디지털 영상처리

OpenCV 인터페이스

학습목표

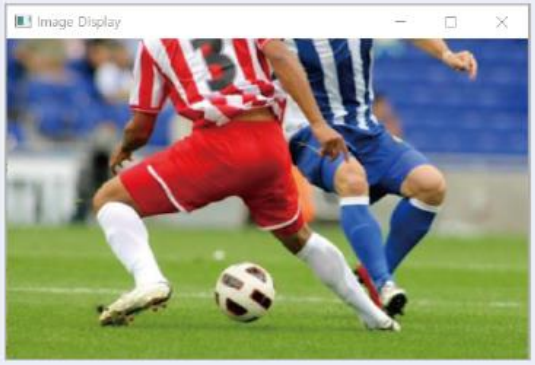
- OpenCV 프로그래밍 소개
- 윈도우 제어
- 이벤트 처리 함수
- 그리기 함수
- 영상파일 처리
- 비디오 처리
- Matplotlib 패키지 활용

영상을 읽고 표시하기

■ 처음 해보는 OpenCV 프로그래밍

영상 파일을 읽고 윈도우에 디스플레이하기

```
01  import cv2 as cv
02  import sys
03
04  img=cv.imread('soccer.jpg')      # 영상 읽기
05
06  if img is None:
07      sys.exit('파일을 찾을 수 없습니다.')
08
09  cv.imshow('Image Display',img)   # 윈도우에 영상 표시
10
11  cv.waitKey()
12  cv.destroyAllWindows()
```



OpenCV에서 영상은 numpy.ndarray 클래스 형의 객체

- **numpy는 다차원 배열을 위한 사실상 표준 모듈**
  - 이런 이유로 OpenCV는 영상을 numpy.ndarray로 표현
  - OpenCV가 다루는 영상은 numpy가 제공하는 다양한 기능(함수)을 사용할 수 있음

OpenCV에서 영상은 numpy.ndarray 클래스 형의 객체

■ numpy는 다차원 배열을 위한 사실상 표준 모듈

- 이런 이유로 OpenCV는 영상을 numpy.ndarray로 표현
- OpenCV가 다루는 영상은 numpy가 제공하는 다양한 기능(함수)을 사용할 수 있음

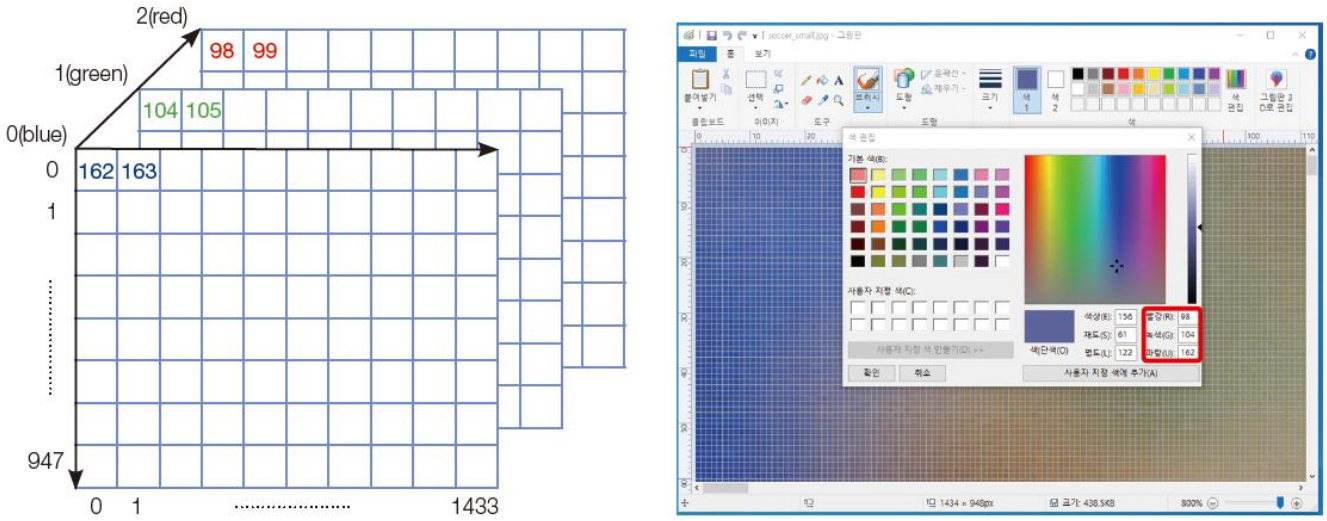
```
In [1]: type(img)
   numpy.ndarray
In [2]: img.shape
   (948,1434,3)
```

OpenCV에서 영상은 numpy.ndarray 클래스 형의 객체

■ 영상의 표현

- 화소의 위치 (r, c) 또는 (y, x)
- 화솟값 조사

```
In [3]: print(img[0,0,0], img[0,0,1], img[0,0,2])   # (0,0) 화소 조사
   162 104 98
In [4]: print(img[0,1,0], img[0,1,1], img[0,1,2])   # (0,1) 화소 조사
   163 105 99
```



(a) 프로그램으로 조사

(b) 그림판으로 조사

그림 2-9 img 객체가 표현하는 영상의 구조와 내용

[프로그래밍 예제3] 웹 캠에서 비디오 읽기

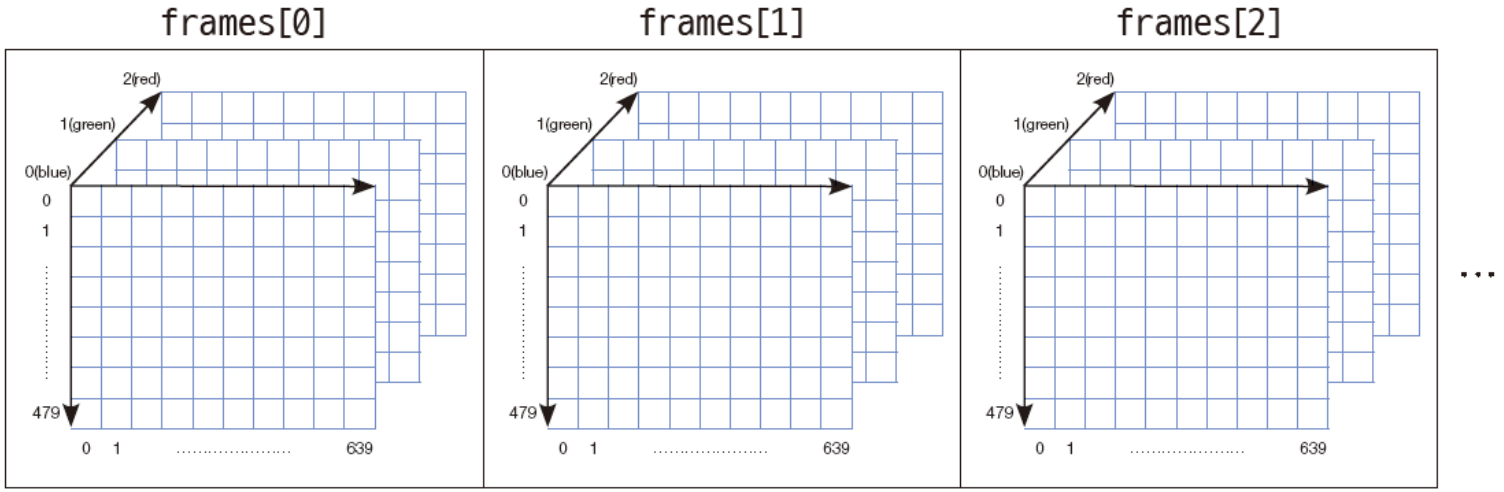
■ 웹 캠에서 비디오 읽기

웹 캠으로 비디오 획득하기

```
01  import cv2 as cv
02  import sys
03
04  cap=cv.VideoCapture(0,cv.CAP_DSHOW)   # 카메라와 연결 시도
05
06  if not cap.isOpened():
07      sys.exit('카메라 연결 실패')
08
09  while True:
10      ret,frame=cap.read()              # 비디오를 구성하는 프레임 획득
11
12      if not ret:
13          print('프레임 획득에 실패하여 루프를 나갑니다.')
14          break
15
16      cv.imshow('Video display',frame)
17
18      key=cv.waitKey(1)                 # 1밀리초 동안 키보드 입력 기다림
19      if key==ord('q'):                 # 'q' 키가 들어오면 루프를 빠져나감
20          break
21
22  cap.release()                         # 카메라와 연결을 끊음
23  cv.destroyAllWindows()
```

[프로그래밍 예제3] 웹 캠에서 비디오 읽기

■ [웹 캠 프로그램]의 자료구조



(a) frames 리스트

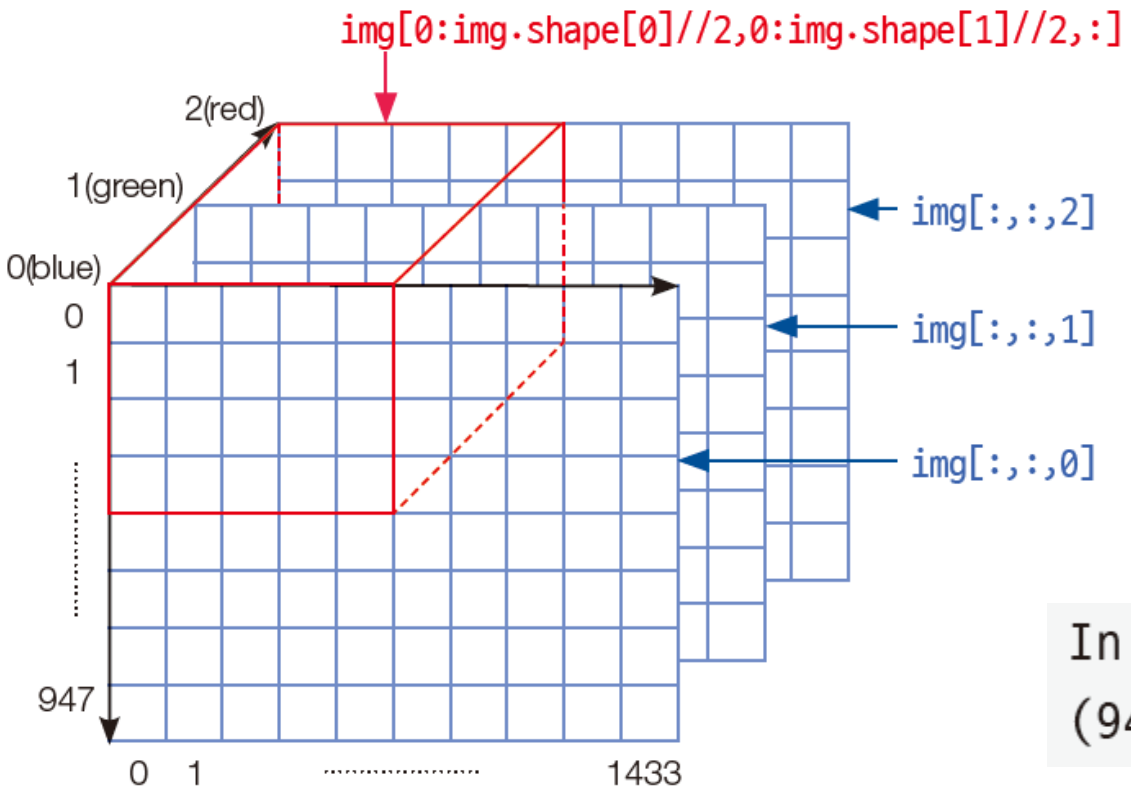
프로그래밍 실습: RGB 채널별로 디스플레이

프로그램 3-1 RGB 컬러 영상을 채널별로 구분해 디스플레이하기

```
01  import cv2 as cv
02  import sys
03
04  img=cv.imread('soccer.jpg')
05
06  if img is None:
07      sys.exit('파일을 찾을 수 없습니다.')
08
09  cv.imshow('original_RGB',img)
10  cv.imshow('Upper left half',img[0:img.shape[0]//2,0:img.shape[1]//2,:])
11  cv.imshow('Center half',img[img.shape[0]//4:3*img.shape[0]//4,img.
              shape[1]//4:3*img.shape[1]//4,:])
12
13  cv.imshow('R channel',img[:,:,2])
14  cv.imshow('G channel',img[:,:,1])
15  cv.imshow('B channel',img[:,:,0])
16
17  cv.waitKey()
18  cv.destroyAllWindows()
```

RGB 채널별로 디스플레이

■ numpy의 슬라이싱 기능을 이용하여 RGB 채널별로 디스플레이




```
In [0]: img.shape
(948,1434,3)
```

그림 3-8 numpy.ndarray의 슬라이싱을 이용한 영상 일부분 자르기([프로그램 3-1]의 10행)

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**

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.destroyAllWindows() → 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

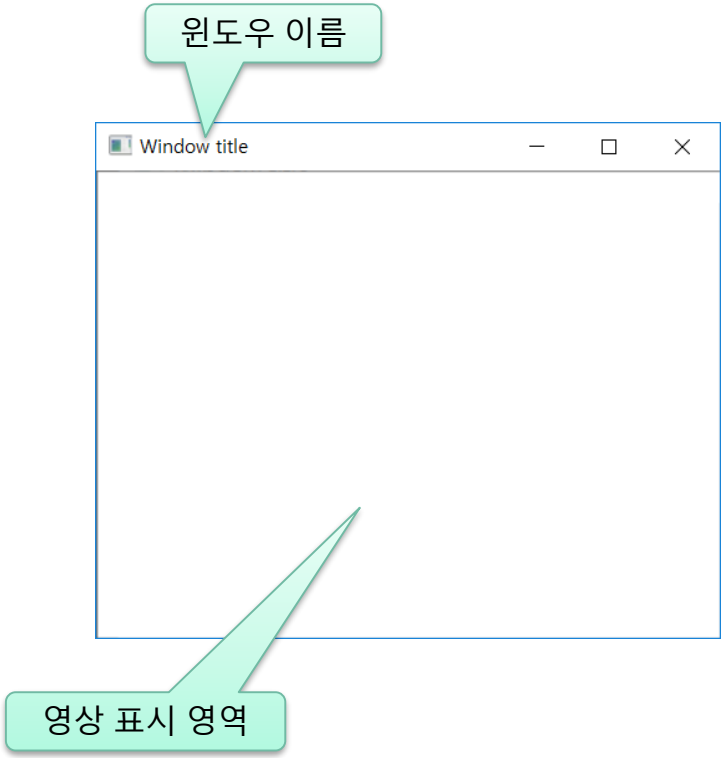
■ x, y 모니터 안에서 이동하려는 위치의 x, y 좌표

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

■ 설명: 윈도우의 크기를 재조정한다.

인수 설명

■ width, height 변경 윈도우의 가로, 세로 크기



윈도우 제어

■ 예제: move_window.py

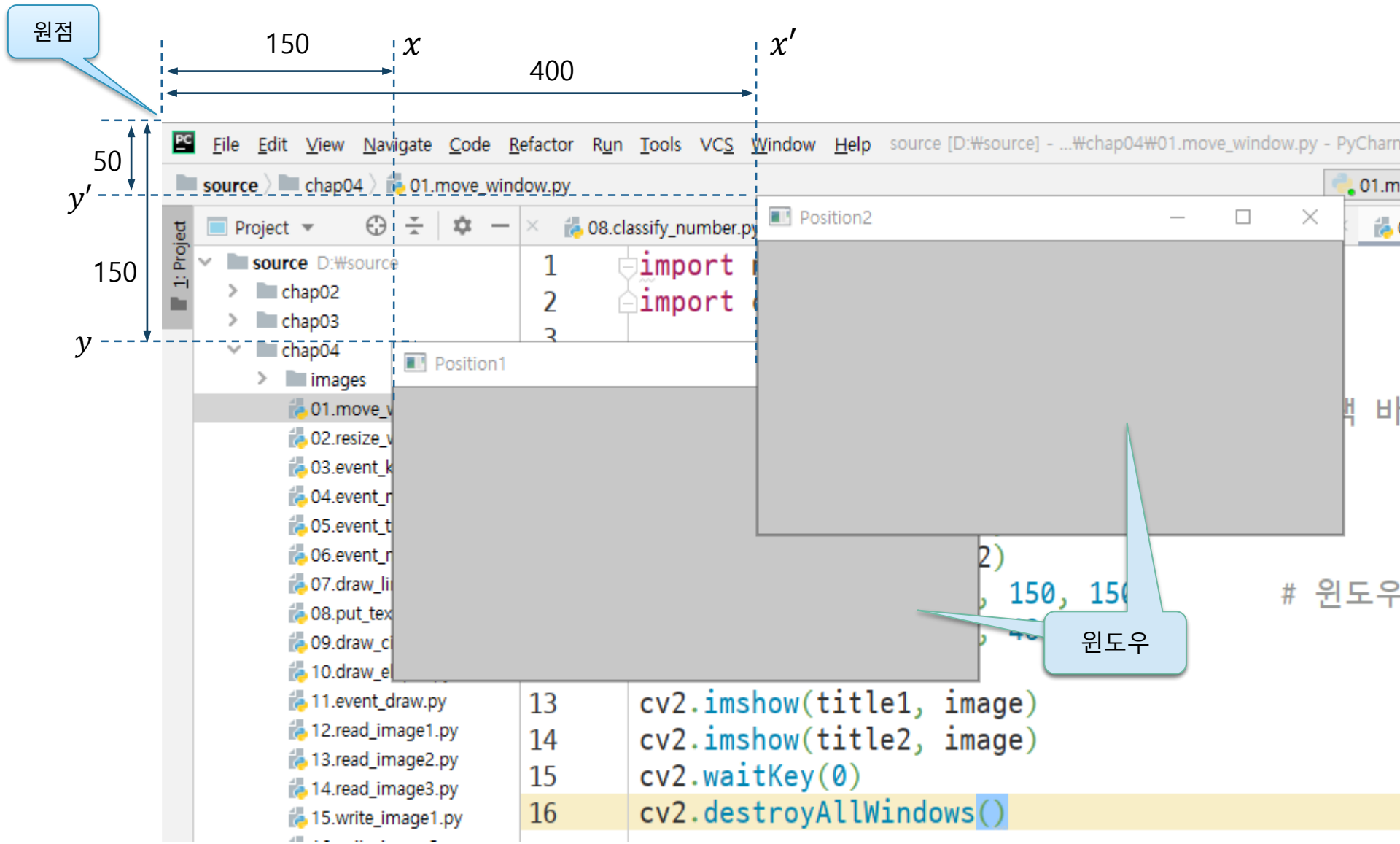
라이브러리 임포트

0 원소 행렬 생성

슬라이스 연산자로 행렬원소값 지정

```
1  import numpy as np
2  import cv2
3
4  image = np.zeros((200,300), np.uint8)
5
6  image[:]=200
7
8  title1, title2 = 'Position1', 'Position2'
9  cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)
10 cv2.moveWindow(title1, 150, 150)
11 cv2.moveWindow(title2, 400, 50)
12
13 cv2.imshow(title1, image)
14 cv2.imshow(title2, image)
15 cv2.waitKey(0)
16 cv2.destroyAllWindows()
```

윈도우 제어


원점
150
x
x′
400
50
y′
150
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
08.classify_number.py
source D:₩source
chap02
chap03
chap04
images
01.move_w
02.resize_w
03.event_k
04.event_m
05.event_t
06.event_m
07.draw_li
08.put_tex
09.draw_ci
10.draw_el
11.event_draw.py
12.read_image1.py
13.read_image2.py
14.read_image3.py
15.write_image1.py
Position1
Position2
윈도우
1 import
2 import
13 cv2.imshow(title1, image)
14 cv2.imshow(title2, image)
15 cv2.waitKey(0)
16 cv2.destroyAllWindows()

윈도우 제어

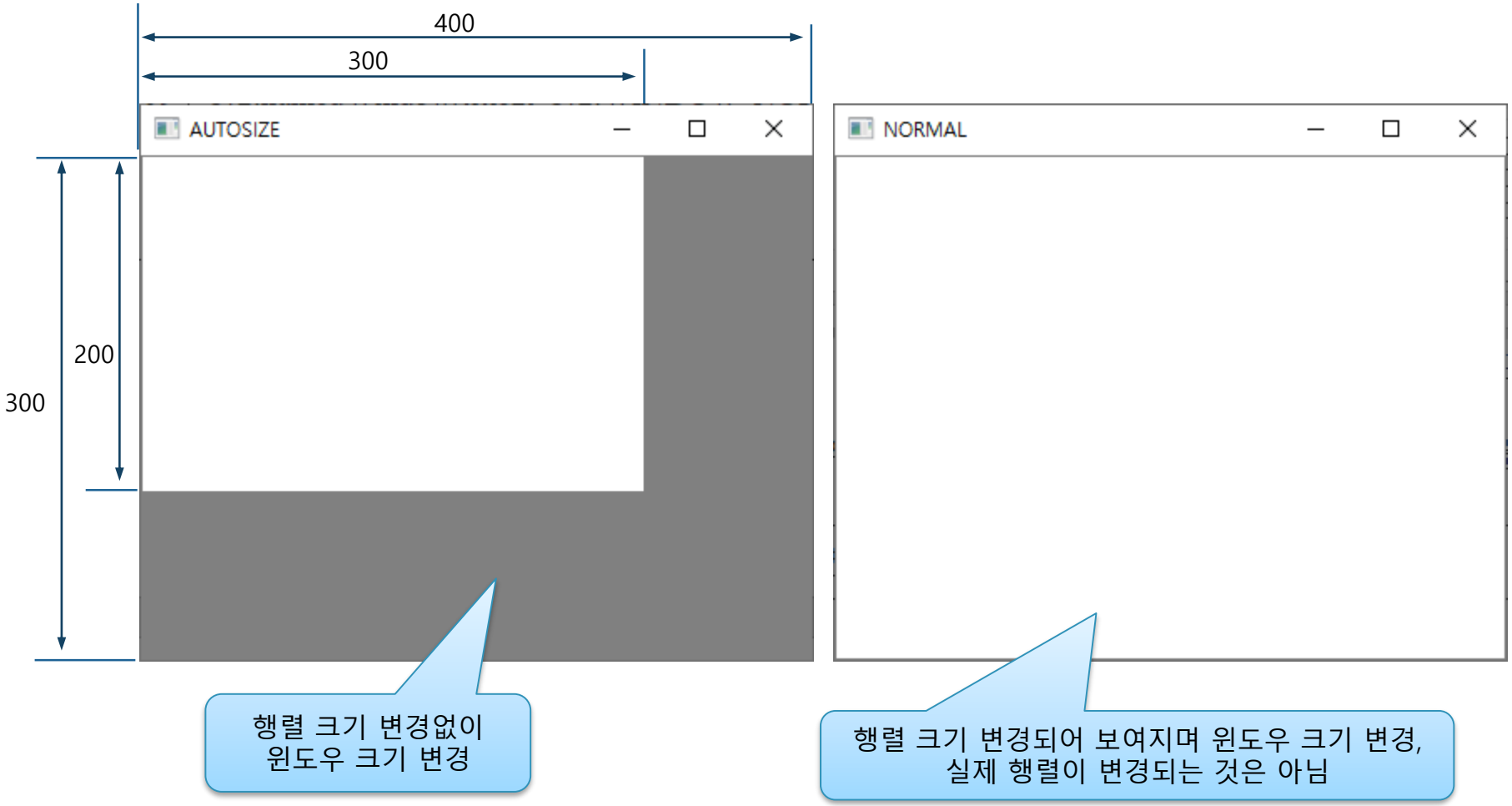
■ WINDOW_AUTOSIZE vs. WINDOW_NORMAL

np.ndarray.fill()
함수로 원소값 지정

```
1   import numpy as np
2   import cv2
3
4   image = np.zeros((200,300), np.uint8)
5
6   image.fill(255)
7
8   title1, title2 = 'AUTOSIZE', 'NORMAL'
9   cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)
10  cv2.namedWindow(title2, cv2.WINDOW_NORMAL)
11
12  cv2.imshow(title1, image)
13  cv2.imshow(title2, image)
14  cv2.resizeWindow(title1, 400, 300)
15  cv2.resizeWindow(title2, 400, 300)
16  cv2.waitKey(0)
17  cv2.destroyAllWindows()
```

윈도우 제어

■ WINDOW_AUTOSIZE vs. WINDOW_NORMAL



이벤트 처리 함수

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

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)	

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

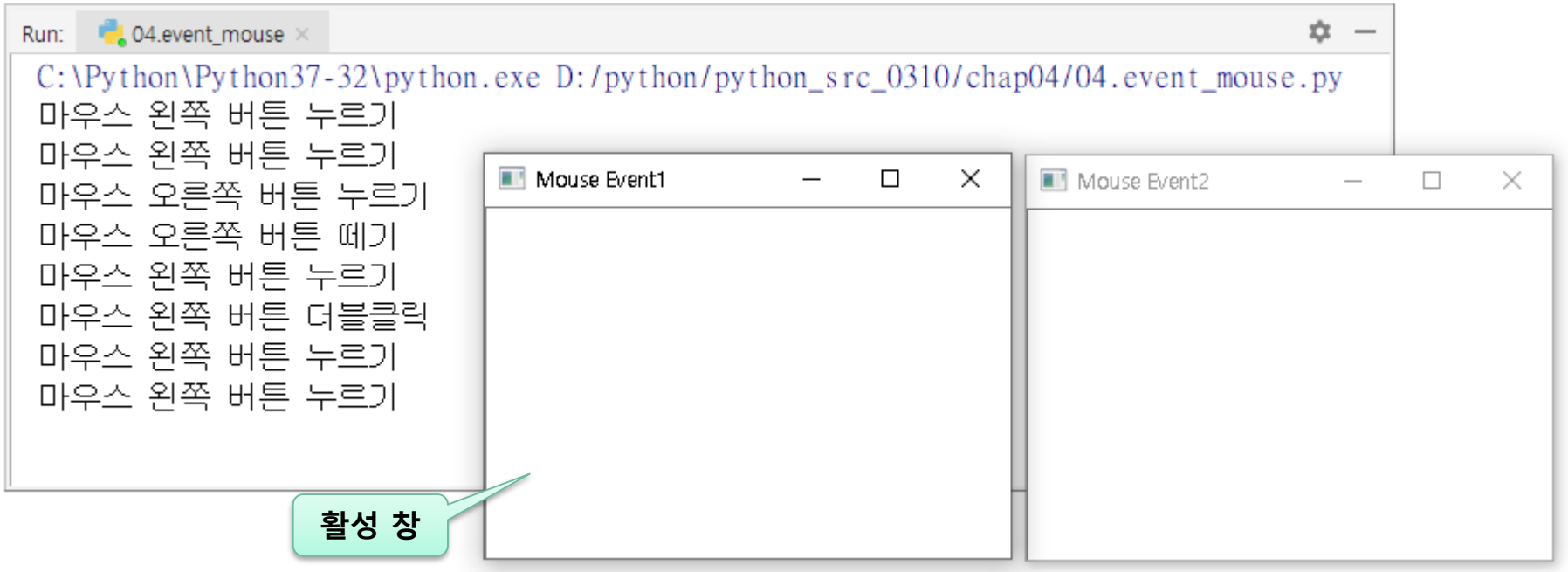
마우스 이벤트 제어

■ 예제: event_mouse.py

```
import numpy as np
import cv2

def onMouse(event, x, y, flags, param):                    # 콜백 함수 - 이벤트 내용 출력
    if event == cv2.EVENT_LBUTTONDOWN:
        print("마우스 왼쪽 버튼 누르기")
    elif event == cv2.EVENT_RBUTTONDOWN:
        print("마우스 오른쪽 버튼 누르기")
    elif event == cv2.EVENT_RBUTTONUP:
        print("마우스 오른쪽 버튼 떼기")
    elif event == cv2.EVENT_LBUTTONDBLCLK:
        print("마우스 왼쪽 버튼 더블클릭")


image = np.full((200, 300), 255, np.uint8)                 # 영상 생성

title1, title2 = "Mouse Event1", "Mouse Event2"     # 윈도우 이름
cv2.imshow(title1, image) # 영상 보기
cv2.imshow(title2, image)

cv2.setMouseCallback('Mouse Event1', onMouse)       # 마우스 콜백 함수

cv2.waitKey(0)                                      # 키 이벤트 대기
cv2.destroyAllWindows()                             # 열린 모든 윈도우 제거
```

마우스 이벤트 제어

■ 실행 결과



트랙바 이벤트 제어

■ 트랙바(trackbar)

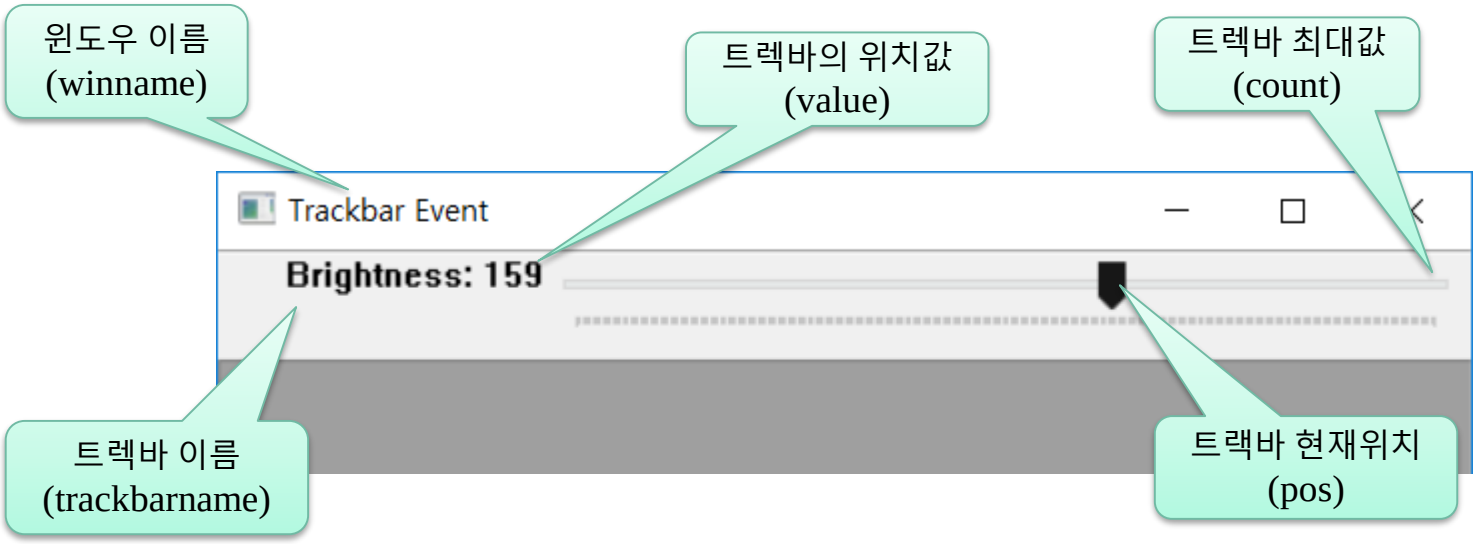
- 일정한 범위에서 특정한 값을 선택할 때 사용하는 일종의 스크롤바 혹은 슬라이더바

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.		

트랙바 이벤트 제어

■ 형식

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



트랙바 이벤트 제어

예제 4.2.3 트랙바 이벤트 사용 – 05.event_trackbar.py

```
01  import numpy as np
02  import cv2
03
04  def onChange(value):                                    # 트랙바 콜백 함수
05      global image, title                                 # 전역 변수 참조
06
07      add_value = value - int(image[0][0])                # 트랙바 값과 영상 화소값 차분
08      print("추가 화소값:", add_value)
09      image = image + add_value                           # 행렬과 스칼라 덧셈 수행
10      cv2.imshow(title, image)
11
12  image = np.zeros((300, 500), np.uint8)                  # 영상 생성
13
14  title = 'Trackbar Event'
15  cv2.imshow(title, image)
16
17  cv2.createTrackbar('Brightness', title, image[0][0], 255, onChange)  # 트랙바 콜백 함수 등록
18  cv2.waitKey(0)
19  cv2.destroyAllWindows()                                 # 열린 모든 윈도우 제거
```

트랙바 이벤트 제어

■ 실행 결과

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

선 그리기

■ cv2.line(img, start, end, color, thickness)

- Start와 End 점을 연결하여 직선을 그림
  - **img** – 그림을 그릴 이미지 파일
  - **start** – 시작 좌표(ex; (0,0))
  - **end** – 종료 좌표(ex; (500. 500))
  - **color** – BGR형태의 Color(ex; (255, 0, 0) -> Blue)
  - **thickness** (*int*) – 선의 두께. pixel

Parameter	내용
img	이미지 파일
pt1	시작점 좌표 (x, y)
pt2	종료점 좌표 (x, y)
color	색상 (blue, green, red) 0 ~ 255
thickness	선 두께 (default 1)
lineType	선 종류 (default cv.Line_8) - LINE_8 : 8-connected line - LINE_4 : 4-connecterd line - LINE_AA : antialiased line
shift	fractional bit (default 0)

사각형 그리기

■ cv2.rectangle(img, start, end, color, thickness)

- top-left corner와 bottom-right corner점을 연결하는 사각형을 그림
  - **img** – 그림을 그릴 이미지 파일
  - **start** – 시작 좌표(ex; (0,0))
  - **end** – 종료 좌표(ex; (500, 500))
  - **color** – BGR형태의 Color(ex; (255, 0, 0) -> Blue)
  - **thickness** (*int*) – 선의 두께. Pixel

- cv2.rectangle(img, rec, color, thickness)
  - rec – 사각형 영역(x, y, w, h)

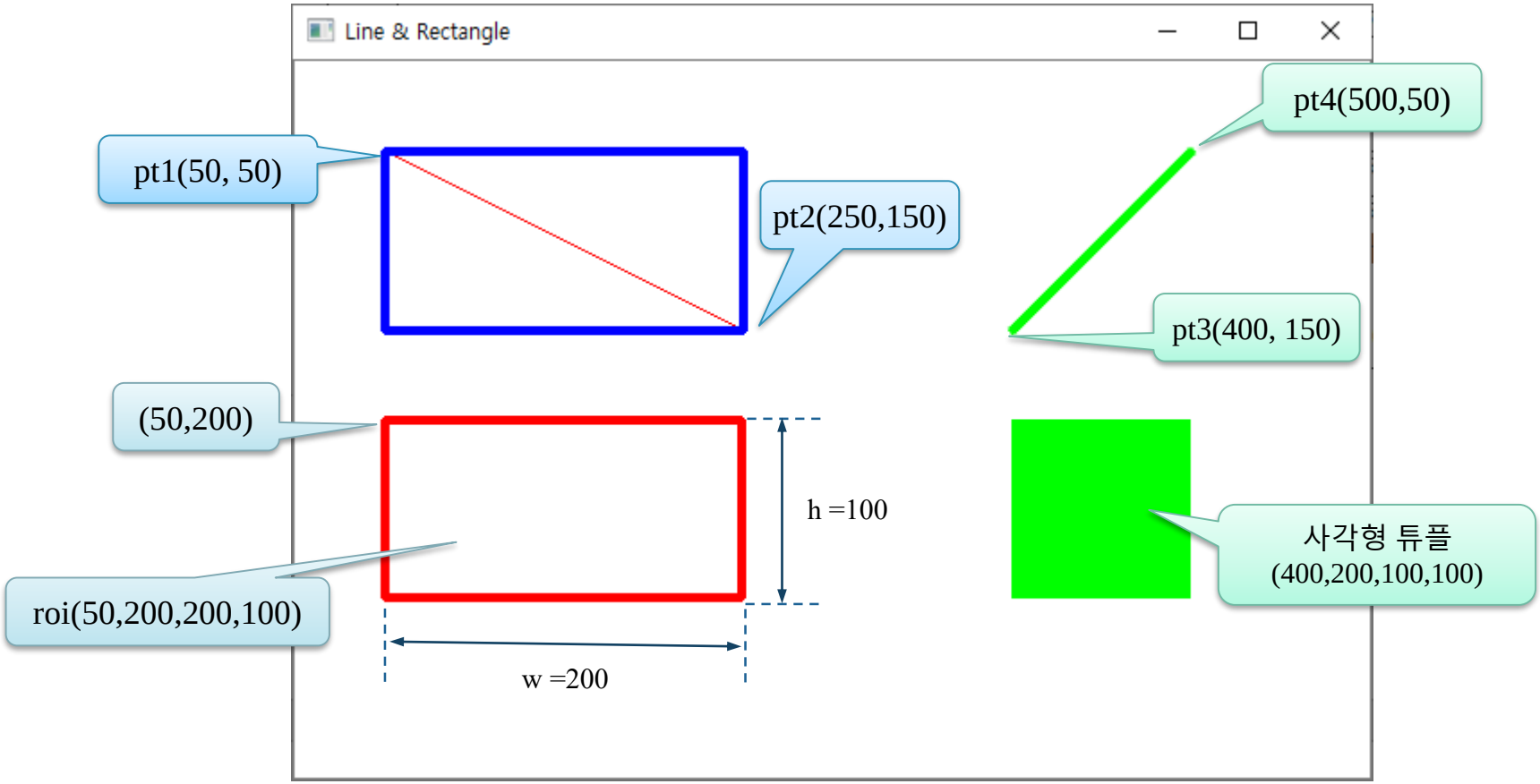
직선 및 사각형 그리기

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

```
01  import numpy as np
02  import cv2
03
04  blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)          # 색상 선언
05  image = np.zeros((400, 600, 3), np.uint8)                         # 3채널 컬러 영상 생성
06  image[:] = (255, 255, 255)                                        # 3채널 흰색
07
08  pt1, pt2 = (50, 50),  (250, 350)                                  # 좌표 선언 – 정수형 튜플
09  pt3, pt4 = (400, 150), (500, 50)
10  roi = (50, 200, 200, 100)                                         # 사각형 영역 – 4원소 튜플
11
12  ## 직선 그리기
13  cv2.line(image, pt1, pt2, red)
14  cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)                  # 계단 현상 감소선
15
16  ## 사각형 그리기
17  cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)               # 4방향 연결선
18  cv2.rectangle(image, roi, red, 3, cv2.LINE_8 )                    # 8방향 연결선
19  cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)     # 내부 채움
20
21  cv2.imshow("Line & Rectangle", image)                             # 윈도우에 영상 표시
22  cv2.waitKey(0)
23  cv2.destroyAllWindows()                                           # 모든 열린 윈도우 닫기
```

직선 및 사각형 그리기

■ 실행결과



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));
```

색상

시작좌표(pt1)

시작좌표(pt2)

3줄 세리프 폰트

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)

-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

글자 쓰기

예제 4.3.2 글자 쓰기 – 08.put_text.py

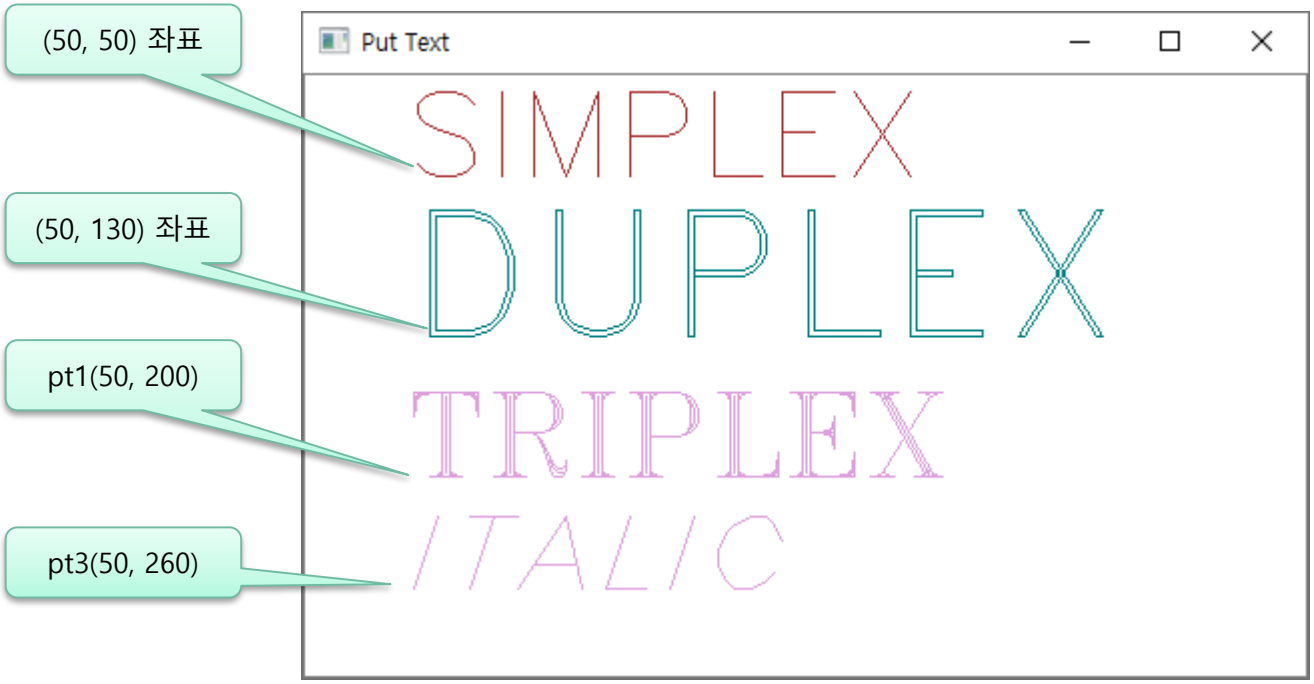
```
01  import numpy as np
02  import cv2
03
04  olive, violet, brown = (128, 128, 0), (221, 160, 221), (42, 42, 165)        # 색상 지정
05  pt1, pt2 =  (50, 230), (50, 310)                                            # 문자열 위치 좌표
06
07  image = np.zeros((350, 500, 3), np.uint8)
08  image.fill(255)
09
10  cv2.putText(image, 'SIMPLEX', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, brown)
11  cv2.putText(image, 'DUPLEX', (50, 130), cv2.FONT_HERSHEY_DUPLEX, 3, olive)
12  cv2.putText(image, 'TRIPLEX', pt1, cv2.FONT_HERSHEY_TRIPLEX, 2, violet)
13  fontFace = cv2.FONT_HERSHEY_PLAIN | cv2.FONT_ITALIC     # 글차체 상수
14  cv2.putText(image, 'ITALIC', pt2, fontFace, 4, violet)
15
16  cv2.imshow('Put Text', image)                                               # 윈도우 이름 지정 및 영상 표시
17  cv2.waitKey(0)                                                              # 키이벤트 대기
```

글자 확대 비율

기울임체 포함

4.3.2 글자 쓰기

- 실행결과



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

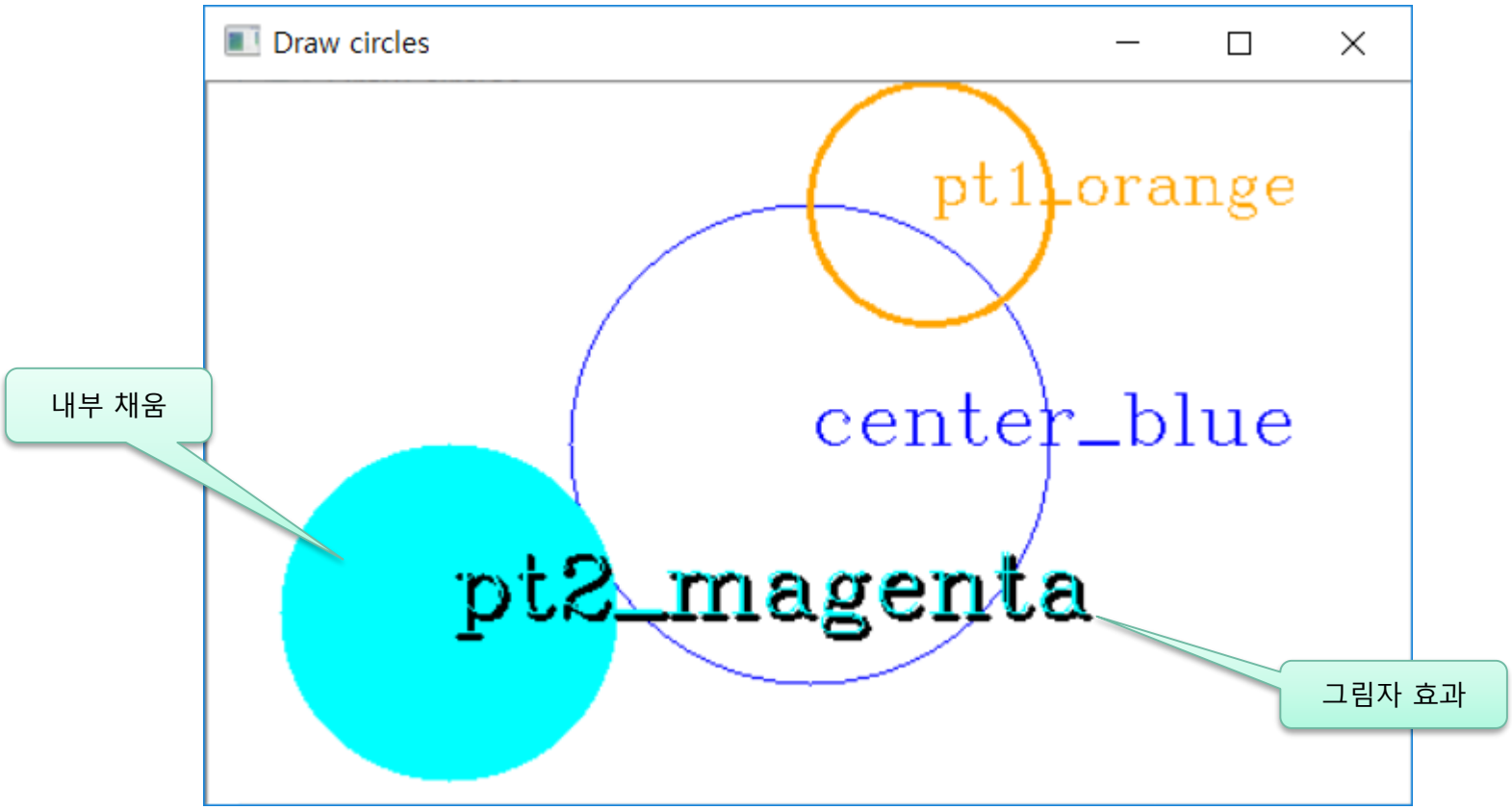
4.3.3 원 그리기

예제 4.3.3 원 그리기 – 09.draw_circle.py

```
01  import numpy as np
02  import cv2
03
04  orange, blue, cyan = (0, 165, 255), (255, 0, 0), (255, 255, 0)
05  white, black = (255, 255, 255), (0, 0, 0)
06  image = np.full((300, 500, 3), white, np.uint8)          # 컬러 영상 생성 및 초기화
07
08  center = (image.shape[1]//2, image.shape[0]//2)          # 영상 중심 좌표 – 역순 구성
09  pt1, pt2 = (300, 50), (100, 220)
10  shade = (pt2[0] + 2, pt2[1] + 2)                         # 그림자 좌표
11
12  cv2.circle(image, center, 100, blue)                     # 원 그리기
13  cv2.circle(image, pt1, 50, orange, 2)
14  cv2.circle(image, pt2, 70, cyan, -1)                     # 원 내부 채움
15
16  font = cv2.FONT_HERSHEY_COMPLEX;
17  cv2.putText(image, 'center_blue', center, font, 1.0, blue)
18  cv2.putText(image, 'pt1_orange', pt1, font, 0.8, orange)
19  cv2.putText(image, 'pt2_cyan', shade, font, 1.2, black, 2)       # 그림자 효과
20  cv2.putText(image, 'pt2_cyan', pt2, font, 1.2, cyan,    1)
21
22  cv2.imshow("Draw circles", image)
23  cv2.waitKey(0)
```

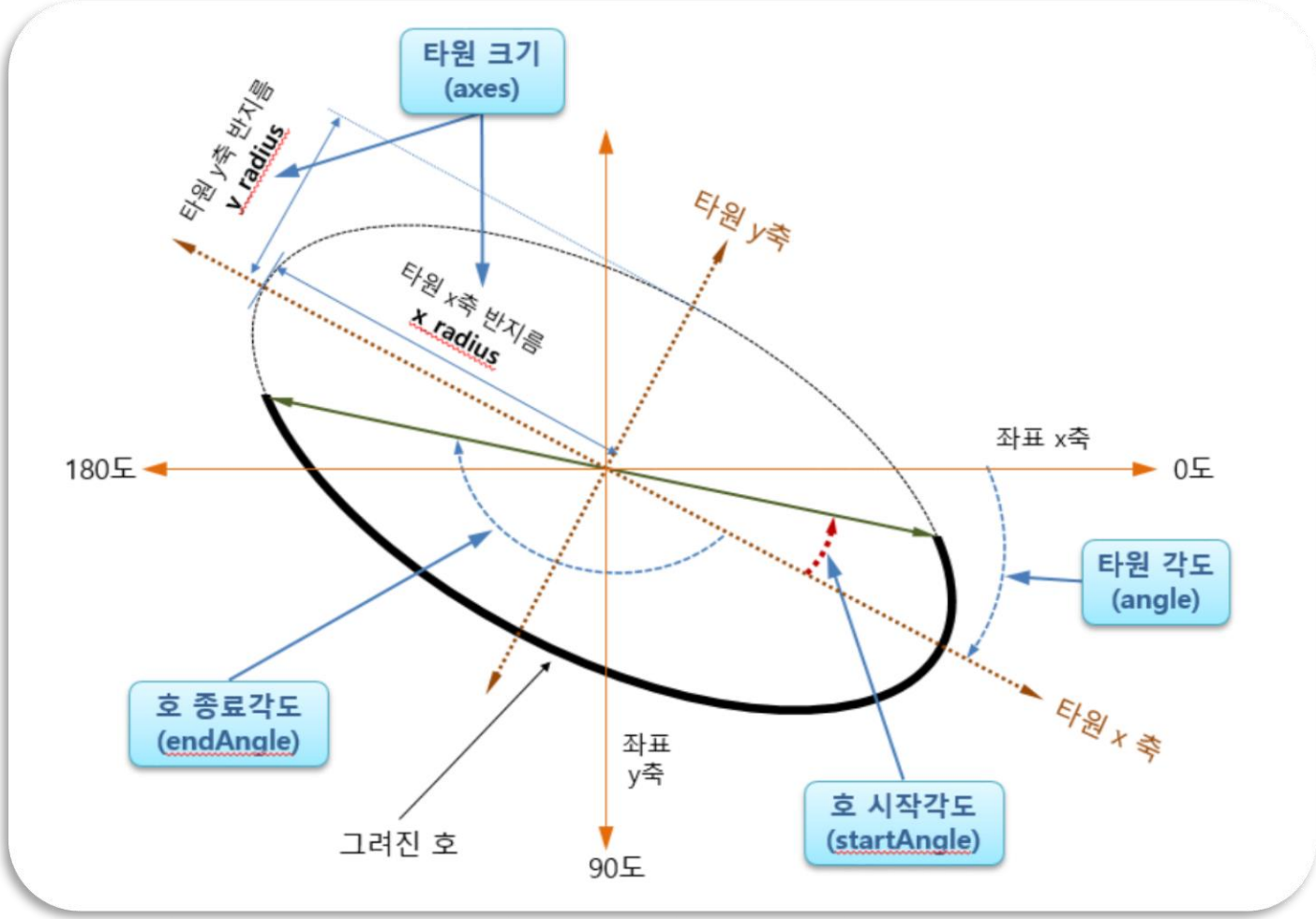
4.3.3 원 그리기

- 실행 결과



4.3.4 타원 그리기

- 타원 그리기 인수 의미



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

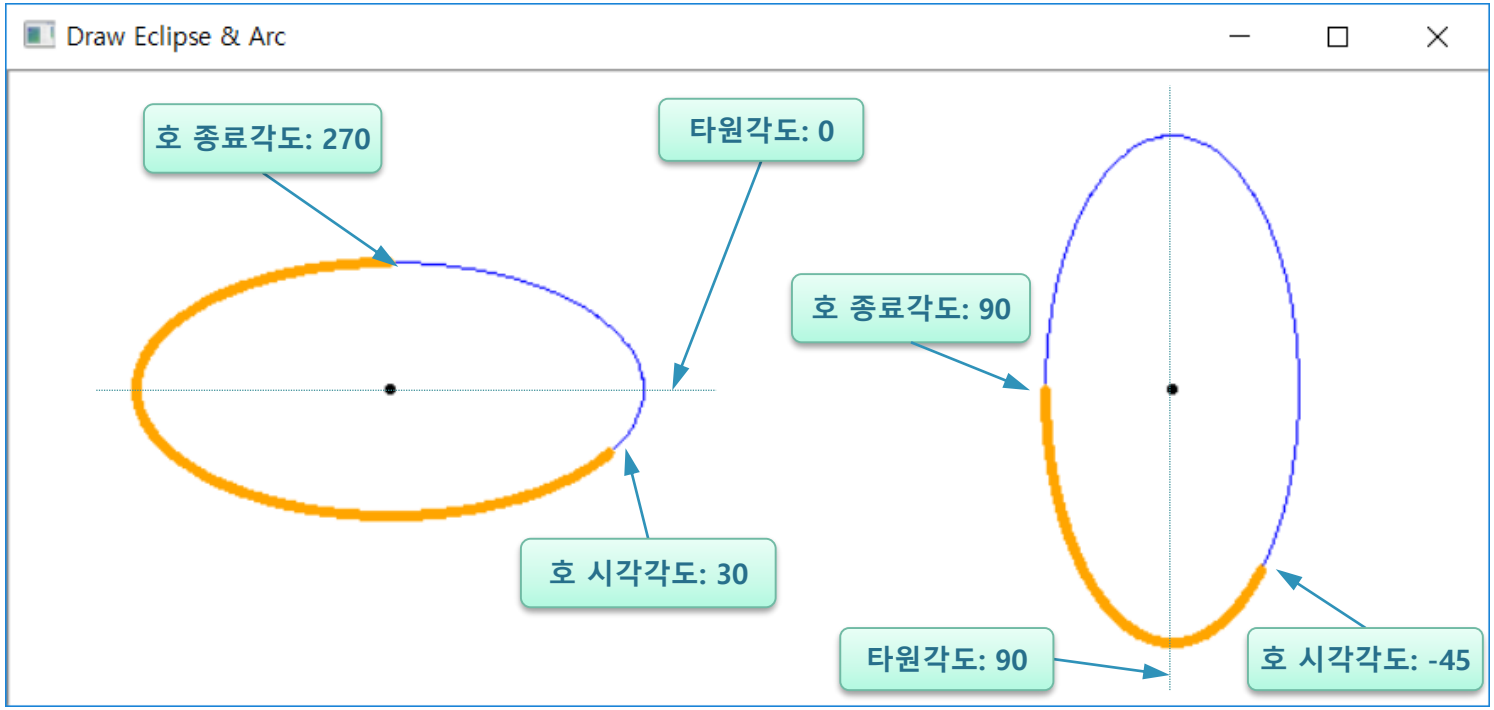
예제 4.3.4 타원 및 호 그리기 – 10.draw_ellipse.py

```
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
```

타원각도: 0

4.3.4 타원 그리기

- 실행 결과



4.4 영상파일 처리

- 영상처리
  - 2차원 데이터에 대한 행렬 연산

- 영상처리 프로그래밍을 한다는 것
  - 영상이라는 2차원 데이터의 원소 값을 개발자가 원하는 방향으로 변경하는 것
  - 영상을 다루려면 기본적으로 영상의 화소 접근, 값 수정, 새로 만들 수 있어야 함

4.4 영상파일 처리

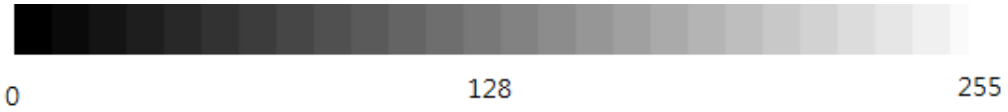
- 화소 (픽셀)
  - 8비트 그레이 레벨 영상 데이터의 밝기 정도

화소의 십진수 값	화소의 이진수 값	표현되는 밝기
0	0000 0000	검정색
⋮	⋮	⋮
63	0011 1111	어두운 회색
⋮	⋮	⋮
127	0111 1111	회색
⋮	⋮	⋮
191	1011 1111	밝은 회색
⋮	⋮	⋮
255	1111 1111	흰색

4.4 영상파일 처리

- 흑백 영상 ?
  - 단어 자체의 의미: 검은색과 흰색의 영상, 의미 안맞음

- 그레이 스케일(gray-scale) 영상 , 명암도 영상
  - 화소값은 0~255의 값을 가지는데 0은 검은색을, 255는 흰색을 의미
  - 0~255 사이의 값들은 다음과 같이 진한 회색에서 연한 회색



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

```
01  import cv2
02
03  def print_matInfo(name, image):                          # 행렬 정보 출력 함수
04      if image.dtype == 'uint8':      mat_type = 'CV_8U'
05      elif image.dtype == 'int8':     mat_type = 'CV_8S'
06      elif image.dtype == 'uint16':   mat_type = 'CV_16U'
07      elif image.dtype == 'int16':    mat_type = 'CV_16S'
08      elif image.dtype == 'float32':  mat_type = 'CV_32F'
09      elif image.dtype == 'float64':  mat_type = 'CV_64F'
10      nchannel = 3 if image.ndim == 3 else 1
11
12      ## depth, channel 출력
13      print("%12s: depth(%s), channels(%s) -> mat_type(%sC%d)"
14            % (name, image.dtype, nchannel, mat_type,  nchannel))
15
16  title1, title2 = 'gray2gray', 'gray2color'                          # 윈도우 이름
17  gray2gray = cv2.imread("images/read_gray.jpg", cv2.IMREAD_GRAYSCALE)    # 명암도
18  gray2color = cv2.imread("images/read_gray.jpg", cv2.IMREAD_COLOR)       # 컬러 영상
19
20  ## 예외처리 –영상파일 읽기 여부 조사
21  if gray2gray is None or gray2color is None :
22      raise Exception("영상파일 읽기 에러")
```

```
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 <module>
    raise Exception("영상파일 읽기 에러")
Exception: 영상파일 읽기 에러
```

4.4.1 영상파일 읽기

```
24  print("행렬 좌표 (100, 100) 화소값")
25  print("%s %s" % (title1, gray2gray[100, 100]))          # 행렬 내 한 화소값 표시
26  print("%s %s\n" % (title2, gray2color[100, 100]))
27
28  print_matInfo(title1, gray2gray)                        # 행렬 정보 출력 함수 호출
29  print_matInfo(title2, gray2color)
30
31  cv2.imshow(title1, gray2gray)                           # 행렬 정보를 영상으로 띄우기
32  cv2.imshow(title2, gray2color)
33  cv2.waitKey(0)
```

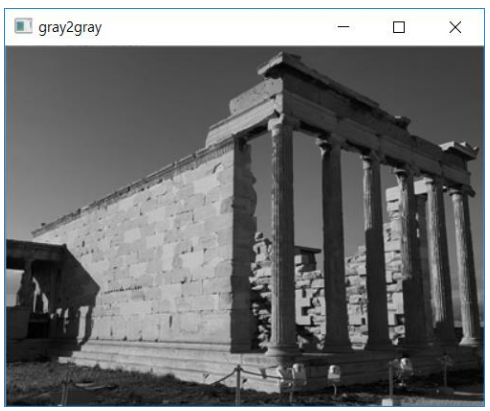
4.4.1 영상파일 읽기

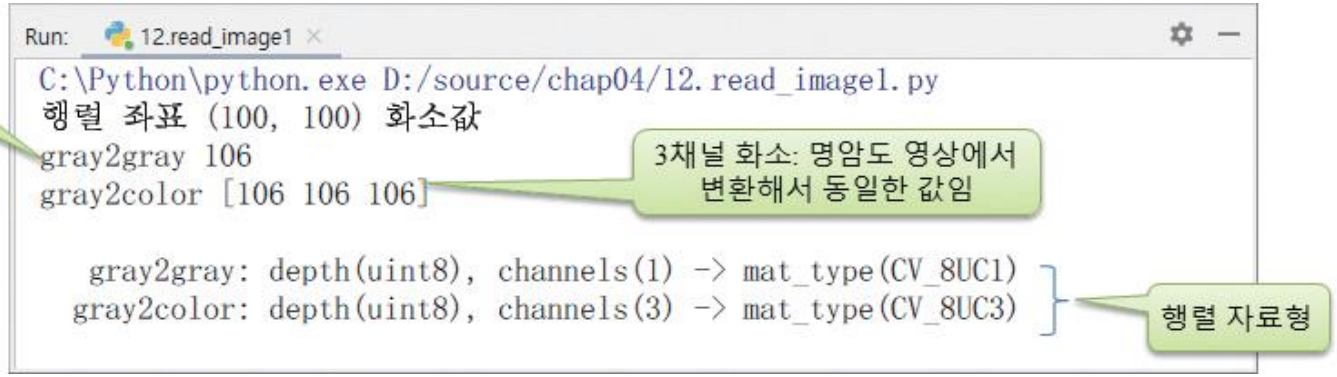
- CV_8U: 8-bit unsigned integer: uchar (0~255)
- CV_8S: 8-bit signed integer: schar (-128~127)
- CV_16U: 16-bit unsigned integer: ushort (0~65535)
- CV_16S: 16-bit signed integer: short (-32768~32767)
- CV_32S: 32-bit signed integer: int (-2147483648~2147483647)
- CV_32F: 32-bit floating-point number: float (-LT_MAX~FLT_MAX, INF, NAN)
- CV_64F: 64-bit floating-point number: double (-DBL_MAX~ DBL_MAX, INF, NAN)
- Multi-channel array: CV_8UC3, CV_8U(3), CV_64FC4, CV_64FC(4)

- CV_8UC1에서
  - CV : 컴퓨터비전의 약자
  - 8 : 비트단위, 하나의 픽셀을 표현하기 위해서 8bits를 활용하겠다는 의미.
  - U : unsigned의 약자 (S : signed, F : floating)
  - C1 : channel-1, 즉 채널 1개를 의미

4.4.1 영상파일 읽기

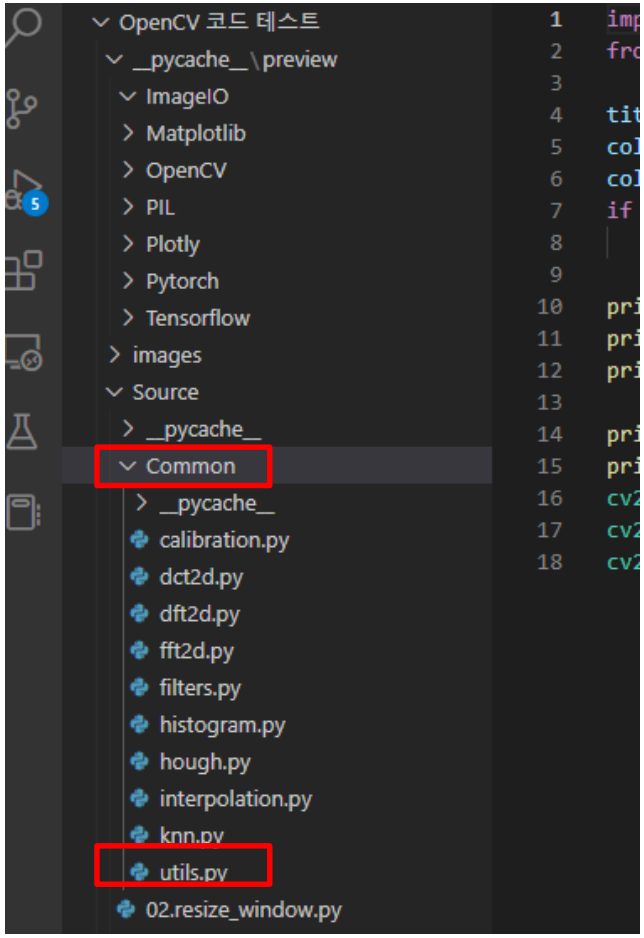
- 실행결과





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
  - '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)
```

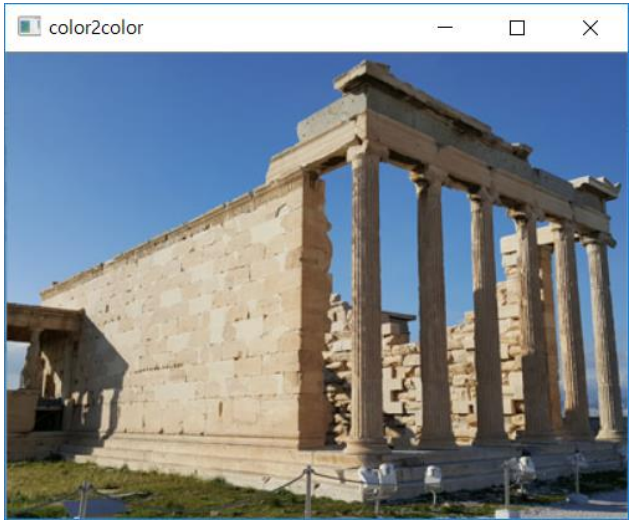
4.4.1 영상파일 읽기

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



4.4.2 행렬을 영상파일로 저장

- cv2.imwrite() 함수
  - 행렬을 영상 파일로 저장
  - 확장자에 따라서 JPG, BMP, PNG, TIF, PPM 등의 영상 파일 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

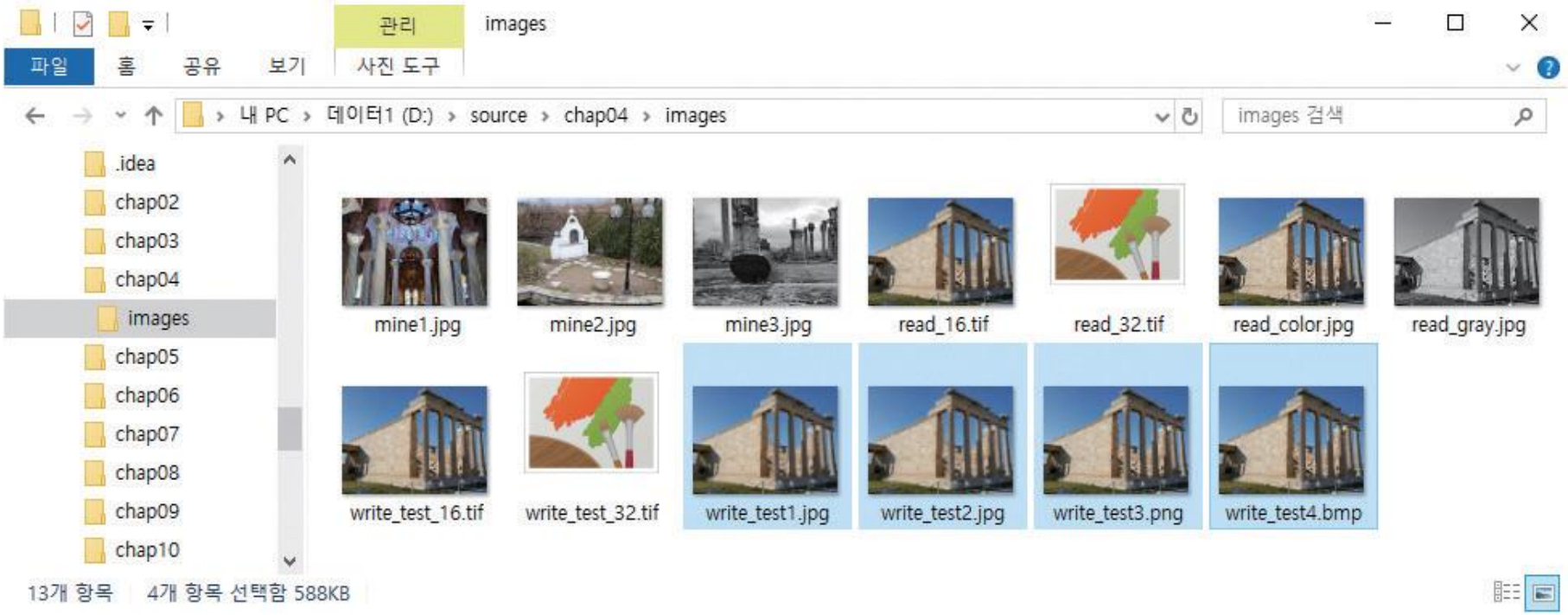
4.4.2 행렬을 영상파일로 저장

예제 4.4.3 행렬 영상 저장1 – 15.write_image1.py

```
01  import cv2
02
03  image = cv2.imread("images/read_color.jpg", cv2.IMREAD_COLOR)
04  if image is None: raise Exception("영상파일 읽기 에러")          # 예외처리
05
06  params_jpg = (cv2.IMWRITE_JPEG_QUALITY, 10)                    # JPEG 화질 설정
07  params_png = [cv2.IMWRITE_PNG_COMPRESSION, 9]                  # PNG 압축 레벨 설정
08
09  ## 행렬을 영상파일로 저장
10  cv2.imwrite("images/write_test1.jpg", image)                   # 디폴트는 95
11  cv2.imwrite("images/write_test2.jpg", image, params_jpg)       # 지정한 화질로 저장
12  cv2.imwrite("images/write_test3.png", image, params_png)
13  cv2.imwrite("iamges/write_test4.bmp", image)                   # BMP 파일로 저장
14  print("저장 완료")
```

4.4.2 행렬을 영상 파일로 저장

- 실행결과



4.5 비디오 처리

- 동영상 파일
  - 초당 30프레임 저장, 압축 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.

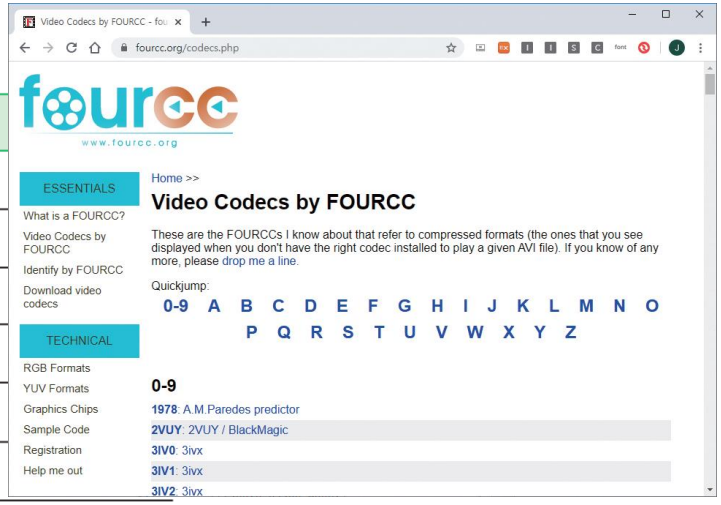
4.5 비디오 처리

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

〈표 4.5.2〉 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()
```

4.5.4 동영상파일 읽기

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
```

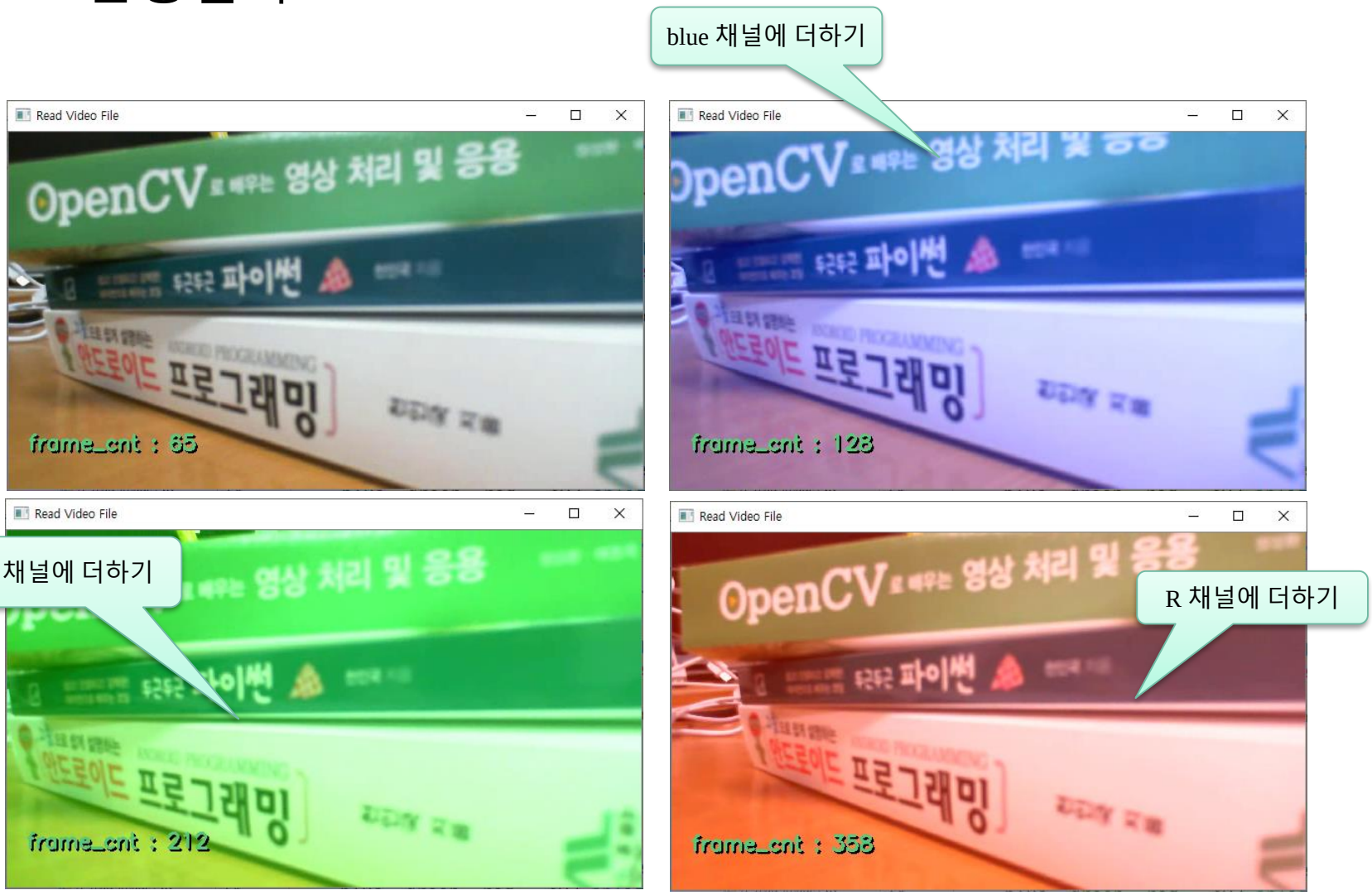
```
11  while True:
12      ret, frame = capture.read()
13      if not ret or cv2.waitKey(delay) >= 0: break       # 프레임 간 지연 시간 지정

14      blue, green, red = cv2.split(frame)                # 컬러 영상 채널 분리
15      frame_cnt += 1
16
17      if 100 <= frame_cnt < 200: cv2.add(blue, 100, blue)          # blue 채널 밝기 증가
18      elif 200 <= frame_cnt < 300: cv2.add(green, 100, green)      # green 채널 밝기 증가
19      elif 300 <= frame_cnt < 400: cv2.add(red  , 100, red)        # red 채널 밝기 증가
20
21      frame = cv2.merge( [blue, green, red] )                      # 단일채널 영상 합성
22      put_string(frame, 'frame_cnt: ', (20, 30), frame_cnt)
23      cv2.imshow("Read Video File", frame)
24  capture.release()
```

4.5.4 동영상파일 읽기

```
11  while True:
12      ret, frame = capture.read()
13      if not ret or cv2.waitKey(delay) >= 0: break      # 프레임 간 지연 시간 지정

14      blue, green, red = cv2.split(frame)               # 컬러 영상 채널 분리
15      frame_cnt += 1
16
17      if 100 <= frame_cnt < 200: cv2.add(blue, 100, blue)        # blue 채널 밝기 증가
18      elif 200 <= frame_cnt < 300: cv2.add(green, 100, green)    # green 채널 밝기 증가
19      elif 300 <= frame_cnt < 400: cv2.add(red  , 100, red)      # red 채널 밝기 증가
20
21      frame = cv2.merge( [blue, green, red] )                    # 단일채널 영상 합성
22      put_string(frame, 'frame_cnt: ', (20, 30), frame_cnt)
23      cv2.imshow("Read Video File", frame)
24  capture.release()
```

4.5.4 동영상파일 읽기

- 실행결과



연습문제1

- **PC 카메라를 통해 받아온 프레임에 다음의 영상처리를 수행하고, 결과 영상을 윈도우에 표시하는 프로그램을 작성하시오**
  - (200, 100)좌표에서 100X200 크기의 관심 영역 지정
  - 관심 영역에서 녹색 성분을 50만큼 증가
  - 관심 영역의 테두리를 두께 3의 빨간색으로 표시

연습문제2

■ **다음의 마우스 이벤트 제어 프로그램을 작성하시오**

- 마우스 오른쪽 버튼 클릭 시 원(클릭 좌표에서 반지름 20화소)을 그린다
- 마우스 왼쪽 버튼 클릭 시 사각형(크기 30 x 30)을 그린다.
- **추가**
  - **트랙바를 추가해서 선의 굵기를 1~10 픽셀로 조절한다.**
  - **트랙바를 추가해서 원의 반지름을 1~ 50픽셀로 조절한다.**