CONTENTS

4.1 윈도우 제어

4.2 이벤트 처리 함수

4.3 그리기 함수

4.4 영상파일 처리

4.5 비디오 처리

4.6 MATPLOTLIB 패키지 활용

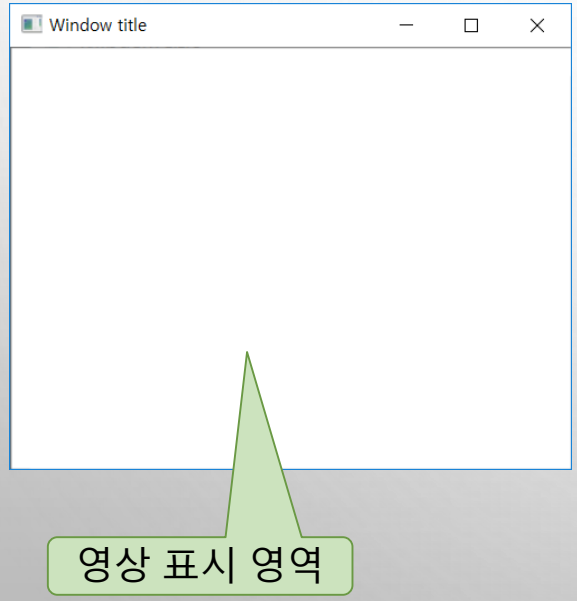
4.1 윈도우 제어

- 영상처리
  - 2차원 행렬에 대한 연산
  - 연산과정에서 행렬 원소 변경
  - 전체 영상에 대한 변화 인지하기 어려움

- 윈도우 영상 표시
  - 영상처리로 적용된 행렬 연산의 의미 이해하기 쉬움

- OpenCV에서는 윈도우(window, 창)가 활성화된 상태에서만 마우스나 키보드 이벤트 감지

4.1 윈도우 제어



함수 설명
cv2.namedWindow(winname[, flags]) → None ■설명: 윈도우 이름을 설정한 후, 해당 이름으로 윈도우 생성

인수 설명

■ winname(str) 윈도우 이름

■ flags(int) 윈도우의 크기 조정

옵션	값	설명
cv2.WINDOW_NORMAL	0	윈도우 크기 재조정 가능
cv2.WINDOW_AUTOSIZE	1	표시될 행렬의 크기에 맞춰 자동 조정

cv2.imshow(winname, mat) → None

■설명: winname 이름의 윈도우에 mat 행렬을 영상으로 표시함. 생성된 윈도우가 없으면, winname 이름으로 윈도우를 생성하고, 영상을 표시한다.

인수 설명

■ mat(numpy.ndarray) 윈도우에 표시되는 영상(행렬이 화소값을 밝기로 표시)

cv2.destroyWindow(winname) → None

■설명: 인수로 지정된 타이틀 윈도우 파괴

cv2.destroyAllWindows() → None

■설명: HighGUI로 생성된 모든 윈도우 파괴

cv2.moveWindow(winname, x, y) → None

■설명: winname 이름의 윈도우를 지정된 위치인 (x, y)로 이동. 이동되는 윈도우의 기준 위치는 좌측 상단임

인수 설명

■ x, y 모니터 안에서 이동하려는 위치의 x, y 좌표

cv2.resizeWindow(winname, width, height) → None

■설명: 윈도우의 크기를 재조정한다.

인수 설명

■ width, height 변경 윈도우의 가로, 세로 크기

4.1 윈도우 제어

◆예제

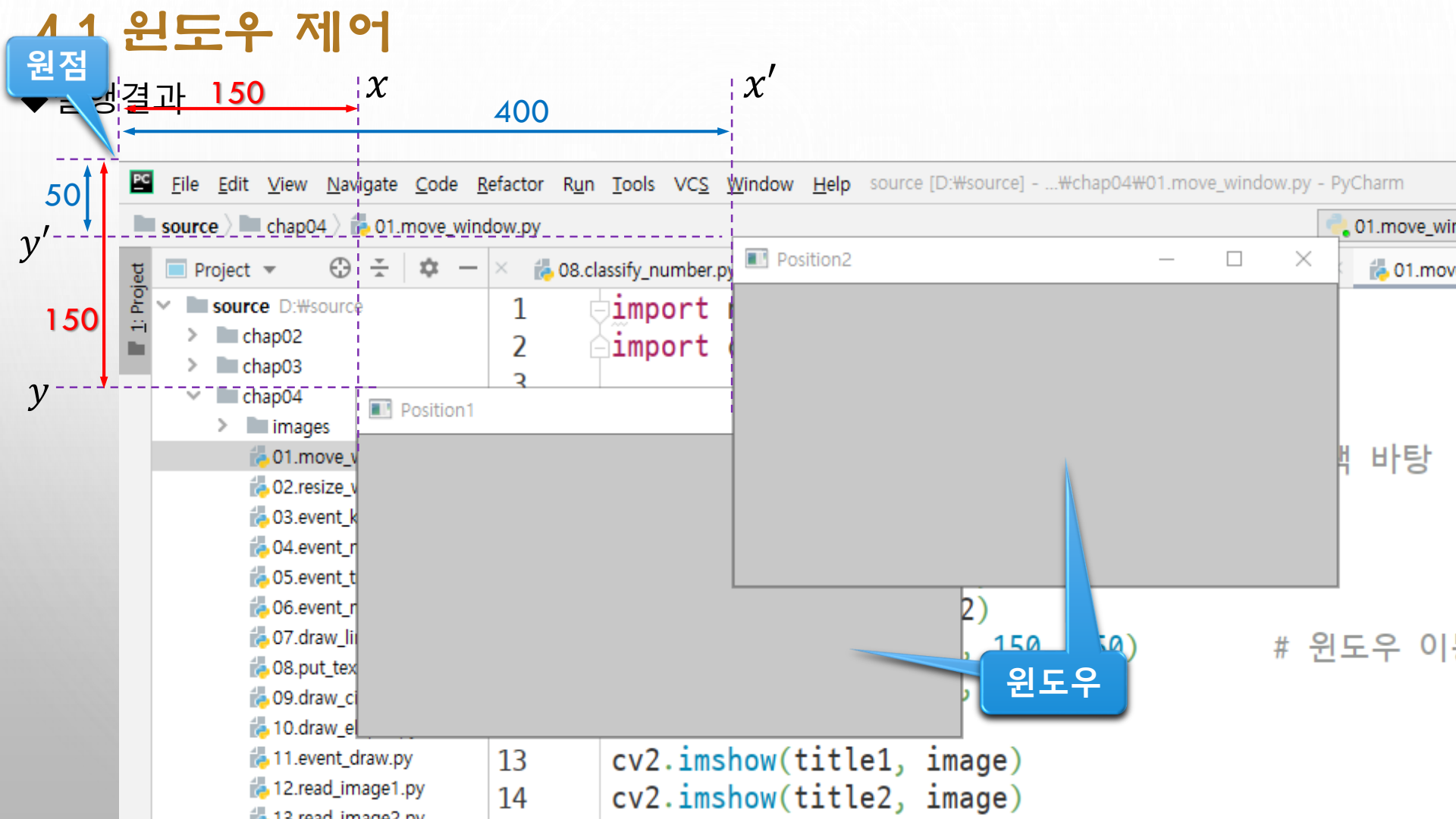
예제 4.1.1 윈도우 이동 – 01.move_window.py

0 원소 행렬 생성

슬라이스 연산자로 행렬 원소값 지정

```
import numpy as np                                  # 넘파이 라이브러리 임포트
import cv2                                          # OpenCV 라리브러리 임포트

image = np.zeros((200, 400), np.uint8)              # 행렬 생성
image[:] = 200                                      # 밝은 회색(200) 바탕 영상 생성

title1, title2 = 'Position1',                       # 윈도우 이름
cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)        # 윈도우 생성 및 크기 조정 옵션
cv2.namedWindow(title2)
cv2.moveWindow(title1, 150, 150)                    # 윈도우 이동 – 위치 지정
cv2.moveWindow(title2, 400, 50)

cv2.imshow(title1, image)                           # 행렬 원소를 영상으로 표시
cv2.imshow(title2, image)
cv2.waitKey(0)                                      # 키 이벤트(key event) 대기
cv2.destroyAllWindows()                             # 열린 모든 윈도우 파괴
```

4.1 윈도우 제어


원점
실행결과
150
400
x
x'
50
150
y'
y
File Edit View Navigate Code Refactor Run Tools VCS Window Help
source [D:₩source] - ...₩chap04₩01.move_window.py - PyCharm
source chap04 01.move_window.py
Project
source D:₩source
chap02
chap03
chap04
images
01.move_w
02.resize_
03.event_k
04.event_r
05.event_t
06.event_r
07.draw_li
08.put_tex
09.draw_ci
10.draw_el
11.event_draw.py
12.read_image1.py
08.classify_number.py
import
import
Position1
Position2
cv2.imshow(title1, image)
cv2.imshow(title2, image)
# 윈도우 이
윈도우

4.1 윈도우 제어

◆ WINDOW_AUTOSIZE vs. WINDOW_NORMAL

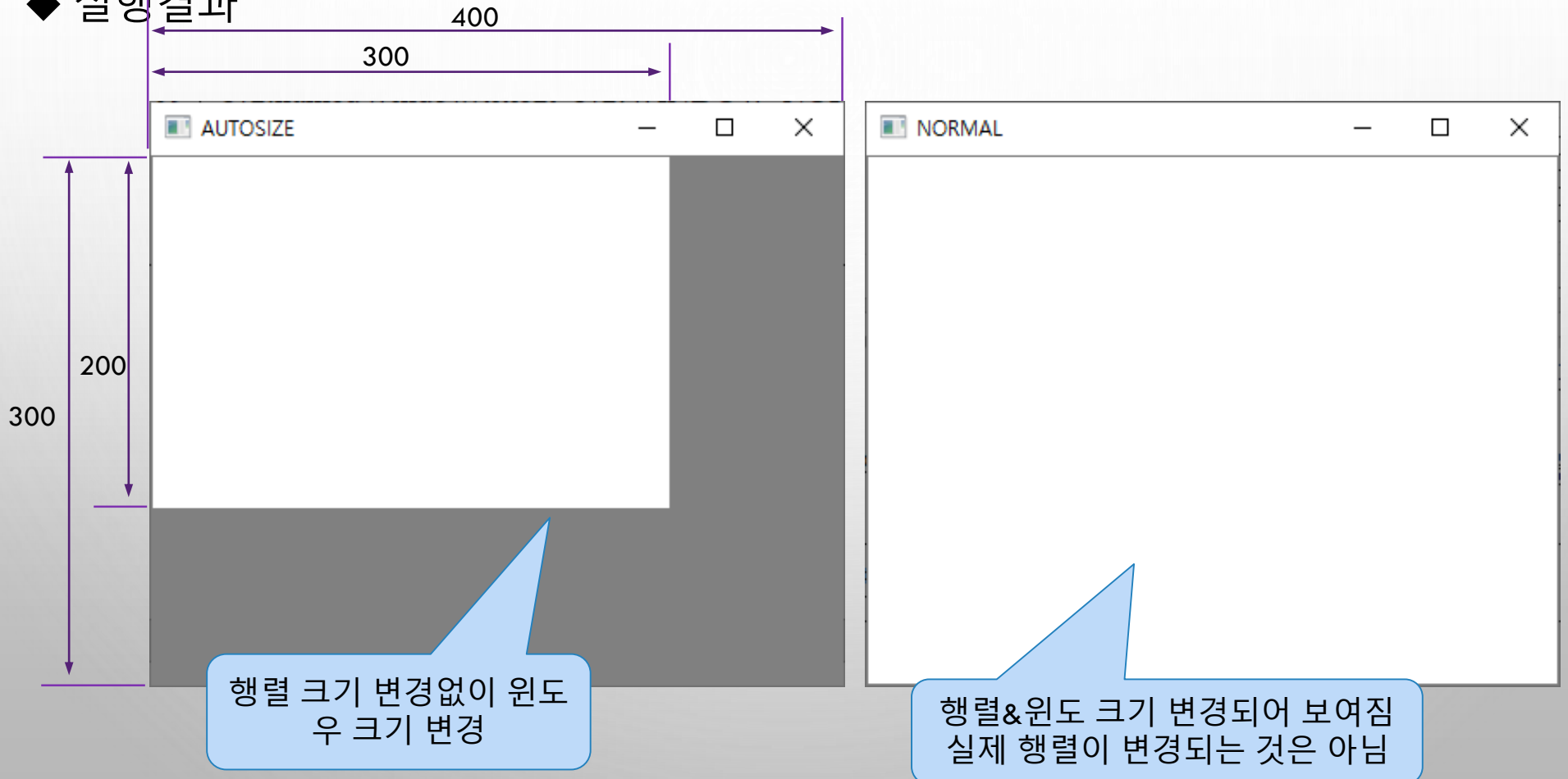
예제 4.1.2 윈도우의 크기 변경 – 02.window_resize.py

np.ndarray.fill() 함수로 원소값 지정

```
import numpy as np
import cv2

image = np.zeros((200, 300), np.uint8)          # ndarray 행렬 생성
image.fill(255)                                 # 모든 원소에 255(흰색) 지정

title1, title2 = 'AUTOSIZE', 'NORMAL'           # 윈도우 이름 변수
cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)    # 윈도우 생성 – 크기변경 불가
cv2.namedWindow(title2, cv2.WINDOW_NORMAL)      # 크기 변경 가능

cv2.imshow(title1, image)                       # 행렬 원소를 영상으로 표시
cv2.imshow(title2, image)
cv2.resizeWindow(title1, 400, 300)              # 윈도우 크기 변경
cv2.resizeWindow(title2, 400, 300)
cv2.waitKey(0)                                  # 키 이벤트(key event) 대기
cv2.destroyAllWindows()                         # 열린 모든 윈도우 제거
```

4.1 윈도우 제어

◆ 실행결과



4.2 이벤트 처리 함수

- 이벤트
  - 프로그램에 의해 감지되고 처리될 수 있는 동작이나 사건
  - 예
    - 사용자가 키보드의 키를 누르는 것
    - 마우스를 움직인다거나 마우스 버튼을 누르는 것
    - 깊이 들어가면 타이머(timer)와 같은 하드웨어 장치가 발생시키는 이벤트
    - 사용자가 자체적으로 정의하는 이벤트
- 이벤트를 처리하기 위해 주로 콜백(callback) 함수 사용
- 콜백 함수
  - 개발자가 시스템 함수를 직접 호출하는 방식
  - 이벤트 발생하거나 특정 시점에 도달했을 때 시스템이 개발자가 등록한 함수 호출
- OpenCV에서도 기본적인 이벤트 처리 함수 지원
  - 키보드 이벤트, 마우스 이벤트, 트랙바(trackbar) 이벤트

4.2.1 키보드 이벤트 제어

◆delay 인수에 따라서 두 가지 모드 동작

함수 설명	
cv2.waitKey([, delay]) → retval ■ 설명: delay(ms: milisccond) 시간만큼 키 입력을 대기하고, 키 이벤트가 발생하면 해당 키 값 반환	
인수 설명	■ delay 지연 시간, ms 단위 – delay≤0 키 이벤트 발생까지 무한 대기 – delay>0 지연 시간 동안 키 입력 대기, 지연 시간 안에 키 이벤트 없으면 –1 반환
cv2.waitKeyEx([, delay]) → retval ■ 설명: cv2.waitKey()와 동일하지만, 전체 키 코드(full key code)를 반환한다. 화살표 키 등을 입력 받을 때 사용 가능 (OpenCV 3.4 이상에서 지원)	

4.2.1 키보드 이벤트 제어

◆예제

예제 4.2.1 키 이벤트 사용 – 03.event_key.py

```
import numpy as np
import cv2

## switch case문을 사전(dictionary)으로 구현
switch_case = {
        ord('a'): "a키 입력",                    # ord() 함수: 문자 → 아스키코드 변환
        ord('b'): "b키 입력",
        0x41: "A키 입력",
        int('0x42', 16): "B키 입력",             # 0x42(16진수) → 10진수 변환
        2424832: "왼쪽 화살표키 입력",           # 0x250000
        2490368: "윗쪽 화살표키 입력",           # 0x260000
        2555904: "오른쪽 화살표키 입력",         # 0x270000
        2621440: "아래쪽 화살표키 입력"          # 0x280000
}

```

4.2.1 키보드 이벤트 제어

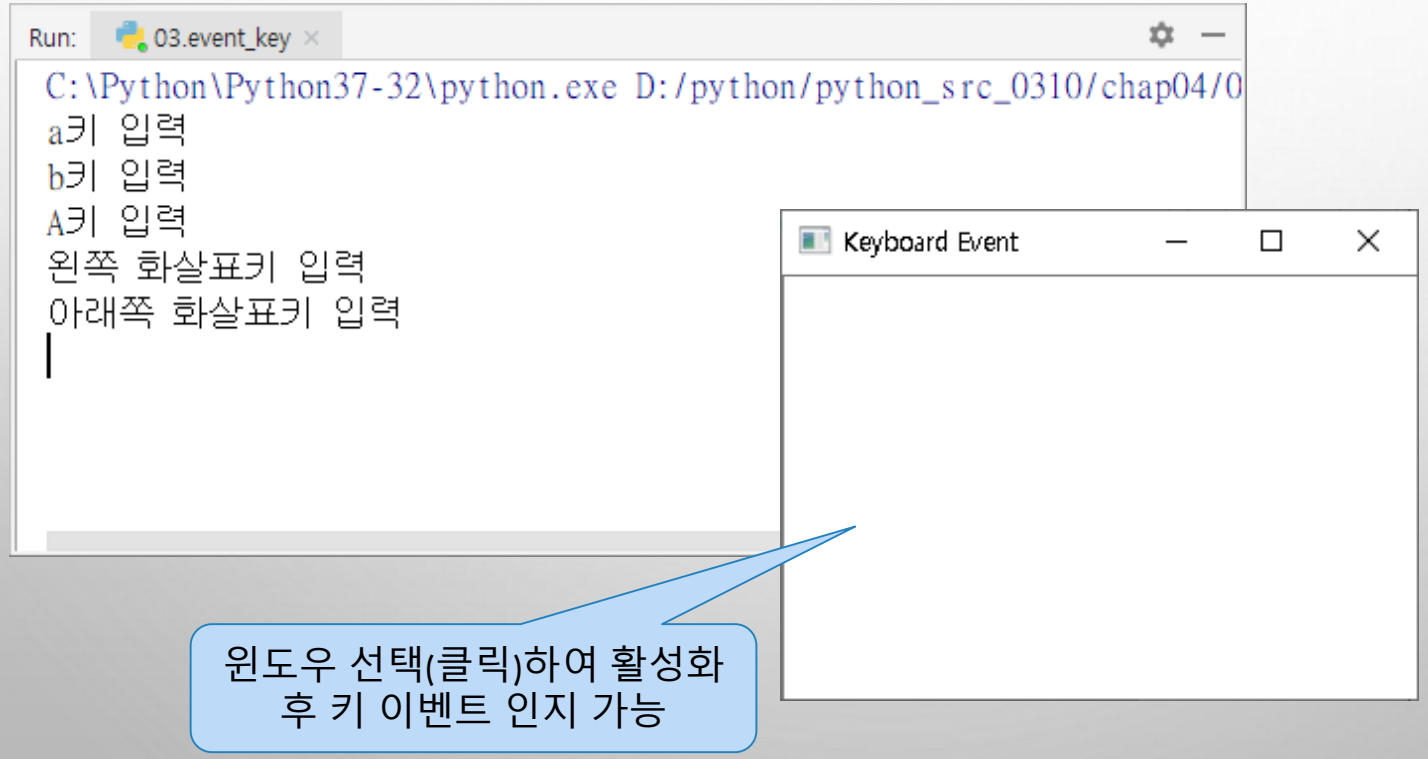
◆예제

```
image = np.ones((200, 300), np.float32)        # 원소값 1인 행렬 생성
cv2.namedWindow('Keyboard Event')              # 윈도우 이름
cv2.imshow("Keyboard Event", image)

while True:                                    # 무한 반복
    key = cv2.waitKeyEx(100)                   # 100ms 동안 키 이벤트 대기
    if key == 27: break                        # ESC 키 누르면 종료

    try:
        result = switch_case[key]
        print(result)
    except KeyError:
        result = -1

cv2.destroyAllWindows()                        # 열린 모든 윈도우 제거
```

4.2.1 키보드 이벤트 제어

◆결과

❖ 키 이벤트를 발생시키려면 "KEYBOARD EVENT" 윈도우를 반드시 선택(클릭)하여 활성화시킨 후, 키보드의 키를 눌러야 한다



4.2.2 마우스 이벤트 제어

함수 설명	
def setMouseCallback(windowName, onMouse, param=None) → None ■ 설명: 사용자가 정의한 마우스 콜백 함수를 시스템에 등록	
인수 설명	■ winname 이벤트 발생을 확인할 윈도우 이름, 문자열 ■ onMouse 마우스 이벤트를 처리하는 콜백 함수 이름(콜백함수) ■ param 이벤트 처리 함수로 전달할 추가적인 사용자 정의 인수
onMouse(event, x, y, flags, param=None) ■ 설명: 발생한 마우스 이벤트에 대한 처리와 제어를 구현하는 콜백 함수. cv2.setMouseCallback() 함수의 두 번째 인수(onMouse)의 구현부, 따라서 이름이 같아야 함. onMouse() 함수의 인수 구조(인수 타입, 인수 순서 등)를 유지해야 함.	
인수 설명	■ event 발생한 마우스 이벤트의 종류 ■ x, y 이벤트 발생 시 마우스 포인터의 x, y 좌표 ■ flags 마우스 버튼과 동시에 특수키([Shift], [Alt], [Ctrl])를 눌렀는지 여부 확인

옵션	값	설명
cv2.EVENT_FLAG_LBUTTON	1	왼쪽 버튼 누르기
cv2.EVENT_FLAG_RBUTTON	2	오른쪽 버튼 누르기
cv2.EVENT_FLAG_MBUTTON	4	중간 버튼 누르기
cv2.EVENT_FLAG_CTRLKEY	8	[Ctrl] 키 누르기
cv2.EVENT_FLAG_SHIFTKEY	16	[Shift] 키 누르기
cv2.EVENT_FLAG_ALTKEY	32	[Alt] 키 누르기

■ param 콜백 함수로 전달하는 추가적인 사용자 정의 인수

〈표 4.2.1〉 마우스 이벤트 종류

옵션	값	설명
cv2.EVENT_MOUSEMOVE	0	마우스 움직임
cv2.EVENT_LBUTTONDOWN	1	왼쪽 버튼 누르기
cv2.EVENT_RBUTTONDOWN	2	오른쪽 버튼 누르기
cv2.EVENT_MBUTTONDOWN	3	중간 버튼 누르기
cv2.EVENT_LBUTTONUP	4	왼쪽 버튼 떼기
cv2.EVENT_RBUTTONUP	5	오른쪽 버튼 떼기
cv2.EVENT_MBUTTONUP	6	중간 버튼 떼기
cv2.EVENT_LBUTTONDBLCLK	7	왼쪽 버튼 더블클릭
cv2.EVENT_RBUTTONDBLCLK	8	오른쪽 버튼 더블클릭
cv2.EVENT_MBUTTONDBLCLK	9	중간 버튼 더블클릭
cv2.EVENT_MOUSEWHEEL	10	마우스 휠
cv2.EVENT_MOUSEHWHEEL	11	마우스 가로 휠

4.2.2 마우스 이벤트 제어

◆예제

예제 4.2.2 마우스 이벤트 사용 – 04.event_mouse.py

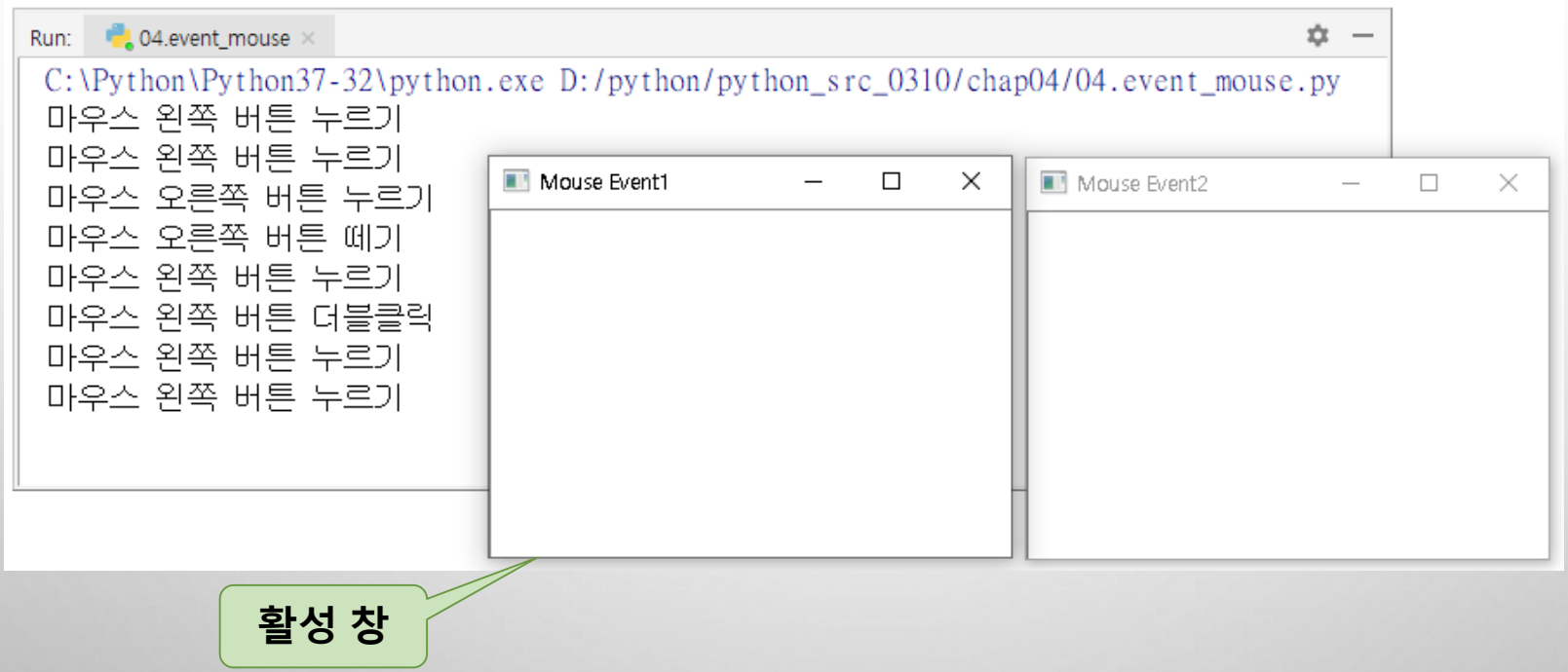
```
01  import numpy as np
02  import cv2
03
04  def onMouse(event, x, y, flags, param):                # 콜백 함수 – 이벤트 내용 출력
05      if event == cv2.EVENT_LBUTTONDOWN:
06          print("마우스 왼쪽 버튼 누르기")
07      elif event == cv2.EVENT_RBUTTONDOWN:
08          print("마우스 오른쪽 버튼 누르기")
09      elif event == cv2.EVENT_RBUTTONUP:
10          print("마우스 오른쪽 버튼 떼기")
11      elif event == cv2.EVENT_LBUTTONDBLCLK:
12          print("마우스 왼쪽 버튼 더블클릭")
13
14  image = np.full((200, 300), 255, np.uint8)             # 초기 영상 생성
15
16  title1, title2 = "Mouse Event1", "Mouse Event2"        # 윈도우 이름
17  cv2.imshow(title1, image)                              # 영상 보기
18  cv2.imshow(title2, image)
19
20  cv2.setMouseCallback(title1, onMouse)                  # 마우스 콜백 함수
21  cv2.waitKey(0)                                         # 키 이벤트 대기
22  cv2.destroyAllWindows()                                # 열린 모든 윈도우 제거
```

마우스 왼쪽 버튼

함수 이름

4.2.2 마우스 이벤트 제어

◆실행 결과



활성 창

4.2.3 트랙바 이벤트 제어

◆ 트랙바(TRACKBAR)

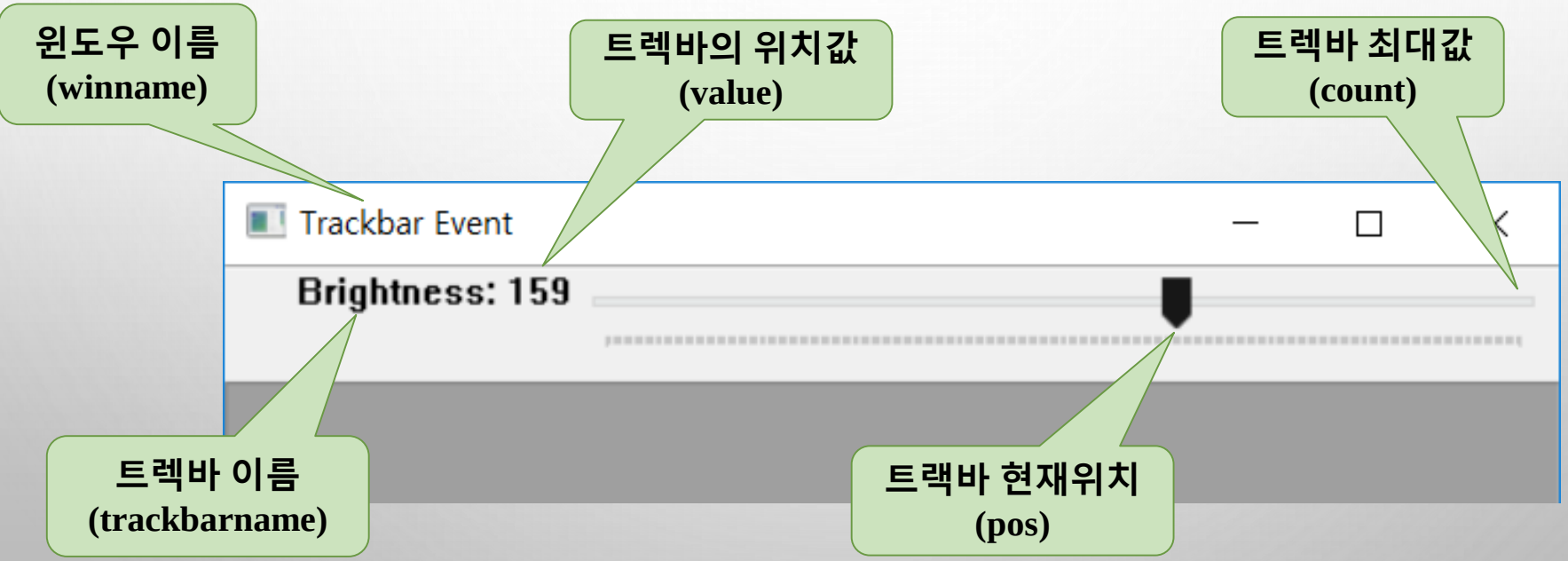
❖ 일정한 범위에서 특정한 값을 선택할 때 사용하는 일종의 스크롤바 혹은 슬라이더바

함수 설명		
cv2.createTrackbar(trackbarname, winname, value count, onChange) → None ■ 설명: 트랙바를 생성한 후, 지정한 윈도우에 추가하는 함수이다.		
인수 설명	■ trackbarname	윈도우에 생성되는 트랙바 이름
	■ winname	트랙바의 부모 윈도우 이름(트랙바 이벤트 발생을 체크하는 윈도우)
	■ value	트랙바 슬라이더의 위치를 반영하는 값(정수)
	■ count	트랙바 슬라이더의 최댓값, 최솟값은 항상 0
	■ onChange	트랙바 슬라이더의 값이 변경될 때 호출되는 콜백 함수
onChange(pos) → None ■ 설명: 트랙바 슬라이더의 위치가 변경될 때마다 호출되는 콜백 함수. cv2.createTrackbar()의 마지막 인수와 이름이 같아야 한다.		
인수 설명	■ pos	트랙바 슬라이더 위치
cv2.getTrackbarPos(trackbarname, winname) → retval ■ 설명: 지정한 트랙바의 슬라이더 위치를 반환한다.		
cv2.setTrackbarPos(trackbarname, winname, pos) → None ■ 설명: 지정한 트랙바의 슬라이더 위치를 설정한다.		

4.2.3 트랙바 이벤트 제어

◆형식

createTrackbar(trackbarname, winname, value, count, onChange , userdata)



4.2.3 트랙바 이벤트 제어

예제 4.2.3 트랙바 이벤트 사용 – 05.event_trackbar.py

```
import numpy as np
import cv2

def onChange(value):                                    # 트랙바 콜백 함수
    global image, title                                 # 전역 변수 참조

    add_value = value - int(image[0][0])                # 트랙바 값과 영상 화소값 차분
    print("추가 화소값:", add_value)
    image[:] = image + add_value                        # 행렬과 스칼라 덧셈 수행
    cv2.imshow(title, image)

image = np.zeros((300, 500), np.uint8)                  # 영상 생성

title = 'Trackbar Event'
cv2.imshow(title, image)

cv2.createTrackbar('Brightness', title, image[0][0], 255, onChange)  # 트랙바 콜백 함수 등록
cv2.waitKey(0)
cv2.destroyAllWindows()                                 # 열린 모든 윈도우 제거
```

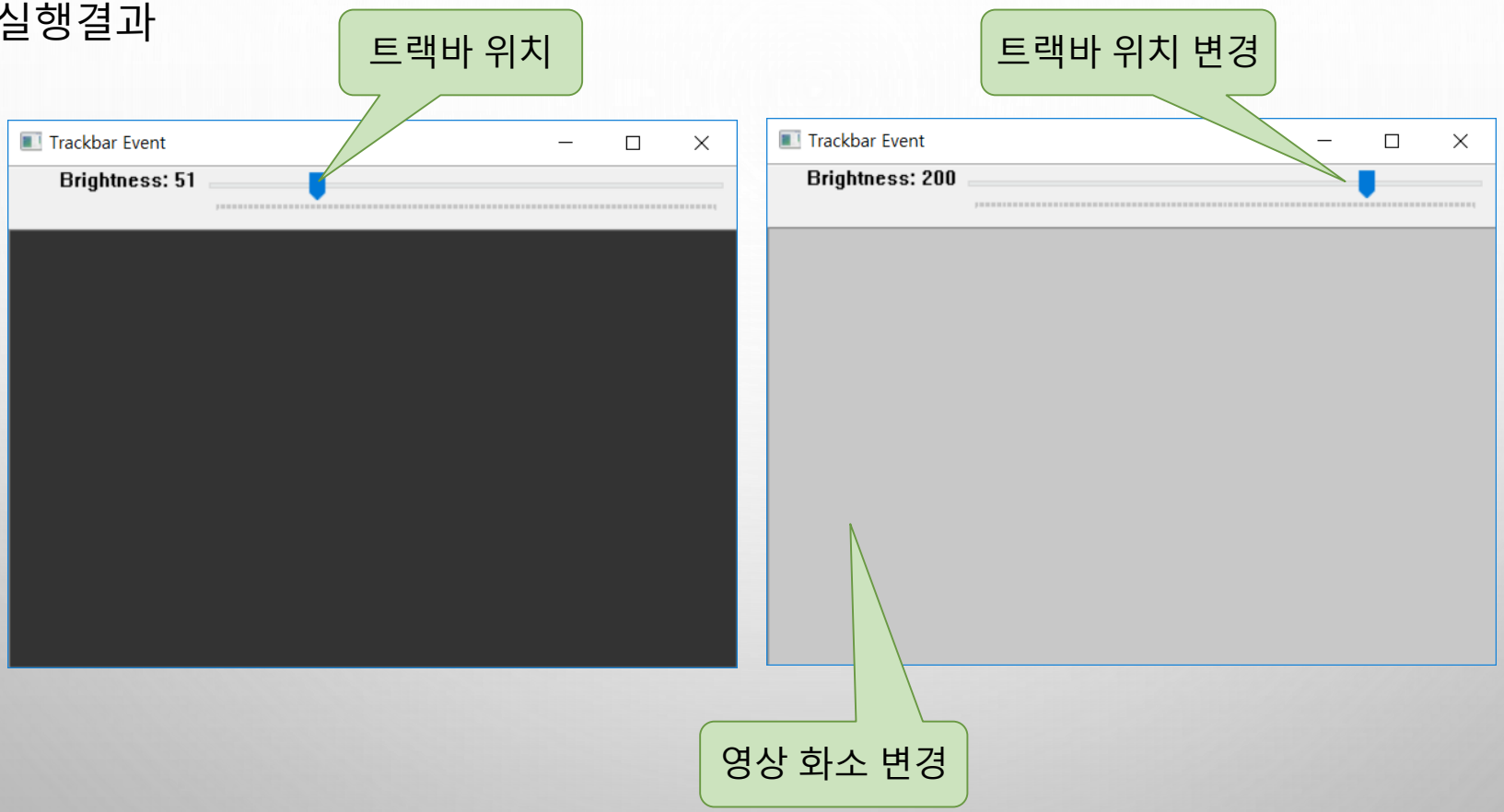
현재값

최대값

콜백함수

4.2.3 트랙바 이벤트 제어

◆실행결과



4.2.3 트랙바 이벤트 제어

◆ 예제

심화예제 4.2.4 마우스 및 트랙바 이벤트 사용 – 06.event_mouse_trackbar.py

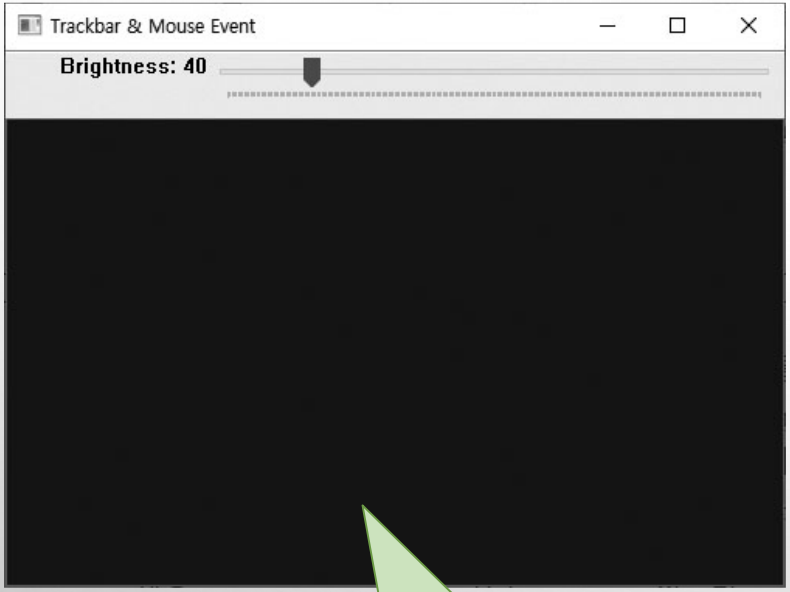
```
import numpy as np
import cv2

def onChange(value): …                                  # 트랙바 콜백 함수 – 소스 표시 생략

def onMouse(event, x, y, flags, param):                 # 마우스 콜백 함수
    global image, bar_name                              # 전역 변수 참조

    if event == cv2.EVENT_RBUTTONDOWN:                  # 마우스 우버튼
        if (image[0][0] < 246):  image[:] = image + add_value
        cv2.setTrackbarPos(bar_name, title, image[0][0])  # 트랙바 위치 변경
        cv2.imshow(title, image)

```

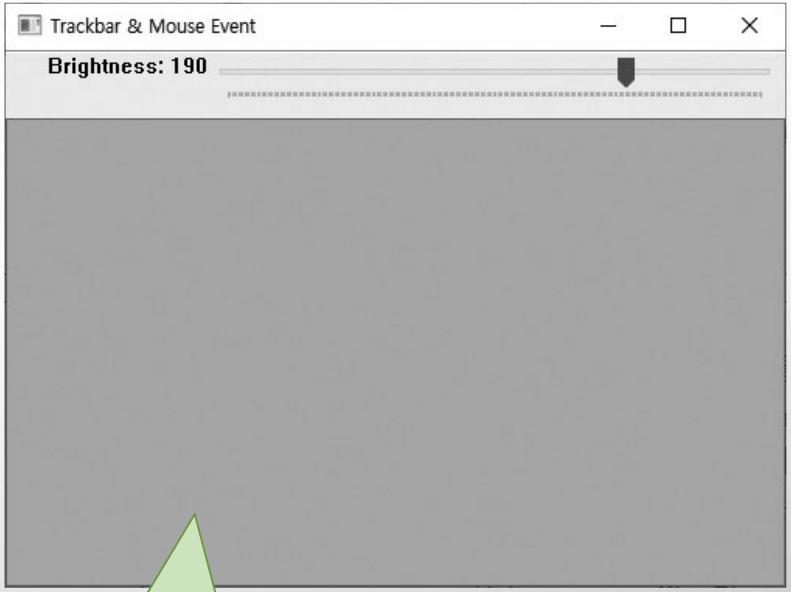
4.2.3 트랙바 이벤트 제어

```
19      elif event == cv2.EVENT_LBUTTONDOWN:                    # 마우스 좌버튼
20          if (image[0][0] >= 10):
21              image[:] = image - 10
22          cv2.setTrackbarPos(bar_name, title, image[0][0])    # 트랙바 위치 변경
23          cv2.imshow(title, image)
24
25  image = np.zeros((300, 500), np.uint8)
26  title = "Trackbar & Mouse Event"                            # 윈도우 이름
27  bar_name = 'Brightness'                                     # 트랙바 이름
28  cv2.imshow(title, image)
29
30  cv2.createTrackbar(bar_name, title, image[0][0], 255, onChange)  # 트랙바 콜백 함수
31  cv2.setMouseCallback(title, onMouse)                        # 마우스 콜백 함수 등록
32  cv2.waitKey(0)                                              # 키 입력 대기
33  cv2.destroyAllWindows()                                     # 모든 윈도우 닫기
```

4.2.3 트랙바 이벤트 제어

◆ 실행결과





마우스 좌버튼 클릭시 어두워짐

마우스 우버튼 클릭시 밝아짐,

4.3 그리기 함수

◆ 영상처리 프로그래밍 과정에서 해당 알고리즘 적용시 결과 확인 필요

❖ 얼굴 검출 알고리즘을 적용했을 때,

- 전체 영상 위에 검출한 얼굴 영역을 사각형이나 원으로 표시

❖ 차선 확인하고자 직선 검출 알고리즘을 적용했을 때,

- 차선을 정확하게 검출했는지 확인하기 위해 도로 영상 위에 선으로 표시

4.3.1 직선 및 사각형 그리기

◆ 함수설명

함수설명	
cv2.line(img, pt1, pt2, color[, thickness[, lineType[, shift]]]) → img ■ 설명: pt1에서 pt2를 잇는 직선 그린다.	
인수 설명	■ img 그릴 대상 행렬(영상) ■ pt1, pt2 시작 좌표와 종료 좌표(정수형 튜플) ■ color 선의 색상(정수형 튜플) ■ thickness 선의 두께, cv2.FILLED일 경우 지정된 색으로 내부를 채움 ■ lineType 선의 형태

옵션	값	설명
cv2.LINE_4	4	4-방향 연결선(4-connected line)
cv2.LINE_8	8	8-방향 연결선(8-connected line)
cv2.LINE_AA	16	계단 현상을 감소시킨 선(antialiased line)

■ shift 입력 좌표(pt1, pt2)에 대해서 오른쪽으로 비트 시프트(〉〉, bit shift) 연산한 결과를 좌표로 지정해서 직선을 그림

cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]]) → img cv2.rectangle(img, rec, color[, thickness[, lineType[, shift]]]) → img ■ 설명: 2가지 방식(두 좌표, 사각형 영역)으로 사각형을 그린다.	
인수 설명	■ pt1, pt2 시작 좌표와 종료 좌표 (정수형) ■ rec 사각형 영역(x, y, w, h)을 나타내는 4 원소 튜플(정수형)

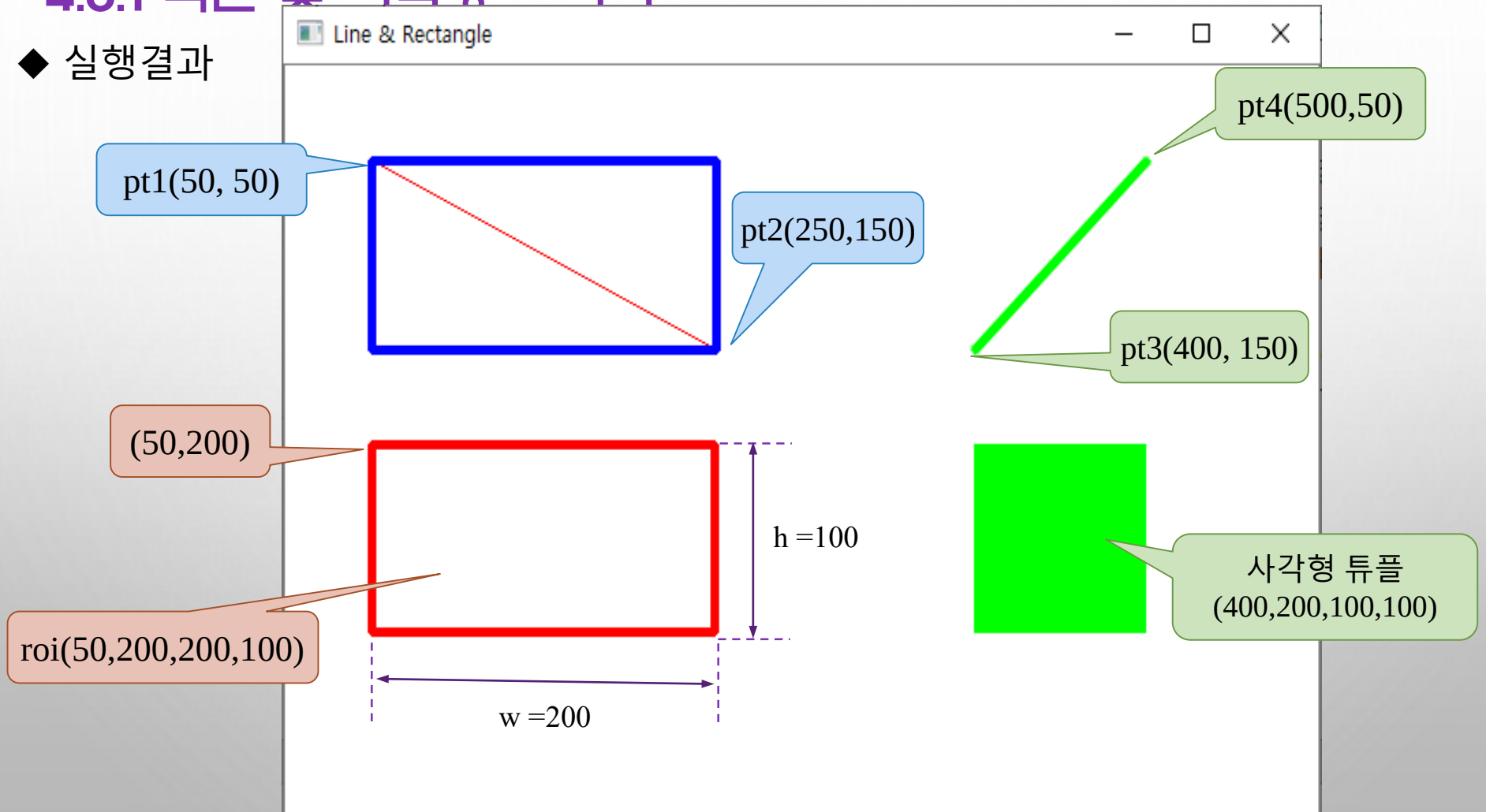
4.3.1 직선 및 사각형 그리기

예제 4.3.1 직선 & 사각형 그리기 – 07.draw_line_rect.py

```
import numpy as np
import cv2

blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)          # 색상 선언
image = np.zeros((400, 600, 3), np.uint8)                          # 3채널 컬러 영상 생성
image[:] = (255, 255, 255)                                         # 3채널 흰색

pt1, pt2 = (50, 50),  (250, 150)                                   # 좌표 선언 – 정수형 튜플
pt3, pt4 = (400, 150), (500, 50)
roi = (50, 200, 200, 100)                                          # 사각형 영역 – 4원소 튜플

## 직선 그리기
cv2.line(image, pt1, pt2, red)
cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)                   # 계단 현상 감소선

## 사각형 그리기
cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)                # 4방향 연결선
cv2.rectangle(image, roi, red, 3, cv2.LINE_8 )                     # 8방향 연결선
cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)      # 내부 채움

cv2.imshow("Line & Rectangle", image)                              # 윈도우에 영상 표시
cv2.waitKey(0)
cv2.destroyAllWindows()                                            # 모든 열린 윈도우 닫
```

4.3.1 직선 및 사각형 그리기

◆ 실행결과



4.3.2 글자 쓰기

- cv2.putText()
  - 행렬의 특정 위치에 원하는 글자를 써서 영상으로 표시하고 싶을 때 사용
- 형식

표시 문자열

2줄 산세리프 폰트

확대 비율

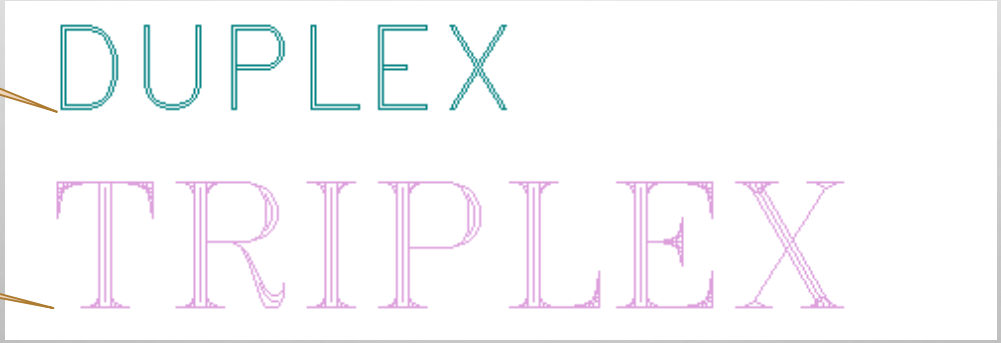
```
putText(image, "DUPLEX", pt1, FONT_HERSHEY_DUPLEX, 2, Scalar(128, 128, 0))
putText(image, "TRIPLEX", pt2, FONT_HERSHEY_TRIPLEX, 3, Scalar(221, 160, 221))
```

3줄 세리프 폰트

색상

시작좌표(pt1)

시작좌표(pt2)



4.3.2 글자 쓰기

함수 설명		
cv2.putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]]) → img ■ 설명: text 문자열을 org 좌표에 color 색상으로 그린다.		
인수 설명	■ img	문자열을 작성할 대상 행렬(영상)
	■ text	작성할 문자열
	■ org	문자열의 시작 좌표, 문자열에서 가장 왼쪽 하단을 의미
	■ fontFace	문자열의 폰트
	■ fontScale	글자 크기 확대 비율
	■ color	글자의 색상
	■ thichness	글자의 굵기
	■ lineTpye	글자 선의 형태
	■ bottomLeftOrigin	영상의 원점 좌표 설정(True- 좌하단 왼쪽, False- 좌상단)

4.3.2 글자 쓰기

◆ 세리프 체

❖ 글자의 획 끝에 날카롭게 튀어나온 글자체

❖ 명조체, 궁서체 등

◆ 산세리프 체

❖ 날카로운 장식선이 없는 글자체

❖ 돋움체 , 고딕체

〈표 4.3.1〉 문자열의 폰트(fontFace)에 대한 옵션과 의미

옵션	값	설명
cv2.FONT_HERSHEY_SIMPLEX	0	중간 크기 산세리프 폰트
cv2.FONT_HERSHEY_PLAIN	1	작은 크기 산세리프 폰트
cv2.FONT_HERSHEY_DUPLEX	2	2줄 산세리프 폰트
cv2.FONT_HERSHEY_COMPLEX	3	중간 크기 세리프 폰트
cv2.FONT_HERSHEY_TRIPLEX	4	3줄 세리프 폰트
cv2.FONT_HERSHEY_COMPLEX_SMALL	5	COMPLEX 보다 작은 크기
cv2.FONT_HERSHEY_SCRIPT_SIMPLEX	6	필기체 스타일 폰트
cv2.FONT_HERSHEY_SCRIPT_COMPLEX	7	복잡한 필기체 스타일
cv2.FONT_ITALIC	16	이탤릭체를 위한 플래그

4.3.2 글자 쓰기

◆ 예제

예제 4.3.2 글자 쓰기 – 08.put_text.py

```
import numpy as np
import cv2

olive, violet, brown = (128, 128, 0), (221, 160, 221), (42, 42, 165)        # 색상 지
pt1, pt2 =  (50, 230), (50, 310)                                  # 문자열 위치 좌표

image = np.zeros((350, 500, 3), np.uint8)
image.fill(255)

cv2.putText(image, 'SIMPLEX', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, brown)
cv2.putText(image, 'DUPLEX', (50, 130), cv2.FONT_HERSHEY_DUPLEX, 3, olive)
cv2.putText(image, 'TRIPLEX', pt1, cv2.FONT_HERSHEY_TRIPLEX, 2, violet)
fontFace = cv2.FONT_HERSHEY_PLAIN | cv2.FONT_ITALIC                # 글자체 상수
cv2.putText(image, 'ITALIC', pt2, fontFace, 4, violet)

cv2.imshow('Put Text', image)                                     # 윈도우 이름 지정 및 영상 표시
cv2.waitKey(0)                                                    # 키이벤트 대기
```

글자 확대 비율

기울임체 포함

4.3.2 글자 쓰기

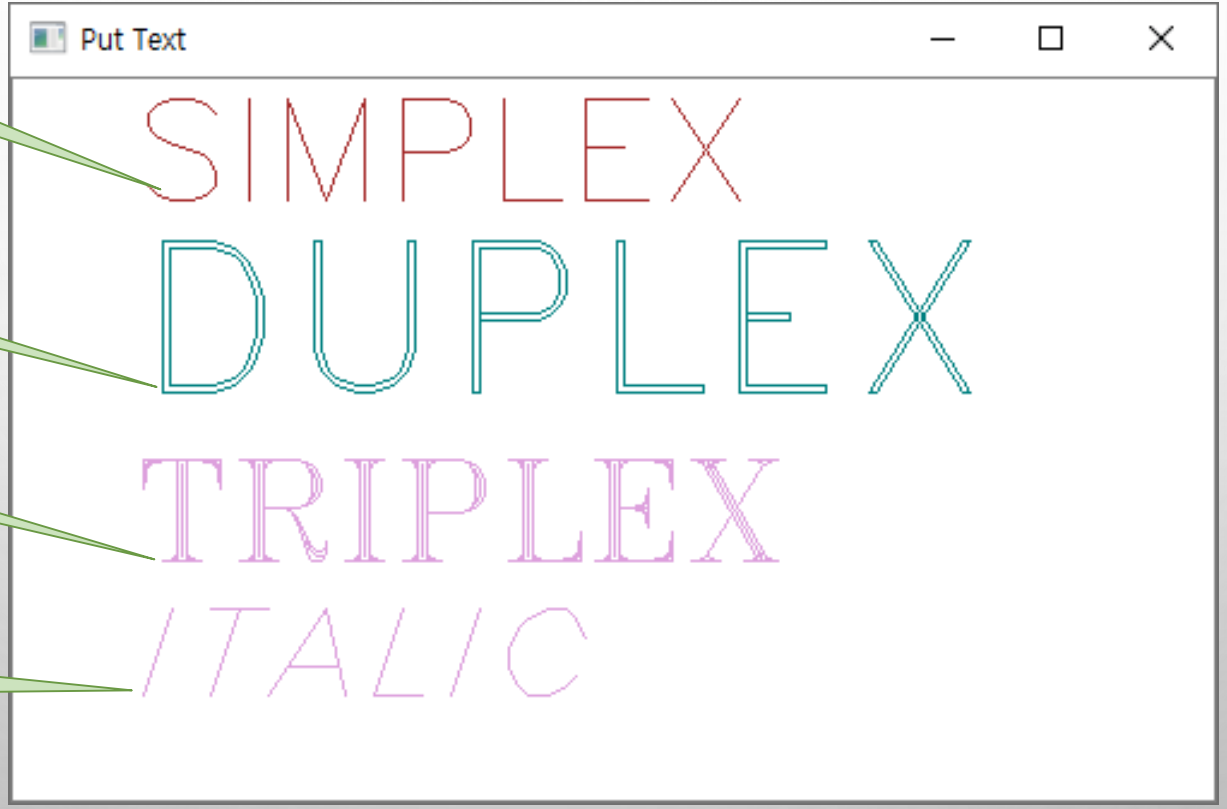
◆ 실행결과

(50, 50) 좌표

(50, 130) 좌표

pt1(50, 200)

pt3(50, 260)



4.3.3 원 그리기

◆ cv2.circle() 함수

❖ 원의 중심 좌표(center), 반지름(radius), 선의 색상(color)은 반드시 지정

함수 설명		
cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]]) → img ■ 설명: center를 중심으로 radius 반지름의 원을 그린다.		
인수 설명	■ img	원을 그릴 대상 행렬(영상)
	■ center	원의 중심 좌표
	■ radius	원의 반지름
	■ color	선의 색상
	■ thickness	선의 두께
	■ lineType	선의 형태 , cv2.line() 함수의 인수와 동일
	■ shift	좌표에 대한 비트 시프트 연산

4.3.3 원

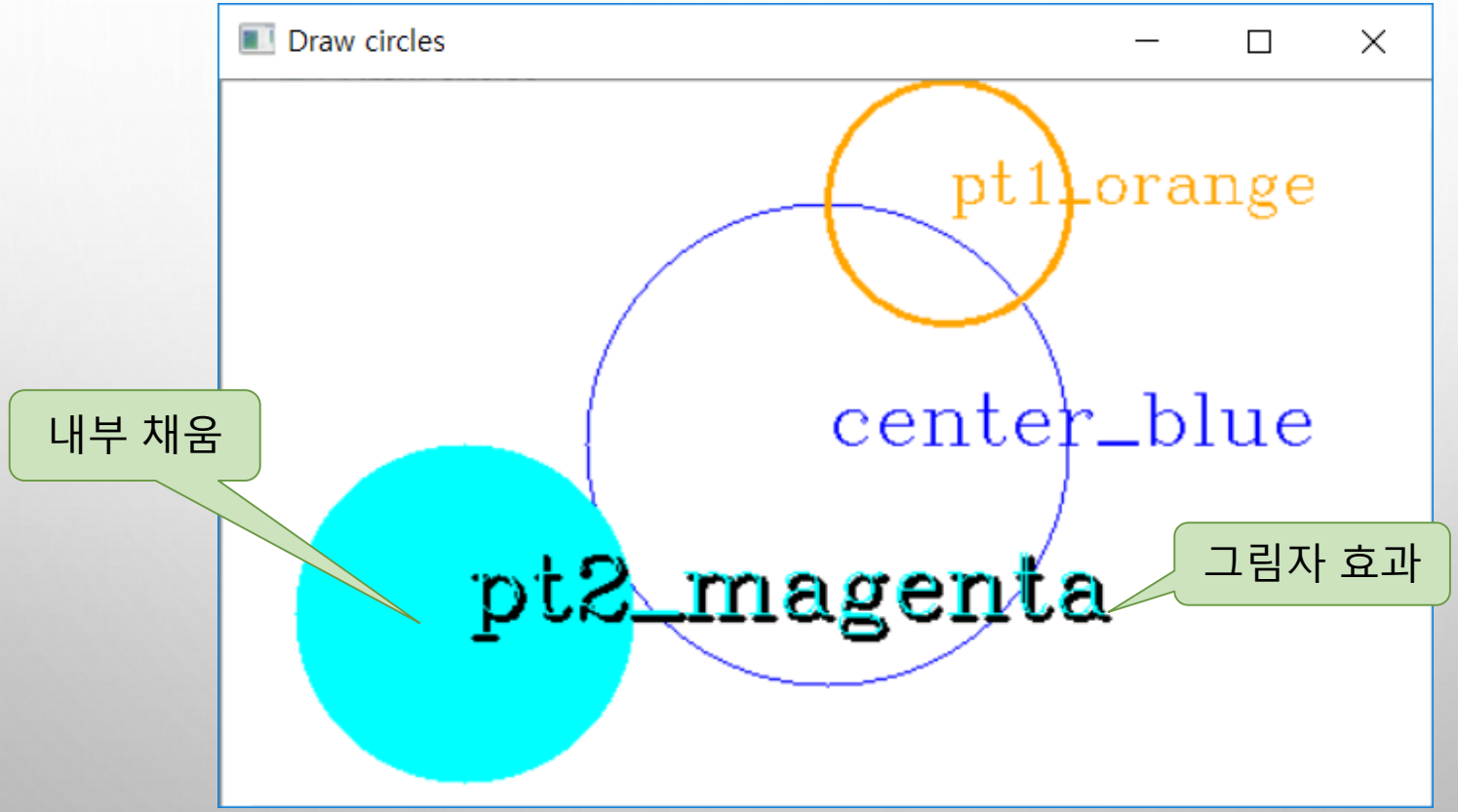
◆ 예제

예제 4.3.3 원 그리기 – 09.draw_circle.py

```
import numpy as np
import cv2

orange, blue, cyan = (0, 165, 255), (255, 0, 0), (255, 255, 0)
white, black = (255, 255, 255), (0, 0, 0)
image = np.full((300, 500, 3), white, np.uint8)          # 컬러 영상 생성 및 초기화

center = (image.shape[1]//2, image.shape[0]//2)          # 영상 중심 좌표 – 역순 구성
pt1, pt2 = (300, 50), (100, 220)
shade = (pt2[0] + 2, pt2[1] + 2)                         # 그림자 좌표

cv2.circle(image, center, 100, blue)                     # 원 그리기
cv2.circle(image, pt1, 50, orange, 2)
cv2.circle(image, pt2, 70, cyan, -1)                     # 원 내부 채움

font = cv2.FONT_HERSHEY_COMPLEX;
cv2.putText(image, 'center_blue', center, font, 1.0, blue)
cv2.putText(image, 'pt1_orange', pt1, font, 0.8, orange)
cv2.putText(image, 'pt2_cyan', shade, font, 1.2, black, 2)        # 그림자 효과
cv2.putText(image, 'pt2_cyan', pt2, font, 1.2, cyan,    1)

cv2.imshow("Draw circles", image)
cv2.waitKey(0)
```

4.3.3 원 그리기

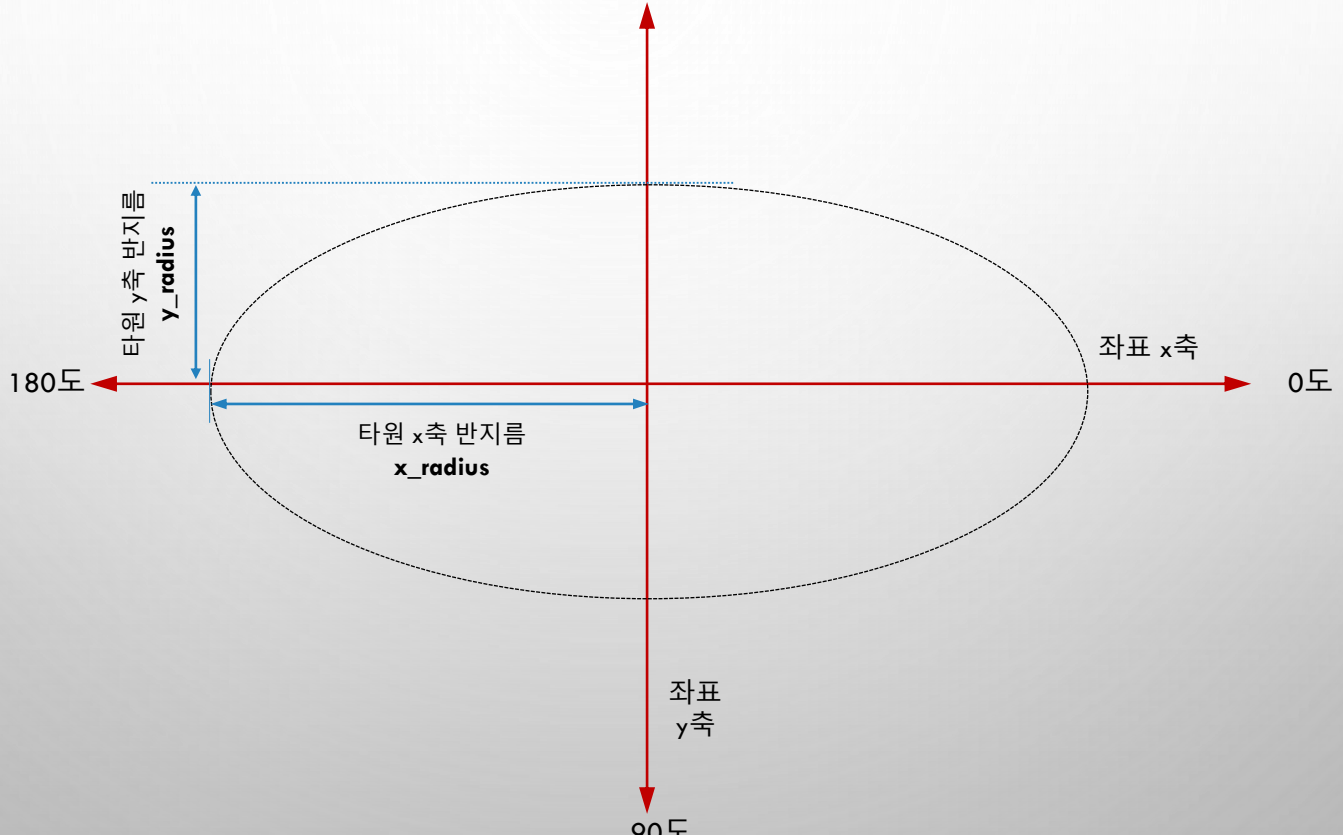
◆ 실행 결과



4.3.4 타원 그리기

◆ 3) 그리기 함수

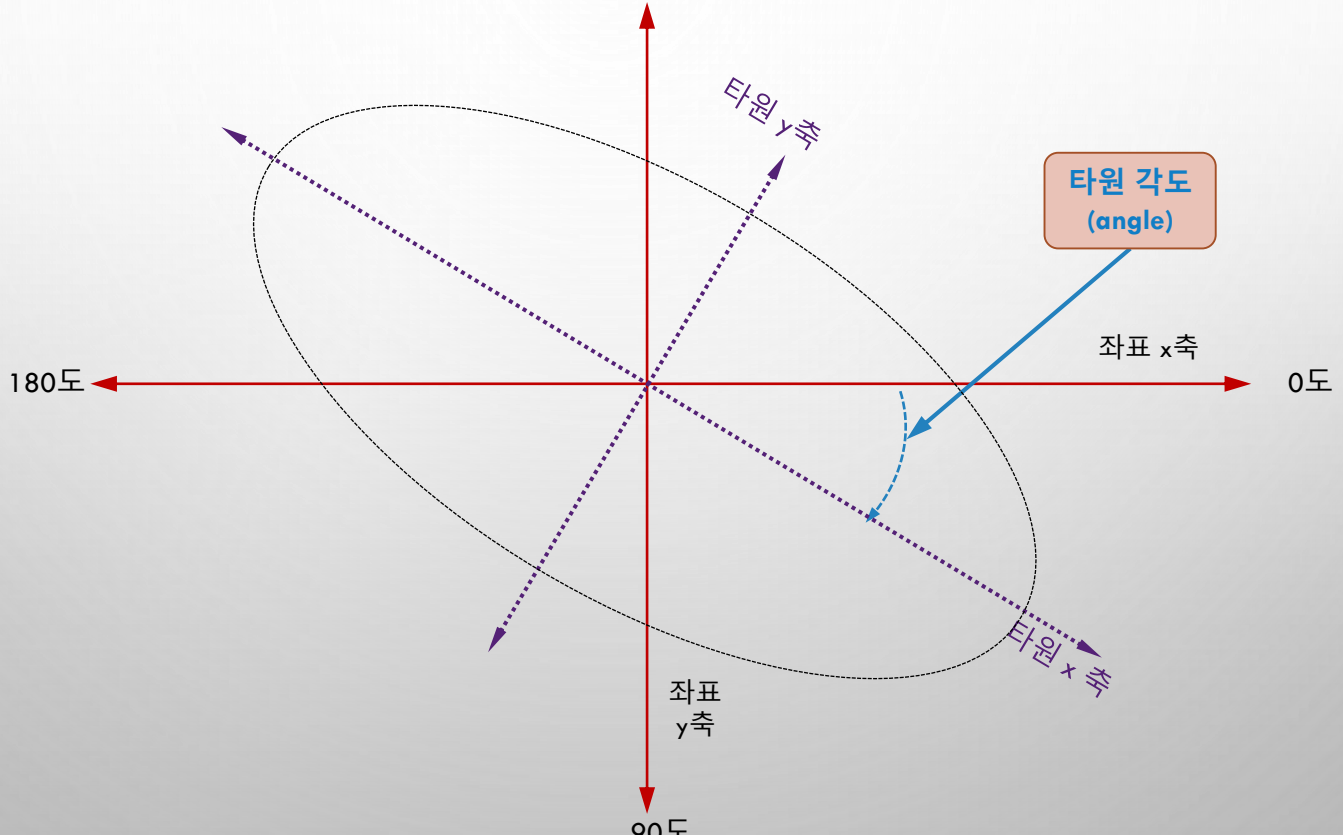
❖ (4) 타원 그리기



4.3.4 타원 그리기

◆ 3) 그리기 함수

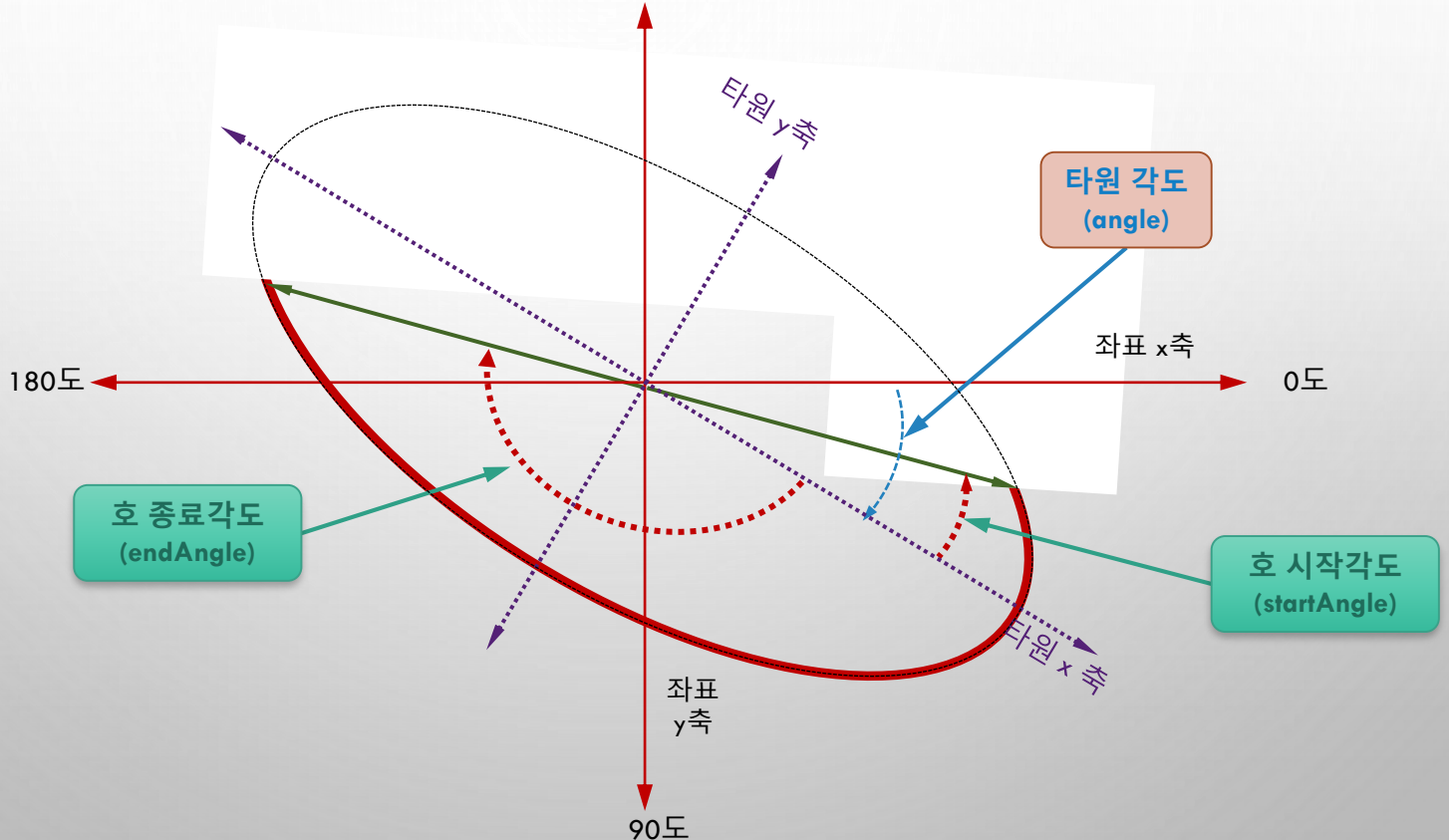
❖ (4) 타원 그리기



4.3.4 타원 그리기

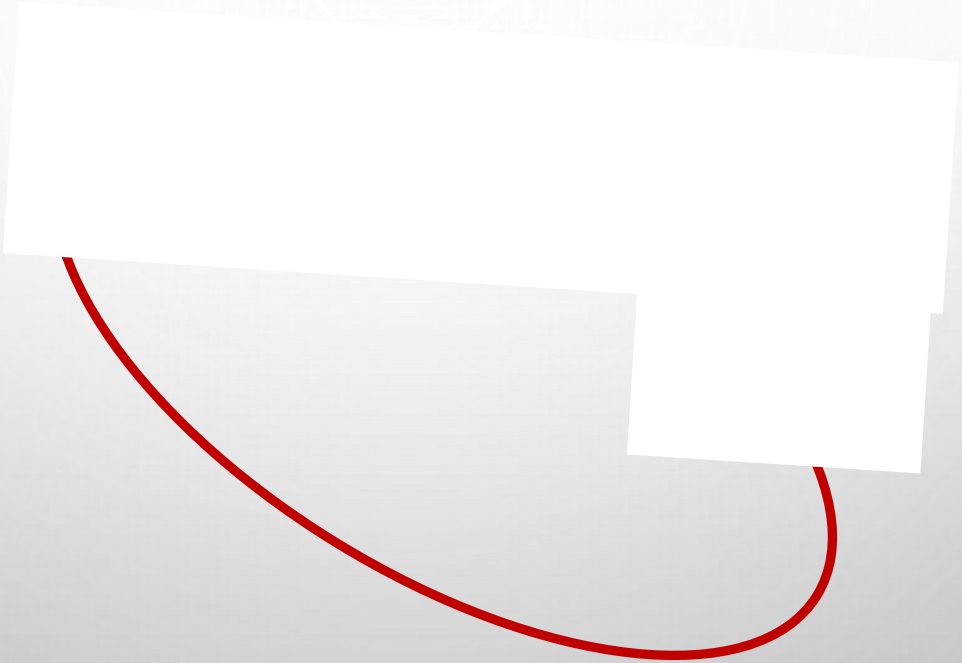
◆ 3) 그리기 함수

❖ (4) 타원 그리기



4.3.4 타원 그리기

- 3) 그리기 함수
  - (4) 타원 그리기



4.3.4 타원 그리기

함수명과 반환형 및 인수 구조		
cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color[, thickness[, lineType[, shift]]]) →img ■설명: center를 중심으로 axes 크기의 타원을 그린다.		
인수 설명	■img	그릴 대상 행렬(영상)
	■center	원의 중심 좌표
	■axes	타원의 절반 크기(x축 반지름, y축 반지름)
	■angle	타원의 각도 (3시 방향이 0도, 시계방향 회전)
	■startAngle	호의 시작 각도
	■endAngle	호의 종료 각도
	■color	선의 색상
	■thickness	선의 두께
	■lineType	선의 형태
	■shift	좌표에 대한 비트 시프트

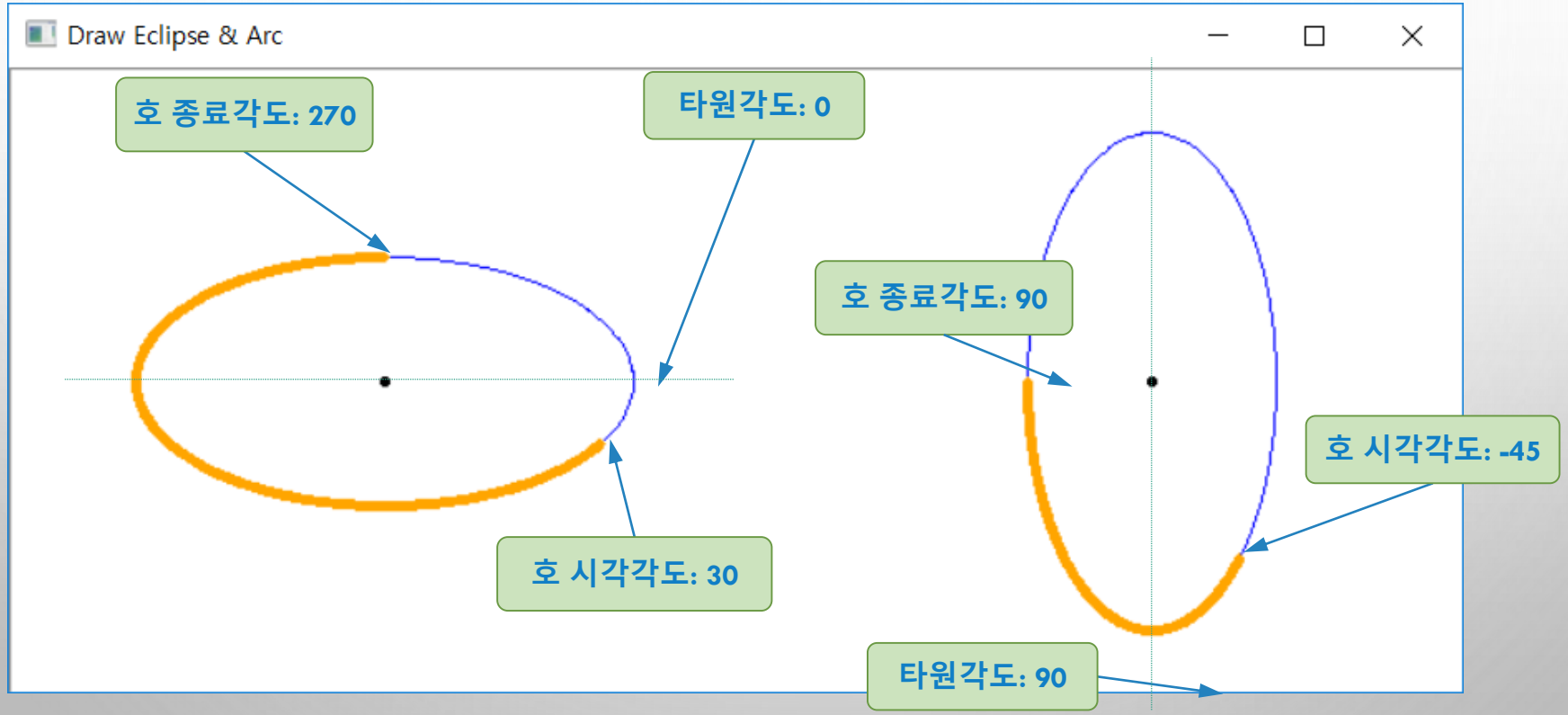
4.3.4 타원 그리기

예제 4.3.4 타원 및 호 그리기 – 10.draw_ellipse.py

```
01  import numpy as np
02  import cv2
03
04  orange, blue, white = (0, 165, 255), (255, 0, 0), (255,255,255) # 색상 지정
05  image = np.full((300, 700, 3), white, np.uint8)          # 3채널 행렬 생성 및 초기화
06
07  pt1, pt2 = (180, 150), (550, 150)                        # 타원 중심점
08  size = (120, 60)                                         # 타원 크기 – 반지름 값임
09
10  cv2.circle(image, pt1, 1, 0, 2)                          # 타원의 중심점(2화소 원) 표시
11  cv2.circle(image, pt2, 1, 0, 2)
12
13  cv2.ellipse(image, pt1, size,  0, 0, 360, blue, 1)       # 타원 그리기
14  cv2.ellipse(image, pt2, size, 90, 0, 360, blue, 1)
15  cv2.ellipse(image, pt1, size,  0, 30, 270, orange, 4)    # 호 그리기
16  cv2.ellipse(image, pt2, size, 90, -45, 90, orange, 4)
17
18  cv2.imshow("문자열", image)
19  cv2.waitKey()                                            # 키입력 대기
```

4.3.4 타원 그리기

◆ 실행 결과

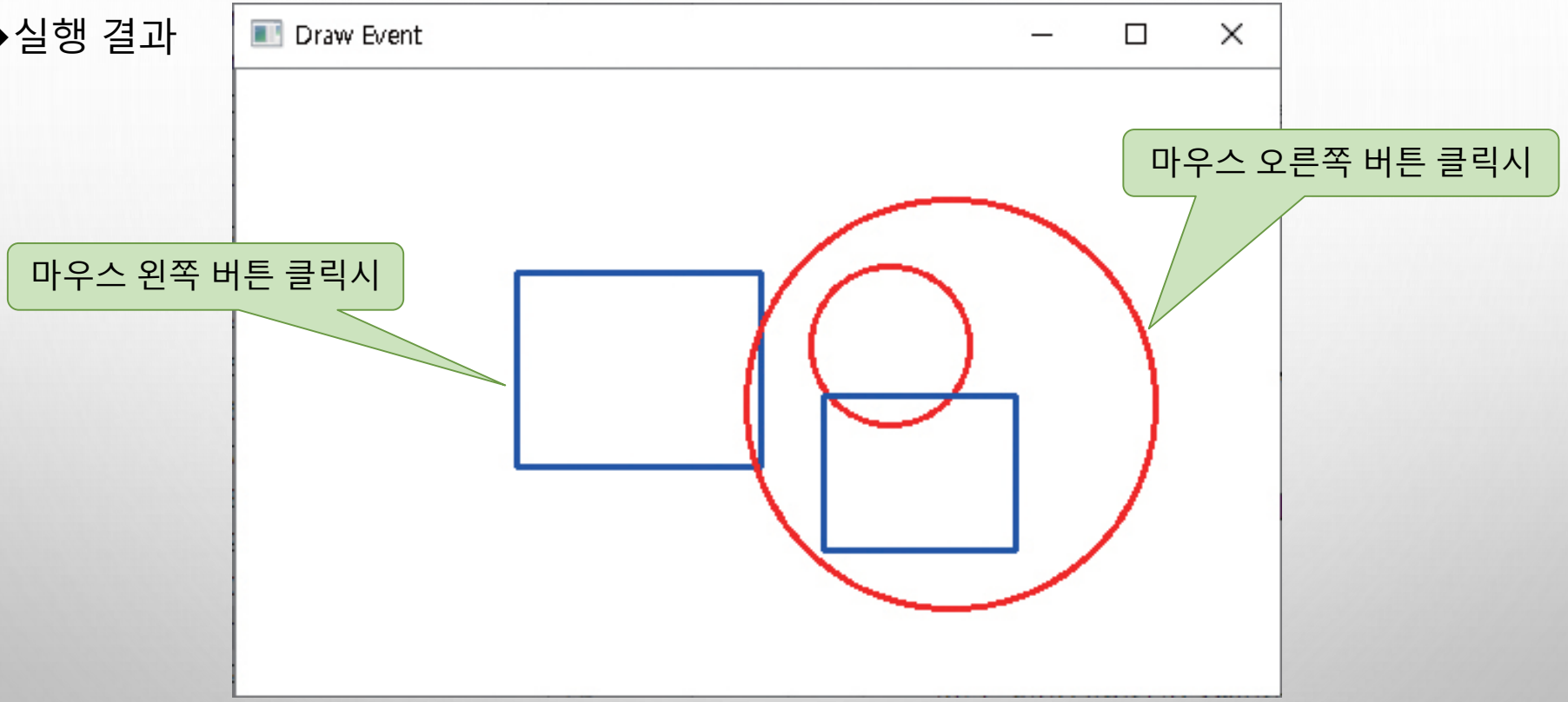


```python
import numpy as np
import cv2

def onMouse(event, x, y, flags, param):
    global title, pt

    if event == cv2.EVENT_LBUTTONDOWN:
        if pt[0] < 0: pt = (x, y)
        else:
            cv2.rectangle(image, pt, (x, y), (255, 0, 0), 2)  # 파란색 사각형
            cv2.imshow(title, image)
            pt = (-1, -1)                                     # 시작 좌표 초기화

    elif event == cv2.EVENT_RBUTTONDOWN:
        if pt[0] < 0: pt = (x, y)
        else:
            dx, dy = pt[0] - x, pt[1] - y                     # 두 좌표 간 간격
            radius = int(np.sqrt(dx*dx + dy*dy))
            cv2.circle(image, pt, radius, (0, 0, 255), 2)     # 빨간색 원
            cv2.imshow(title, image)
            pt = (-1, -1)                                     # 시작 좌표 초기화
```

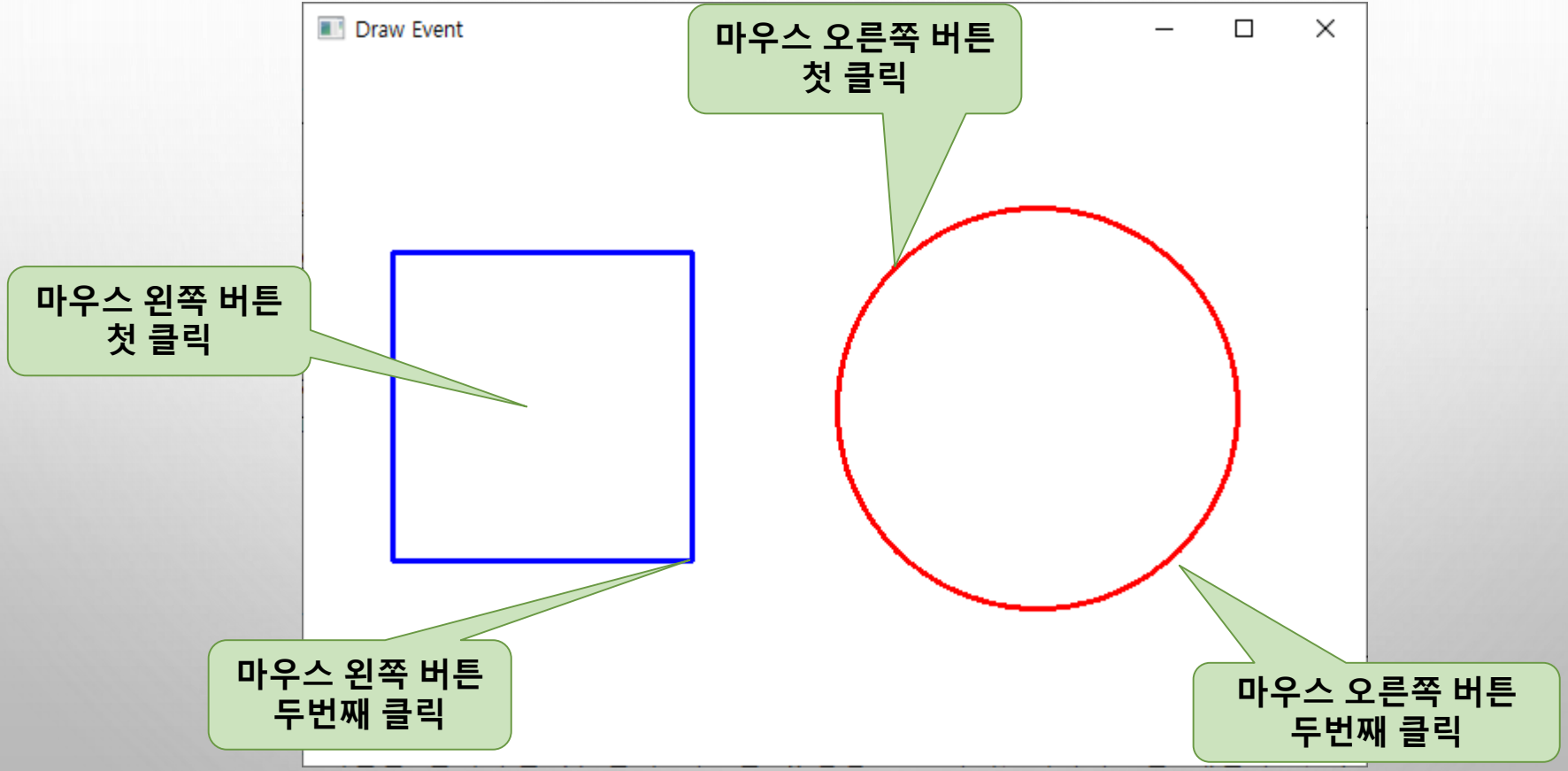
```python
image = np.full((300, 500, 3), (255, 255, 255), np.uint8)

pt = (-1, -1)
title = "Draw Event"
cv2.imshow(title, image)
cv2.setMouseCallback(title, onMouse)
cv2.waitKey(0)
```

심화 예제

◆실행 결과



심화 예제2


Draw Event
마우스 오른쪽 버튼
첫 클릭
마우스 왼쪽 버튼
첫 클릭
마우스 왼쪽 버튼
두번째 클릭
마우스 오른쪽 버튼
두번째 클릭

심화 예제2 - 정답

```
elif event == cv2.EVENT_RBUTTONDOWN:
    if pt[0] < 0: pt = (x, y)
    else:
        dx, dy = x-pt[0],  y-pt[1]
        center = (pt[0]+dx//2, pt[1]+dy//2)
        center = (pt[0] + x) // 2, (pt[1] + y) // 2
        center = np.add(pt, (x,y))//2


        radius = int(np.sqrt(dx * dx + dy * dy))//2

        cv2.circle(image, center, radius, (0,0,255), 2)
        cv2.imshow(title, image)
        pt = (-1, -1)
```

4.4 영상파일 처리

◆ 영상처리 과정

- ❖ 영상 파일을 읽어 들여 행렬에 저장
- ❖ 행렬 연산 과정에서 행렬의 원소를 필요할 때마다 눈으로 직접 확인
- ❖ 처리된 결과 행렬을 영상 파일로 저장 →

◆ 영상파일을 처리해 주는 함수와 활용 방법 필수

함수 설명		
cv2.imread(filename[, flags]) → retval ■ 설명: 지정한 영상파일로부터 영상을 적재한 후, 행렬로 반환한다.		
인수 설명	■ filename	적재할 영상파일 이름(디렉터리 구조 포함)
	■ flags	적재한 영상을 행렬로 반환될 때 컬러 타입을 결정하는 상수
cv2.imwrite(filename, img[, params]) → retval ■ 설명: img 행렬을 지정한 영상파일로 저장한다.		
인수 설명	■ filename	저장할 영상파일 이름(디렉터리 구조 포함), 확장자명에 따라 영상파일 형식 결정
	■ img	저장하고자 하는 행렬(영상)
	■ params	압축 방식에 사용되는 인수 쌍(paramId, paramValue)

4.4 영상파일 처리

〈표 4.4.1〉 행렬의 컬러 타입 결정 상수

옵션	값	설명
cv2.IMREAD_UNCHANGED	-1	입력 파일에 정의된 타입의 영상을 그대로 반환(알파(alpha) 채널 포함)
cv2.IMREAD_GRAYSCALE	0	명암도(grayscale) 영상으로 변환하여 반환
cv2.IMREAD_COLOR	1	컬러 영상으로 변환하여 반환
cv2.IMREAD_ANYDEPTH	2	입력 파일에 정의된 깊이(depth)에 따라 16비트/32비트 영상으로 변환, 설정되지 않으면 8비트 영상으로 변환
cv2.IMREAD_ANYCOLOR	4	입력 파일에 정의된 타입의 영상을 반환

4.4.1 영상파일 읽기

예제 4.4.1 영상파일 읽기1 – 12.read_image1.py

```
import cv2

def print_matInfo(name, image):                          # 행렬 정보 출력 함수
    if image.dtype == 'uint8':      mat_type = 'CV_8U'
    elif image.dtype == 'int8':     mat_type = 'CV_8S'
    elif image.dtype == 'uint16':   mat_type = 'CV_16U'
    elif image.dtype == 'int16':    mat_type = 'CV_16S'
    elif image.dtype == 'float32':  mat_type = 'CV_32F'
    elif image.dtype == 'float64':  mat_type = 'CV_64F'
    nchannel = 3 if image.ndim == 3 else 1

    ## depth, channel 출력
    print("%12s: depth(%s), channels(%s) -> mat_type(%sC%d)"
        % (name, image.dtype, nchannel, mat_type,  nchannel))

title1, title2 = 'gray2gray', 'gray2color'                          # 윈도우 이름
gray2gray = cv2.imread("images/read_gray.jpg", cv2.IMREAD_GRAYSCALE)     # 명암도
gray2color = cv2.imread("images/read_gray.jpg", cv2.IMREAD_COLOR)        # 컬러 영상

```

4.4.1 영상파일 읽기

```python
## 예외처리 –영상파일 읽기 여부 조사
if gray2gray is None or gray2color is None :
    raise Exception("영상파일 읽기 에러")

print("행렬 좌표 (100, 100) 화소값")
print("%s %s" % (title1, gray2gray[100, 100]))          # 행렬 내 한 화소값 표시
print("%s %s\n" % (title2, gray2color[100, 100]))

print_matInfo(title1, gray2gray)                        # 행렬 정보 출력 함수 호출
print_matInfo(title2, gray2color)

cv2.imshow(title1, gray2gray)                           # 행렬 정보를 영상으로 띄우기
cv2.imshow(title2, gray2color)
cv2.waitKey(0)
```

```
Run: 12.read_image1
C:\Python\python.exe D:/source/chap04/12.read_image1.py
Traceback (most recent call last):
  File "D:/source/chap04/12.read_image1.py", line 23, in <mo
    raise Exception("영상파일 읽기 에러")
Exception: 영상파일 읽기 에러
```

4.4.1 영상파일 읽기

◆실행결과

명암도 영상
(1채널 화소)

```
Run:  12.read_image1
C:\Python\python.exe D:/source/chap04/12.read_image1.py
행렬 좌표 (100, 100) 화소값
gray2gray 106
gray2color [106 106 106]

   gray2gray: depth(uint8), channels(1) -> mat_type(CV_8UC1)
  gray2color: depth(uint8), channels(3) -> mat_type(CV_8UC3)
```

3채널 화소: 명암도 영상에서 변환해서 동일한 값임

행렬 자료형

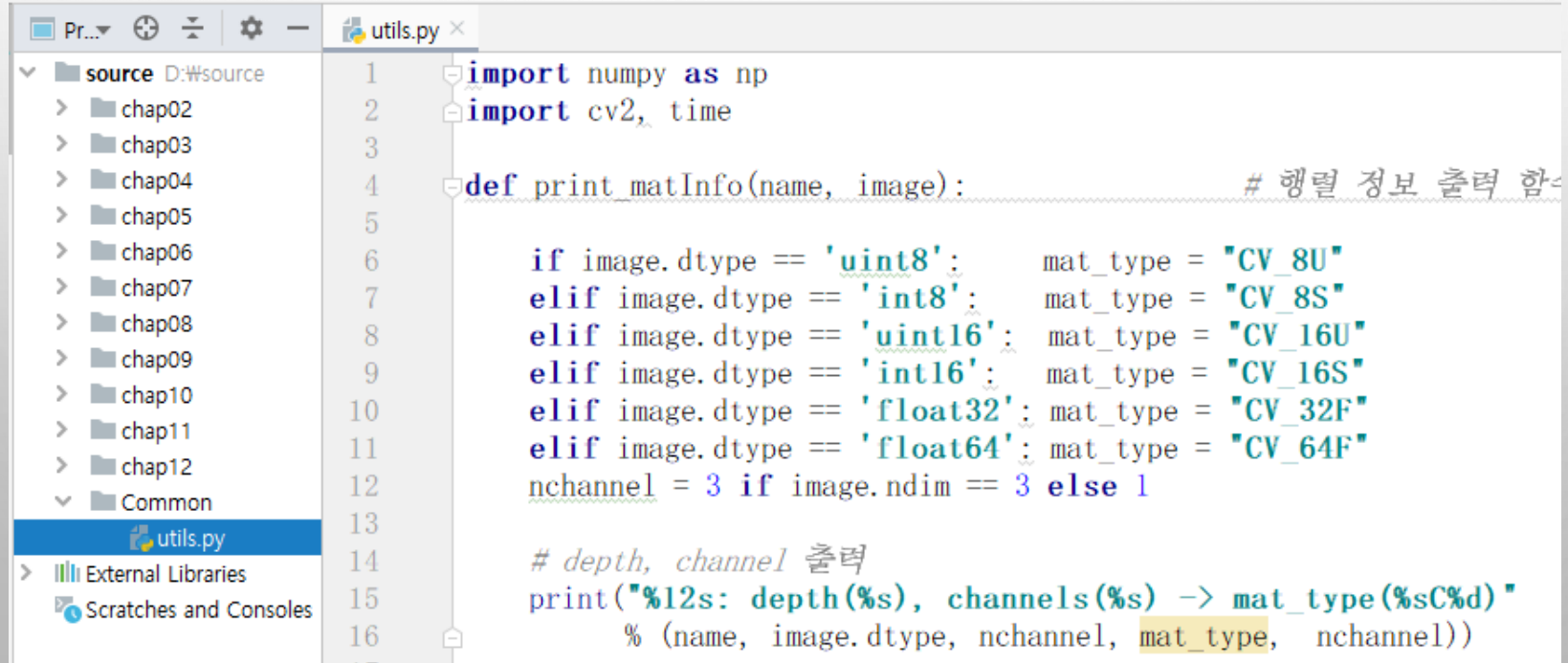
모듈 임포트 하기

◆ 모듈 라이브러리 파일 만들기

❖ 프로젝트 루트 폴더(source)에서 마우스 우버튼 눌러 'Common' 이름으로 폴더 생성

- 루트 폴더에 'Common' 폴더를 두는 것 - 다른 챕터의 소스 파일들에서도 모듈의 쉬운 사용위해

❖ 'Common' 폴더에서 파이썬 소스 파일(utils.py) 추가



4.4.1 영상파일 읽기

예제 4.4.2 영상파일 읽기(컬러) – 13.read_image2.py

저자 생성 모듈의 함수 임포트

```
01  import cv2
02  from Common.utils import print_matInfo        # 행렬 정보 출력 함수 임포트
03
04  title1, title2 = 'color2gray', 'color2color'
05  color2gray = cv2.imread("images/read_color.jpg", cv2.IMREAD_GRAYSCALE)
06  color2color = cv2.imread("images/read_color.jpg", cv2.IMREAD_COLOR)
07  if color2gray is None or color2color is None:          # 예외처리
08      raise Exception("영상파일 읽기 에러")
09
10  print("행렬 좌표(100, 100) 화소값")
11  print("%s %s" % (title1, color2gray[100, 100]))          # 한 화소값 표시
12  print("%s %s\n" % (title2, color2color[100, 100]))
13
14  print_matInfo(title1, color2gray)                        # 행렬 정보 출력
15  print_matInfo(title2, color2color)
16  cv2.imshow(title1, color2gray)                           # 행렬 정보 영상으로 띄우기
17  cv2.imshow(title2, color2color)
18  cv2.waitKey(0)
```

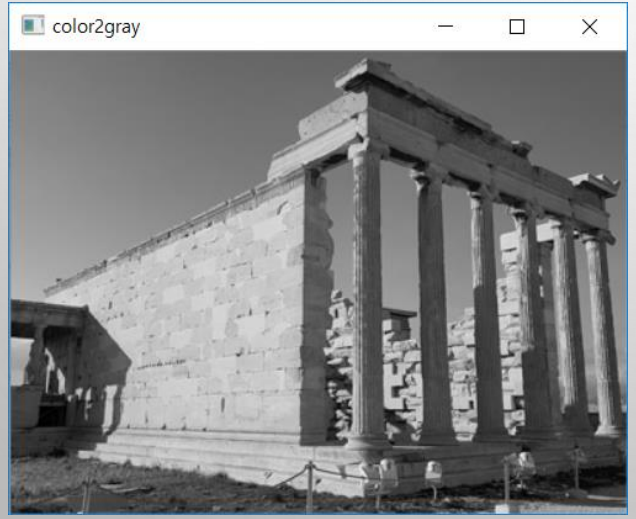
4.4.1 영상파일 읽기

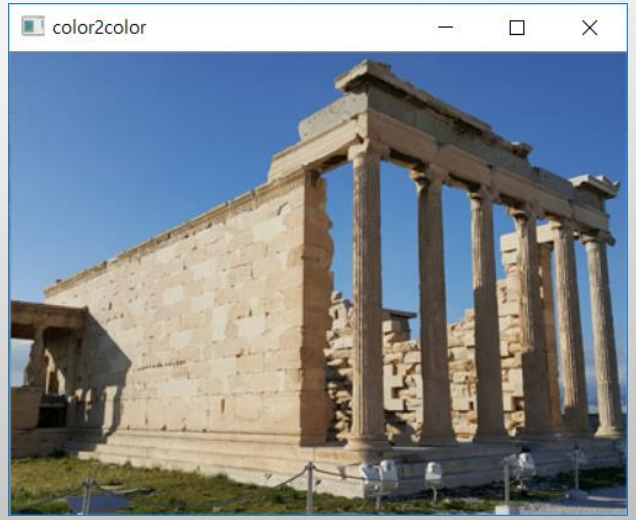
◆실행결과

컬러 영상도 명암도 타입으로 읽으면 1채널 영상이 됨

```
Run: 13.read_image2
C:\Python\python.exe D:/source/chap04/13.read_image2.py
행렬 좌표 (100, 100) 화소값
color2gray 137
color2color [197 145  98]

  color2gray: depth(uint8), channels(1) -> mat_type(CV_8UC1)
 color2color: depth(uint8), channels(3) -> mat_type(CV_8UC3)
```

3채널 컬러 영상 화소값





4.4.1 영상파일 읽기

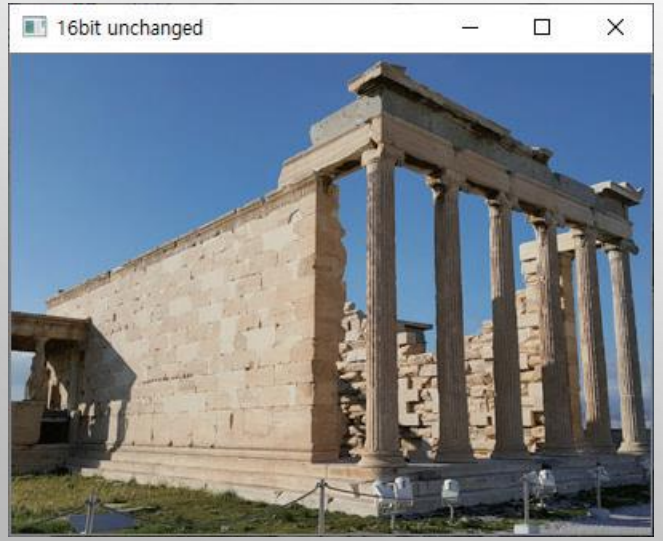
예제 4.4.3 영상파일 읽기 – 14.read_image3.py

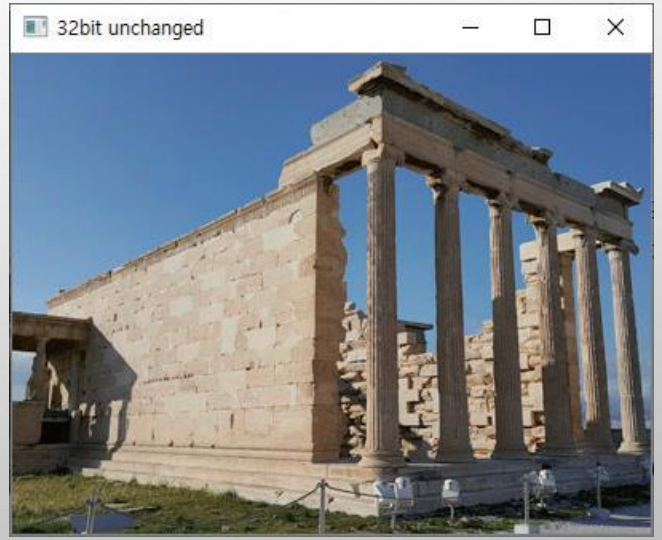
```
import cv2
from Common.utils import print_matInfo                # 행렬 정보 출력 함수 임포트

title1, title2 = '16bit unchanged', '32bit unchanged'          # 윈도우 이름
color2unchanged1 = cv2.imread("images/read_16.tif", cv2.IMREAD_UNCHANGED)
color2unchanged2 = cv2.imread("images/read_32.tif", cv2.IMREAD_UNCHANGED)
if color2unchanged1 is None or color2unchanged2 is None:       # 영상파일 예외처리
    raise Exception("영상파일 읽기 에러")

print("16/32비트 영상 행렬 좌표(10, 10) 화소값")
print(title1, '원소 자료형 ', type(color2unchanged1[10][10][0]))     # 원소 자료형
print(title1, '화소값(3원소) ', color2unchanged1[10, 10] )          # 한 화소 값 표시
print(title2, '원소 자료형 ', type(color2unchanged2[10][10][0]))
print(title2, '화소값(3원소) ', color2unchanged2[10, 10] )
print()

print_matInfo(title1, color2unchanged1)                         # 행렬 정보 출력
print_matInfo(title2, color2unchanged2)
cv2.imshow(title1, color2unchanged1)                  # 16비트 정수형 행렬 영상 표시
cv2.imshow(title2, color2unchanged2)                            # 정수형 변환 후 표시
cv2.waitKey(0)
```

4.4.1 영상파일 읽기

◆실행결과

```
Run:  14.read_image3
C:\Python\python.exe D:/source/chap04/14.read_image3.py
16/32비트 영상 행렬 좌표 (10, 10) 화소값
16bit unchanged 원소 자료형  <class 'numpy.uint16'>
16bit unchanged 화소값(3원소)  [48573 34438 23387]
32bit unchanged 원소 자료형  <class 'numpy.float32'>
32bit unchanged 화소값(3원소)  [0.7456989  0.52237624 0.36376417]

16bit unchanged: depth(uint16), channels(3) -> mat_type(CV_16UC3)
32bit unchanged: depth(float32), channels(3) -> mat_type(CV_32FC3)
```





4.4.2 행렬을 영상파일로 저장

◆CV2.IMWRITE() 함수

❖ 행렬을 영상파일로 저장

❖ 확장자에 따라서 JPG, BMP, PNG, TIF, PPM 등의 영상파일 포맷 저장 가능

〈표 4.4.3〉 압축 방식에 사용되는 params 인수 튜플(paramId, paramValue)의 예시

paramId	paramValue (기본값)	설명
cv2.IMWRITE_JPEG_QUALITY	0~100 (95)	JPG 파일 화질, 높은 값일수록 화질 좋음
cv2.IMWRITE_PNG_COMPRESSION	0~9 (3)	PNG 파일 압축 레벨, 높은 값일수록 용량은 적어지고, 압축 시간이 길어짐
cv2.IMWRITE_PXM_BINARY	0 or 1 (1)	PPM, PGM 파일의 이진 포맷 설정

4.4.2 행렬을 영상파일로 저장

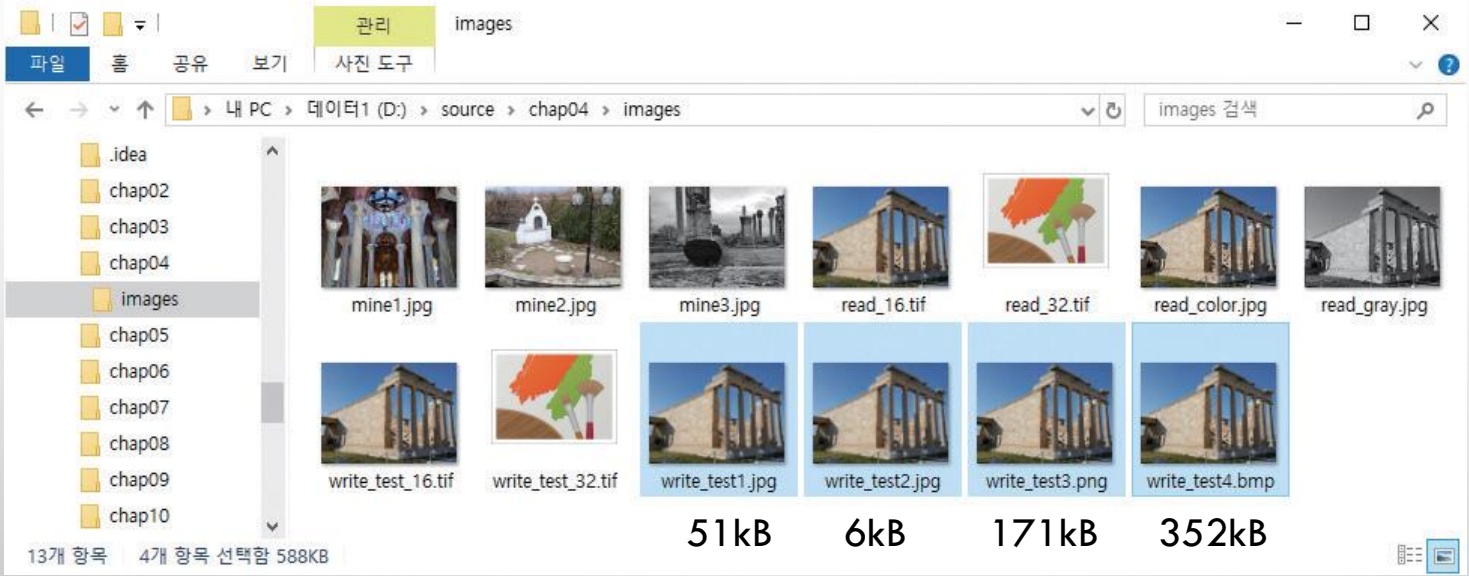
◆예제

예제 4.4.3 행렬 영상 저장1 – 15.write_image1.py

```
import cv2

image = cv2.imread("images/read_color.jpg", cv2.IMREAD_COLOR)
if image is None: raise Exception("영상파일 읽기 에러")          # 예외처리

params_jpg = (cv2.IMWRITE_JPEG_QUALITY, 10)                     # JPEG 화질 설정
params_png = [cv2.IMWRITE_PNG_COMPRESSION, 9]                   # PNG 압축 레벨 설정

## 행렬을 영상파일로 저장
cv2.imwrite("images/write_test1.jpg", image)                    # 디폴트는 95
cv2.imwrite("images/write_test2.jpg", image, params_jpg)        # 지정한 화질로 저장
cv2.imwrite("images/write_test3.png", image, params_png)
cv2.imwrite("iamges/write_test4.bmp", image)                    # BMP 파일로 저장
print("저장 완료")
```

4.4.2 행렬을 영상파일로 저장

◆ 실행결과



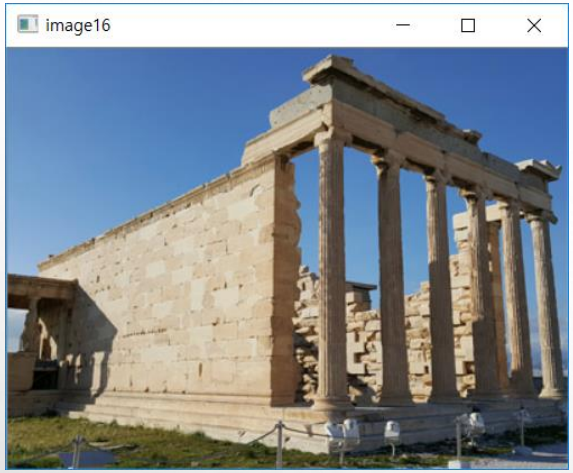
심화 예제

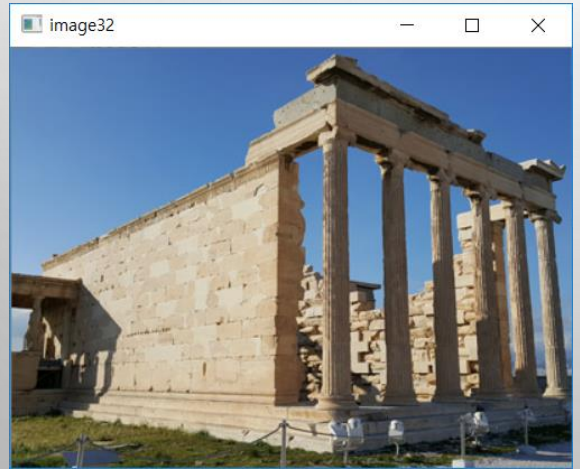
◆ 16비트 32

심화예제 4.4.4 행렬 영상 저장2 – 16.write_image2.py

```
import numpy as np
import cv2

image8 = cv2.imread("images/read_color.jpg", cv2.IMREAD_COLOR)
if image8 is None: raise Exception("영상파일 읽기 에러")          # 영상파일 예외처리

image16 = np.uint16(image8 * (65535/255))       # 형변환 및 화소 스케일 조정
image32 = np.float32(image8 * (1/255))

## 화소값 확인 – 관심 영역((10, 10) 위치에서 2행, 3열) 출력
print("image8 행렬의 일부\n %s\n"  % image8[10:12, 10:13])
print("image16 행렬의 일부\n %s\n" % image16[10:12, 10:13])
print("image32 행렬의 일부\n %s\n" % image32[10:12, 10:13])

cv2.imwrite('images/write_test_16.tif', image16)          # 16비트 행렬 저장
cv2.imwrite('images/write_test_32.tif', image32)          # 32비트 행렬 저정

cv2.imshow('image16', image16)          # 영상 표시
cv2.imshow('image32', (image32*255).astype('uint8'))
cv2.waitKey(0)
```

심화 예제





3개
화소

첫번째 행

두번째 행

```
Run:  16.write_image2
C:\Python\python.exe
 D:/source/chap04/16.write_image2.py
image8 행렬의 일부
 [[[189 134  91]
  [189 134  91]
  [189 134  91]]

 [[189 134  91]
  [189 134  91]
  [189 134  91]]]

image16 행렬의 일부
 [[[48573 34438 23387]
  [48573 34438 23387]
  [48573 34438 23387]]

 [[48573 34438 23387]
  [48573 34438 23387]
  [48573 34438 23387]]]

image32 행렬의 일부
 [[[0.7411765  0.5254902  0.35686275]
  [0.7411765  0.5254902  0.35686275]
  [0.7411765  0.5254902  0.35686275]]

 [[0.7411765  0.5254902  0.35686275]
  [0.7411765  0.5254902  0.35686275]
  [0.7411765  0.5254902  0.35686275]]]
```

4.5 비디오 처리

◆ 동영상 파일

❖ 초당 30프레임 저장, 압출 필요 → 압축 코덱(codec) 사용함

◆ OpenCV는 동영상을 처리할 수 있는 클래스 제공

VideoCapture 클래스 함수 설명		
cv2.VideoCapture() → 〈VideoCapture object〉 cv2.VideoCapture(filename) → 〈VideoCapture object〉 cv2.VideoCapture(device) → 〈VideoCapture object〉 ■ 설명: 생성자, 3가지 VideoCapture 객체 선언 방법을 지원한다.		
인수 설명	■ filename	개방할 동영상파일의 이름 혹은 영상 시퀀스
	■ device	개방할 동영상 캡처 장치의 ID(카메라 한대만 연결되면 0을 지정)
cv2.VideoCapture.open(filename) → retval cv2.VideoCapture.open(device) → retval ■ 설명: 동영상 캡처를 위한 동영상파일 혹은 캡처 장치를 개방한다.		
cv2.VideoCapture.isOpened() → retval ■ 설명: 캡처 장치의 연결 여부를 반환한다.		
cv2.VideoCapture.release() → None ■ 설명: 동영상파일이나 캡처 장치를 해제한다. (클래스 소멸자에 의해서 자동으로 호출되므로 명시적으로 수행하지 않아도 됨)		

4.5 비디오 처리

cv2.VideoCapture.get(propId) → retval

■ 설명: 비디오 캡처의 속성 식별자로 지정된 속성의 값을 반환한다. 캡처 장치가 제공하지 않는 속성은 0을 반환한다.

인수 설명	■ propId	속성 식별자 – 〈표 4.5.1〉에서 정리

cv2.VideoCapture.set(propId, value) → retval

■ 설명: 지정된 속성 식별자로 비디오 캡처의 속성을 설정한다.

인수 설명	■ propId	속성 식별자
	■ value	속성 값

cv2.VideoCapture.grab() → retval

■ 설명: 캡처 장치나 동영상파일에서 다음 프레임을 잡는다.

retrieve([, image[, flag]]) → retval, image

■ 설명: grab()으로 잡은 프레임을 디코드해서 image 행렬로 전달한다.

인수 설명	■ image	잡은 프레임이 저장되는 행렬
	■ flag	프레임 인덱스

cv2.VideoCapture.read([image]) → retval, image

■ 설명: 캡처 장치나 동영상파일에서 다음 프레임을 잡아 디코드해서 image 행렬로 전달한다.

4.5 비디오 처리

VideoWriter 클래스 설명		
cv2.VideoWriter([filename, fourcc, fps, frameSize[, isColor]]) → 〈VideoWriter object〉 cv2.VideoWriter.open(filename, fourcc, fps, frameSize[, isColor]) → retval		
인수 설명	■ filename	출력 동영상파일의 이름
	■ fourcc	프레임 압축에 사용되는 코덱의 4-문자
	■ fps	생성된 동영상 프레임들의 프레임률(초당 프레임수)
	■ frameSize	동영상 프레임의 크기(가로×세로)
	■ isColor	True면 컬러 프레임으로 인코딩, False면 명암도 프레임으로 인코딩
cv2.VideoWriter.isOpened() → retval ■ 설명: 캡처 장치나 동영상파일이 열려있는지 확인한다.		
cv2.VideoWriter.write(image) → None ■ 설명: image 프레임을 파일로 저장한다.		
cv2.VideoWriter.open() ■ 설명: 영상을 동영상파일의 프레임으로 저장하기 위해 동영상파일을 개방한다. 인수는 생성자의 인수와 동일하다.		
cv2.VideoWriter.isOpened() ■ 설명: 동영상파일 저장을 위해 VideoWriter 객체의 개방 여부를 확인한다.		

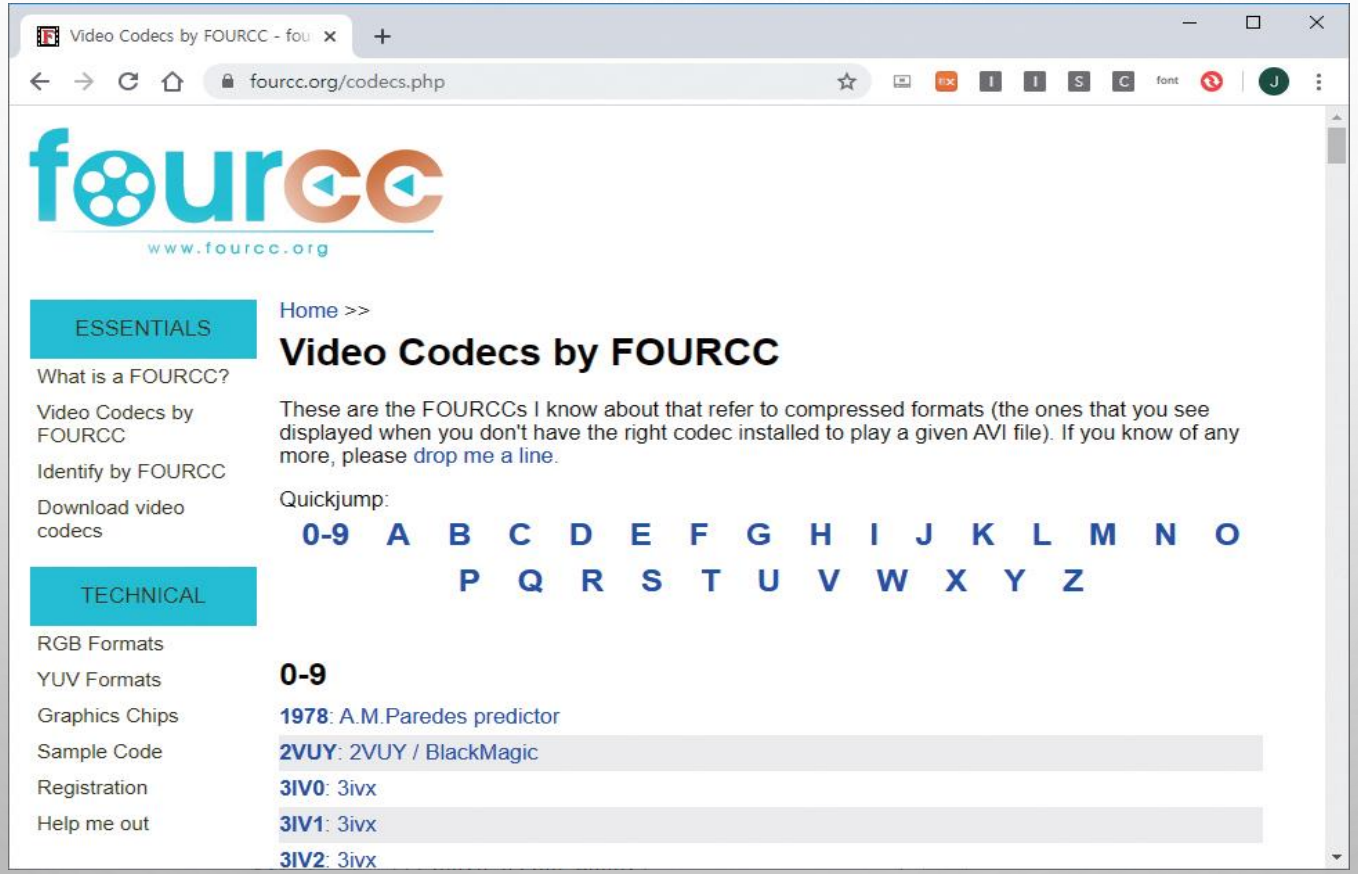
4.5 비디오 처리

〈표 4.5.1〉 카메라의 주요 속성 식별자

속성 상수	설명
cv2.CAP_PROP_POS_MSEC	동영상파일의 현재 위치인 밀리초(millisecond)
cv2.CAP_PROP_POS_FRAMES	캡처되는 프레임의 번호
cv2.CAP_PROP_POS_AVI_RATIO	동영상파일의 상대적 위치 (0 – 시작, 1 – 끝)
cv2.CAP_PROP_FRAME_WIDTH	프레임의 너비
cv2.CAP_PROP_FRAME_HEIGHT	프레임의 높이
cv2.CAP_PROP_FPS	초당 프레임 수
cv2.CAP_PROP_FOURCC	코덱의 4문자
cv2.CAP_PROP_FRAME_COUNT	동영상파일의 총 프레임 수
cv2.CAP_PROP_FORMAT	cv2.VideoCatpure.retrieve()이 반환하는 행렬 포맷
cv2.CAP_PROP_BRIGHTNESS	카메라에서 영상의 밝기
cv2.CAP_PROP_CONTRAST	카메라에서 영상의 대비
cv2.CAP_PROP_SATURATION	카메라에서 영상의 포화도
cv2.CAP_PROP_HUE	카메라에서 영상의 색상
cv2.CAP_PROP_GAIN	카메라에서 영상의 Gain
cv2.CAP_PROP_EXPOSURE	카메라에서 노출
cv2.CAP_PROP_AUTOFOCUS	자동 초점 조절

4.5 비디오 처리

https://www.fourcc.org/codecs.php



4.5 비디오 처리

◆주요 코덱 문자

속성 상수	설명
cv2.VideoWriter_fourcc('D', 'I', 'V', '4') cv2.VideoWrite_fourcc(*'DIV4')	DivX MPEG-4
cv2.VideoWriter_fourcc('D', 'I', 'V', '5') cv2.VideoWrite fourcc(*'DIV5')	Div5
cv2.VideoWriter_fourcc('D', 'I', 'V', 'X') cv2.VideoWrite_fourcc(*'DIVX')	DivX
cv2.VideoWriter_fourcc('D', 'X', '5', '0') cv2.VideoWrite_fourcc(*'DX50')	DivX MPEG-4
cv2.VideoWriter_fourcc('F', 'M', 'P', '4') cv2.VideoWrite_fourcc(*'FMP4')	FFMpeg
cv2.VideoWriter_fourcc('I', 'Y', 'U', 'V') cv2.VideoWrite_fourcc(*'TYUV')	IYUV
cv2.VideoWriter_fourcc('M', 'J', 'P', 'G') cv2.VideoWrite_fourcc(*'MJPG')	Motion JPEG codec
cv2.VideoWriter_fourcc'M', 'P', '4', '2') cv2.VideoWrite_fourcc(*'MP42')	MPEG4 v2
cv2.VideoWriter_fourcc('M', 'P', 'E', 'G') cv2.VideoWrite_fourcc(*'MPEG')	MPEG codecs
cv2.VideoWriter_fourcc('X', 'V', 'I', 'D') cv2.VideoWrite_fourcc(*'XVID')	XVID codecs
cv2.VideoWriter_fourcc('X', '2', '6', '4') cv2.VideoWrite_fourcc(*'X264')	H.264/AVC codecs
-1	코덱 선택 대화상자 띄움

4.5.1 카메라에서 프레임 읽기

예제 4.5.1 카메라 프레임 읽기 – 17.read_pccamera.py

```
01  import cv2
02
03  def put_string(frame, text, pt, value, color=(120, 200, 90) ):  # 문자열 출력 함수
04      text += str(value)
05      shade = (pt[0] + 2, pt[1] + 2)
06      font = cv2.FONT_HERSHEY_SIMPLEX
07      cv2.putText(frame, text, shade, font, 0.7, (0, 0, 0), 2)      # 그림자 효과
08      cv2.putText(frame, text, pt    , font, 0.7, color, 2)         # 글자 적기
09
10  capture = cv2.VideoCapture(0)                          # 0번 카메라 연결
11  if capture.isOpened() == False:                        # 카메라 연결 예외처리
12       raise Exception("카메라 연결 안됨")
13
14  ## 카메라 속성 획득 및 출력
15  print("너비 %d" % capture.get(cv2.CAP_PROP_FRAME_WIDTH))
16  print("높이 %d" % capture.get(cv2.CAP_PROP_FRAME_HEIGHT))
17  print("노출 %d" % capture.get(cv2.CAP_PROP_EXPOSURE))
18  print("밝기 %d" % capture.get(cv2.CAP_PROP_BRIGHTNESS))
19
```

Run: 17.read_pccamera

```
C:\Python\python.exe D:/source/chap04/17.read_pccamera.py
Traceback (most recent call last):
  File "D:/source/chap04/17.read_pccamera.py", line 11, in <module>
    if capture.isOpened() == False: raise Exception("카메라 연결 안됨")
Exception: 카메라 연결 안됨
```

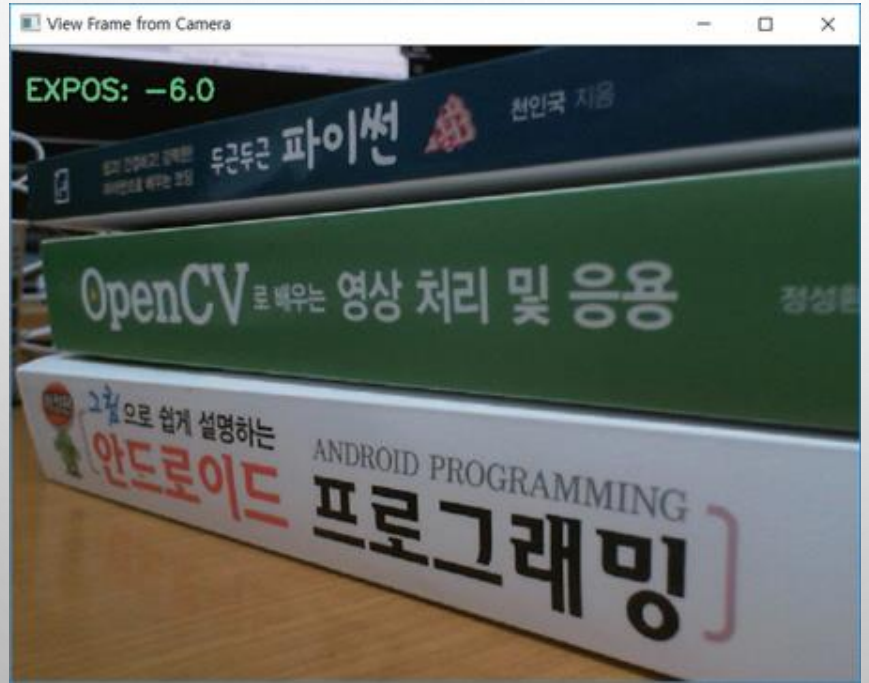
4.5.1 카메라에서 프레임 읽기

```
14  ## 카메라 속성 획득 및 출력
15  print("너비 %d" % capture.get(cv2.CAP_PROP_FRAME_WIDTH))
16  print("높이 %d" % capture.get(cv2.CAP_PROP_FRAME_HEIGHT))
17  print("노출 %d" % capture.get(cv2.CAP_PROP_EXPOSURE))
18  print("밝기 %d" % capture.get(cv2.CAP_PROP_BRIGHTNESS))
19
20  while True                                          # 무한 반복
21      ret, frame = capture.read()                     # 카메라 영상 받기
22      if not ret: break
23      if cv2.waitKey(30) >= 0: break                  # 종료 조건 - 스페이스바 키
24
25      exposure = capture.get(cv2.CAP_PROP_EXPOSURE)   # 노출 속성 획득
26      put_string(frame, 'EXPOS: ', (10, 40), exposure)
27      title = "View Frame from Camera"
28      cv2.imshow(title, frame)                        # 윈도우에 영상 띄우기
29  capture.release()
```

4.5.1 카메라에서 프레임 읽기

◆ 실행결과

```
Run: 17.read_pccamera
C:\Python\python.exe D:/source/chap04/17.read_pccamera.py
너비 640
높이 480
노출 -6
밝기 143
```



4.5.2 카메라 속성 설정하기

◆예제

심화예제 4.5.2 카메라 속성 설정 – 18.set_camera_attr.py

```
01  import cv2
02  from Common.utils import put_string                    # 함수 재사용 위한 임포트
03
04  def zoom_bar(value):                                   # 줌 조절 콜백 함수
05      global capture
06      capture.set(cv2.CAP_PROP_ZOOM, value)              # 줌 설정
07
08  def focus_bar(value):                                  # 초점조절 콜백 함수
09      global capture
10      capture.set(cv2.CAP_PROP_FOCUS, value)
11
12  capture = cv2.VideoCapture(0)                          # 0번 카메라 연결
13  if capture.isOpened() == False: raise Exception("카메라 연결 안됨")        # 예외처리
15  capture.set(cv2.CAP_PROP_FRAME_WIDTH, 400)             # 카메라 프레임 너비
16  capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 300)            # 카메라 프레임 높이
17  capture.set(cv2.CAP_PROP_AUTOFOCUS, 0)                 # 자동초점 중지
18  capture.set(cv2.CAP_PROP_BRIGHTNESS, 100)              # 프레임 밝기 초기화
```

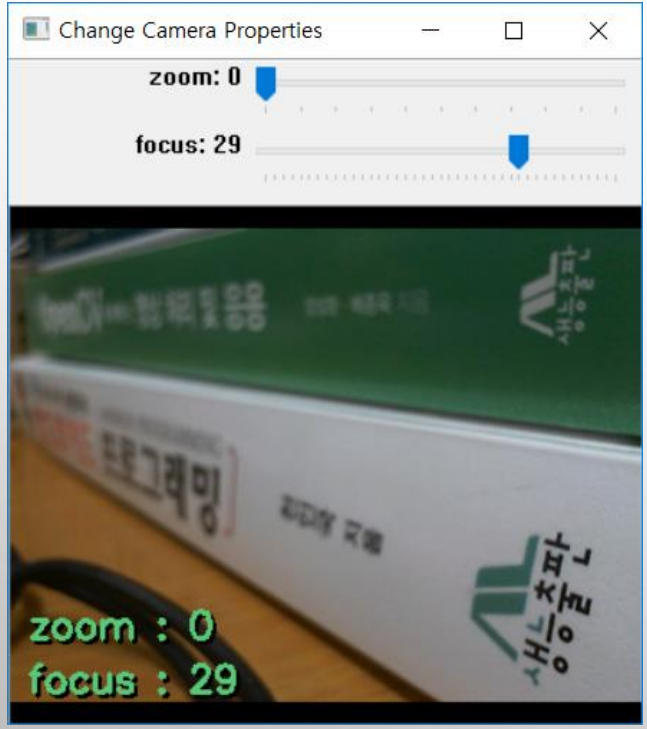
4.5.2 카메라 속성 설정하기

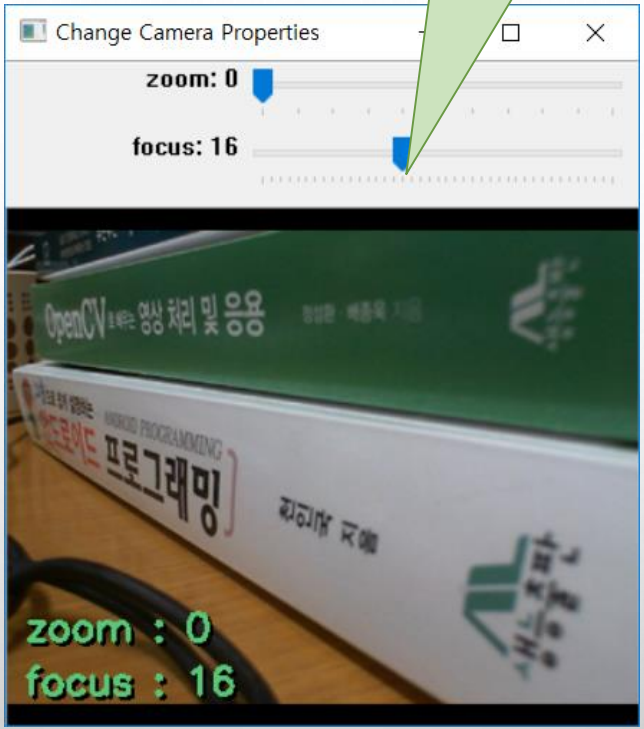
◆예제

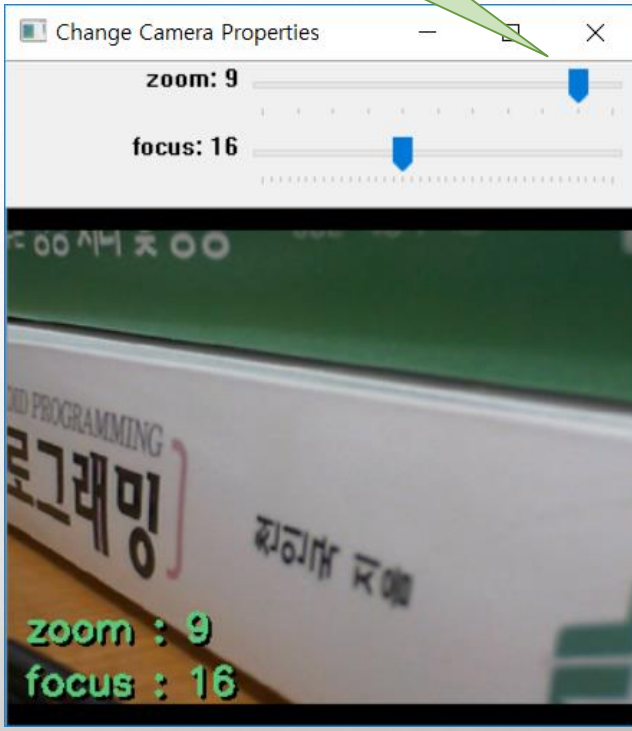
```
20  title = "Change Camera Properties"                    # 윈도우 이름 지정
21  cv2.namedWindow(title)                                # 윈도우 생성
22  cv2.createTrackbar('zoom', title, 0, 10, zoom_bar)    # 줌 트랙바
23  cv2.createTrackbar('focus', title, 0, 40, focus_bar)  # 포커스 트랙바

25  while True:
26      ret, frame = capture.read()                       # 카메라 영상 받기
27      if not ret: break
28      if cv2.waitKey(30) >= 0: break
29
30      zoom = int(capture.get(cv2.CAP_PROP_ZOOM))        # 카메라 속성 가져오기
31      focus = int(capture.get(cv2.CAP_PROP_FOCUS))
32      put_string(frame, 'zoom : ', (10, 240), zoom)     # 줌 값 표시
33      put_string(frame, 'focus : ', (10, 270), focus)   # 초점 값 표시
34      cv2.imshow(title, frame)
35
36  capture.release()                                     # 비디오 캡쳐 메모리 해제
```

4.5.2 카메라 속성 설정하기

◆실행결과







4.5.3 카메라 프레임을 동영상파일로 저장

◆예제 - 카메라 영상 실시간 저장 방법

예제 4.5.3 카메라 프레임을 동영상파일로 저장 – 19.write_camera

```
import cv2

capture = cv2.VideoCapture(0)                          # 0번
if capture.isOpened() == False: raise Exception("카메라 연결 안됨")

fps = 29.97                                            # 초당 프레임 수
delay = round(1000/fps)                                # 프레임 간 지연 시간
size = (640, 360)                                      # 동영상파일 해상도
fourcc = cv2.VideoWriter_fourcc(*'DX50')               # 압축 코덱 설정

## 카메라 속성 실행창에 출력
print("width × height: ", size )
print("VideoWriterfourcc: %s" % fourcc)
print("delay: %2d ms" % delay)
print("fps: %.2f" % fps)
```

```
capture.set(cv2.CAP_PROP_ZOOM, 1)
capture.set(cv2.CAP_PROP_FOCUS, 0)
capture.set(cv2.CAP_PROP_FRAME_WIDTH, size[0]) #
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, size[1])
```

4.5.3 카메라 프레임을 동영상파일로 저장

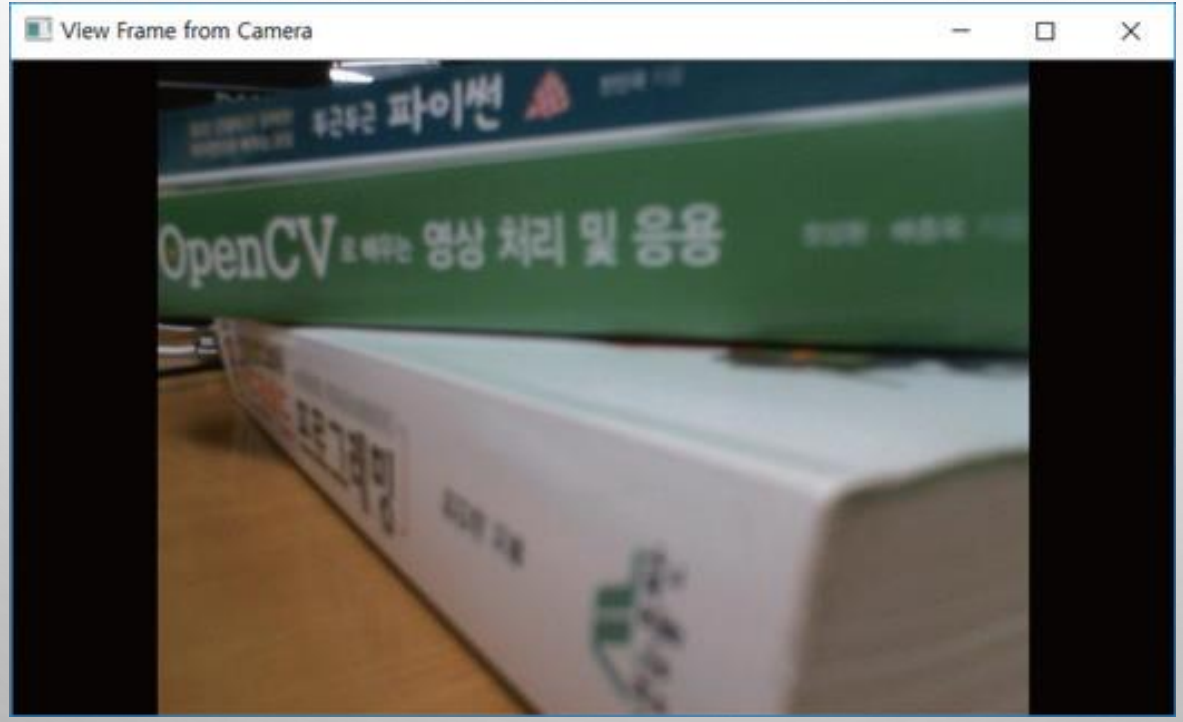
◆예제 - 카메라 영상 실시간 저장 방법

```
22  ## 동영상파일 개방 및 코덱, 해상도 설정
23  writer = cv2.VideoWriter("images/video_file.avi", fourcc, fps, size)
24  if writer.isOpened() == False: raise Exception("동영상파일 개방 안됨")
25
26  while True:
27      ret, frame = capture.read()              # 카메라 영상 받기
28      if not ret: break
29      if cv2.waitKey(delay) >= 0: break
30
31      writer.write(frame)                      # 프레임을 동영상으로 저장
32      cv2.imshow("View Frame from Camera", frame)
33
34  writer.release()
35  capture.release()
```

4.5.3 카메라 프레임을 동영상파일로 저장

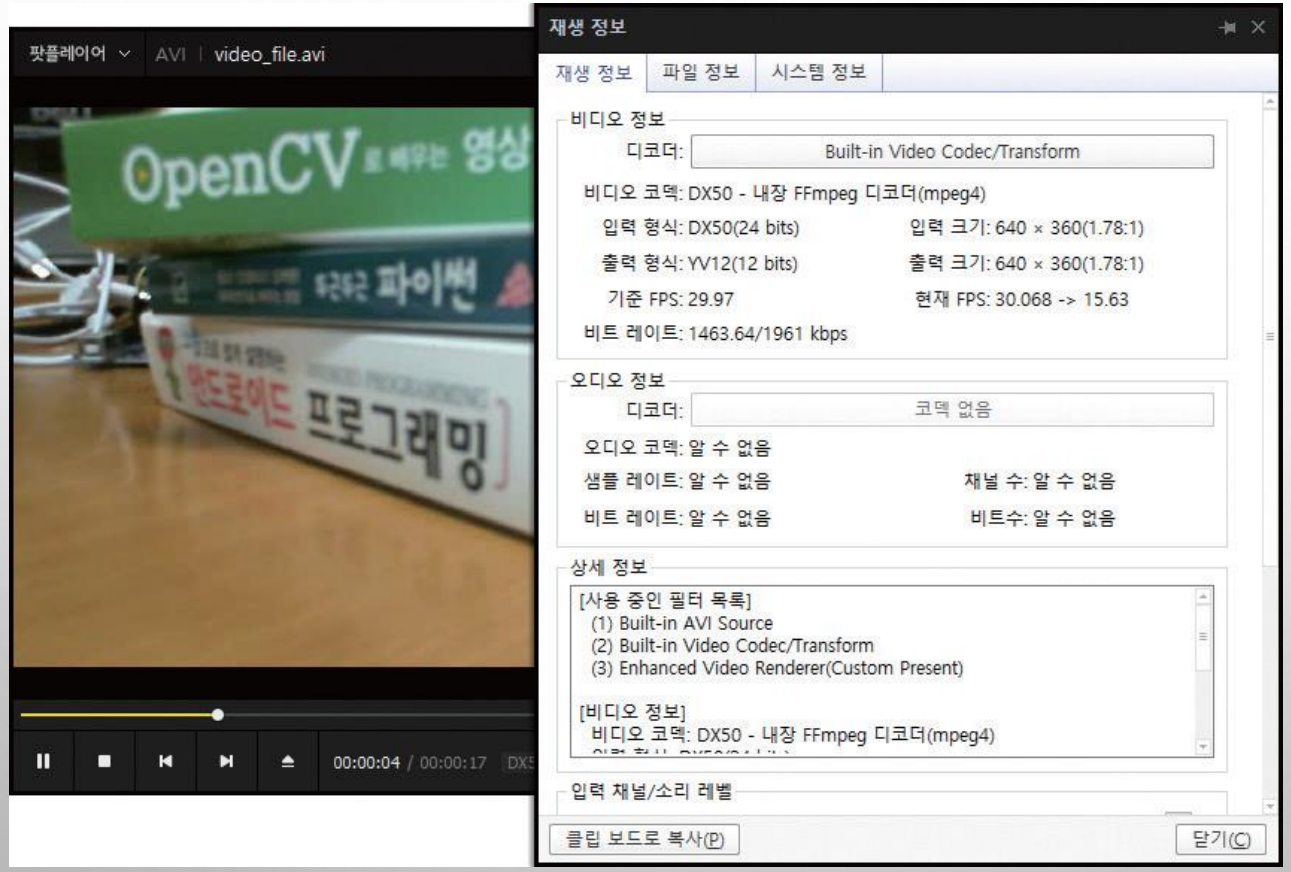
◆실행결과

```
Run: 19.write_camera_frame
C:\Python\python.exe D:/source/chap04/19.write_camera_frame.py
프레임 해상도: (640, 360)
압축코덱 숫자: 808802372
delay: 33 ms
fps: 29.97
```



4.5.3 카메라 프레임을 동영상파일로 저장

◆동영상 재생기(POTPLAYER)에서 동영상 정보 확인



4.5.4 동영상파일 읽기

◆예제 – 동영상 파일 읽어 프레임별 영상처리 하기

〈표 4.5.3〉 프레임 번호에 따른 영상처리 예시

프레임 번호	영상처리
1 ~ 99	아무런 영상처리를 적용하지 않음
100 ~ 199	프레임별 화소의 파란색 성분에 100을 더해서 영상을 더 푸르게 만듦
200 ~ 299	프레임별 화소의 녹색 성분에 100을 더해서 영상을 더 녹색으로 만듦
300 ~ 399	프레임별 화소의 빨간색 성분에 100을 더해서 영상을 더 빨갛게 만듦

예제 4.5.4 동영상파일 읽기 – 20.read_video_file.py

```
01  import cv2
02  from Common.utils import put_string                    # 글쓰기 함수 임포트
03
04  capture = cv2.VideoCapture("images/video_file.avi")    # 동영상파일 개방
05  if not capture.isOpened(): raise Exception("동영상파일 개방 안됨")          # 예외 처리
06
07  frame_rate = capture.get(cv2.CAP_PROP_FPS)             # 초당 프레임 수
08  delay = int(1000 / frame_rate)                         # 지연 시간
09  frame_cnt = 0                                          # 현재 프레임 번호
```

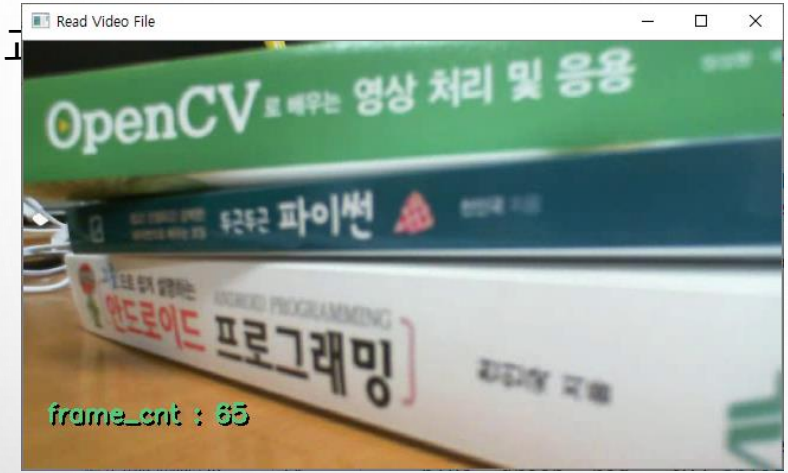
4.5.4 동영상파일 읽기

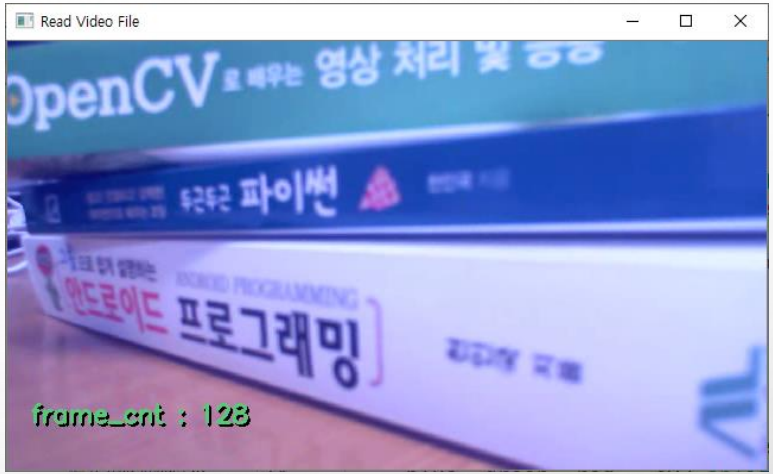
◆예제 – 동영상 파일 읽어 프레임별 영상처리 하기

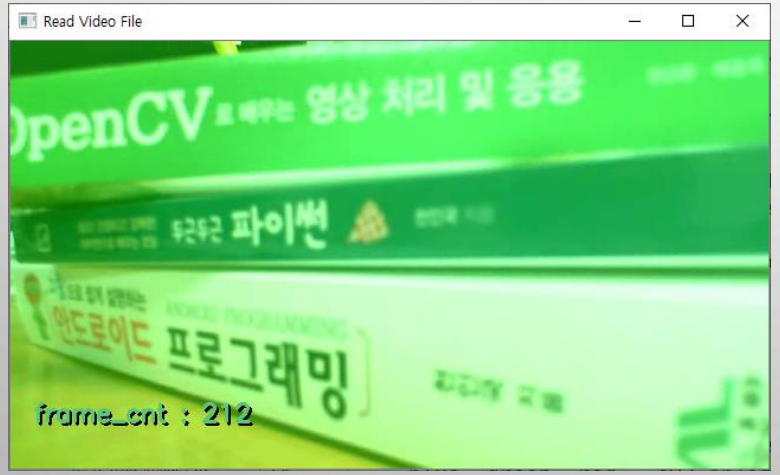
```
while True:
    ret, frame = capture.read()
    if not ret or cv2.waitKey(delay) >= 0: break      # 프레임 간 지연 시간 지정
    blue, green, red = cv2.split(frame)               # 컬러 영상 채널 분리
    frame_cnt += 1

    if 100 <= frame_cnt < 200: cv2.add(blue, 100, blue)        # blue 채널 밝기 증가
    elif 200 <= frame_cnt < 300: cv2.add(green, 100, green)    # green 채널 밝기 증가
    elif 300 <= frame_cnt < 400: cv2.add(red  , 100, red)      # red 채널 밝기 증가

    frame = cv2.merge( [blue, green, red] )                    # 단일채널 영상 합성
    put_string(frame, 'frame_cnt: ', (20, 30), frame_cnt)
    cv2.imshow("Read Video File", frame)
capture.release()
```

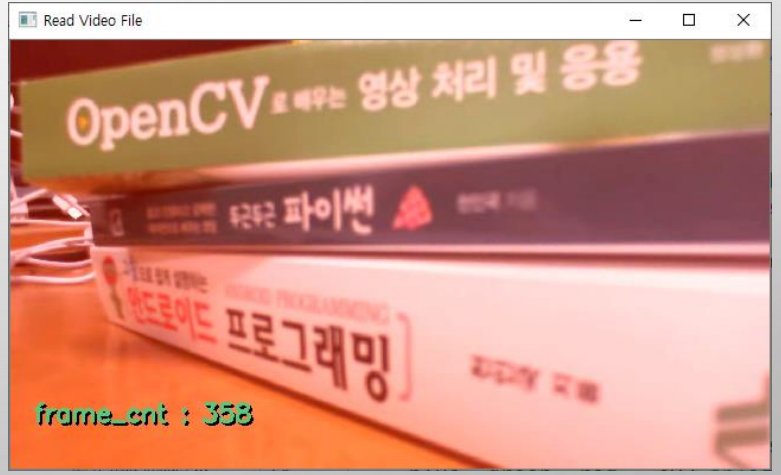
4.5.4 동영상파일 읽기

◆실행결과









4.6 Matplotlib 패키지 활용

◆MATPLOTLIB 라이브러리

❖ 파이썬에서 데이터를 차트나 그래프로 그려주는 라이브러리

◆PYPLOT 모듈

❖ MATPLOTLIB 라이브러리의 하나의 모듈

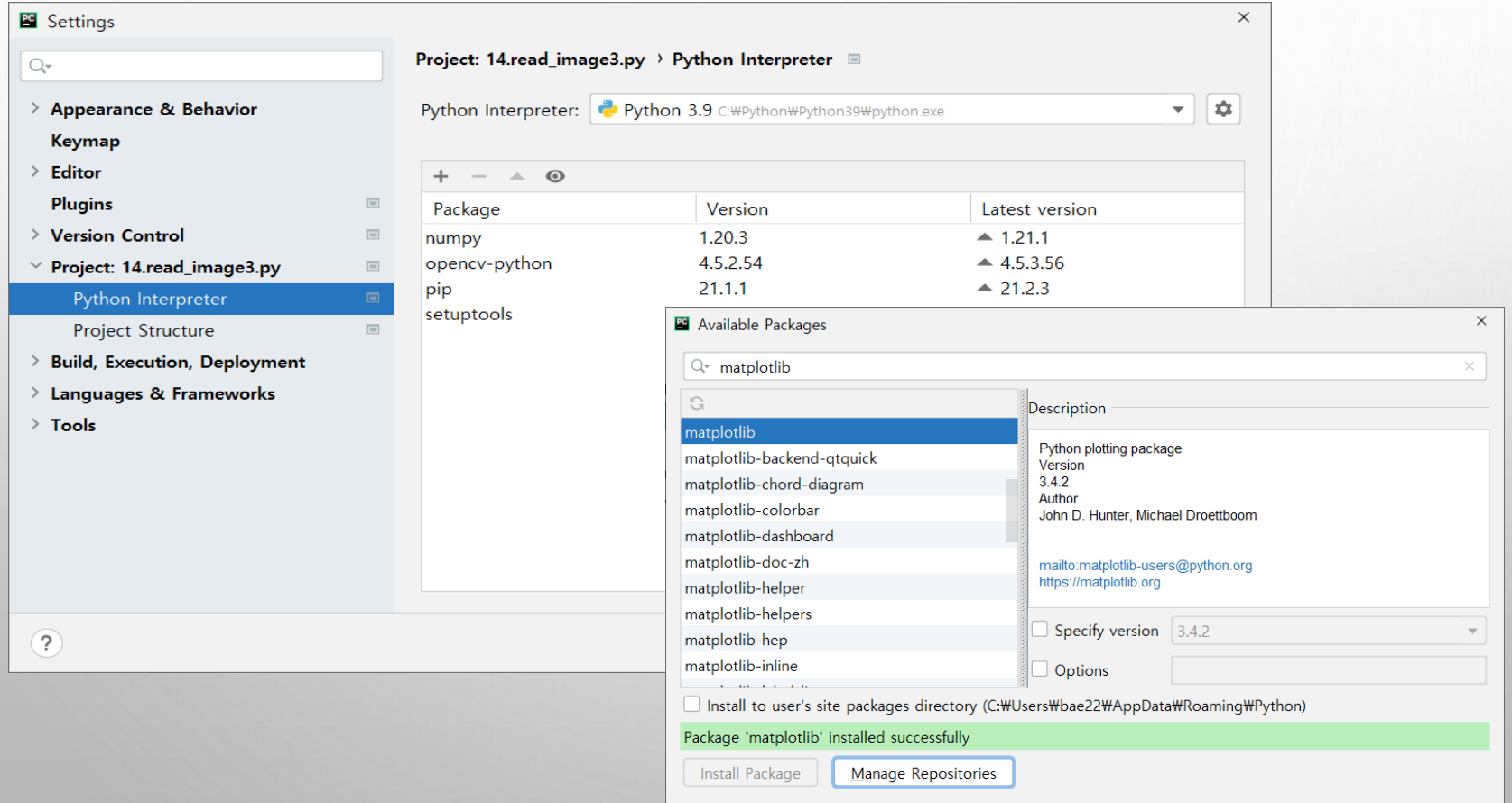
❖ 매트랩의 수치해석 소프트웨어의 시각화 명령을 거의 그대로 사용가능한 명령어 집합 제공

❖ 주피터 노트북이나 구글의 COLAB 에서 간단히 결과 확인할 때

- OPENCV의 CV2.IMSHOW() 함수를 사용하지 않고 MATPLOTLIB 패키지를 사용하는 것이 편리

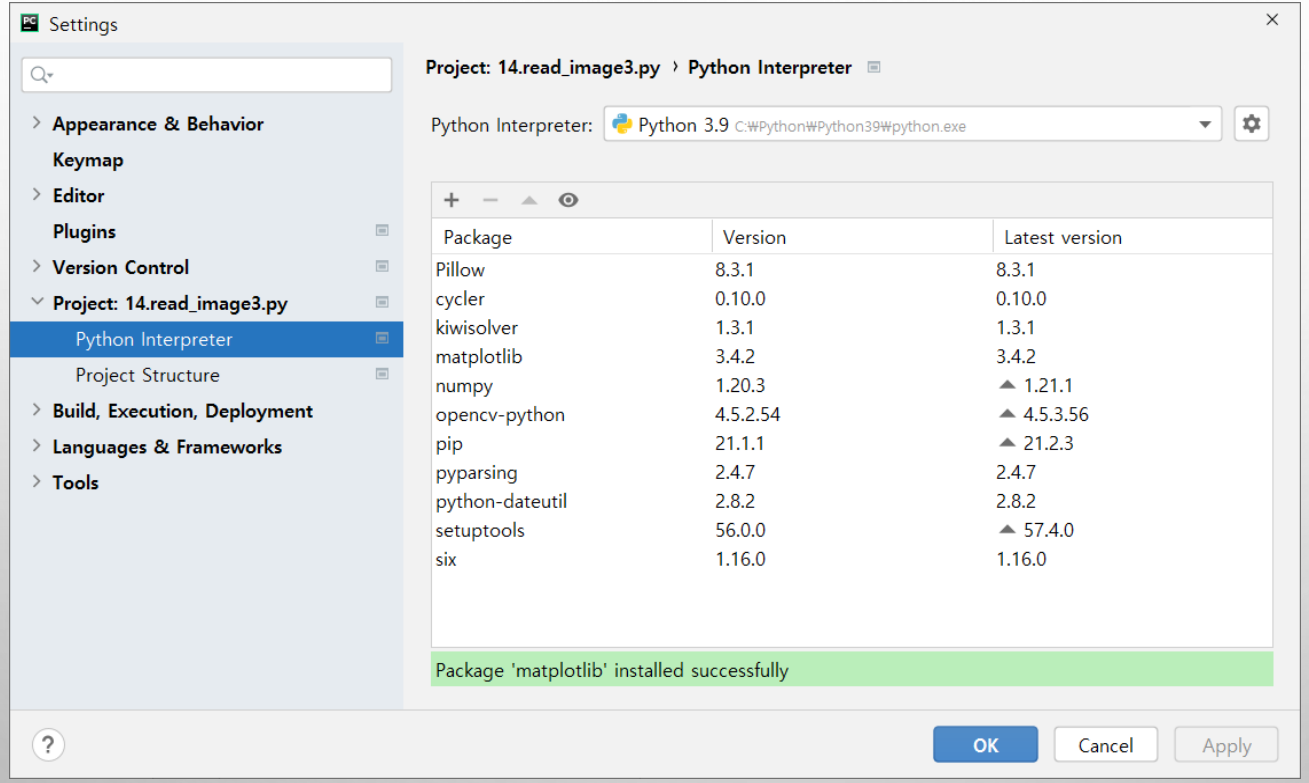
4.6 Matplotlib 패키지 활용

◆설치 : SETTINGS 윈도우 [CTRL+ALT+S]



4.6 Matplotlib 패키지 활용

◆설치 : SETTINGS 윈도우 [CTRL+ALT+S]



4.6 Matplotlib 패키지 활용

◆모듈내 메서드 설명

함수 설명		
pyplot.figure(num=None, figsize=None, dip=None) → matplotlib.figure.Figure ■ 설명: 그림(Figure) 객체를 생성해서 플롯(plot)을 그릴 수 있게 한다.		
인수 설명	■ num	그림 이름, 정수 or 문자열(지정 안하면 자동 증가 숫자)
	■ figsize	그림 크기(가로, 세로), 인치 단위(실수형)
	■ dpi	그림 해상도(정수형)
pyplot.imshow(X, cmap=None, norm=None, aspect=None, interpolation=None, alpha=None) → (matplotlib.image.AxesImage) ■ 설명: 데이터 X를 그래프에 영상으로 그려준다.		
인수 설명	■ X	그리려는 데이터(ndarray 객체), 2차원 혹은 3차원
	■ cmap	컬러 맵, 정해진 컬러 조합에 따라서 색상 매칭
	■ aspect	그리지는 영상의 종횡비
	■ interpolation	그리지는 영상의 보간 방법(예: 'nearest', 'bilinear', 'bicubic')
	■ alpha	투명도(0: 완전 투명, 1: 완전 불투명)

4.6 Matplotlib 패키지 활용

◆모듈내 메서드 설명

pyplot.plot(*arg, scalex=None, scaley=None, **kwargs) → [matplotlib.lines.Line2D]

■ 설명: x에 대해서 y값으로 그래프(선, 마크)를 그려준다.

인수 설명

■ *arg [x], y, [fmt] 순으로 인수 작성, []는 생략 가능 표시

■ fmt 선을 나타내는 문자열('[color][marker][line]')

color(색)		marker		line		line	
'b'	파란색	'.'	점	's'	사각형	'-'	실선
'g'	초록색	','	픽셀	'p'	오각형	'--'	파선
'r'	빨간색	'o'	원	'h'	육각형	'-.'	파선 점선
'c'	청록색	'v'	삼각형1	'*'	별모양	':'	점선
'm'	자주색	'^'	삼각형2	'+'	더하기		
'y'	노란색	'>'	삼각형3	'x'	x모양		
'k'	검정색	'1'	삼각뿔1	'd'	다이아몬드		
'w'	흰색	'2'	삼각뿔2	'-'	수평라인		

4.6 Matplotlib 패키지 활용

◆모듈내 메서드 설명

pyplot.subplot(*args, **kwargs) → matplotlib.axes._subplots.AxesSubplot ■ 설명: 현재 그림에 서브 플롯을 추가한다.	
인수 설명	■ *args 행, 열, 순번 지정 – nrows, ncols, index 콤머 분리숫자, 예시〉 3, 3, 3 – pos 세 자리 숫자, 예시〉 333 – axes AxesSubplot 객체
pyplot.subplots(nrows=1, ncols=1, **fig_kw) → tuple(Figure, ndarray[AxesSubplot , ..]) ■ 설명: 그림 객체와 서브 플롯들(ndarray)을 생성한다.	
인수 설명	■ nrows 서브 플롯 격자의 행수 ■ ncols=1, 서브 플롯 격자의 열수 ■ **fig_kw 추가 키워드 인수 – figsize 등 지정
pyplot.axis(*args, **kwargs) → tuple([축범위], matplotlib.text.Text) ■ 설명: 플롯의 축 정보 조정한다.	
	■ *args [xmin, xmax, ymin, ymax]를 리스트로 작성하여 지정 ■ xmin, xmax, ymin, ymax x, y축 범위 ■ **kwargs option 지정

4.6 Matplotlib 패키지 활용

◆ 모듈내 메서드 설명

pyplot.title(label, fontdict=None, loc='center') → matplotlib.text.Text ■ 설명: 서브 플롯의 제목을 지정한다.		
인수 설명	■ label	그림 제목
	■ fontdict	라벨의 모양 편집, 사전 자료형
	■ loc	제목의 위치 ('center', 'left', 'right')
	■ pad	그림과의 간격
pyplot.tight_layout(pad=1.08, h_pad=None, w_pad=None) → None ■ 설명: 자동으로 명시된 여백(1.08)으로 하위 그래프 파라미터를 조정한다.		
인수 설명	■ pad	그림 여백 설정, float
	■ h_pad	서브 그림의 상하 여백, float
	■ w_pad	서브 그림의 좌우 여백, float
pyplot.show() → None ■ 설명: 그림 객체를 윈도우에 띄운다		
pyplot.savefig(*args, **kwargs) ■ 설명: 그림 객체를 영상파일로 저장한다.		
	■ *args	파일명(fname) 지정
	■ **kwargs	추가 키워드 인수

4.6 Matplotlib 패키지 활용

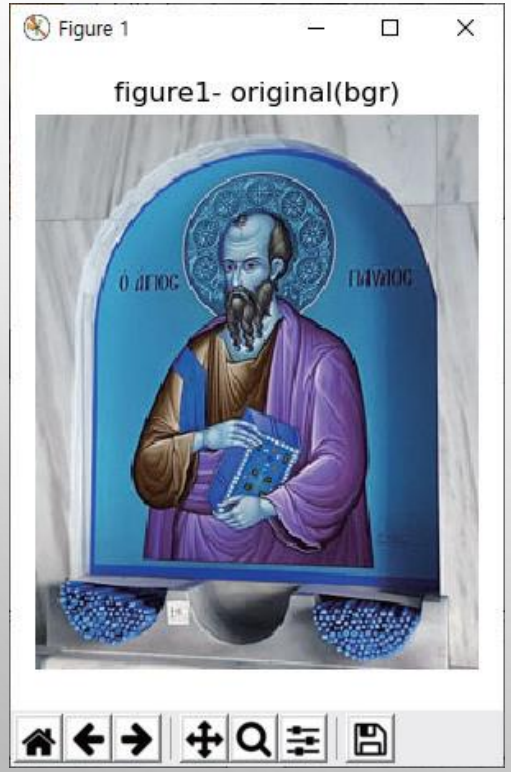
◆예제

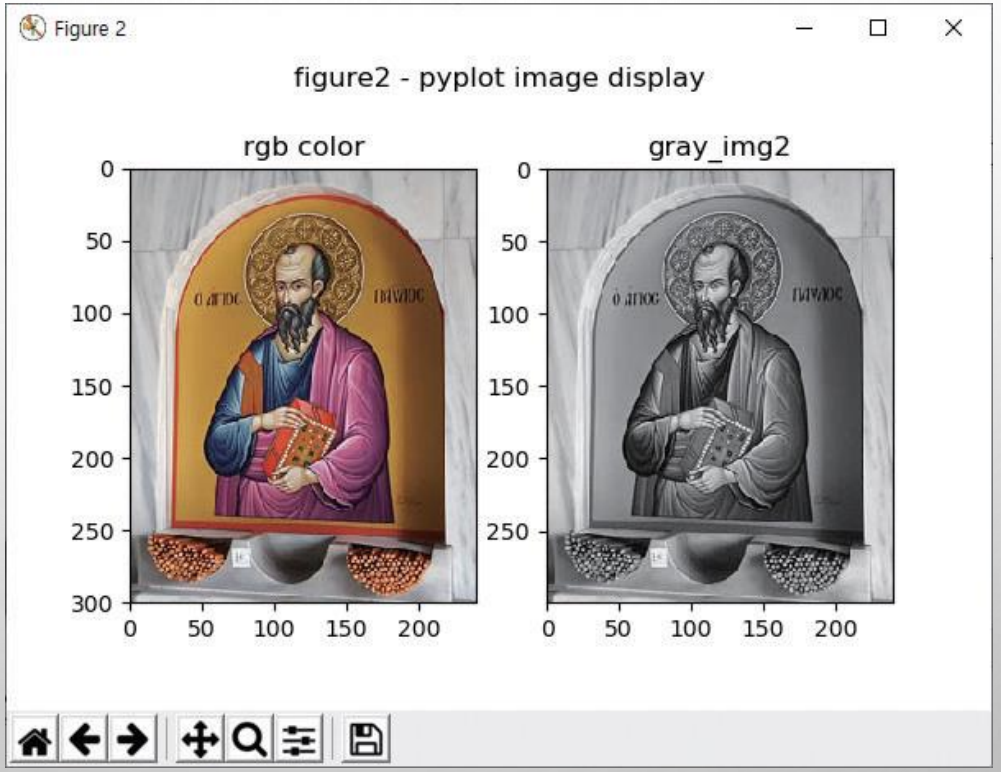
예제 4.6.1 Matplotlib 라이브러리로 영상 표시 – 21.matplotlib.py

```
01  import cv2
02  import matplotlib.pyplot as plt                                  # pyplot 모듈 임포트
03
04  image  = cv2.imread("images/matplot.jpg", cv2.IMREAD_COLOR)      # 영상 읽기
05  if image is None: raise Exception("영상파일 읽기 에러")            # 예외처리
06
07  rows, cols = image.shape[:2]                                     # 영상 크기 정보
08  rgb_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)        # 컬러 공간변환
09  gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)      # 명암도 영상 변환

11  fig = plt.figure(num=1, figsize=(3,4))                  # 그림 생성
12  plt.imshow(image), plt.title('figure1- original(bgr)')  # 그림표시 및 제목
13  plt.axis('off'), plt.tight_layout()                     # 축 없음, 여백 없음
14
15  fig = plt.figure(figsize=(6,4))                         # 그림 생성
16  plt.suptitle('figure2- pyplot image display')           # 전체 제목 지정
17  plt.subplot(1,2,1), plt.imshow(rgb_img)                 # 서브 플롯 그림
18  plt.axis([0,cols, rows,0]), plt.title('rgb color')      # 축 범위, 서브 플롯 제목
19  plt.subplot(1,2,2), plt.imshow(gray_img, cmap='gray')   # 서브 플롯 그림, 명암도로 표시
20  plt.title('gray_img2')
21  plt.show()                                              # 전체 그림 띄우기
```

4.6 Matplotlib 패키지 활용

◆실행결과





4.6 Matplotlib 패키지 활용

◆여러 서브 플롯을 나타내는 다른 방법

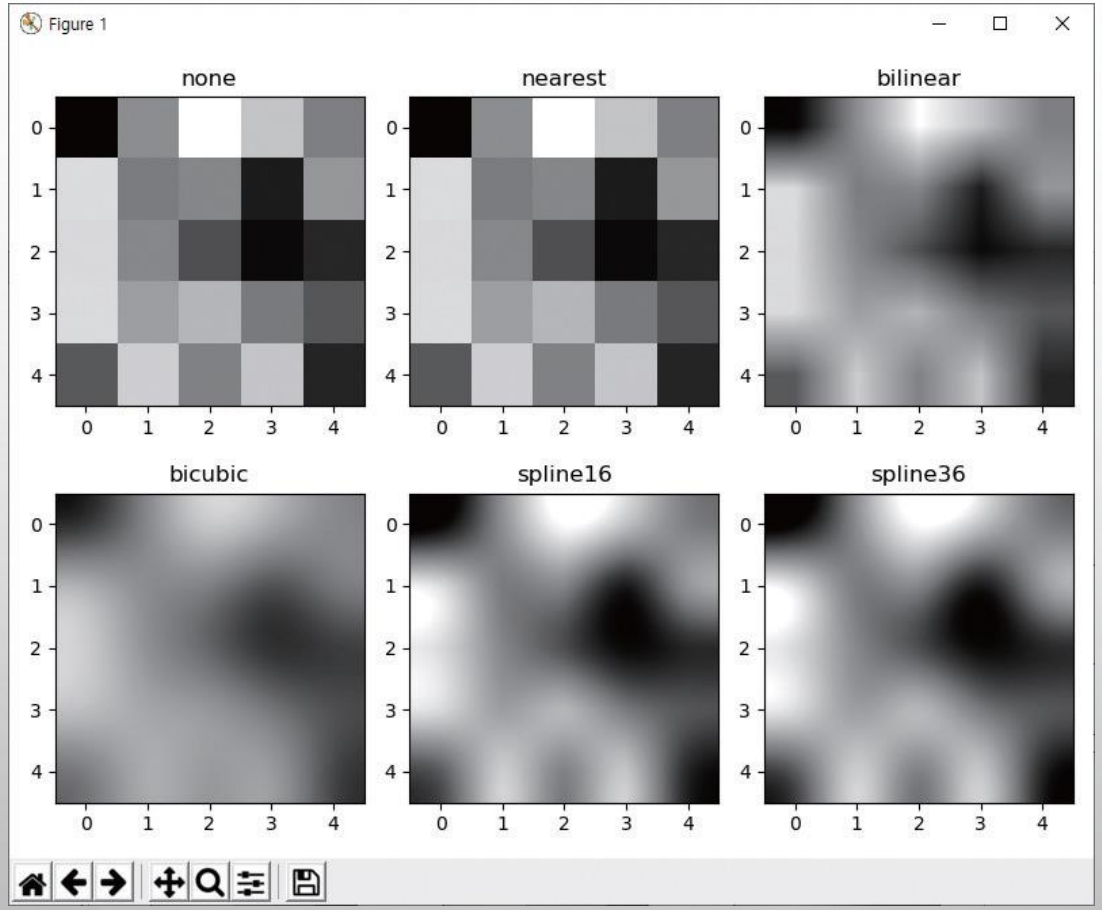
예제 4.6.2 서브플롯 그리기 및 보간 적용 – 22.interpolation.py

```
01  import matplotlib.pyplot as plt
02  import numpy as np
03
04  methods = ['none', 'nearest', 'bilinear', 'bicubic', 'spline16', 'spline36' ] # 보간 방법
05  grid = np.random.rand(5, 5)                                          # 0~1사이 임의 난수 생성
06
07  fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(8, 6))            # 서브 플롯들 생성
08
09  for ax, method in zip(axs.flat, methods):                            # 서브 플롯 순회
10      ax.imshow(grid, interpolation=method, cmap='gray')               # 명암도 영상 표시
11      ax.set_title(method)                                             # 서브 플롯 제목
12  plt.tight_layout(), plt.show()                                       # 여백 없음, 윈도우 띄움
```

여백 제거

4.6 Matplotlib 패키지 활용

◆실행결과



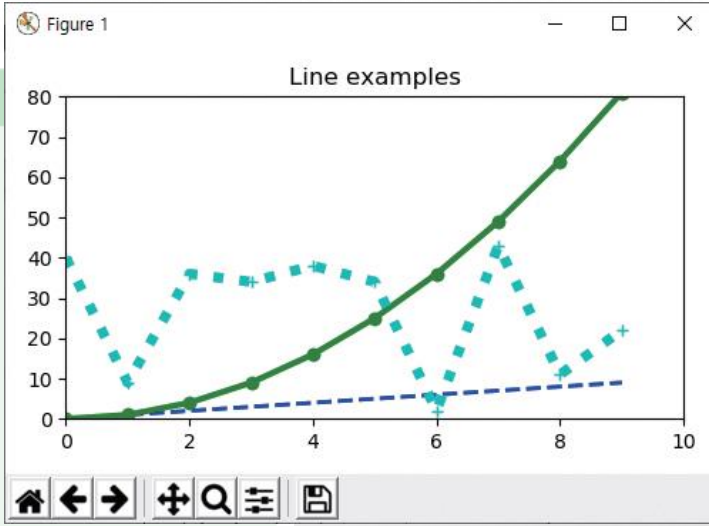
4.6 Matplotlib 패키지 활용

예제 4.6.3 그래프 그리기 – 23.plot.py

```
01  import matplotlib.pyplot as plt
02  import numpy as np
03
04  x = np.arange(10)                                  # x축 데이터
05  y1 = np.arange(10)                                 # y축 데이터
06  y2 = np.arange(10)**2
07  y3 = np.random.choice(50, size=10)
08
09  plt.figure(figsize=(5,3))                          # 그림 객체 생성 – 그래프 크기(단위inch)
10  plt.plot(x, y1, 'b--', linewidth=2)                # 선 스타일 지정 – 파란색, 파선
11  plt.plot(x, y2, 'go-', linewidth=3)                # 녹색, 원 마크, 실선
12  plt.plot(x, y3, 'c+:', linewidth=5)                # 청록색, +마크, 점선
13
14  plt.title('Line examples')                         # 그래프 제목
15  plt.axis([0,10, 0,80])                             # 축 범위
16  plt.tight_layout()
17  plt.savefig(fname='sample.png', dpi=300)           # 그림 저장
18  plt.show()                                         # 윈도우 표시
```

선 스타일

선두께



단원 요약

- 콜백 함수
  - 개발자가 함수를 직접 호출하는 것이 아니라, 시스템이 개발자가 등록한 함수를 호출하는 방식
  - cv2.setMouseCallback() 함수와 cv2.createTrackbar() 함수 사용
- 윈도우 제어
  - cv2.namedWindow() – 이름 지정
  - cv2.imshow() 함수 - 지정 윈도우에 행렬을 영상 표시
- cv2.waitKey() 함수
  - 지정된 대기 시간 동안 키보드 키를 입력 받을 수 있는 함수
  - 키 이벤트를 처리하거나 윈도우 창을 바로 닫지 않고 대기
- 도형 그리기 함수
  - cv2.line(), cv2.rectangle() 함수, cv2.circle(), cv2.ellipse()
  - 타원 및 호를 그려주는 cv2.ellipse() 함수
- cv2.imread() 함수
  - 영상파일을 읽어서 영상 데이터인 행렬로 반환
  - 컬러 타입 인수(flag)를 활용해서 명암도 영상, 컬러 영상, 16비트/32비트 영상으로 로드

단원 요약

- 행렬을 영상파일로 저장하기 위해서는 cv2.imwrite() 함수 이용
  - 확장자로 저장할 파일의 포맷을 결정하며, 추가 인수인 params로 압축을 통한 화질 지정
- cv2.VideoCapture 클래스
  - 동영상파일 or PC캠의 연속 프레임을 처리
  - cv2.VideoCapture.open() 메서드 – 동영상파일을 개방
  - cv2.VideoCapture.read() 메서드 - 동영상 시퀀스에서 하나의 프레임을 행렬로 가져옴
  - cv2.VideoCapture.get() / cv2.VideoCapture.set() 메서드 - 카메라 속성 정보 확인, 변경
- cv2.VideoWriter 클래스
  - 연속적인 행렬 데이터를 동영상파일로 저장
  - cv2.VideoWriter.open() 메서드 - 저장할 동영상파일을 개방
  - cv2.VideoWriter().write() 메서드로 연속적인 행렬을 저장
- Matplotlib 라이브러리의 pyplot 모듈
  - ndarray 객체의 데이터 시각화 ,
  - imshow() 함수는 2차원 데이터를 영상으로 표시
  - plot() 함수는 1차원 데이터를 그래프로 표시
  - subplot() 와 subplots() 함수 - 여러 개의 서브 플롯을 하나의 그림 객체에 표시