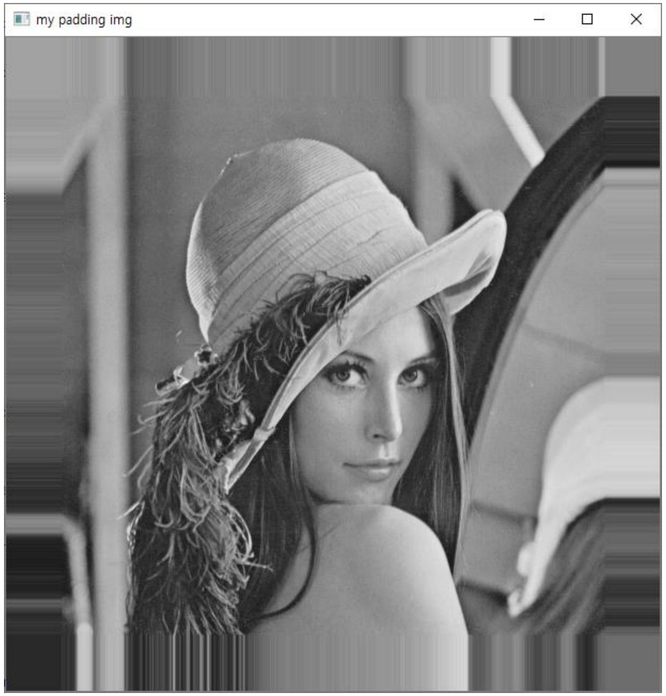
zero padding

Padding

- **Repetition padding**
  - 가장 자리의 값을 복사해옴



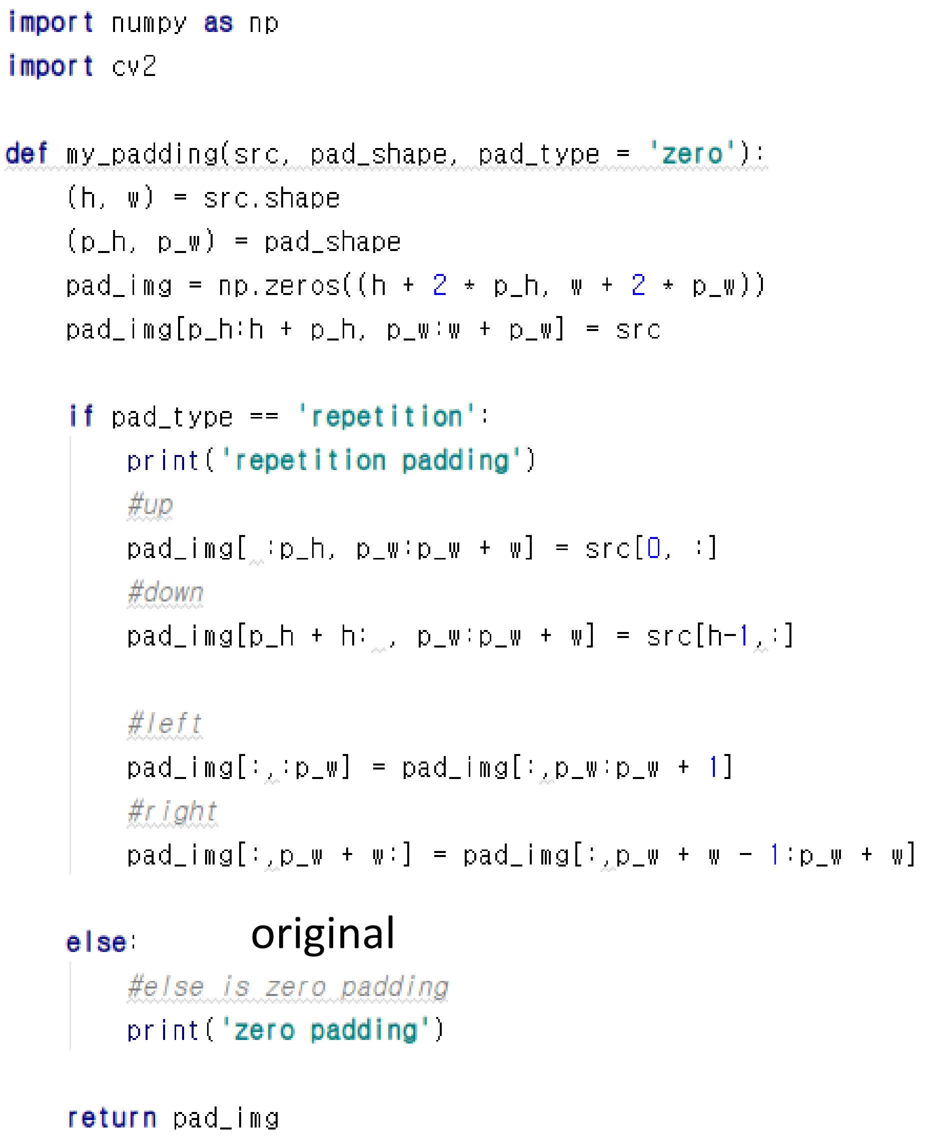
original



repetition padding

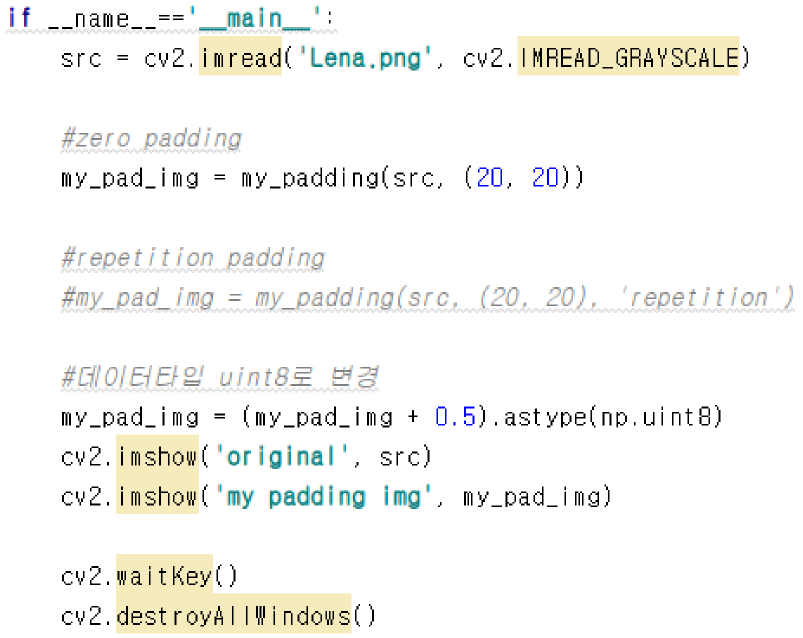
Padding

- 실습



```python
import numpy as np
import cv2

def my_padding(src, pad_shape, pad_type = 'zero'):
    (h, w) = src.shape
    (p_h, p_w) = pad_shape
    pad_img = np.zeros((h + 2 * p_h, w + 2 * p_w))
    pad_img[p_h:h + p_h, p_w:w + p_w] = src

    if pad_type == 'repetition':
        print('repetition padding')
        #up
        pad_img[ :p_h, p_w:p_w + w] = src[0, :]
        #down
        pad_img[p_h + h: , p_w:p_w + w] = src[h-1,:]

        #left
        pad_img[:,:p_w] = pad_img[:,p_w:p_w + 1]
        #right
        pad_img[:,p_w + w:] = pad_img[:,p_w + w - 1:p_w + w]

    else:
        #else is zero padding
        print('zero padding')

    return pad_img
```

original



```python
if __name__=='__main__':
    src = cv2.imread('Lena.png', cv2.IMREAD_GRAYSCALE)

    #zero padding
    my_pad_img = my_padding(src, (20, 20))

    #repetition padding
    #my_pad_img = my_padding(src, (20, 20), 'repetition')

    #데이터타입 uint8로 변경
    my_pad_img = (my_pad_img + 0.5).astype(np.uint8)
    cv2.imshow('original', src)
    cv2.imshow('my padding img', my_pad_img)

    cv2.waitKey()
    cv2.destroyAllWindows()
```

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현

$$\underset{\text{output}}{h[m,n]} = \sum_{k,l} \underset{\text{filter}}{g[k,l]} \; \underset{\text{image (signal)}}{f[m+k,n+l]}$$

0	0	0	0	0
15	16	0	0	0
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

Image

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

filter

$$m = ?$$
$$n = ?$$
$$k = ?$$
$$l = ?$$

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현

$$h[m,n]=\sum_{k,l} g[k,l]\; f[m+k,n+l]$$

output filter image (signal)

	0	0	0	0	0	
	15	16	0	0	0	
	10	11	12	13	14	
	5	6	7	8	9	
	0	1	2	3	4	

Image

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

filter

$$h[0,0] = ?$$

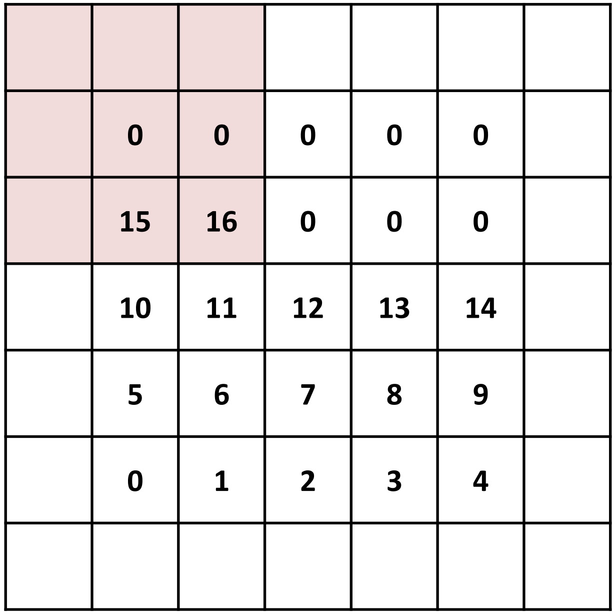
?				

output

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현

$$\underset{\text{output}}{h[m,n]} = \sum_{k,l} \underset{\text{filter}}{g[k,l]}\ \underset{\text{image (signal)}}{f[m+k,n+l]}$$



	0	0	0	0	0	
	15	16	0	0	0	
	10	11	12	13	14	
	5	6	7	8	9	
	0	1	2	3	4	

Image

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

filter

$$h[0,0] = \sum_{k=0}^{2}\sum_{l=0}^{2} g[k,l]\, f[k,l]$$

?				

output

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현

$$\underset{\text{output}}{h[m,n]} = \sum_{k,l} \underset{\text{filter}}{g[k,l]}\ \underset{\text{image (signal)}}{f[m+k,n+l]}$$

	0	0	0	0	0	
	15	16	0	0	0	
	10	11	12	13	14	
	5	6	7	8	9	
	0	1	2	3	4	

Image = f

(0,0) 1/9	(0,1) 1/9	(0,2) 1/9
(1,0) 1/9	(1,1) 1/9	(1,2) 1/9
(2,0) 1/9	(2,1) 1/9	(2,2) 1/9

Filter = g

$$h[0,0] = \sum_{k=0}^{2} \sum_{l=0}^{2} g[k,l]\, f[k,l]$$

?				

Output = h

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현

$$\underset{\text{output}}{h[m,n]} = \sum_{k,l} \underset{\text{filter}}{g[k,l]}\ \underset{\text{image (signal)}}{f[m+k,n+l]}$$

(0,0)	(0,1)	(0,2)	(0,3)			
(1,0)	(1,1) 0	(1,2) 0	(1,3) 0	0	0	
(2,0)	(2,1) 15	(2,2) 16	(2,3) 0	0	0	
	10	11	12	13	14	
	5	6	7	8	9	
	0	1	2	3	4	

Image

(0,0) 1/9	(0,1) 1/9	(0,2) 1/9
(1,0) 1/9	(1,1) 1/9	(1,2) 1/9
(2,0) 1/9	(2,1) 1/9	(2,2) 1/9

filter

$h[0,0] =$
$g[0,0]f[0,0]$ +
$g[0,1]f[0,1]$ +
$g[0,2]f[0,2]$ +
$g[1,0]f[1,0]$ +
… +
$g[2,1]f[2,1]$ +
$g[2,2]f[2,2]$

?				

output

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현

$$h[m,n] = \sum_{k,l} g[k,l]\; f[m+k,n+l]$$

output filter image (signal)

(0,0)	(0,1)	(0,2)	(0,3)			
(1,0)	(1,1) 0	(1,2) 0	(1,3) 0	0	0	
(2,0)	(2,1) 15	(2,2) 16	(2,3) 0	0	0	
	10	11	12	13	14	
	5	6	7	8	9	
	0	1	2	3	4	

Image

(0,0) 1/9	(0,1) 1/9	(0,2) 1/9
(1,0) 1/9	(1,1) 1/9	(1,2) 1/9
(2,0) 1/9	(2,1) 1/9	(2,2) 1/9

filter

3	?			

output

$$h[0,1] = \sum_{k=0}^{2}\sum_{l=0}^{2} g[k,l]\, f[k,1+l]$$

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

output filter image (signal)

(0,0)	(0,1)	(0,2)	(0,3)			
(1,0)	(1,1) 0	(1,2) 0	(1,3) 0	0	0	
(2,0)	(2,1) 15	(2,2) 16	(2,3) 0	0	0	
	10	11	12	13	14	
	5	6	7	8	9	
	0	1	2	3	4	

Image

(0,0) 1/9	(0,1) 1/9	(0,2) 1/9
(1,0) 1/9	(1,1) 1/9	(1,2) 1/9
(2,0) 1/9	(2,1) 1/9	(2,2) 1/9

filter

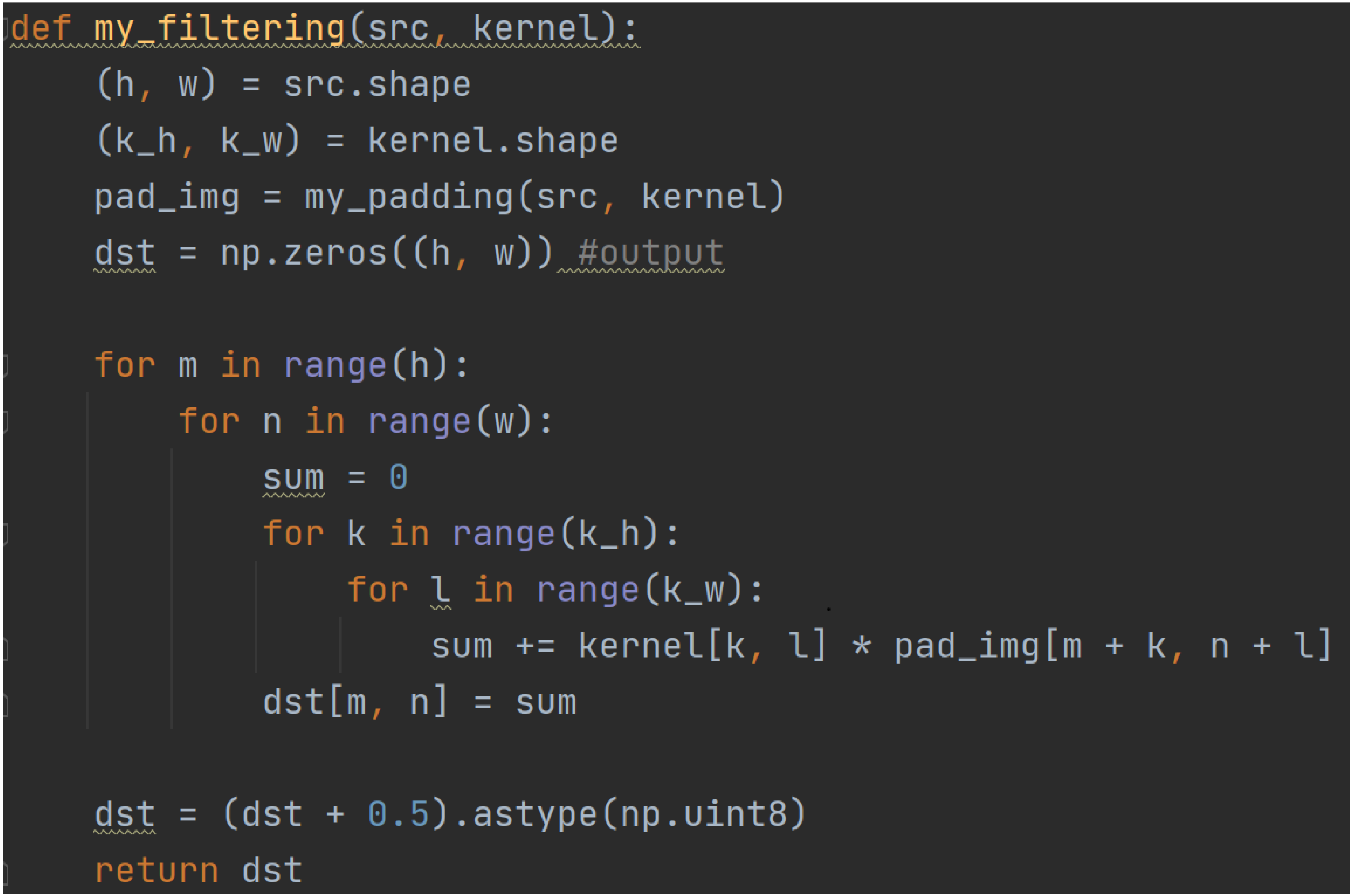
$h[0,1] = g[0,0]f[0,1]$ +
$g[0,1]f[0,2]$ +
$g[0,2]f[0,3]$ +
$g[1,0]f[1,1]$ +
… +
$g[2,1]f[2,2]$ +
$g[2,2]f[2,3]$

3	?			

output

Filtering

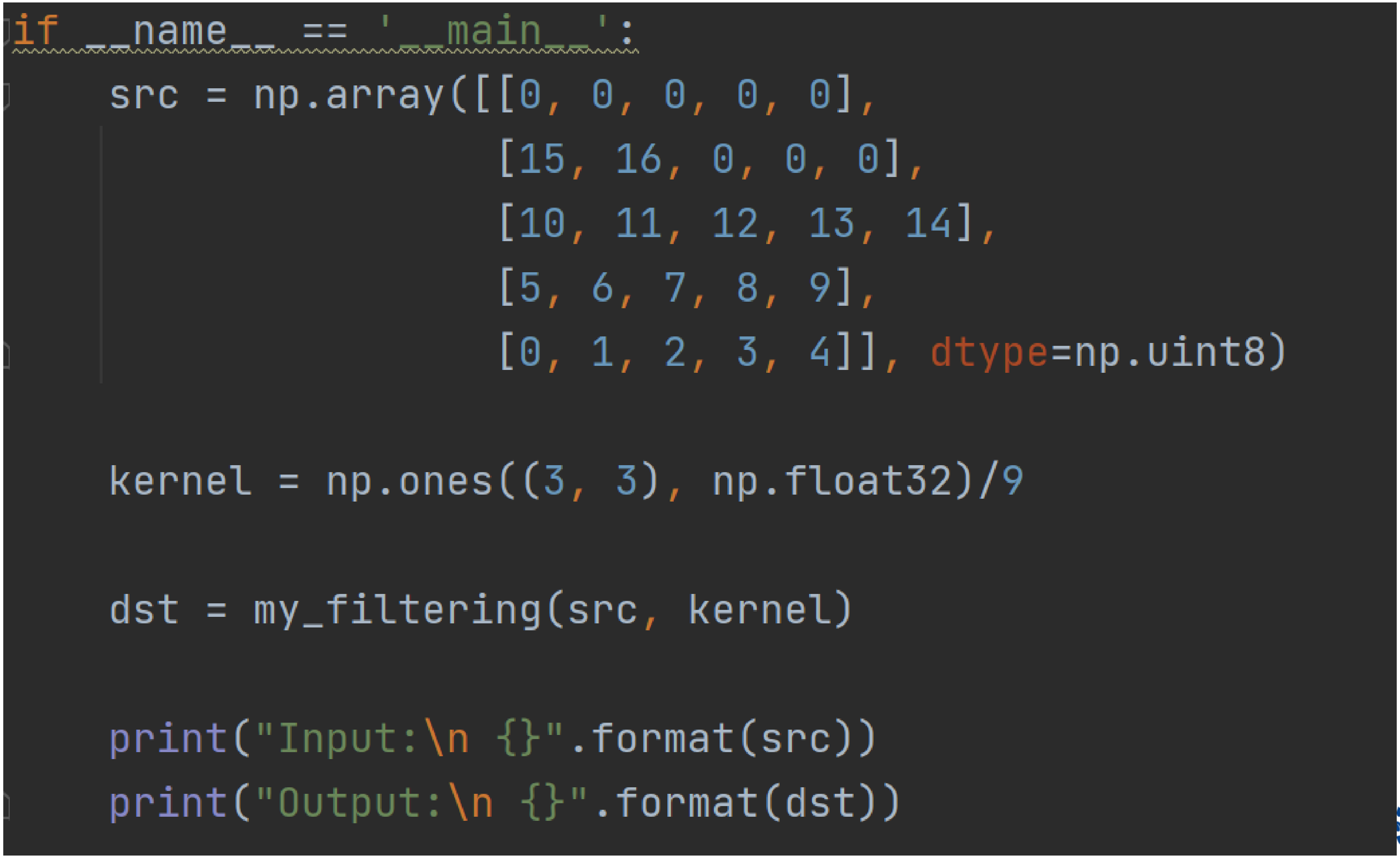
- **실습2**
  - Filtering을 하는 함수를 구현



```
def my_filtering(src, kernel):
    (h, w) = src.shape
    (k_h, k_w) = kernel.shape
    pad_img = my_padding(src, kernel)
    dst = np.zeros((h, w)) #output

    for m in range(h):
        for n in range(w):
            sum = 0
            for k in range(k_h):
                for l in range(k_w):
                    sum += kernel[k, l] * pad_img[m + k, n + l]
            dst[m, n] = sum

    dst = (dst + 0.5).astype(np.uint8)
    return dst
```

Filtering

- **실습2**
  - Filtering을 하는 함수를 구현



```python
if __name__ == '__main__':
    src = np.array([[0, 0, 0, 0, 0],
                    [15, 16, 0, 0, 0],
                    [10, 11, 12, 13, 14],
                    [5, 6, 7, 8, 9],
                    [0, 1, 2, 3, 4]], dtype=np.uint8)

    kernel = np.ones((3, 3), np.float32)/9

    dst = my_filtering(src, kernel)

    print("Input:\n {}".format(src))
    print("Output:\n {}".format(dst))
```

Gaussian filter

- **2D Gaussian filter**

  - $\mathrm{G}_{\sigma} = \frac{1}{2\pi\sigma^{2}}\mathrm{e}^{-\frac{(\mathrm{x}^{2}+\mathrm{y}^{2})}{2\sigma^{2}}}$

x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
σ : Gaussian 분포의 표준편차

n = mask의 행or열 길이// 2
ex) mask의 크기가 5이면
n = 5//2 = 2

5 x 5 Gaussian filter σ = 1

0.0029	0.0133	0.0219	0.0133	0.0029
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0029	0.0133	0.0219	0.0133	0.0029

Gaussian filter

- **2D Gaussian filter**
  - $\mathrm{G}_\sigma = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{(\mathrm{x}^2+\mathrm{y}^2)}{2\sigma^2}}$

x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
σ : Gaussian 분포의 표준편차

n = mask의 행or열 길이// 2
ex) mask의 크기가 5이면
n = 5//2 = 2

5 x 5 Gaussian filter $\sigma = 1$

$\frac{1}{sum}\cdot$

0.0029	0.0133	0.0219	0.0133	0.0029
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0029	0.0133	0.0219	0.0133	0.0029

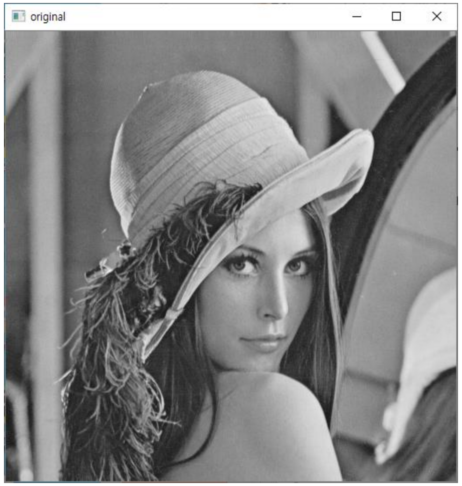
밝기 유지를 위해 총합은 1

Gaussian filter

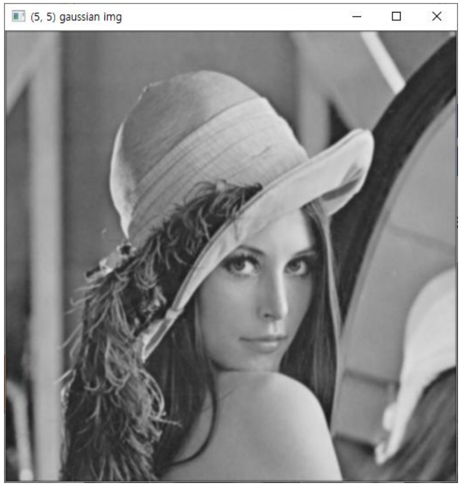
- **2D Gaussian filter**

  - $G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$

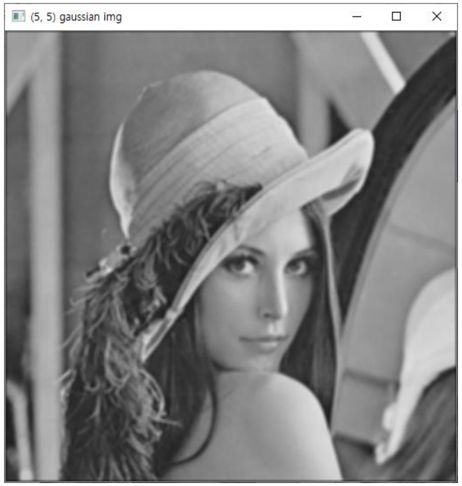
x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
σ : Gaussian 분포의 표준편차



original



5 x 5 Gaussian filter
sigma=1



5 x 5 Gaussian filter
sigma = 3

Gaussian filter

- **1D Gaussian filter**
  - Separability of the Gaussian filter

2D convolution
(center location only)

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array}$$

The filter factors
into a product of 1D
filters:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

Perform convolution
along rows:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline & 11 & \\ \hline & 18 & \\ \hline & 18 & \\ \hline \end{array}$$

Gaussian filter

- **1D Gaussian filter**
  - Separability of the Gaussian filter
  - $G_\sigma = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}$
    - 1×5 Gaussian kernel

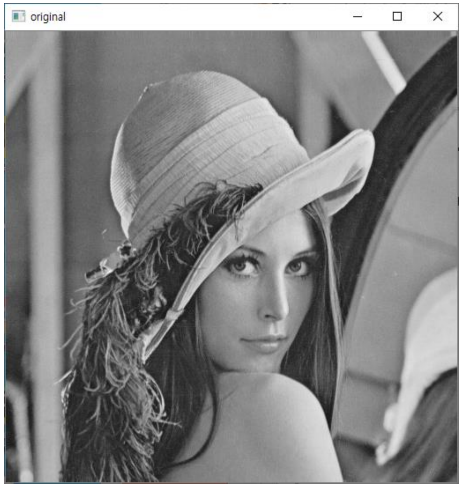
0.0544	0.2442	0.4026	0.2442	0.0544

  - $G_\sigma = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{y^2}{2\sigma^2}}$
    - 5×1 Gaussian kernel

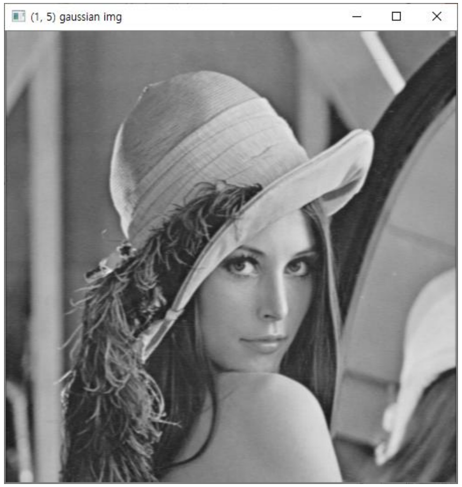
0.0544
0.2442
0.4026
0.2442
0.0544

Gaussian filter

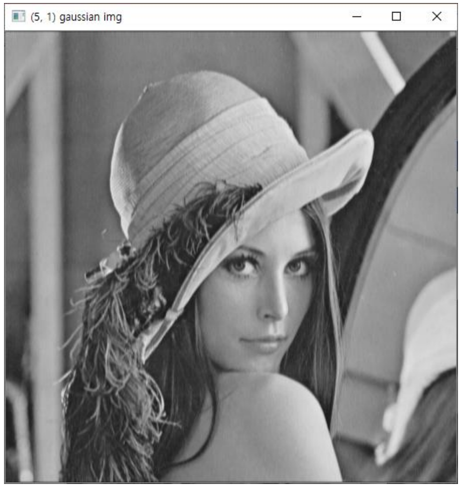
- **1D Gaussian filter**
  - Separability of the Gaussian filter
  - $G_\sigma = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}$
    - 1×5 Gaussian kernel



original



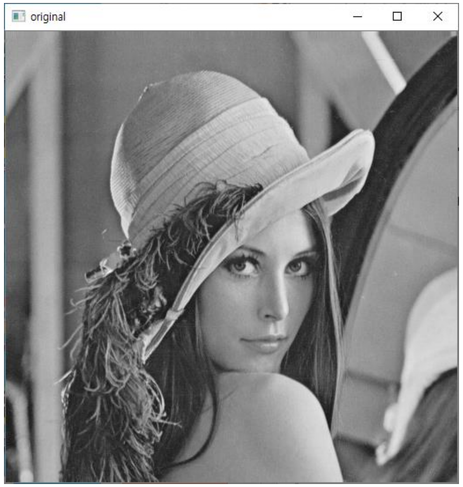
1 x 5 Gaussian filter
sigma=1



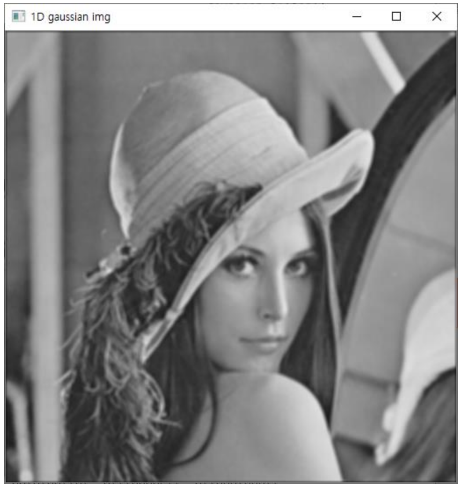
1 x 5 Gaussian filter
sigma = 3

Gaussian filter

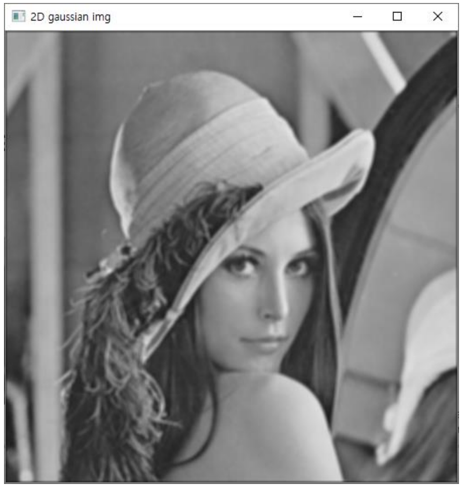
- **1D Gaussian filter**
  - Separability of the Gaussian filter
  - $G_\sigma = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{y^2}{2\sigma^2}}$
    - 5×1 Gaussian kernel



original



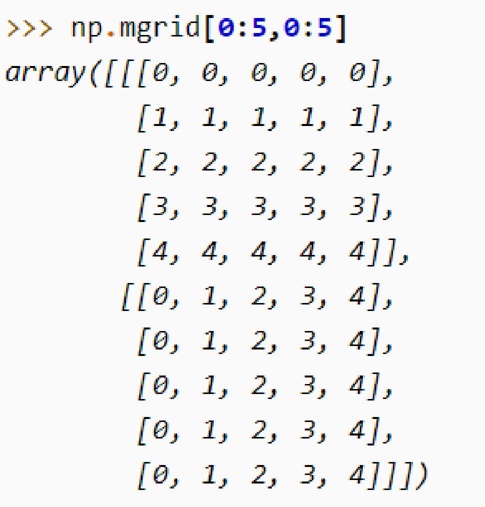
(5 x 1), (1 x 5) Gaussian filter
sigma=3



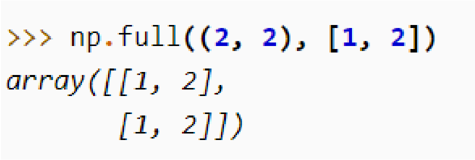
(5 x 5) Gaussian filter
sigma = 3

과제

- **Separability of the Gaussian**
  - 1D Gaussian filter vs. 2D Gaussian filter
    - 1D Gaussian과 2D Gaussian을 코드로 작성하고 실행 속도 비교
  - Gaussian filter를 만드는 코드를 작성하기
    - Gaussian filter의 크기와 sigma값을 변경해보고 결과 확인하기
    - 필요 함수
      - np.mgrid: 격자 그리드(meshgrid)를 생성하는 함수
      - np.sqrt: 로그 함수
      - np.exp: 지수 함수
      - np.full



```
>>> np.mgrid[0:5,0:5]
array([[[0, 0, 0, 0, 0],
        [1, 1, 1, 1, 1],
        [2, 2, 2, 2, 2],
        [3, 3, 3, 3, 3],
        [4, 4, 4, 4, 4]],
       [[0, 1, 2, 3, 4],
        [0, 1, 2, 3, 4],
        [0, 1, 2, 3, 4],
        [0, 1, 2, 3, 4],
        [0, 1, 2, 3, 4]]])
```



```
>>> np.full((2, 2), [1, 2])
array([[1, 2],
       [1, 2]])
```