AI TRAINING SERIES

Day 05

실습

GStreamer 실습

GStreamer 로 카메라 프리뷰하기

1. GStreamer 및 필수 플러그인 설치

```
$ sudo apt install -y \
  gstreamer1.0-tools \
  gstreamer1.0-plugins-base \
  gstreamer1.0-plugins-good \
  gstreamer1.0-plugins-bad \
  gstreamer1.0-plugins-ugly \
  gstreamer1.0-libav
```

2. 설치 확인

```
# 설치된 GStreamer 버전 확인
$ gst-launch-1.0 --version
---------------------------------------
# 정상출력 결과 (version 은 다를 수 있음)
gst-launch-1.0 version 1.22.0
GStreamer 1.22.0
https://tracker.debian.org/pkg/gstreamer1.0
```

3. GStreamer 플러그인 확인

```
# GStreamer 플러그인 중 v4l2src 플러그인 확인
$ gst-inspect-1.0 v4l2src
```

4. 간단한 프리뷰 파이프라인 실행

```
$ gst-launch-1.0 -v \
  v4l2src device=/dev/video0 ! \
  videoconvert ! \
  autovideosink
```

5. 카메라 인풋 format 을 MJPEG 으로 변경하기

```
$ gst-launch-1.0 -v \
  v4l2src device=/dev/video0 ! \
    image/jpeg, width=1280, height=720, framerate=30/1 ! \
  jpegdec ! \
  videoconvert ! \
  autovideosink
```

v4l-utils 실습

1. v4l-utils 설치하자. 설치하는 법 (터미널 창을 열고 아래 명령어 실행)

```
$ sudo apt install v4l-utils -y
```

2. USB webcam 을 Target Device 에 연결하자.

3. USB webcam 이 잘 인식되어 UVC kernel 드라이버에서 정상적으로 올라왔는지 확인

```
$ ls /dev/video*
```

4. 정상적으로 로드 됐으면 아래와 같이 출력되며 여기서 video0 장치가 주로 사용되는 장치임

```
$ /dev/video0    /dev/video1
```

5. 아래 명령어들을 실행하며 어떤 기능인지 확인해 보자

```
# 지원되는 이미지 포멧, 해상도, FPS 정보를 확인
$ v4l2-ctl --device=/dev/video0 --list-formats-ext
```

```
# 지원되는 Tuning Parameters 를 확인
$ v4l2-ctl --device=/dev/video0 --list-ctrls-menus
```

```
# 튜닝 파라미터를 읽어오기 (예: 밝기)
$ v4l2-ctl --device=/dev/video0 --get-ctrl brightness
```

```
# 튜닝 파라미터를 쓰기 (예: 밝기)
$ v4l2-ctl --device=/dev/video0 –set-ctrl brightness=0
```

v4l-utils 실습


```
$ v4l2-ctl --device=/dev/video0 --list-ctrls-menus
```

```
User Controls

                     brightness 0x00980900 (int)    : min=0 max=255 step=1 default=128 value=200
                       contrast 0x00980901 (int)    : min=0 max=255 step=1 default=32 value=32
                     saturation 0x00980902 (int)    : min=0 max=255 step=1 default=32 value=32
        white_balance_automatic 0x0098090c (bool)   : default=1 value=1
                           gain 0x00980913 (int)    : min=0 max=255 step=1 default=64 value=192
              power_line_frequency 0x00980918 (menu)   : min=0 max=2 default=2 value=2 (60 Hz)
                                                         0: Disabled
                                                         1: 50 Hz
                                                         2: 60 Hz
          white_balance_temperature 0x0098091a (int)    : min=0 max=10000 step=10 default=4000 value=5230 flags=inactive
                         sharpness 0x0098091b (int)    : min=0 max=255 step=1 default=24 value=24
            backlight_compensation 0x0098091c (int)    : min=0 max=1 step=1 default=0 value=0

Camera Controls

                  auto_exposure 0x009a0901 (menu)   : min=0 max=3 default=3 value=3 (Aperture Priority Mode)
                                                        1: Manual Mode
                                                        3: Aperture Priority Mode
              exposure_time_absolute 0x009a0902 (int)    : min=1 max=10000 step=1 default=166 value=31 flags=inactive
          exposure_dynamic_framerate 0x009a0903 (bool)   : default=0 value=1
```

밝기: 화면 전체 밝기 조절. 값이 클수록 영상이 밝아짐. (Digital gain)

대비: 밝은 부분과 어두운 부분의 차이를 조절. 높이면 뚜렷해짐

채도: 색상의 선명도 조절. 낮추면 흑백에 가까워지고, 높이면 색상이 더 진해짐

자동 색온도 조절: 자동으로 색온도를 맞춰줌. 켜져 있으면 카메라가 주변 조명에 따라 색을 자동 조정.

밝기: 영상 신호 증폭 정도. 낮추면 노이즈 적지만 어두워지고, 높이면 밝지만 노이즈 증가 (Analog gain)

전원주파수: 조명 깜박임 방지를 위한 주파수 설정. 지역에 따라 50Hz (유럽) 또는 60Hz (미국,한국) 설정

선명도: 영상의 선명함 정도 조절. 높으면 경계가 더 뚜렷해짐

역광보정: 밝은 배경에서 인물 등 피사체가 너무 어둡지 않도록 밝기 자동 보정

노출 자동모드: 노출 자동 조절 모드 선택

- Manual Mode: 직접 노출값 조절
- Aperture Priority Mode: 조리개 우선 자동 조절

노출 시간 조절. Manual 모드에서만 사용 가능

v4l-utils 실습

1. AWB (Auto White Balance) 기능을 끄고 Manual 하게 색온도를 변경해 보자

```
# 값이 낮을수록 푸른 빛 (차가운 색), 높을수록 붉은 빛 (따뜻한 색)
$ v4l2-ctl --device=/dev/video0 --set-ctrl white_balance_automatic=0
$ v4l2-ctl --device=/dev/video0 --set-ctrl white_balance_temperature=100
$ v4l2-ctl --device=/dev/video0 --set-ctrl white_balance_temperature=10000
```

2. Sharpness 값을 min 값과 max 값을 써서 비교해 보자

```
# 값이 높으면 경계가 더 뚜렷해 짐
$ v4l2-ctl --device=/dev/video0 --set-ctrl sharpness=0
$ v4l2-ctl --device=/dev/video0 --set-ctrl sharpness=255
```

3. AE (Auto Exposure) 기능을 끄고 Manual 하게 Exposure 변경해 보자

```
# 값이 높으면 노출 시간이 길어져 빛을 더 많이 받아 밝아지나 FPS 떨어질 수 있음
$ v4l2-ctl --device=/dev/video0 --set-ctrl auto_exposure=1
$ v4l2-ctl --device=/dev/video0 --set-ctrl exposure_time_absolute=0
$ v4l2-ctl --device=/dev/video0 --set-ctrl exposure_time_absolute=300
$ v4l2-ctl --device=/dev/video0 --set-ctrl exposure_time_absolute=1000
```

v4l-utils 실습

4. 카메라 Preview 를 Gstreamer 를 사용해 키고 아래 설정들을 바로 적용시키는 script 를 만들기

- auto_exposure 기능 끄기 (manual mode)
- exposure_time_absolute 값 150 으로 설정
- gain 값 100 으로 설정
- Contrast 값 50 으로 설정

```
#!/bin/bash

# 1. auto_exposure 끄기 (Manual Mode: 1)
v4l2-ctl -d /dev/video0 --set-ctrl=auto_exposure=1

# 2. exposure_time_absolute 값 150 으로 설정
v4l2-ctl -d /dev/video0 --set-ctrl=exposure_time_absolute=150

# 3. gain 값 100 으로 설정
v4l2-ctl -d /dev/video0 --set-ctrl=gain=100

# 4. contrast 값 50 으로 설정
v4l2-ctl -d /dev/video0 --set-ctrl=contrast=50

# 5. gst-launch-1.0 실행 (카메라 스트림 출력)
gst-launch-1.0 -v \
  v4l2src device=/dev/video0 ! \
  image/jpeg,width=1280,height=720,framerate=30/1 ! \
  jpegdec ! \
  videoconvert ! \
  autovideosink
```

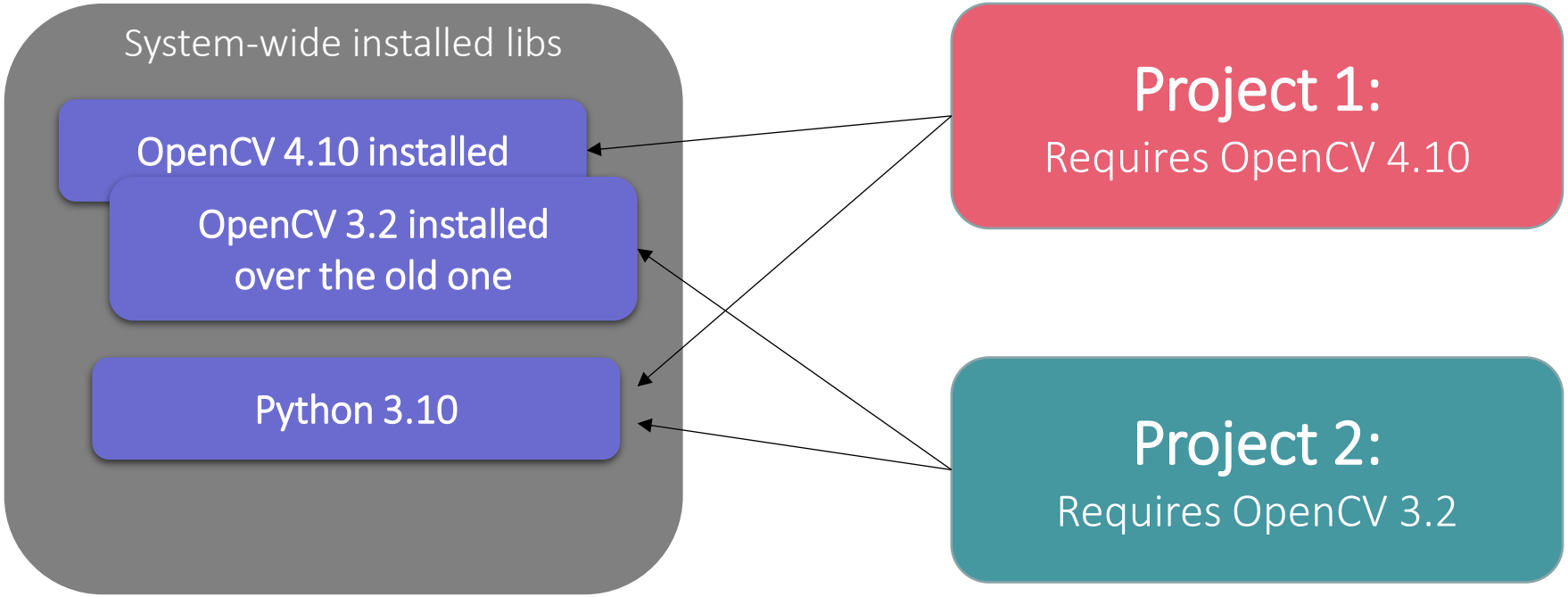
OpenCV 실습

Python 가상환경에 대해

Problem ⇉ Dependency Conflicts

왜 가상 환경 사용을 권장하나?

Project 1 의 개발환경을 설치 후 프로젝트를 수행하다가 Project 2 의 개발환경을 설치하면 OpenCV 버전 conflict 발생!!



OpenCV 실습

Python 가상환경 venv 로 만들어 opencv-python 설치하기

```
# venv 로 python 가상환경을 생성, 가상환경 이름을 venv_opencv 로 선언
python -m venv venv_opencv

# 생성한 가상환경 activate
source venv_opencv/bin/activate

# pip 를 최신으로 upgrade
pip install -U pip

# OpenCV 를 python 에서 사용하기 위한 package 설치
pip install opencv-python
```

OpenCV 실습

1. OpenCV 를 활용해 1280x720, 3 Channel(BGR) 포멧의 빈 스케치북을 만들자.

```python
import cv2
import numpy as np

# 1. 720p 해상도의 빈 스케치북 (검정 바탕)
height, width = 720, 1280
sketchbook = np.zeros((height, width, 3), dtype=np.uint8) # 초기값은 모두 0, 즉 검정색

window_name = "스케치북" # 윈도우 이름 설정
cv2.namedWindow(window_name, cv2.WINDOW_NORMAL) # OpenCV 창 생성

# 1. 검정색으로 화면 보여주기
cv2.imshow(window_name, sketchbook)
cv2.setWindowTitle(window_name, "검정색 스케치북")
cv2.waitKey(1000)

# 2. 흰색으로 화면 전체 칠하기
sketchbook[:] = (255, 255, 255) # BGR 순서
cv2.imshow(window_name, sketchbook)
cv2.setWindowTitle(window_name, "흰색 스케치북")
cv2.waitKey(1000) # 1초 대기

# 3. 빨간색으로 화면 전체 칠하기
sketchbook[:] = (0, 0, 255) # 빨간색 (OpenCV는 BGR)
cv2.imshow(window_name, sketchbook)
cv2.setWindowTitle(window_name, "빨간색 스케치북")
cv2.waitKey(1000)

# 4. 종료
cv2.destroyAllWindows()
```

OpenCV 실습

2. 직선, 원, 사각형, 다각형 도형을 빈 스케치북 위에 각각 자유로운 크기와 위치에 그려보자

```
cv2.line(img, pt1, pt2, color, thickness)
```

- img: 그릴 대상 이미지 (NumPy 배열)
- pt1, pt2: 시작점과 끝점 좌표 (튜플)
- color: 선 색상 (B, G, R)
- thickness: 선 두께 (픽셀 단위)
- 예: cv2.line(img, (100, 100), (400, 100), (0, 255, 0), 3) # 녹색 선

```
cv2.circle(img, center, radius, color, thickness)
```

- img: 그릴 대상 이미지 (NumPy 배열)
- center: 중심 좌표 (튜플)
- radius: 반지름
- thickness: -1이면 내부 채움
- 예: cv2.circle(img, (640, 360), 50, (0, 0, 255), 5) # 빨간 테두리 원

```
cv2.rectangle(img, pt1, pt2, color, thickness)
```

- img: 그릴 대상 이미지 (NumPy 배열)
- pt1: 왼쪽 위 꼭짓점
- pt2: 오른쪽 아래 꼭짓점
- thickness: 선 두께. -1이면 내부 채움
- cv2.rectangle(img, (100, 200), (300, 400), (255, 0, 0), -1) # 파란색 채운 사각형

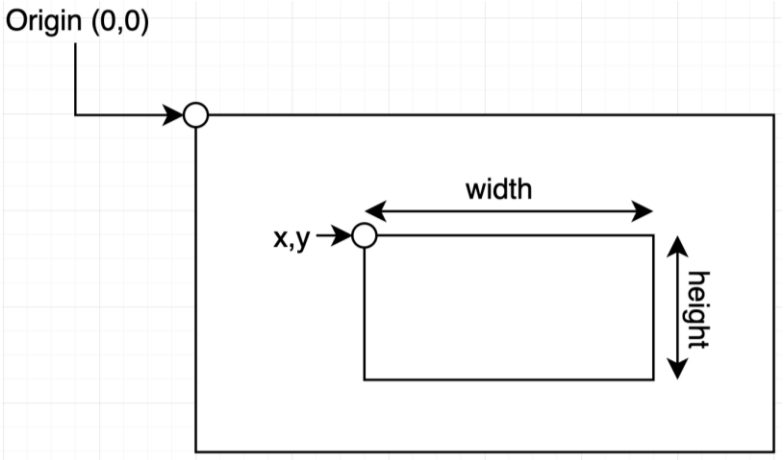
```
cv2.polylines(img, [pts], isClosed, color, thickness)
```

- img: 그릴 대상 이미지 (NumPy 배열)
- pts: 꼭짓점 좌표들의 NumPy 배열 (정수형, 2차원)
- isClosed: True이면 시작점과 끝점 자동 연결
- color, thickness: 색상과 두께
- 예: pts = np.array([[100, 100], [200, 300], [400, 200]], np.int32)
  cv2.polylines(img, [pts], isClosed=True, color=(255, 255, 0), thickness=2)

OpenCV 실습

2. 직선, 원, 사각형, 다각형 도형을 빈 스케치북 위에 각각 자유로운 크기와 위치에 그려보자 (cont.)

```python
import cv2
import numpy as np

# 1. 720p 해상도의 빈 스케치북 (검정 바탕)
height, width = 720, 1280
sketchbook = np.zeros((height, width, 3), dtype=np.uint8) # 초기값은 모두 0, 즉 검정색

# 2. 직선 그리기 - 시작점 (100, 100) → 끝점 (400, 300), 색상은 초록색, 두께는 3픽셀
cv2.line(sketchbook, (100, 100), (400, 300), color=(0, 255, 0), thickness=3)

# 3. 원 그리기 - 중심점 (600, 150), 반지름 60, 색상은 빨간색, 두께는 5픽셀
cv2.circle(sketchbook, center=(600, 150), radius=60, color=(0, 0, 255), thickness=5)

# 4. 사각형 그리기 - 왼쪽 위 (800, 200) → 오른쪽 아래 (1100, 400), 색상은 파란색, 채워진 형태
cv2.rectangle(sketchbook, (800, 200), (1100, 400), color=(255, 0, 0), thickness=-1)

# 5. 다각형 그리기 (삼각형 예제) - 꼭짓점 좌표 리스트 정의
pts = np.array([[500, 500], [700, 600], [600, 700]], np.int32)
cv2.polylines(sketchbook, [pts], isClosed=True, color=(255, 255, 0), thickness=4)

# 6. 결과 출력
cv2.namedWindow("스케치북", cv2.WINDOW_NORMAL) # 크기 조절 가능한 창 생성
cv2.resizeWindow("스케치북", height, width) # 창 크기 강제 설정
cv2.imshow("스케치북", sketchbook)
cv2.waitKey(0) # 별도의 키 입력이 있을때까지 imshow 로 생성된 창을 열어둠
cv2.destroyAllWindows()
```



< OpenCV 에서 2D 이미지 좌표 개념 >

OpenCV 실습

3. Text 를 출력해 보자 (English only, 한글은 다른 방식 필요)

```
cv2.putText(img, text, org, fontFace, fontScale, color, thickness, lineType)
```

- img: 그릴 대상 이미지 (NumPy 배열)
- text: 출력할 문자열
- org: 문자열이 시작될 좌표 (왼쪽 아래 기준) – e.g., (x, y)
- fontFace: 폰트 종류
- fontScale: 폰트 크기 배율 (1.0 = 기본 크기)
- color: 글자 색상 (B, G, R)
- thickness: 글자 두께 (선 굵기)
- lineType: 선 스타일
- 예: cv2.putText(img, 'Hello OpenCV', (50, 150), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, 0), 2, cv2.LINE_AA)

```
cv2.FONT_HERSHEY_SIMPLEX        # 가장 일반적인 산세리프
cv2.FONT_HERSHEY_PLAIN          # 얇고 단순한 폰트
cv2.FONT_HERSHEY_DUPLEX         # SIMPLEX보다 두껍고 선명
cv2.FONT_HERSHEY_COMPLEX        # 복잡한 글꼴
cv2.FONT_HERSHEY_TRIPLEX        # COMPLEX보다 두꺼운 느낌
cv2.FONT_HERSHEY_COMPLEX_SMALL  # 작은 사이즈의 복잡한 글꼴
cv2.FONT_HERSHEY_SCRIPT_SIMPLEX # 필기체 느낌
cv2.FONT_HERSHEY_SCRIPT_COMPLEX # 복잡한 필기체
cv2.FONT_ITALIC                 # 이탤릭체는 다른 폰트와 조합 사용
```

```
cv2.LINE_8  # 8-connected line (기본 선, 가장 일반적인 방식)
cv2.LINE_AA # Antialiased line (부드러운 선, 텍스트나 곡선에 추천)
```

OpenCV 실습

3. Text 를 모든 폰트로 출력해 보자 (English only, 한글은 다른 방식 필요)

```python
import cv2
import numpy as np

# 캔버스 생성 (배경: 흰색)
img = np.ones((900, 1280, 3), dtype=np.uint8) * 255
font_defs = [# 폰트와 이름
    (cv2.FONT_HERSHEY_SIMPLEX, "FONT_HERSHEY_SIMPLEX"),
    (cv2.FONT_HERSHEY_PLAIN, "FONT_HERSHEY_PLAIN"),
    (cv2.FONT_HERSHEY_DUPLEX, "FONT_HERSHEY_DUPLEX"),
    (cv2.FONT_HERSHEY_COMPLEX, "FONT_HERSHEY_COMPLEX"),
    (cv2.FONT_HERSHEY_TRIPLEX, "FONT_HERSHEY_TRIPLEX"),
    (cv2.FONT_HERSHEY_COMPLEX_SMALL, "FONT_HERSHEY_COMPLEX_SMALL"),
    (cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, "FONT_HERSHEY_SCRIPT_SIMPLEX"),
    (cv2.FONT_HERSHEY_SCRIPT_COMPLEX, "FONT_HERSHEY_SCRIPT_COMPLEX")
]
colors = [ # 폰트 색
    (255, 0, 0), (0, 128, 0), (0, 0, 255), # Blue / Green / Red
    (255, 128, 0), (128, 0, 128), (0, 255, 255), # Orange / Purple / Cyan
    (255, 0, 255), (0, 0, 0)  # Magenta / Black
]

y = 60
for i, (font, name) in enumerate(font_defs):
    color = colors[i % len(colors)]
    # 일반 버전
    cv2.putText(img, f"{name}", (50, y), font, 1.0, color, 2, cv2.LINE_AA)
    # 이탤릭 버전 (OR 연산)
    italic_font = font | cv2.FONT_ITALIC
    cv2.putText(img, f"{name} + ITALIC", (650, y), italic_font, 1.0, color, 2, cv2.LINE_AA)
    y += 80

cv2.imshow("Font vs Italic Font", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OpenCV 실습

4. 이미지를 로드하고 ‘s’ 키 입력 시 파일로 저장하기

```
cv2.imread(filename, flags=IMREAD_COLOR)
```

- filename: 불러올 이미지 경로 (예: "image.jpg")
- flags: 읽는 방식 지정

```
cv2.IMREAD_COLOR      # (1, 기본값) - 컬러 이미지로 읽음 (알파 채널 무시)
cv2.IMREAD_GRAYSCALE # (0) - 흑백 이미지로 읽음
cv2.IMREAD_UNCHANGED # (-1) - 원본 유지 (알파 값 포함)
```

```
cv2.imwrite(filename, img[, params])
```

- filename: 저장할 파일 경로 (예: "output.jpg")
- img: 저장할 이미지 (numpy 배열)
- params(optional): 포멧별 저장 옵션 (예: JPEG 압축률)
- 예: cv2.imwrite("output.jpg", img, [cv2.IMWRITE_JPEG_QUALITY, 90]) # 0~100
- 예: cv2.imwrite("output.png", img, [cv2.IMWRITE_PNG_COMPRESSION, 3]) # 0~9

```
cv2.imshow(winname, mat)
```

- winname: 창 이름 (예: "My Image")
- mat: 저장할 이미지 (numpy 배열)

```
cv2.waitKey(delay)
```

- 키 값을 기다리며 창을 유지시킴
- delay: 대기 시간 (ms 단위, 0이면 무한 대기)
- return: 입력된 키보드의 ASCII 코드(int)

OpenCV 실습

4. 이미지를 로드하고 ‘s’ 키 입력 시 파일로 저장하기

```python
import cv2

# 이미지 읽기 - 파일과 경로는 각자 맞게 수정해야 함
img = cv2.imread("res/card.jpg")

# 이미지가 잘 로드 됐는지 확인
if img is None:
    print("이미지를 불러올 수 없습니다.")
    exit()

# 이미지 보여주기
cv2.imshow("Loaded Image", img)

# 키 입력 대기 (무한 대기)
key = cv2.waitKey(0)

# 's' 키를 누르면 이미지 저장
if key == ord('s'): # ord 는 ASCII(아스키) 코드 값을 정수로 반환하는 Python 내장 함수
    cv2.imwrite("saved_image.jpg", img)
    print(“Saved as a saved_image.jpg file!!")

cv2.destroyAllWindows()
```

OpenCV 실습

5. 이미지 크기를 2배 scale up / down 하고 좌우 대칭, 좌우 90도 회전을 해보자

```
cv2.resize(src, dsize, fx, fy, interpolation)
```

- src: 입력 이미지
- dsize: 출력 이미지 크기 ((width, height) 튜플)
- fx, fy: x, y 방향의 비율 (dsize 대신 사용 가능) 지정
- interpolation: 보간 방법 (확대/축소 시 품질 결정)
- 예: resized = cv2.resize(img, None, fx=2.0, fy=2.0, interpolation=cv2.INTER_LINEAR) # 2배 확대

```
cv2.INTER_NEAREST    # 최근접 이웃 (가장 빠름, 품질 낮음)
cv2.INTER_LINEAR     # 기본값, 양선형 보간 (보통 확대에 사용)
cv2.INTER_AREA       # 축소에 추천, 영역 보간
```

```
cv2.flip(src, flipCode)
```

- src: 입력 이미지
- flipCode: 1 -> 좌우반전, 0 -> 상하반전, -1 -> 상하좌우반전
- 예: flipped = cv2.flip(img, 1) # 좌우반전

```
cv2.rotate(src, rotateCode)
```

- src: 입력 이미지
- rotateCode: 회전 방향
- 예: rotated = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE) # 90도 시계방향 회전

```
cv2.ROTATE_90_CLOCKWISE         # 90도 시계방향
Cv2.ROTATE_90_COUNTERCLOCKWISE  # 90도 반시계 방향
cv2.ROTATE_180                  # 180도
```

OpenCV 실습

5. 이미지 크기를 2배 scale up / down 하고 좌우 대칭, 좌우 90도 회전을 해보자

```python
import cv2

# 이미지 로드
img = cv2.imread("res/card.jpg")

# 이미지 로드 실패 시 종료
if img is None:
    print("이미지를 불러올 수 없습니다.")
    exit()

# 1. 이미지 크기 확대 (2배)
scaled_up = cv2.resize(img, None, fx=2.0, fy=2.0, interpolation=cv2.INTER_LINEAR)
# 2. 이미지 크기 축소 (1/2배)
scaled_down = cv2.resize(img, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_AREA)
# 3. 좌우 대칭 (flipCode=1)
flipped = cv2.flip(img, 1)
# 4. 90도 회전 (시계 방향)
rotated_90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)

# 결과 이미지 보기
cv2.imshow("Original", img)
cv2.imshow("Scaled Up x2", scaled_up)
cv2.imshow("Scaled Down x0.5", scaled_down)
cv2.imshow("Flipped (Left-Right)", flipped)
cv2.imshow("Rotated 90 Clockwise", rotated_90)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

THANK YOU