# 디지털 영상처리

## OpenCV 인터페이스

# 학습목표

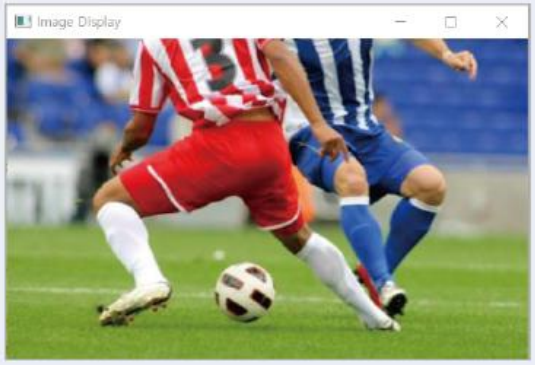
- OpenCV 프로그래밍 소개
- 윈도우 제어
- 이벤트 처리 함수
- 그리기 함수
- 영상파일 처리
- 비디오 처리

# 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

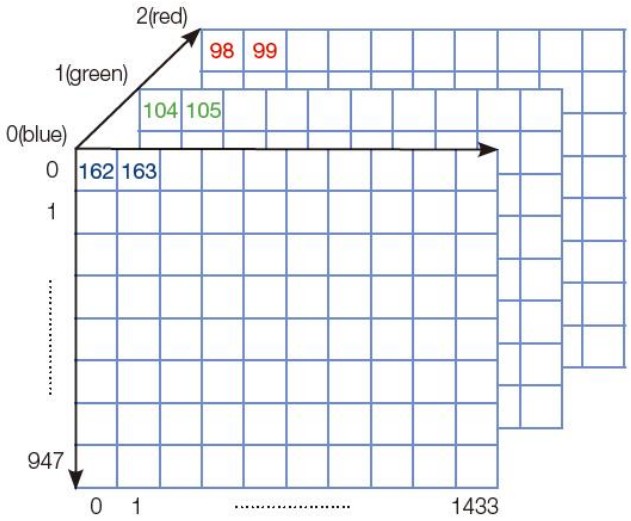
```
In [1]: type(img)
   numpy.ndarray
In [2]: img.shape
   (948,1434,3)
```

# OpenCV에서 영상은 numpy.ndarray 클래스 형의 객체

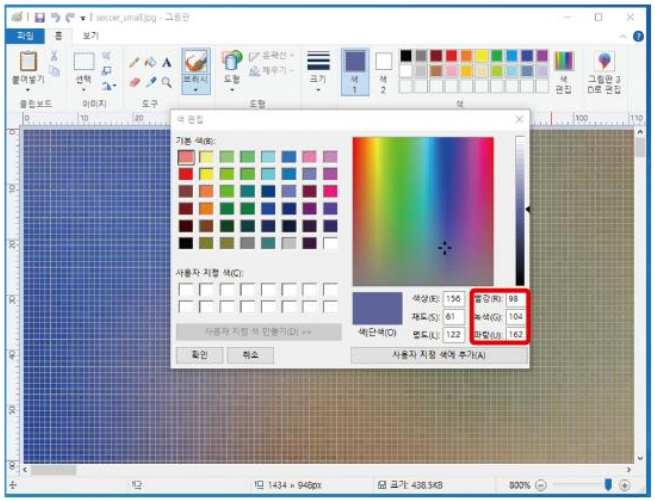
## ■ 영상의 표현

- 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

img 객체가 표현하는 영상의 구조와 내용

# [프로그래밍 예제3] 웹 캠에서 비디오 읽기

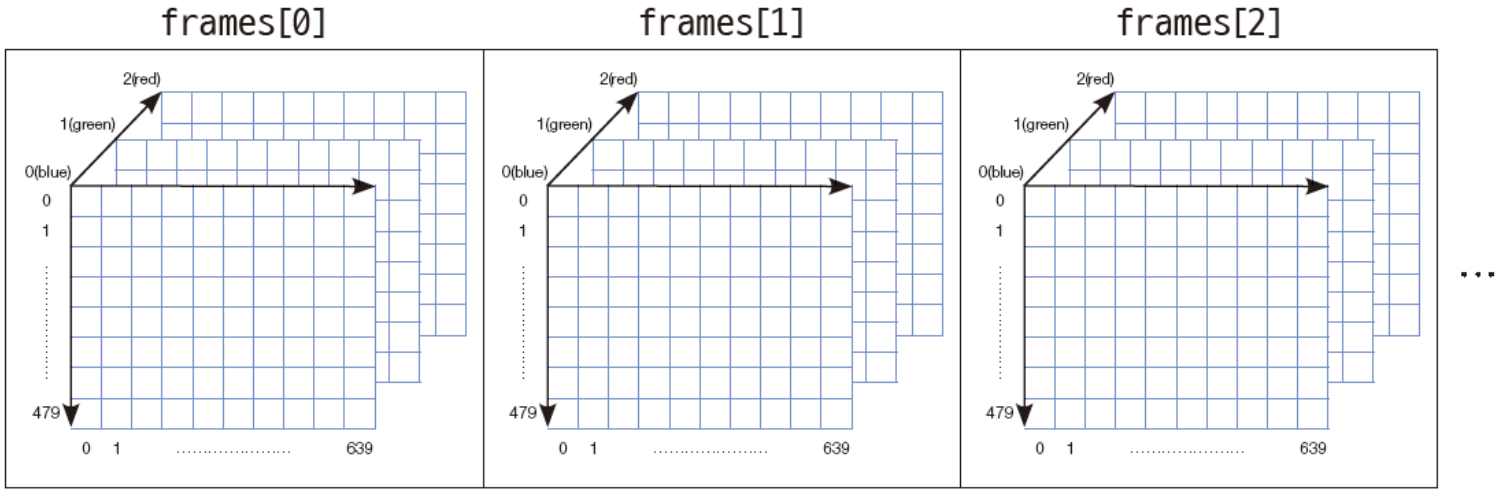
## ■ 웹 캠에서 비디오 읽기

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

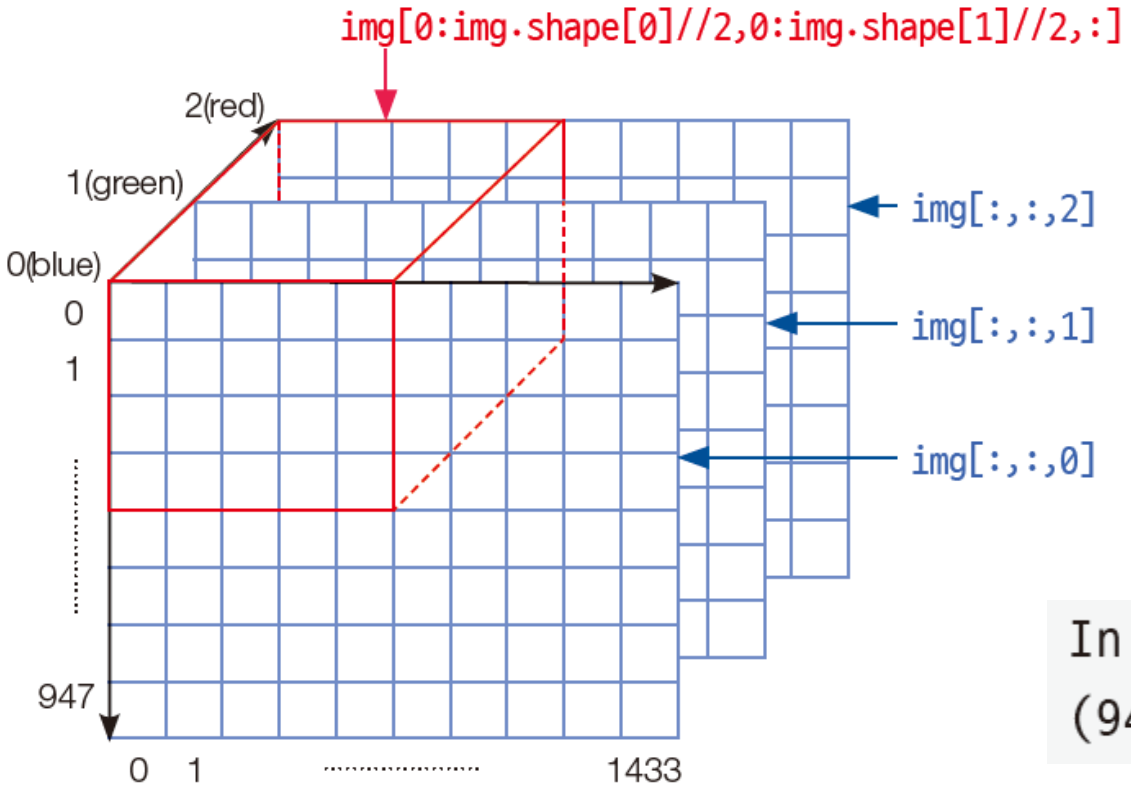
# 프로그래밍 실습: RGB 채널별로 디스플레이

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)
```

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

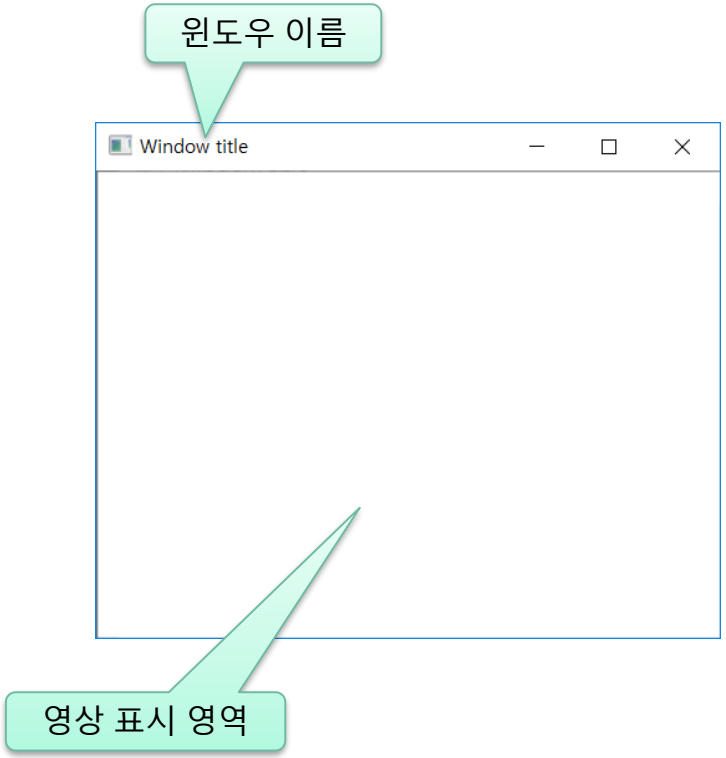
■ x, y 모니터 안에서 이동하려는 위치의 x, y 좌표

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

■ 설명: 윈도우의 크기를 재조정한다.

인수 설명

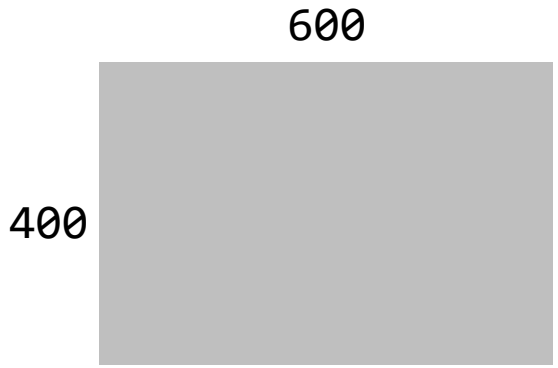
■ width, height 변경 윈도우의 가로, 세로 크기



# 윈도우 제어

OpenCV 테스트

```
import numpy as np
import cv2

image = np.zeros((400, 600), np.uint8)
image.fill(128)

cv2.imshow("Window title", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



# 윈도우 제어

■ 예제: move_window.py

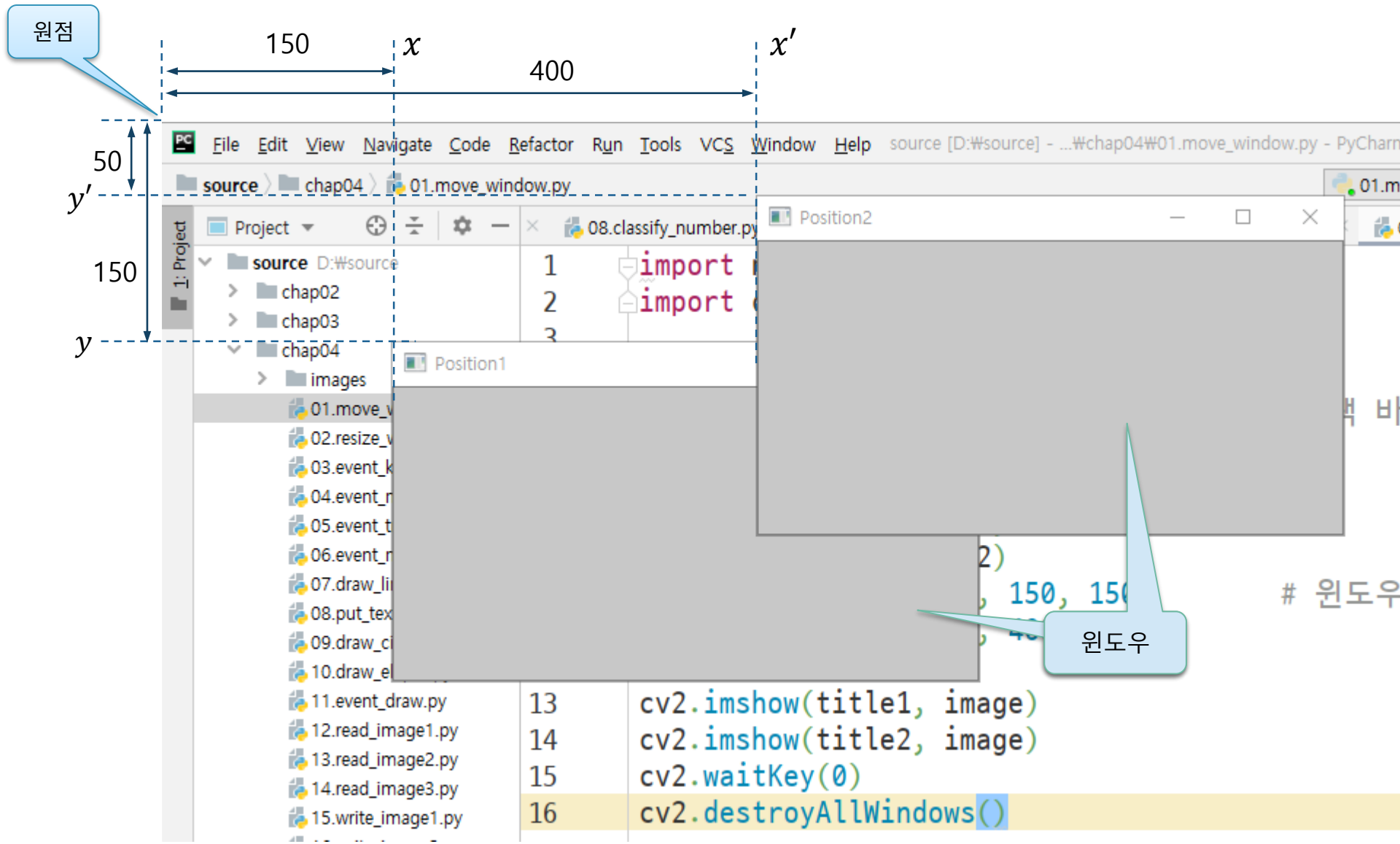
라이브러리 임포트

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

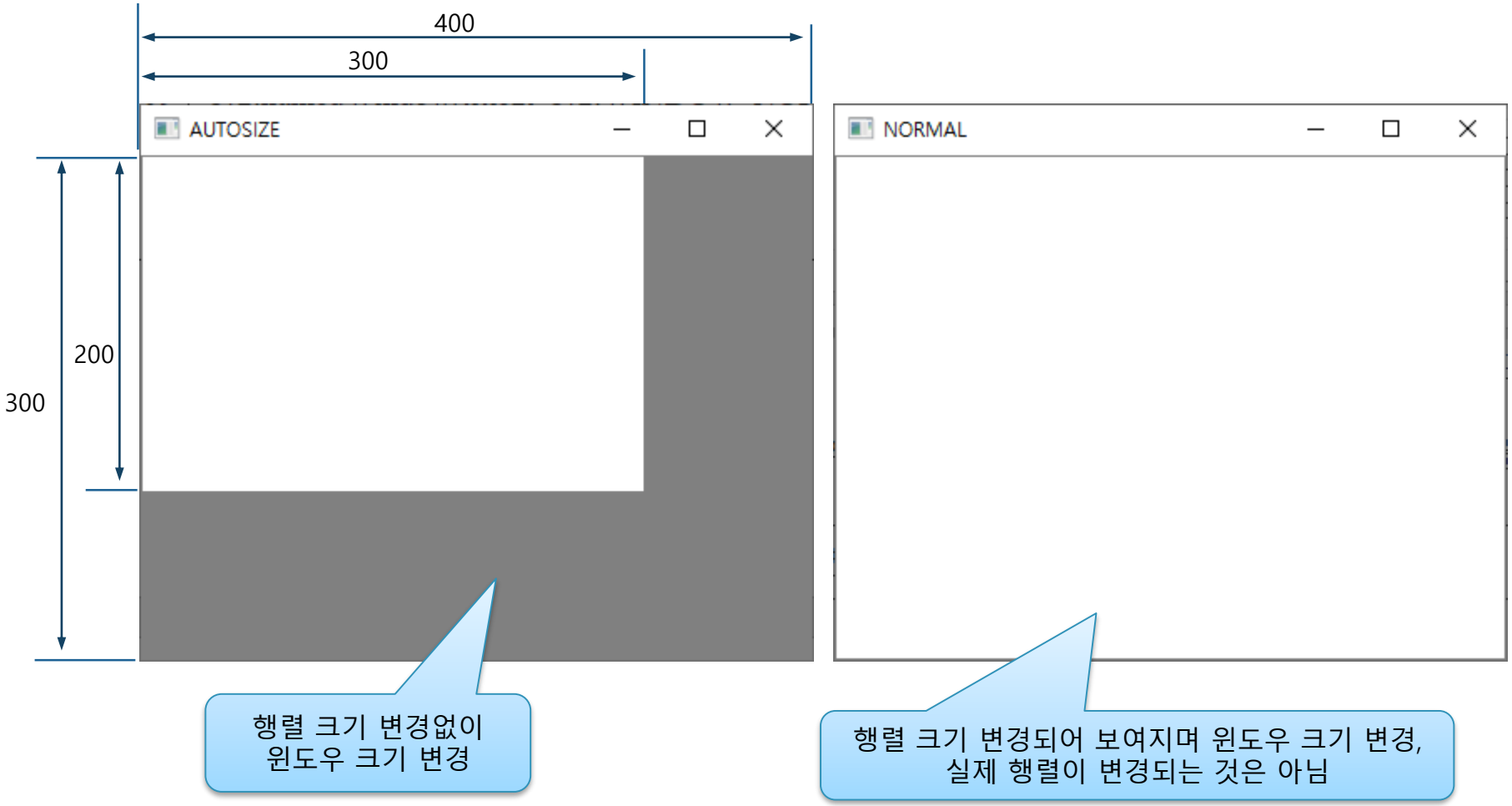
## ■ WINDOW_AUTOSIZE vs. WINDOW_NORMAL

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

- 사용자가 키보드의 키를 누르는 것
- 마우스를 움직이거나 마우스 버튼을 누르는 것
- 타이머와 같은 하드웨어 장치가 발생시키는 이벤트
- 사용자가 자체적으로 정의하는 이벤트

# 이벤트 처리 함수

## 콜백 함수

- 프로그래밍에서 특정 이벤트가 발생했을 때 호출되는 함수
- 이벤트를 처리하기 위해 사용
- 개발자가 시스템 함수를 직접 호출하는 방식
- 이벤트 발생 시 시스템이 개발자가 등록한 함수를 호출

# 이벤트 처리 함수

## OpenCV에서의 이벤트 처리

- 기본적인 이벤트 처리 함수 지원
- 키보드 이벤트
- 마우스 이벤트
- 트랙바(trackbar) 이벤트

## 이벤트 루프

- 프로그램이 이벤트를 기다리고 처리하는 구조
- 이벤트 큐에서 이벤트를 가져와 처리하는 방식

# 이벤트 처리 함수

## 이벤트 핸들러 등록

- 특정 이벤트 발생 시 호출될 함수를 미리 등록

예 cv2.setMouseCallback()을 사용해 마우스 이벤트 처리

# 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 |

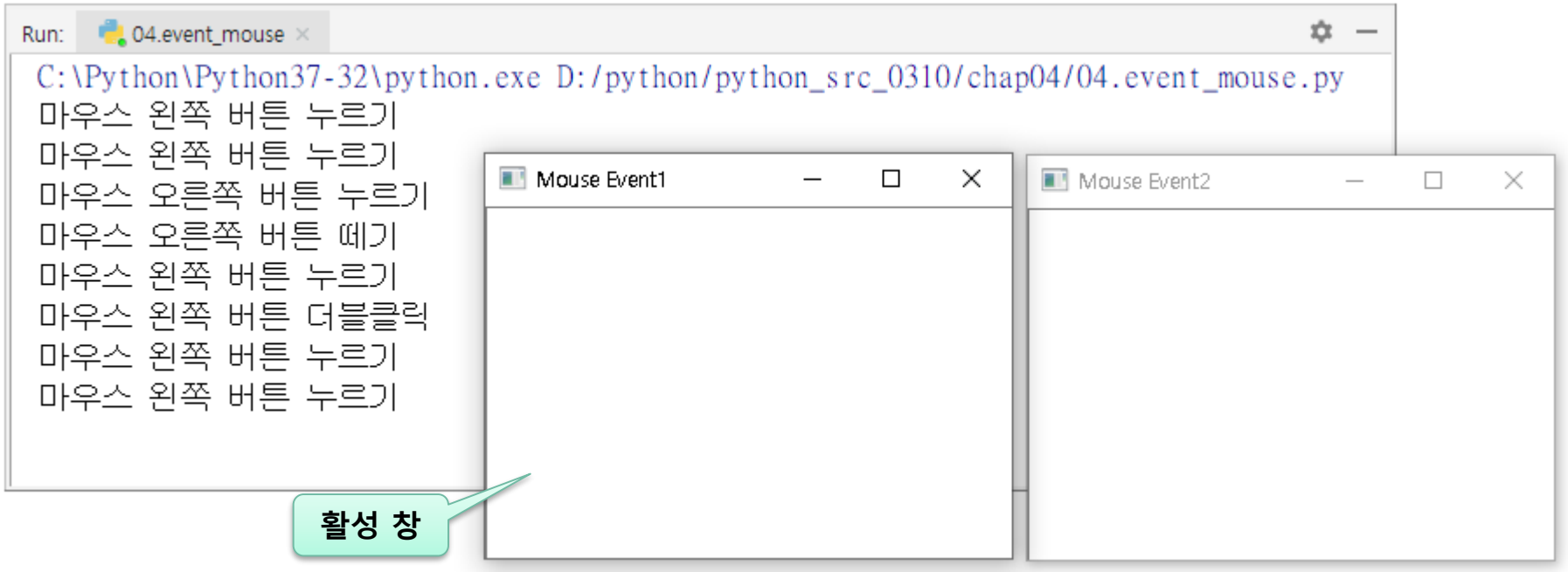
# 마우스 이벤트 제어

## ■ 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)

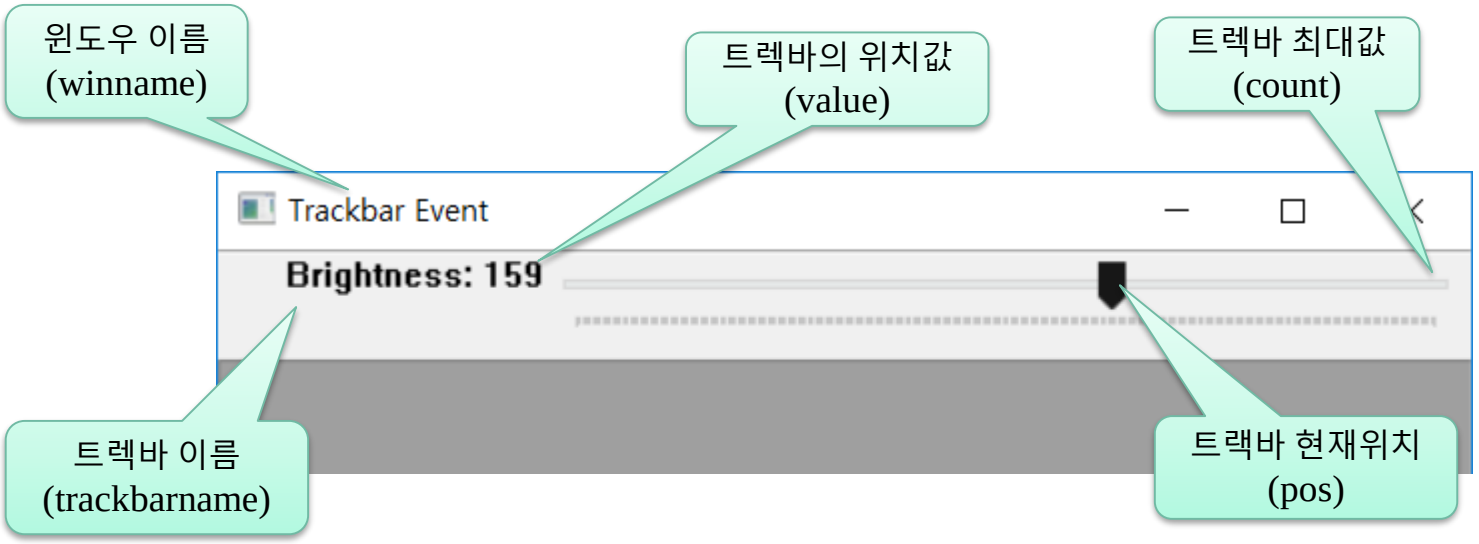
- 일정한 범위에서 특정한 값을 선택할 때 사용하는 일종의 스크롤바 혹은 슬라이더바

| 함수 설명 | | |
|---|---|---|
| cv2.createTrackbar(trackbarname, winname, value count, onChange) → None<br>■ 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<br>■ 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<br>■ 설명: 지정한 트랙바의 슬라이더 위치를 반환한다. | | |
| cv2.setTrackbarPos(trackbarname, winname, pos) → None<br>■ 설명: 지정한 트랙바의 슬라이더 위치를 설정한다. | | |

# 트랙바 이벤트 제어

## ■ 형식

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



# 트랙바 이벤트 제어

```
import numpy as np
import cv2

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

    add_value = value - initial_value                   # 트랙바 값과 초기값 차분
    print("추가 화소값:", add_value)

    image[:] = np.clip(image + add_value, 0, 255)       # 행렬과 스칼라 덧셈 수행, 화소값 클리핑
    cv2.imshow(title, image)

initial_value = 127
image = np.full((400, 600), initial_value, np.uint8)   # 초기값으로 영상 생성

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

cv2.createTrackbar("Brightness", title, initial_value, 255, onChange)

cv2.waitKey(0)
cv2.destroyWindow(title)
```

# 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

| Paremeter | 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)<br>• LINE_8 : 8-connected line<br>• LINE_4 : 4-connecterd line<br>• 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)

사각형 영역(x, y, w, h)

# 직선 및 사각형 그리기

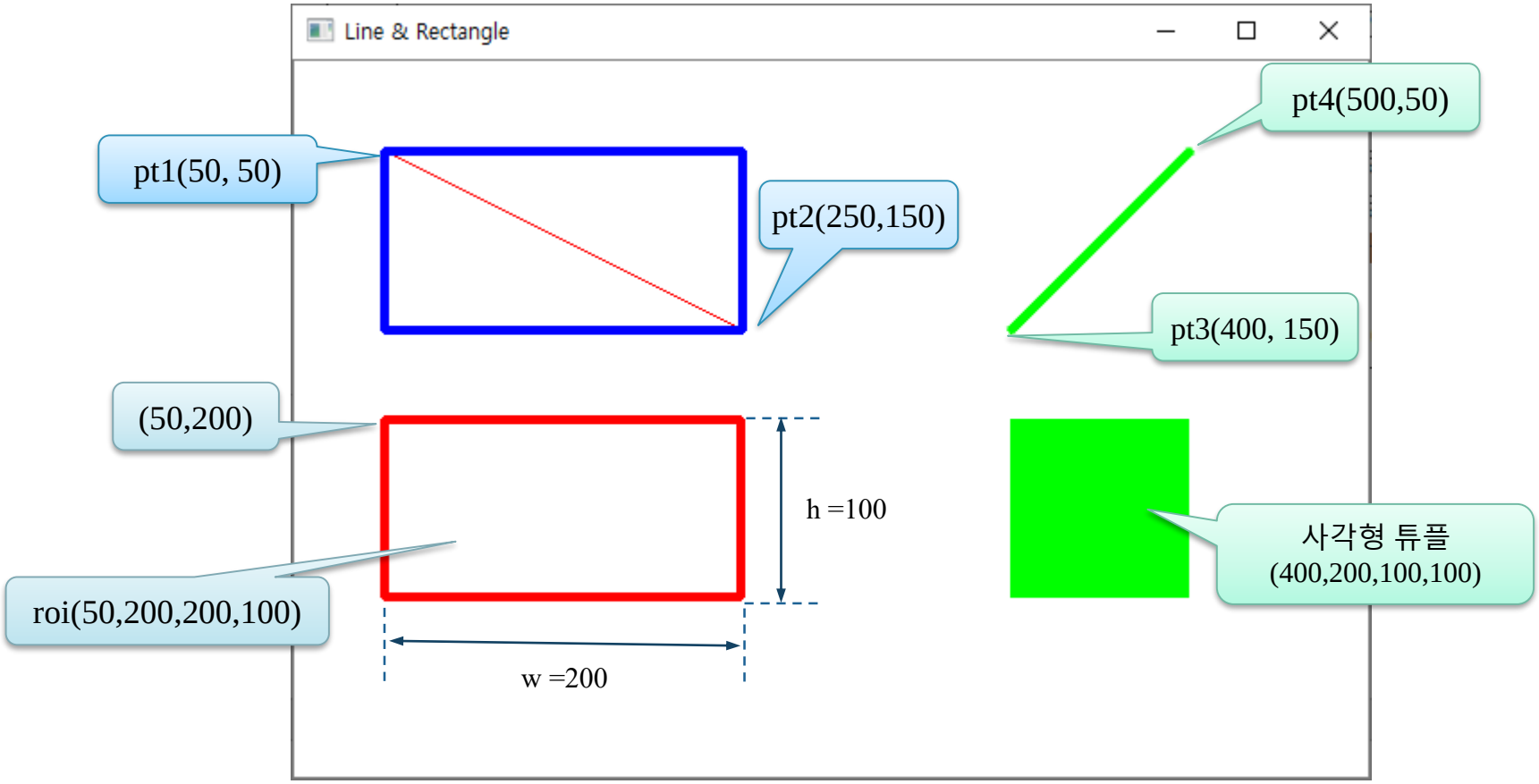
```
import numpy as np
import cv2

# 색상 선언
yellow, cyan, magenta = (0, 255, 255), (255, 255, 0), (255, 0, 255)

# 3채널 컬러 영상 생성, 초기화: 회색 배경
image = np.zeros((500, 700, 3), np.uint8)
image[:] = (200, 200, 200)

# 좌표 선언 – 정수형 튜플
pt1, pt2 = (100, 100), (300, 200)
pt3, pt4 = (500, 200), (600, 100)
roi = (100, 300, 300, 150)
```

# 직선 및 사각형 그리기

```
# 직선 그리기
cv2.line(image, pt1, pt2, cyan, 2, cv2.LINE_8)       # 8방향 연결선
cv2.line(image, pt3, pt4, magenta, 4, cv2.LINE_AA)  # 계단 현상이 감소한 선

# 사각형 그리기
cv2.rectangle(image, pt1, pt2, yellow, 3, cv2.LINE_4)  # 4방향 연결선
cv2.rectangle(image, roi, cyan, 3, cv2.LINE_8)              # 내부 채움
cv2.rectangle(image, (500, 300, 150, 200), magenta, cv2.FILLED) # 내부 채움

# 윈도우에 영상 표시
cv2.imshow('Line & Rectangle', image)
cv2.waitKey(0)
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

확대 비율

```
putText(image, "TEST", point1, cv2.FONT_HERSHEY_DUPLEX, 4, Scalar(100, 100, 100));
```

2줄 산세리프 폰트

# 글자 쓰기

| 함수 설명 | | |
|---|---|---|
| cv2.putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]]) → img<br>■ 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

```
import numpy as np
import cv2

# 색상 선언
olive, violet, brown, blue, red = (128, 128, 0), (221, 160, 221), (42, 42, 165),
(255, 0, 0), (0, 0, 255)

# 문자열 위치 좌표
pt1, pt2 = (50, 150), (50, 250)

# 3채널 컬러 영상 생성 및 초기화: 흰색 배경
image = np.zeros((400, 600, 3), np.uint8)
image.fill(255)
```

# 글자 쓰기

```
# 텍스트 출력
cv2.putText(image, "HELLO", (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, blue, 2)
cv2.putText(image, "WORLD", (50, 200), cv2.FONT_HERSHEY_DUPLEX, 2, red, 2)
cv2.putText(image, "OPEN CV", pt1, cv2.FONT_HERSHEY_TRIPLEX, 3, violet, 2)
fontFace = cv2.FONT_HERSHEY_PLAIN | cv2.FONT_ITALIC  # 글자체 상수
cv2.putText(image, "PYTHON", pt2, fontFace, 3, olive, 2)

# 윈도우에 영상 표시
cv2.imshow("Put Text", image)
cv2.waitKey(0)  # 키 이벤트 대기
cv2.destroyAllWindows()  # 모든 열린 윈도우 닫기
```

# 글자 쓰기

# 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<br>■ 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 |

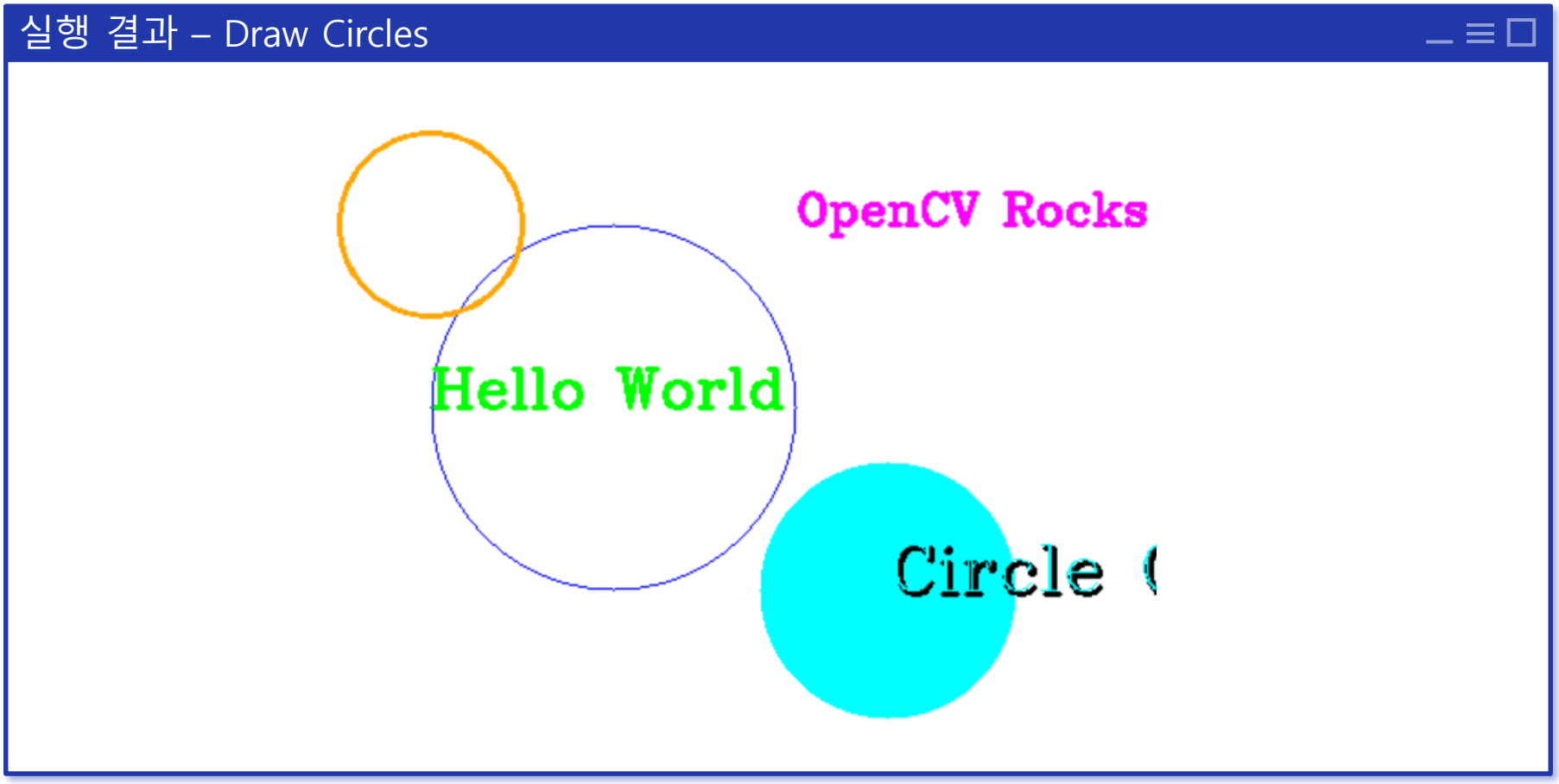
# 4.3.3 원 그리기

```
import numpy as np
import cv2

# 색상 선언
orange, blue, cyan = (0, 165, 255), (255, 0, 0), (255, 255, 0)
white, black, green, magenta = (255, 255, 255), (0, 0, 0), (0, 255, 0), (255, 0, 255)

# 컬러 영상 생성 및 초기화: 크기 (400, 600)
image = np.full((400, 600, 3), white, np.uint8)

# 새로운 동그라미 위치 좌표
center = (300, 200)  # 영상의 중심 좌표
pt1, pt2 = (200, 100), (450, 300)
shade = (pt2[0] + 2, pt2[1] + 2)  # 그림자 좌표
```

# 4.3.3 원 그리기

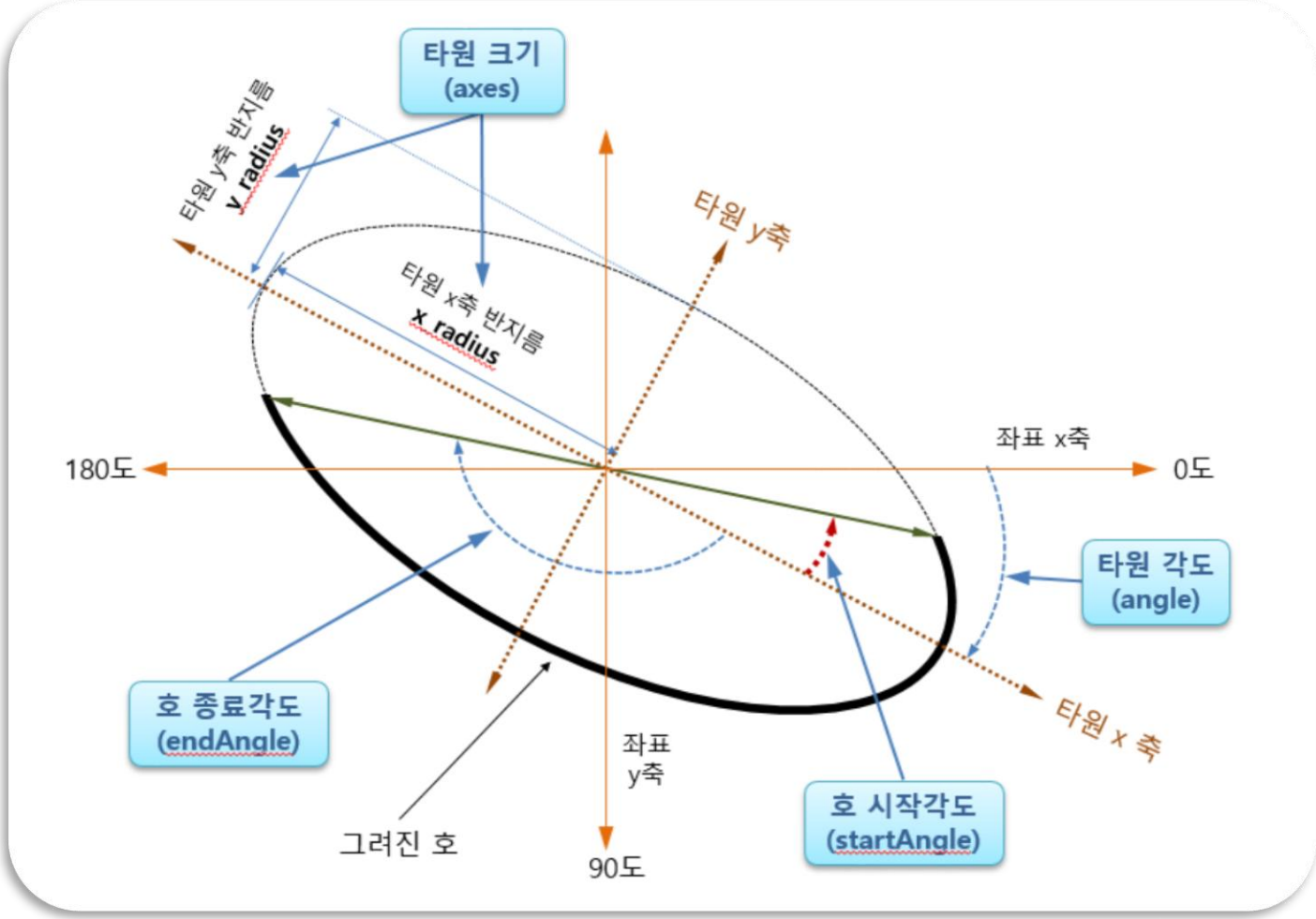
```
# 원 그리기
cv2.circle(image, center, 100, blue)  # 중심에 큰 원
cv2.circle(image, pt1, 50, orange, 2)  # 좌측 상단에 작은 원
cv2.circle(image, pt2, 70, cyan, -1)  # 우측 하단에 채워진 큰 원

# 텍스트 출력
font = cv2.FONT_HERSHEY_COMPLEX
cv2.putText(image, "Hello World", (200, 200), font, 1.0, green, 2)
cv2.putText(image, "OpenCV Rocks", (400, 100), font, 0.8, magenta, 2)
cv2.putText(image, "Circle Cyan", shade, font, 1.2, black, 2)  # 그림자 효과
cv2.putText(image, "Circle Cyan", pt2, font, 1.2, cyan, 1)

# 윈도우에 영상 표시
title = "Draw Circles"
cv2.namedWindow(title)
cv2.imshow(title, image)
cv2.waitKey(0)
cv2.destroyAllWindows()  # 모든 열린 윈도우 닫기
```

# 원 그리기



# 타원 그리기

- 타원 그리기 인수 의미



# 타원 그리기

| 함수명과 반환형 및 인수 구조 | | |
|---|---|---|
| cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color[, thickness[, lineType[, shift]]]) →img<br>■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 |

# 타원 그리기

```
import numpy as np
import cv2

# 색상 지정
green, red, yellow = (0, 255, 0),(0, 0, 255),(0, 255, 255)
white = (255, 255, 255)

# 3채널 컬러 영상 생성 및 초기화: 크기 (400, 800)
image = np.full((400, 800, 3), white, np.uint8)

# 새로운 타원 중심점과 크기
pt1, pt2 = (200, 200), (600, 200)
size = (150, 80)

# 타원의 중심점(2화소 원) 표시
cv2.circle(image, pt1, 1, 0, 2)
cv2.circle(image, pt2, 1, 0, 2)
```

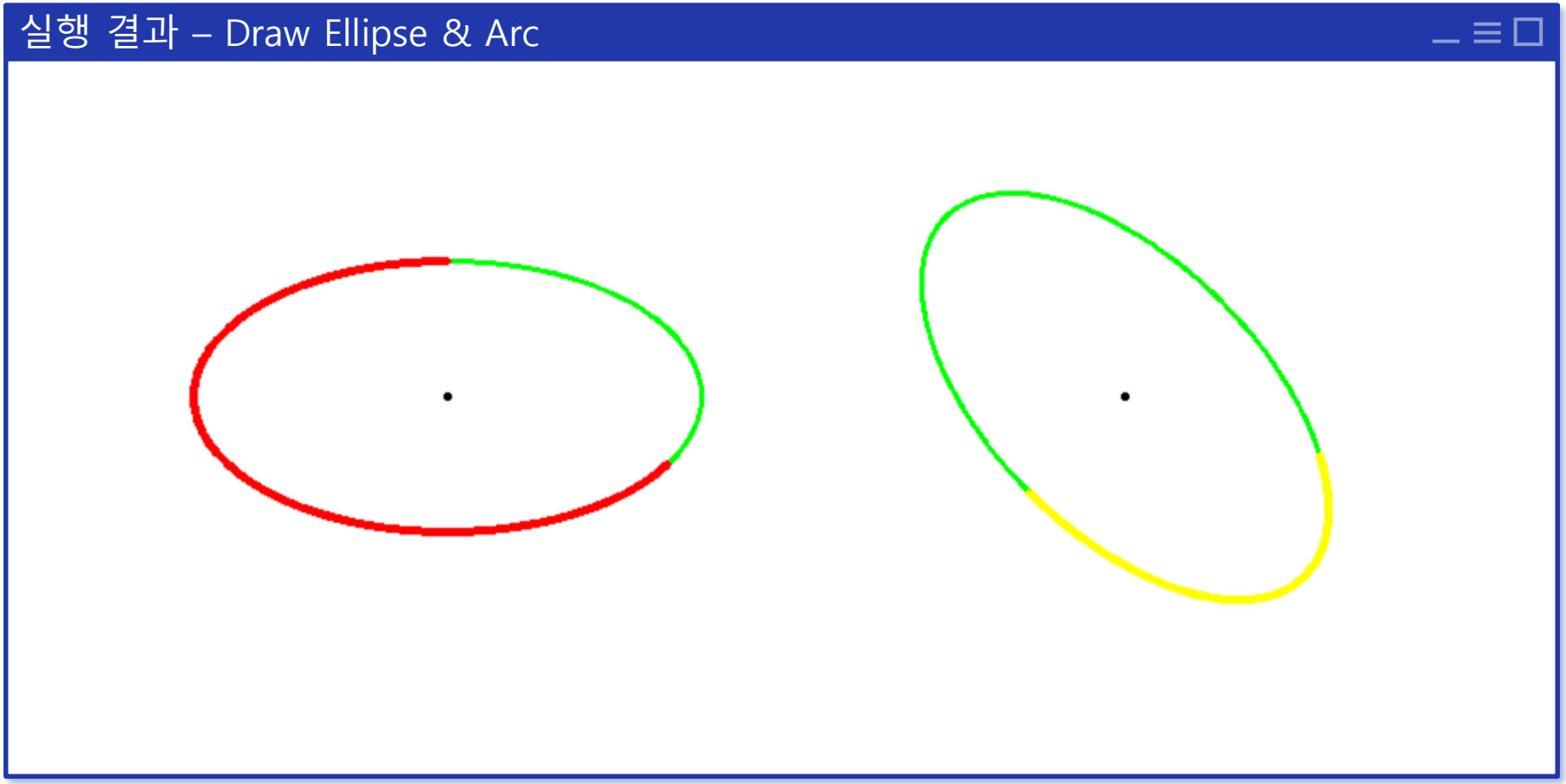
| 항목 | 설명 |
|---|---|
| 목적 | 이미지에 타원(ellipse)을 그리기 위해 사용 |
| 기본 사용법 | `cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color, thickness=1, lineType=cv2.LINE_8, shift=0)` |
| 매개변수 | `img`: 타원을 그릴 이미지 (NumPy 배열)<br>`center`: 타원의 중심 좌표 (튜플, `(x, y)` 형식)<br>`axes`: 타원의 두 축 길이 (튜플, `(major_axis, minor_axis)` 형식)<br>`angle`: 타원의 회전 각도 (실수, 단위: 도)<br>`startAngle`: 타원의 호(arc) 시작 각도 (실수, 단위: 도)<br>`endAngle`: 타원의 호(arc) 끝 각도 (실수, 단위: 도)<br>`color`: 타원의 색상 (튜플, `(B, G, R)` 형식)<br>`thickness`: 타원의 두께 (정수, 기본값: `1`, `-1`로 설정 시 타원 내부 채움)<br>`lineType`: 라인 타입 (예: `cv2.LINE_8`, 기본값: `cv2.LINE_8`)<br>`shift`: 중심 좌표와 축 길이 값의 소수점 자리 수 (정수, 기본값: `0`) |

# 타원 그리기

```
# 타원 그리기
cv2.ellipse(image, pt1, size, 0, 0, 360, green, 2)
cv2.ellipse(image, pt2, size, 45, 0, 360, green, 2)

# 호 그리기
cv2.ellipse(image, pt1, size, 0, 30, 270, red, 4)
cv2.ellipse(image, pt2, size, 45, -45, 90, yellow, 4)

# 윈도우에 영상 표시
cv2.imshow("Draw Ellipse & Arc", image)
cv2.waitKey(0)  # 키입력 대기
cv2.destroyAllWindows()  # 모든 열린 윈도우 닫기
```

| 항목 | 설명 |
| --- | --- |
| 목적 | 이미지에 타원(ellipse)을 그리기 위해 사용 |
| 기본 사용법 | `cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color, thickness=1, lineType=cv2.LINE_8, shift=0)` |
| 매개변수 | `img`: 타원을 그릴 이미지 (NumPy 배열)<br>`center`: 타원의 중심 좌표 (튜플, `(x, y)` 형식)<br>`axes`: 타원의 두 축 길이 (튜플, `(major_axis, minor_axis)` 형식)<br>`angle`: 타원의 회전 각도 (실수, 단위: 도)<br>`startAngle`: 타원의 호(arc) 시작 각도 (실수, 단위: 도)<br>`endAngle`: 타원의 호(arc) 끝 각도 (실수, 단위: 도)<br>`color`: 타원의 색상 (튜플, `(B, G, R)` 형식)<br>`thickness`: 타원의 두께 (정수, 기본값: `1`, `-1`로 설정 시 타원 내부 채움)<br>`lineType`: 라인 타입 (예: `cv2.LINE_8`, 기본값: `cv2.LINE_8`)<br>`shift`: 중심 좌표와 축 길이 값의 소수점 자리 수 (정수, 기본값: `0`) |

# 타원 그리기



# 영상파일 처리

## 영상처리

2차원 데이터에 대한 행렬 연산

## 영상처리 프로그래밍

- 2차원 데이터인 영상의 각 픽셀 값을 원하는 방식으로 조작하는 과정
- 픽셀에 접근하고 값을 변경하며, 새로운 영상을 생성할 수 있어야 함
- 영상 데이터를 다루기 위한 기본적인 기술 습득 필요

# 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 |

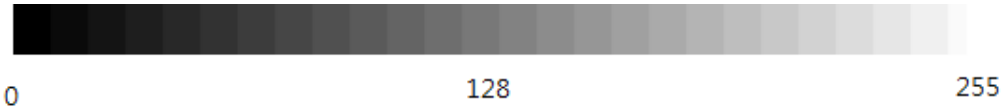
# 영상파일 처리

## 흑백 영상

- 단어 자체의 의미 : 검은색과 흰색의 영상, 의미 안 맞음

## 그레이 스케일(gray-scale) 영상 혹은 명암도 영상

- 화소 값은 0~255의 값을 가지는데 0은 검은색을,
  255는 흰색을 의미
- 0~255 사이의 값들은 다음과 같이 진한 회색에서 연한 회색



# 영상파일 읽기

영상처리 과정

영상파일을 불러와 행렬 형태로 저장 → 행렬 연산 중 필요한 경우 원소 값을 직접 확인 → 처리된 행렬을 다시 영상파일로 저장

- 영상파일을 처리해 주는 함수와 활용 방법 필수

| 함수 설명 | | |
|---|---|---|
| cv2.imread(filename[, flags]) → retval<br>■ 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<br>■ 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) |

# 영상파일 읽기

〈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 |

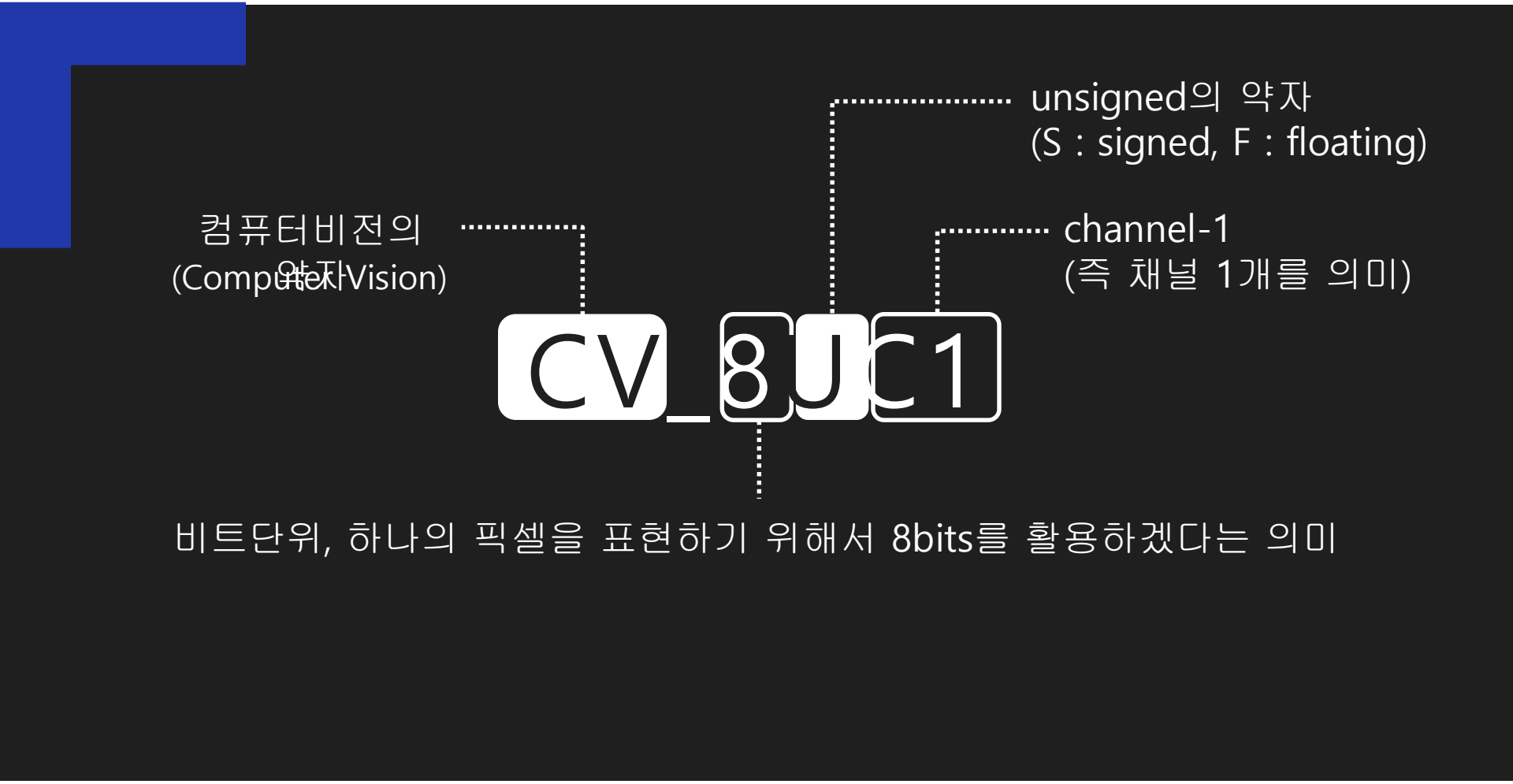
# 영상파일 읽기

## 주요 데이터 타입 상수

| | |
|---|---|
| CV_8U (8-bit Unsigned) | 8비트 부호 없는 정수형 데이터 타입으로, 각 픽셀 값은 0에서 255 사이의 값 |
| CV_8S (8-bit Signed) | 8비트 부호 있는 정수형 데이터 타입으로, 각 픽셀 값은 -128에서 127 사이의 값 |
| CV_16U (16-bit Unsigned) | 16비트 부호 없는 정수형 데이터 타입으로, 각 픽셀 값은 0에서 65,535 사이의 값 |
| CV_16S (16-bit Signed) | 16비트 부호 있는 정수형 데이터 타입으로, 각 픽셀 값은 -32,768에서 32,767 사이의 값 |
| CV_32S (32-bit Signed) | 32비트 부호 있는 정수형 데이터 타입으로, 매우 큰 숫자 범위를 다룰 수 있음 |
| CV_32F (32-bit Float) | 32비트 부동 소수점 데이터 타입으로, 실수형 데이터를 표현 |
| CV_64F (64-bit Double) | 64비트 부동 소수점 데이터 타입으로, 매우 높은 정밀도의 실수형 데이터를 표현 |

# 영상파일 읽기

## 주요 데이터 타입 상수



# 영상파일 읽기

```
import cv2

def print_matInfo(name, image):
    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))
```

# 영상파일 읽기

```
# 새로운 윈도우 이름
title1, title2 = "16bit Image", "32bit Image"

# 이미지 읽기
image16bit = cv2.imread("images/read_16.tif", cv2.IMREAD_UNCHANGED)
image32bit = cv2.imread("images/read_32.tif", cv2.IMREAD_UNCHANGED)

if image16bit is None or image32bit is None:
    raise Exception("영상파일 읽기 에러")
```

# 영상파일 읽기

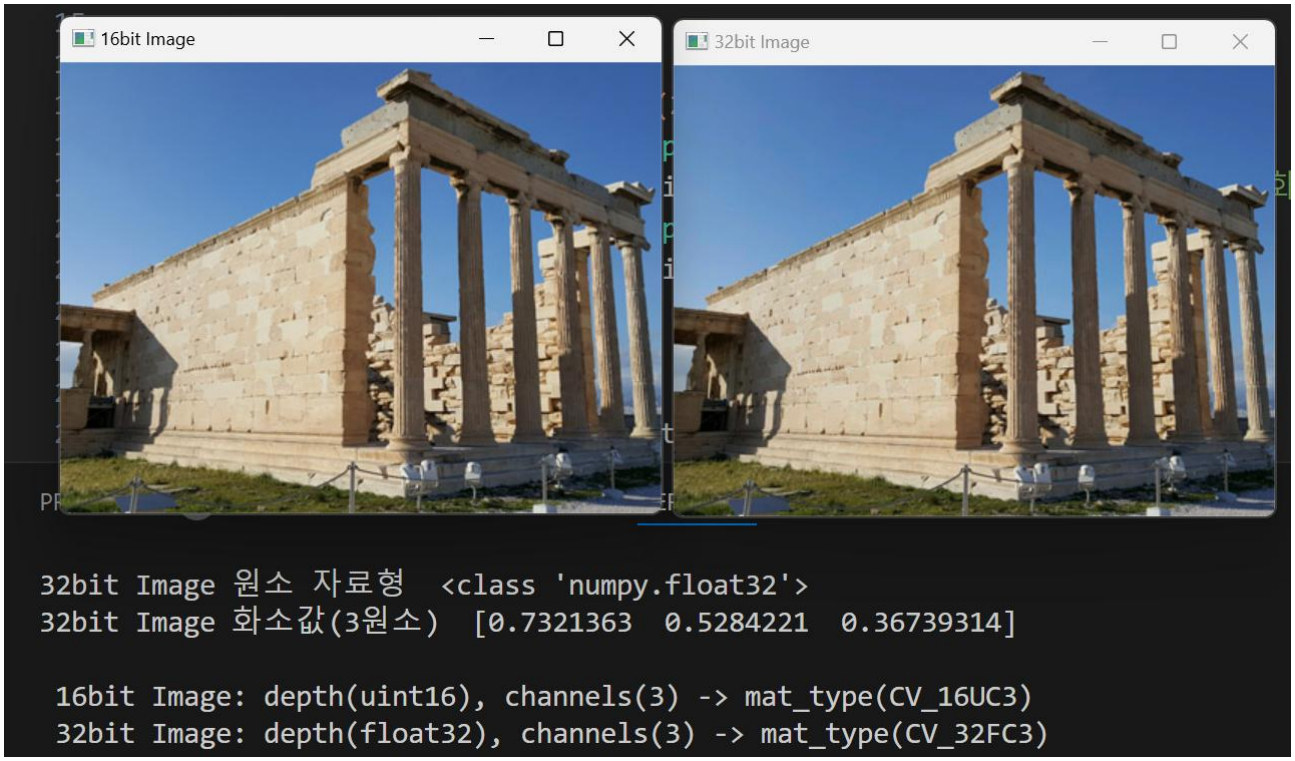
```
# 새로운 화소의 위치
pixel_pos = (20, 20)

# 화소 정보 출력
print("16/32비트 영상 행렬 좌표 (20, 20) 화소값")
print(title1, "원소 자료형 ", type(image16bit[pixel_pos][0]))  # 원소 자료형
print(title1, "화소값(3원소) ", image16bit[pixel_pos])  # 행렬 내 한 화소 값 표시
print(title2, "원소 자료형 ", type(image32bit[pixel_pos][0]))
print(title2, "화소값(3원소) ", image32bit[pixel_pos])
print()
```

# 영상파일 읽기

```
# 행렬 정보 출력
print_matInfo(title1, image16bit)
print_matInfo(title2, image32bit)

# 이미지 표시
cv2.imshow(title1, image16bit)
cv2.imshow(title2, (image32bit * 255).astype('uint8'))
cv2.waitKey(0)
cv2.destroyAllWindows()  # 모든 열린 윈도우 닫기
```

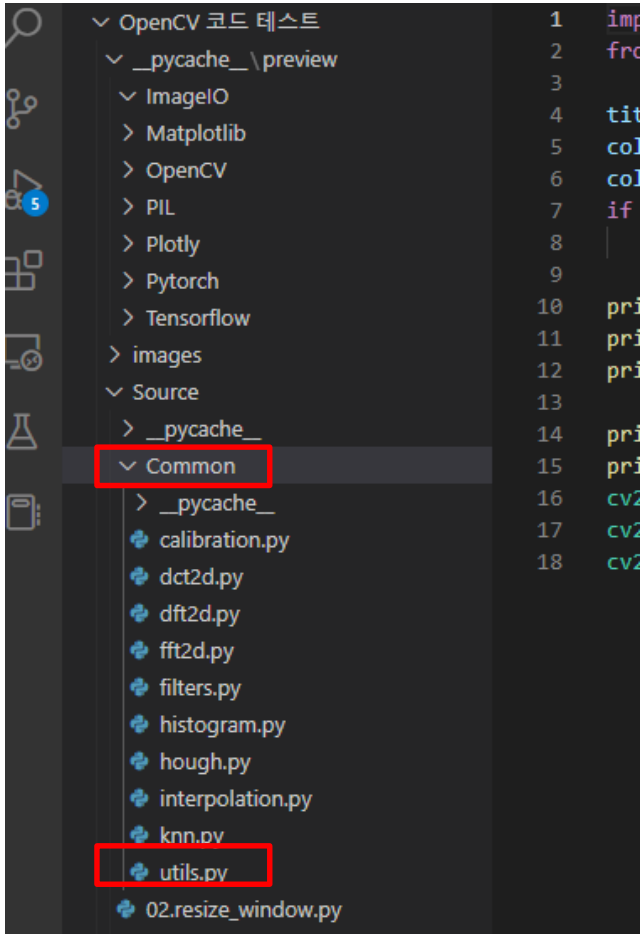
# 영상파일 읽기

- 실행결과


16bit Image
32bit Image
32bit Image 원소 자료형 <class 'numpy.float32'>
32bit Image 화소값(3원소) [0.7321363 0.5284221 0.36739314]
16bit Image: depth(uint16), channels(3) -> mat_type(CV_16UC3)
32bit Image: depth(float32), channels(3) -> mat_type(CV_32FC3)

# 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



# 행렬을 영상파일로 저장

## cv2.imwrite() 함수

- 행렬을 영상파일로 저장
- 확장자에 따라서 JPG, BMP, PNG, TIF, PPM 등의 영상파일 포맷 저장 가능

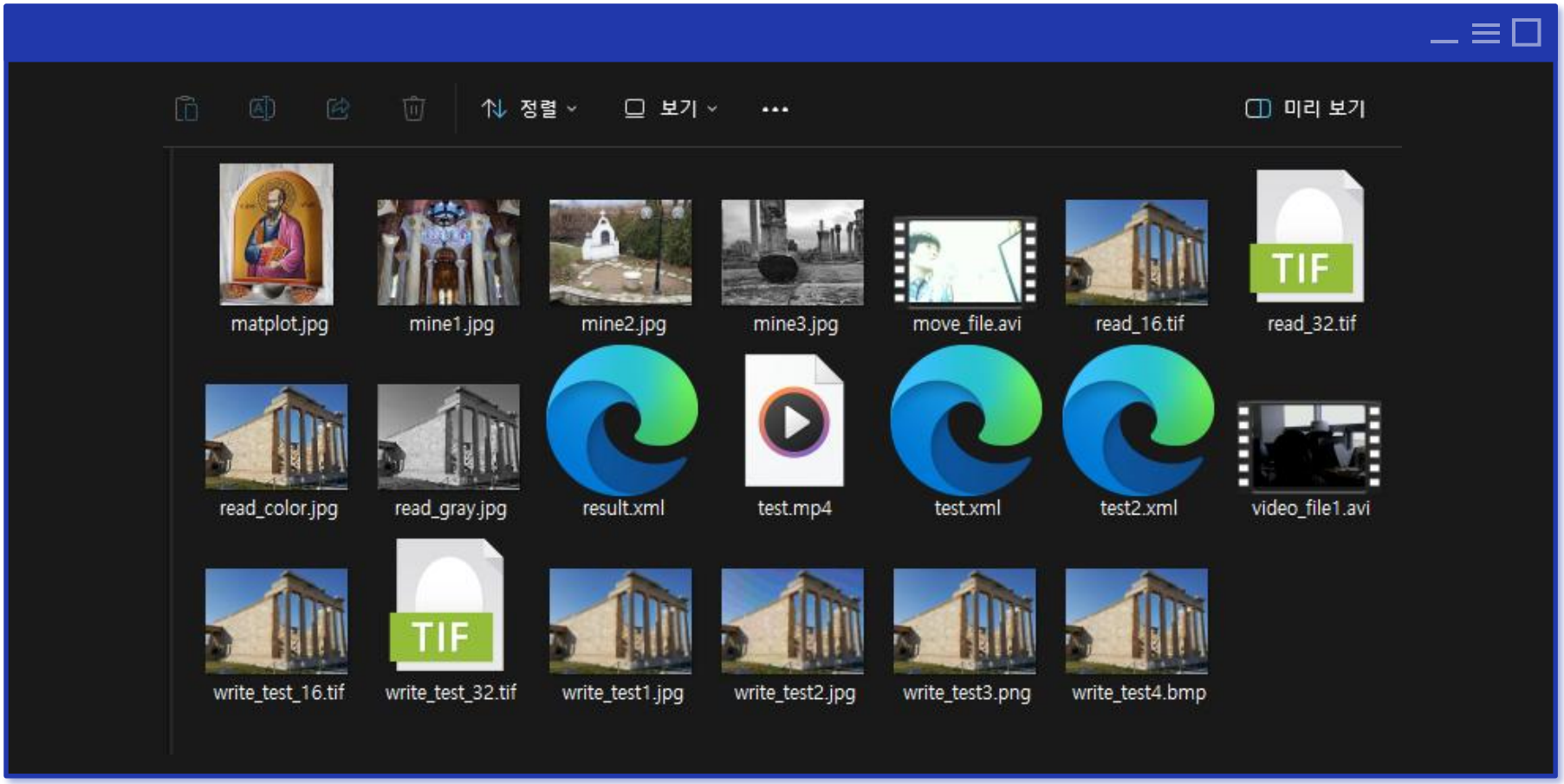
| | |
|---|---|
| JPEG | 'cv2.IMWRITE_JPEG_QUALITY'<br>JPEG 이미지의 품질을 설정(0-100 사이의 정수, 기본값: 95) |
| PNG | 'cv2.IMWRITE_PNG_COMPRESSION'<br>PNG 이미지의 압축 레벨을 설정(0-9 사이의 정수, 기본값: 3) |
| WEBP | 'cv2.IMWRITE_WEBP_QUALITY'<br>WEBP 이미지의 품질을 설정(0-100 사이의 정수, 기본값: 100) |

# 행렬을 영상파일로 저장

```
import cv2

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

# 새로운 JPEG 화질 설정 및 PNG 압축 레벨 설정
params_jpg = (cv2.IMWRITE_JPEG_QUALITY, 50)                    # JPEG 화질 설정
params_png = [cv2.IMWRITE_PNG_COMPRESSION, 5]                  # PNG 압축 레벨 설정

# 행렬을 영상파일로 저장
cv2.imwrite("output_images/high_quality.jpg", image)                    # 디폴트는 95
cv2.imwrite("output_images/medium_quality.jpg", image, params_jpg)      # 지정 화질로 저장
cv2.imwrite("output_images/medium_compression.png", image, params_png)
cv2.imwrite("output_images/image_output.bmp", image)   # BMP 파일로 저장

print("저장 완료")
```

# 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><br>cv2.VideoCapture(filename) → <VideoCapture object><br>cv2.VideoCapture(device) → <VideoCapture object><br>■ 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<br>cv2.VideoCapture.open(device) → retval<br>■ 설명: 동영상 캡처를 위한 동영상파일 혹은 캡처 장치를 개방한다. | | |
| cv2.VideoCapture.isOpened() → retval<br>■ 설명: 캡처 장치의 연결 여부를 반환한다. | | |
| cv2.VideoCapture.release() → None<br>■ 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.

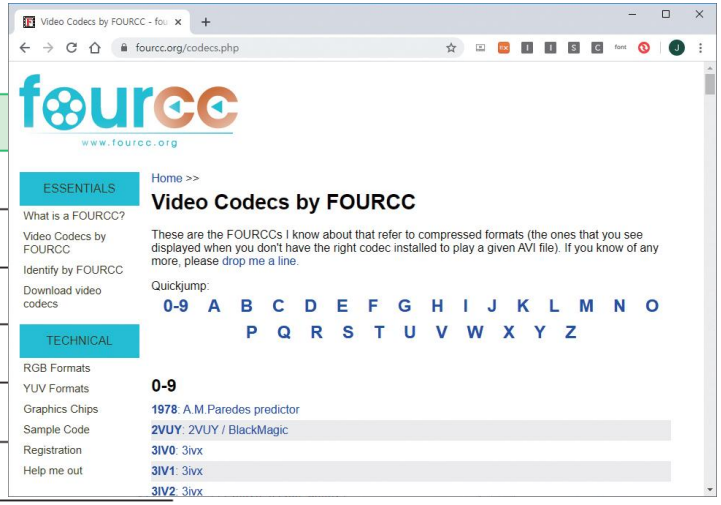
# 4.5 비디오 처리

| VideoWriter 클래스 설명 | | |
|---|---|---|
| cv2.VideoWriter([filename, fourcc, fps, frameSize[, isColor]]) → <VideoWriter object><br>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<br>■ 설명: 캡처 장치나 동영상파일이 열려있는지 확인한다. | | |
| cv2.VideoWriter.write(image) → None<br>■ 설명: image 프레임을 파일로 저장한다. | | |
| cv2.VideoWriter.open()<br>■ 설명: 영상을 동영상파일의 프레임으로 저장하기 위해 동영상파일을 개방한다. 인수는 생성자의 인수와 동일하다. | | |
| cv2.VideoWriter.isOpened()<br>■ 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')<br>cv2.VideoWrite_fourcc(*'DIV4') | DivX MPEG-4 |
| cv2.VideoWriter_fourcc('D', 'I', 'V', '5')<br>cv2.VideoWrite_fourcc(*'DIV5') | Div5 |
| cv2.VideoWriter_fourcc('D', 'I', 'V', 'X')<br>cv2.VideoWrite_fourcc(*'DIVX') | DivX |
| cv2.VideoWriter_fourcc('D', 'X', '5', '0')<br>cv2.VideoWrite_fourcc(*'DX50') | DivX MPEG-4 |
| cv2.VideoWriter_fourcc('F', 'M', 'P', '4')<br>cv2.VideoWrite_fourcc(*'FMP4') | FFMpeg |
| cv2.VideoWriter_fourcc('I', 'Y', 'U', 'V')<br>cv2.VideoWrite_fourcc(*'TYUV') | IYUV |
| cv2.VideoWriter_fourcc('M', 'J', 'P', 'G')<br>cv2.VideoWrite_fourcc(*'MJPG') | Motion JPEG codec |
| cv2.VideoWriter_fourcc'M', 'P', '4', '2')<br>cv2.VideoWrite_fourcc(*'MP42') | MPEG4 v2 |
| cv2.VideoWriter_fourcc('M', 'P', 'E', 'G')<br>cv2.VideoWrite_fourcc(*'MPEG') | MPEG codecs |
| cv2.VideoWriter_fourcc('X', 'V', 'I', 'D')<br>cv2.VideoWrite_fourcc(*'XVID') | XVID codecs |
| cv2.VideoWriter_fourcc('X', '2', '6', '4')<br>cv2.VideoWrite_fourcc(*'X264') | H.264/AVC codecs |
| -1 | 코덱 선택 대화상자 띄움 |

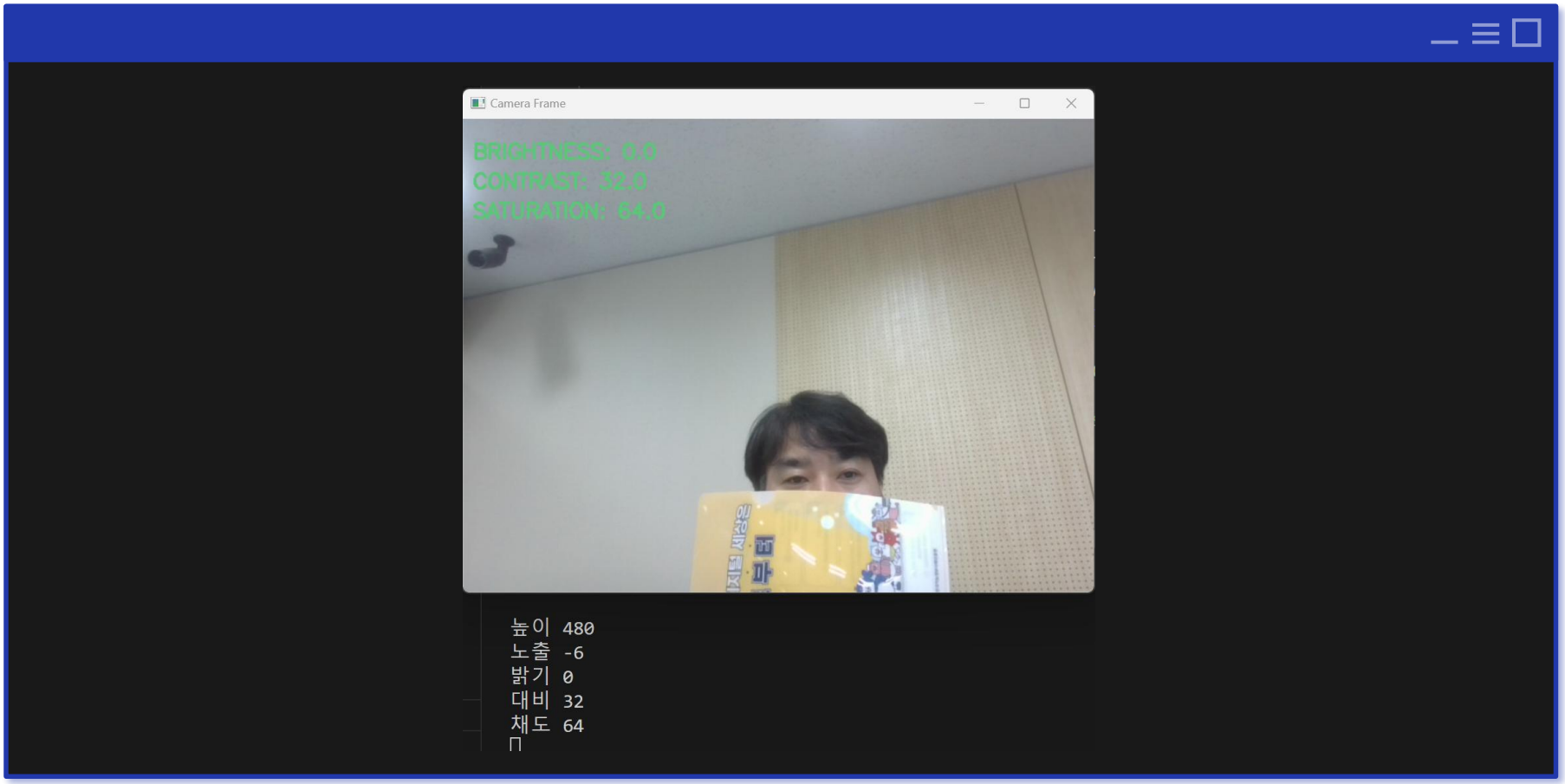
# 카메라에서 프레임 읽기

```
import cv2

def display_info(frame, text, pt, value, color=(120, 200, 90)):
    text += str(value)
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(frame, text, pt, font, 0.7, color, 2)

capture = cv2.VideoCapture(0)  # 0번 카메라 연결
if not capture.isOpened():
    raise Exception("카메라 연결 실패")

# 카메라 속성 획득 및 출력
print("너비 %d" % capture.get(cv2.CAP_PROP_FRAME_WIDTH))
print("높이 %d" % capture.get(cv2.CAP_PROP_FRAME_HEIGHT))
print("노출 %d" % capture.get(cv2.CAP_PROP_EXPOSURE))
print("밝기 %d" % capture.get(cv2.CAP_PROP_BRIGHTNESS))
print("대비 %d" % capture.get(cv2.CAP_PROP_CONTRAST))
print("채도 %d" % capture.get(cv2.CAP_PROP_SATURATION))
```

# 카메라에서 프레임 읽기

```
while True:  # 무한 반복
    ret, frame = capture.read()  # 카메라 영상 받기
    if not ret:
        break
    if cv2.waitKey(30) >= 0:
        break

    brightness = capture.get(cv2.CAP_PROP_BRIGHTNESS)
    contrast = capture.get(cv2.CAP_PROP_CONTRAST)
    saturation = capture.get(cv2.CAP_PROP_SATURATION)
    display_info(frame, "BRIGHTNESS: ", (10, 40), brightness)
    display_info(frame, "CONTRAST: ", (10, 70), contrast)
    display_info(frame, "SATURATION: ", (10, 100), saturation)
    title = "Camera Frame"
    cv2.imshow(title, frame)  # 윈도우에 영상 띄우기

capture.release()
cv2.destroyAllWindows()  # 모든 열린 윈도우 닫기
```

# 카메라에서 프레임 읽기


Camera Frame
BRIGHTNESS: 0.0
CONTRAST: 32.0
SATURATION: 64.0
높이 480
노출 -6
밝기 0
대비 32
채도 64

# 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
```

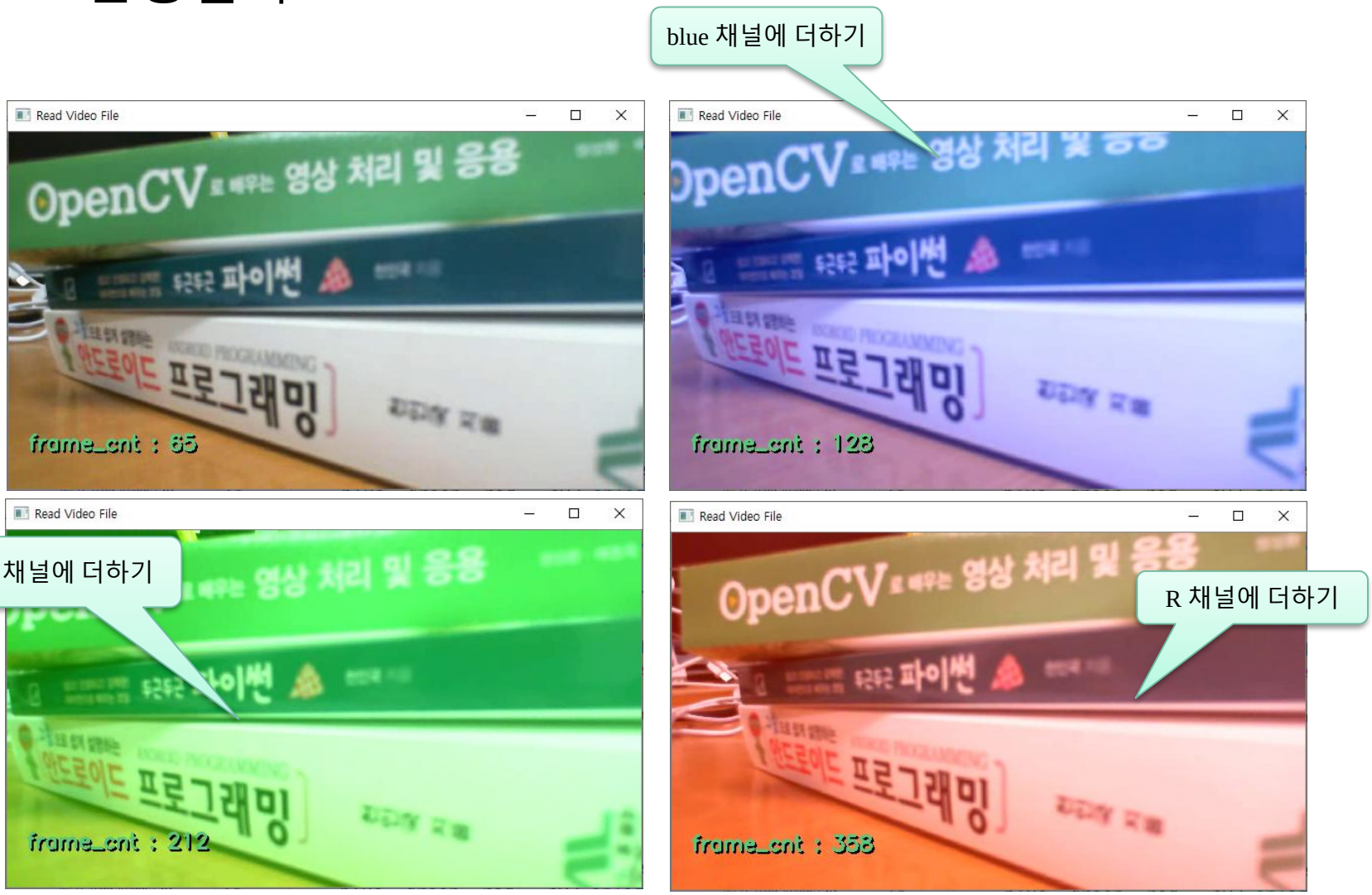
```
11  while True:
12      ret, frame = capture.read()
13      if not ret or cv2.waitKey(delay) >= 0: break       # 프레임 간 지연 시간 지정

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.**