# 임베디드 보드 실습과 응용 프로그램

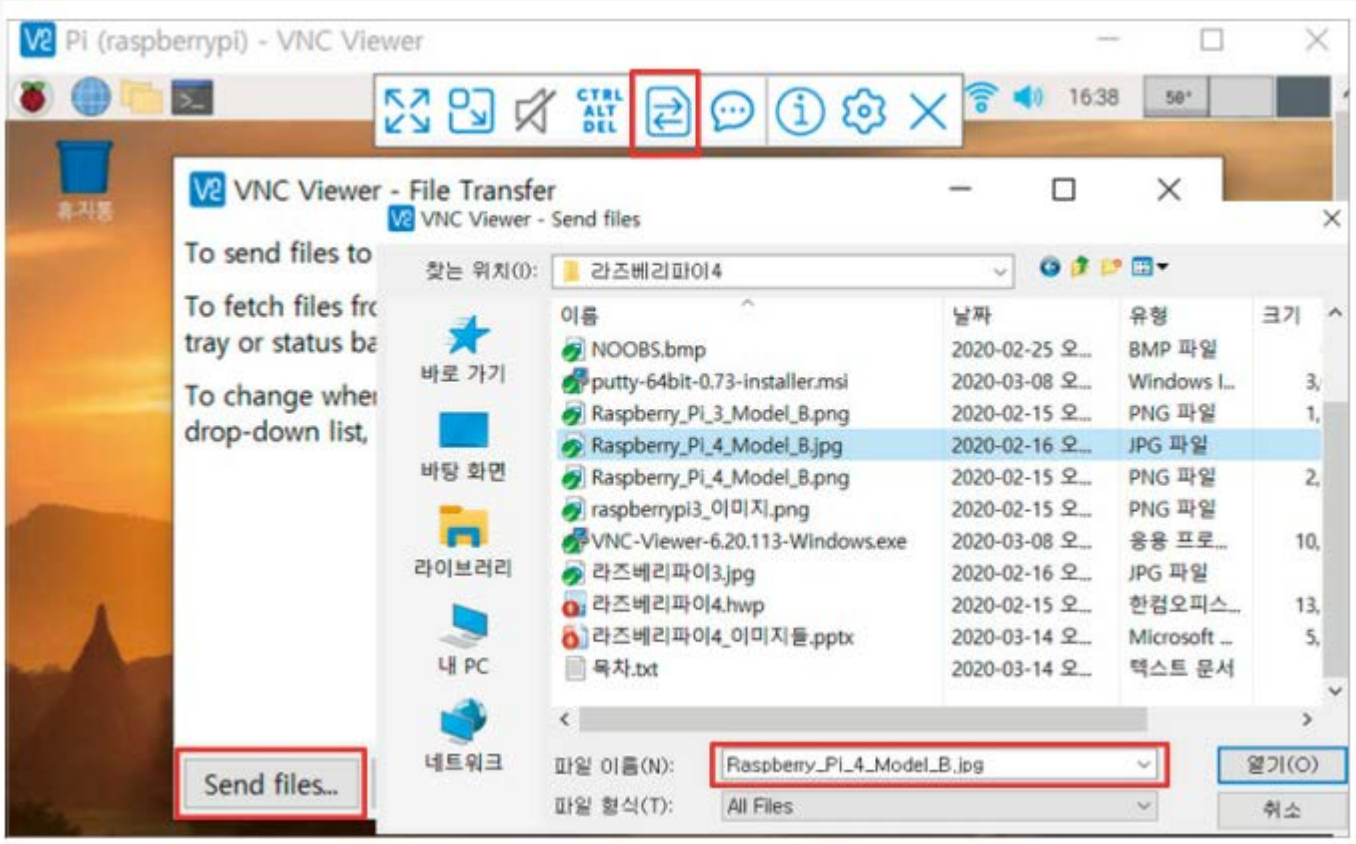
## Chapter 6.

**인공지능 라이브러리 활용하기**
**(얼굴 인식)**

최영근
010-5898-3202

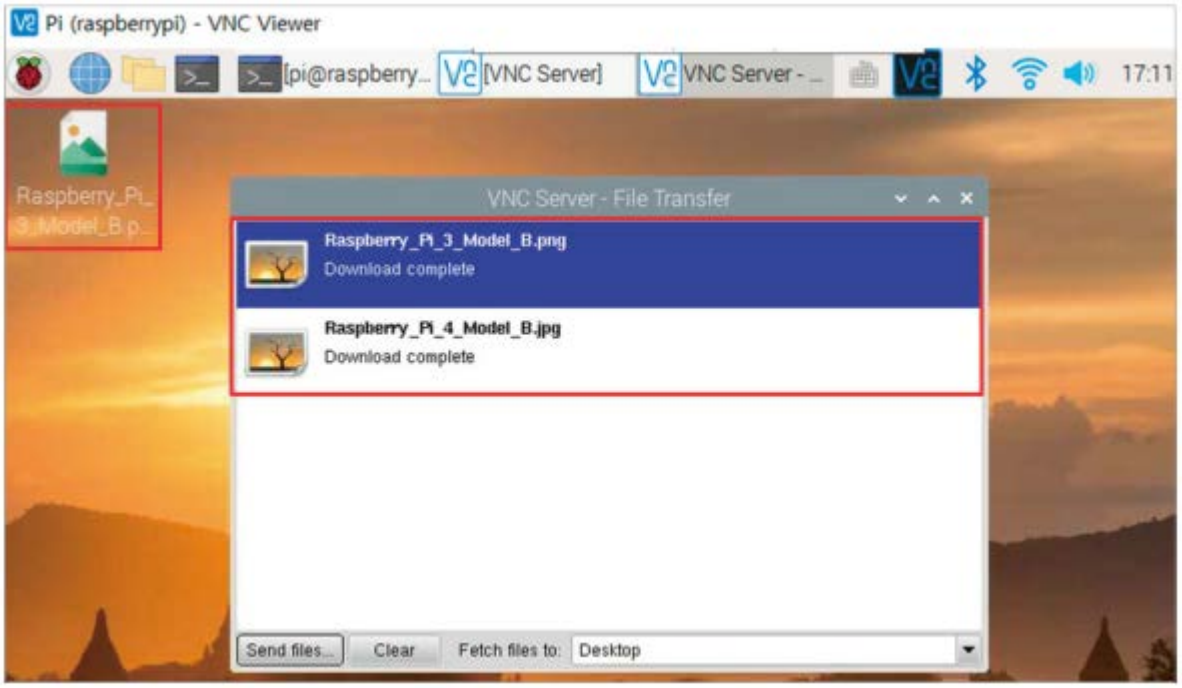
# PC – 라즈베리 파이 간 파일 전송

- **PC -> 라즈베리 파이**
  - VNC Viewer 상단에 마우스를 위치시키면 아래와 같이 파일 전송 아이콘이 생김. 좌측 하단의 Send files 버튼을 누르면 파일 선택 가능



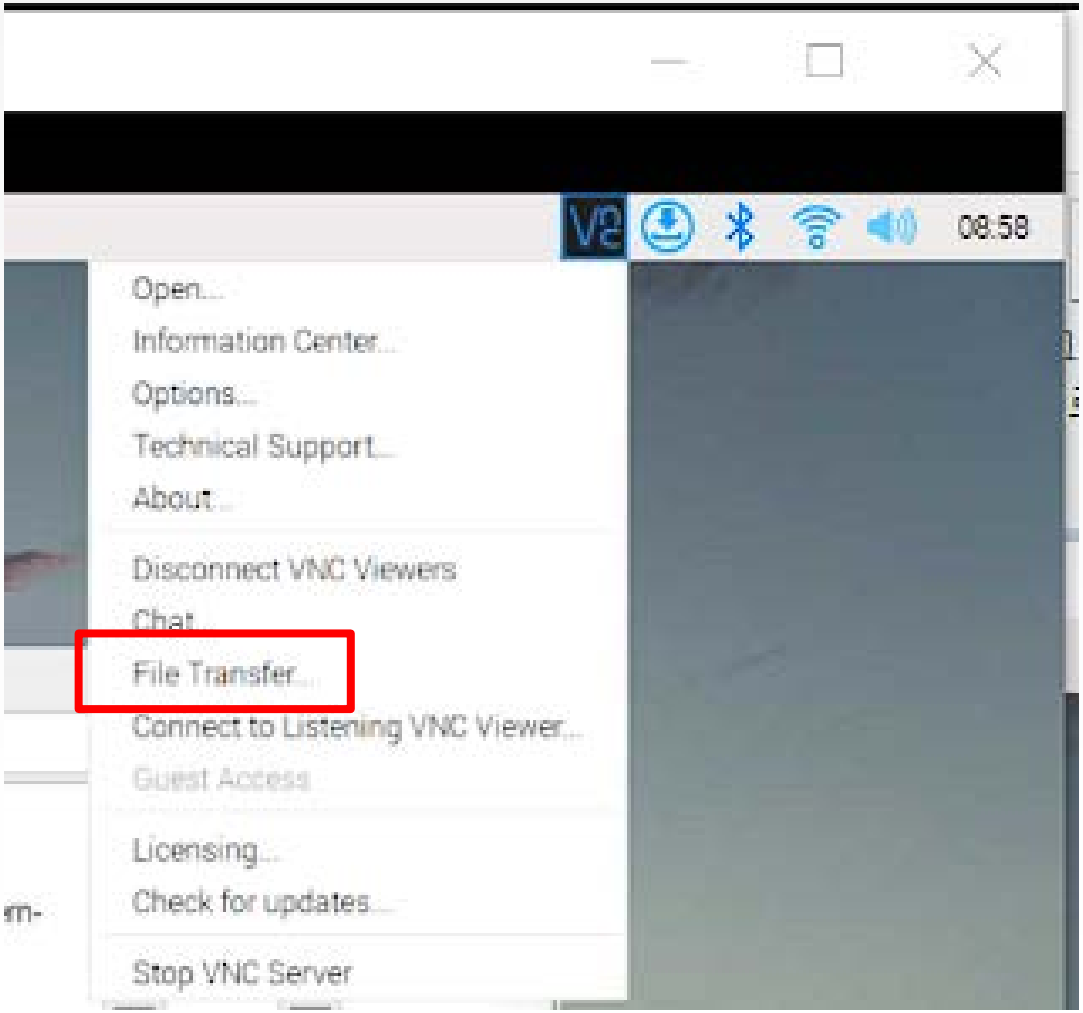
# PC – 라즈베리 파이 간 파일 전송

- **PC -> 라즈베리 파이**
  - 바탕 화면에 파일이 전송됨



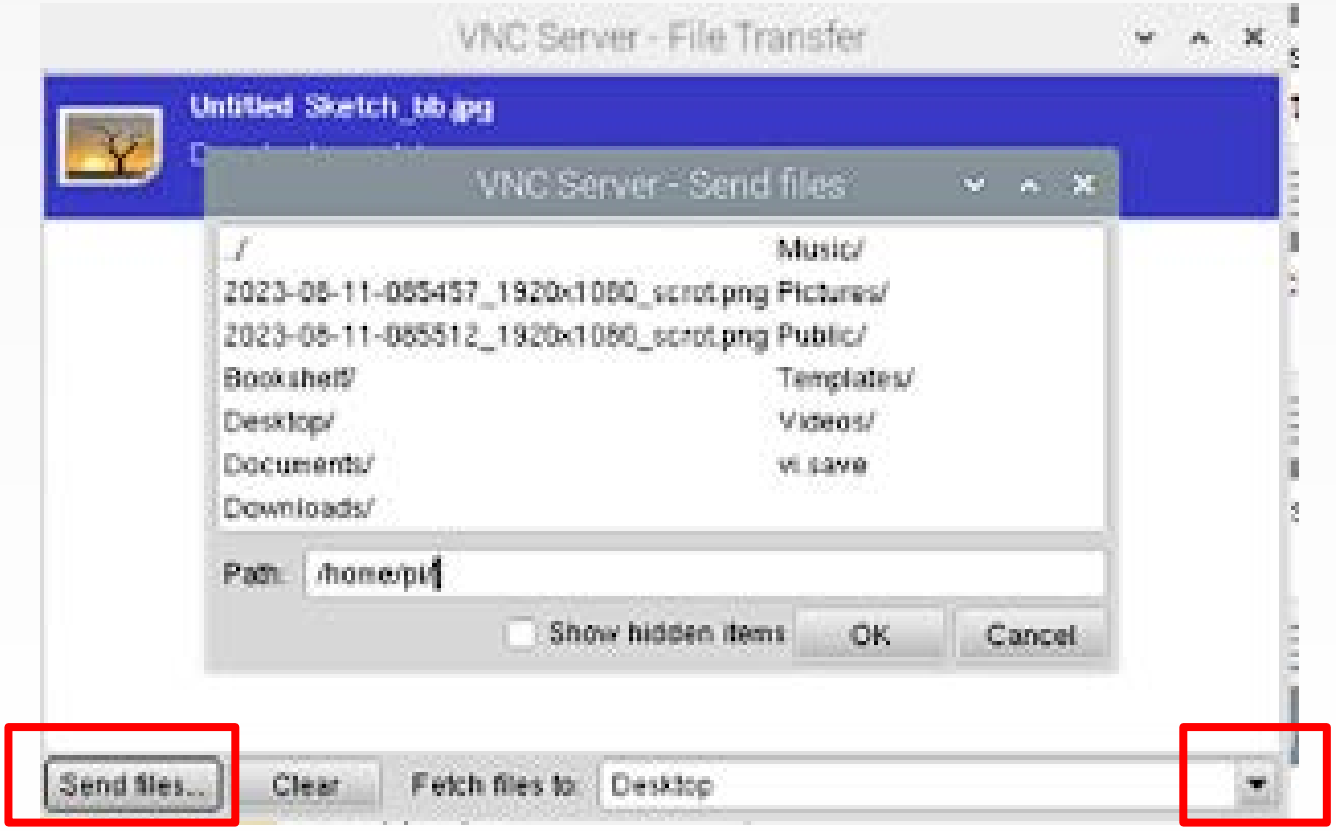
# PC – 라즈베리 파이 간 파일 전송

- **라즈베리 파이 -> PC**
  - 우측 상단의 VNC Viewer 아이콘을 우클릭해서 File Transfer 선택



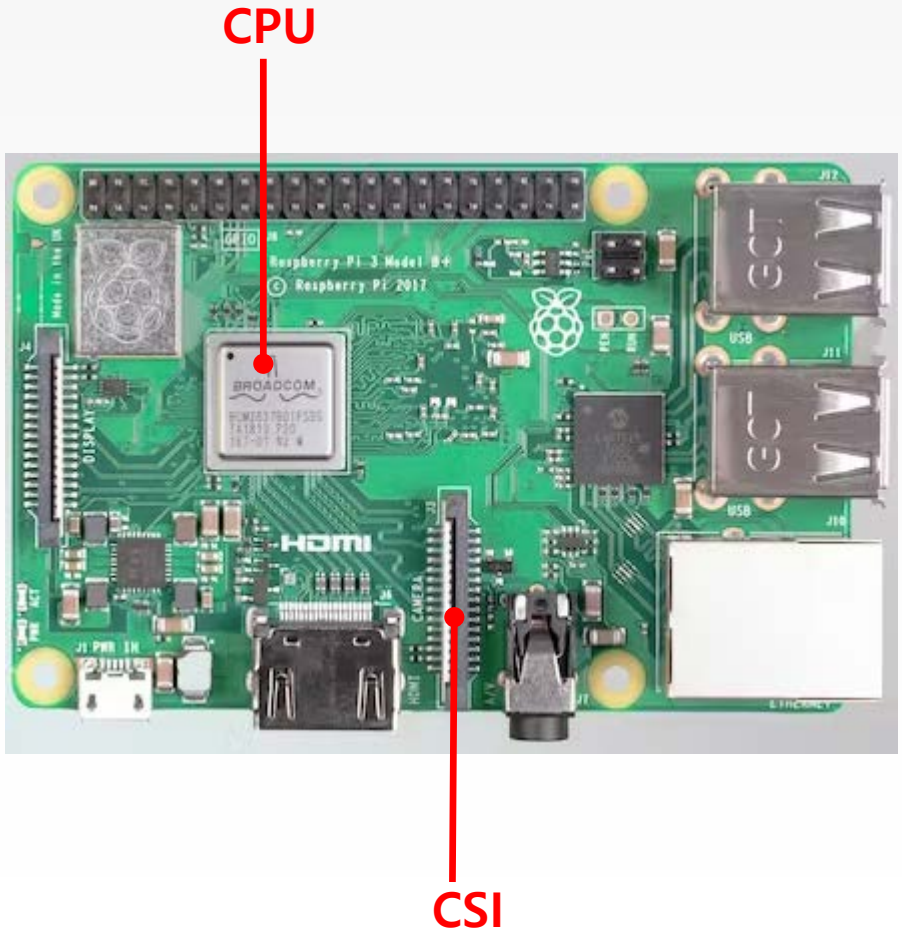
# PC – 라즈베리 파이 간 파일 전송

- **라즈베리 파이 -> PC**
  - Fetch files to 오른쪽의 역삼각형 아이콘을 클릭해서 경로를 지정한 후 좌측 하단의 Send files 버튼을 클릭하면 파일 선택 가능



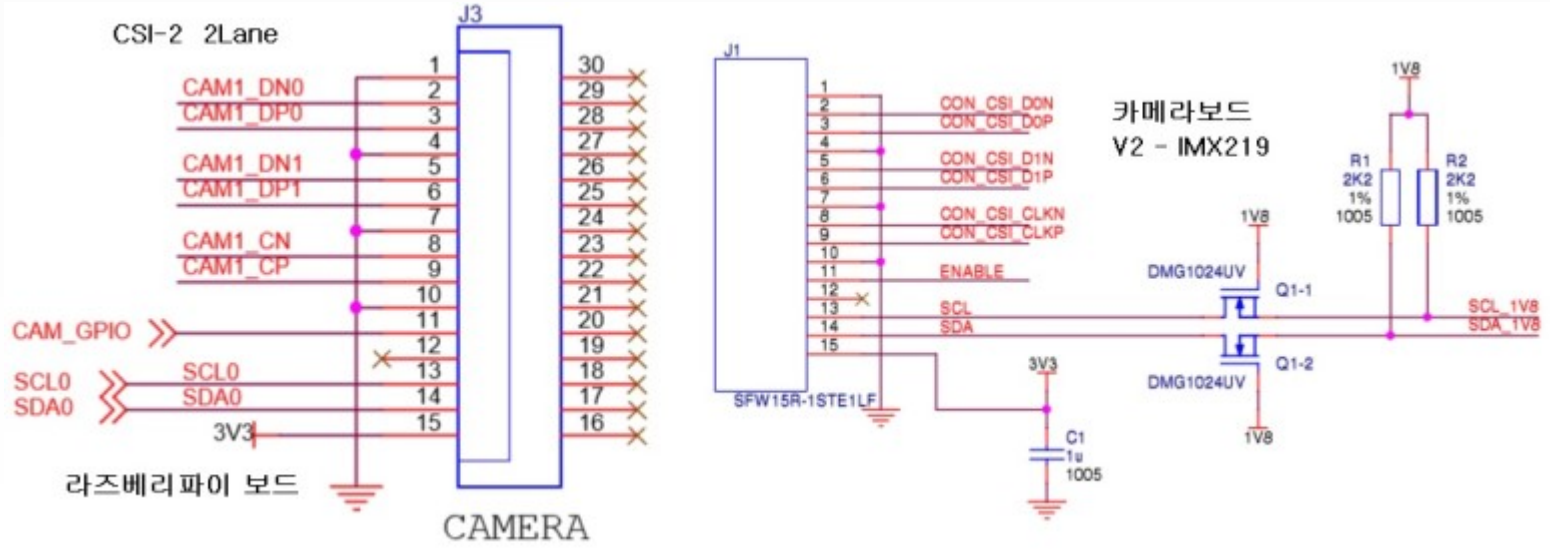
# 카메라 모듈

- **라즈베리 파이 카메라 모듈**
  - 라즈베리 파이에는 라즈베리 파이 카메라를 CSI 포트에 연결해서 사용할 수 있음



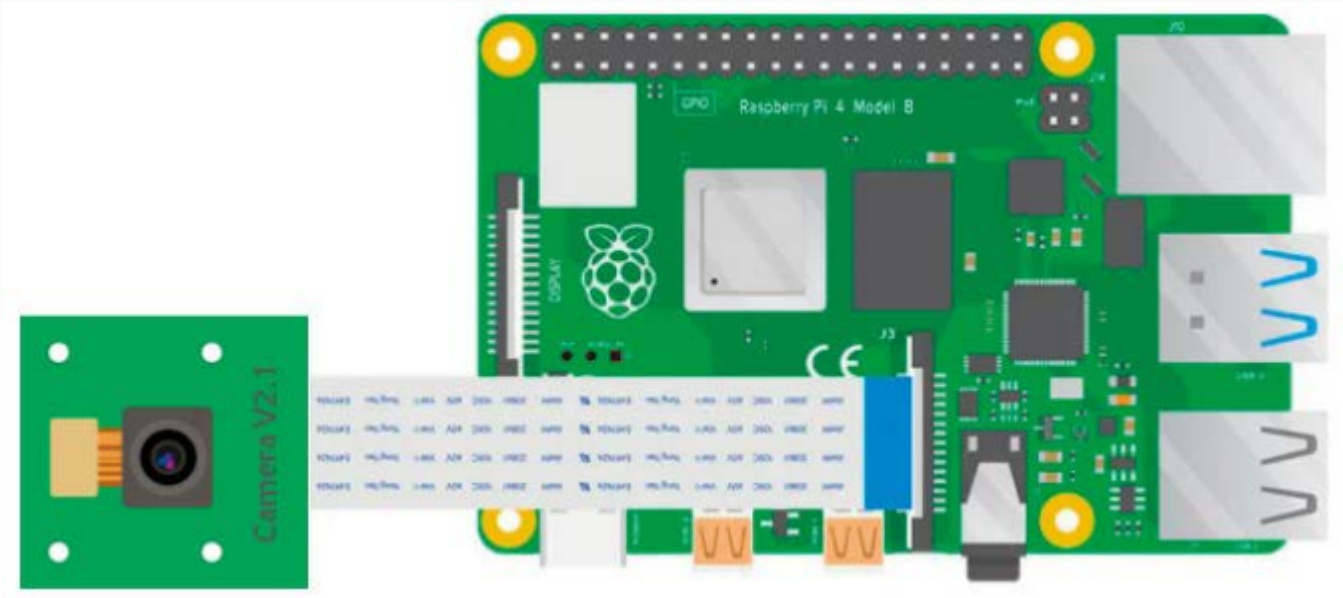
# 카메라 모듈

- **CSI 포트**
  - 2003년 삼성, 인텔, 노키아, 텍사스 인스트루먼츠 등의 기업들이 모바일 및 IoT 기기에서 각종 프로세서와 주변 기기들에 대한 인터페이스 표준을 개발하기 위해 MIPI(Mobile Industry Processor Interface) 설립
  - MIPI에서 결정한 대표적인 인터페이스가 DSI(Display Serial Interface), CSI(Camera Serial Interface)
  - CSI는 커넥터나 케이블의 종류를 지정한 것이 아니라 집적회로 내의 연결을 규정



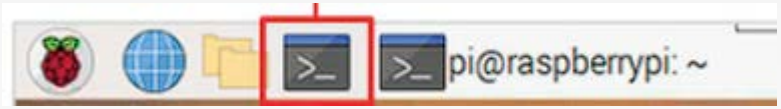
# 카메라 모듈

- **결선**
  - 20pin 케이블에 top, bottom 방향이 있으므로 방향을 맞춰서 연결
  - 커넥터 양쪽 끝의 고정 캡을 잡고 살짝 들어올려서 케이블을 끼워넣은 다음 다시 양쪽 고정 캡을 살짝 눌러주면 연결됨

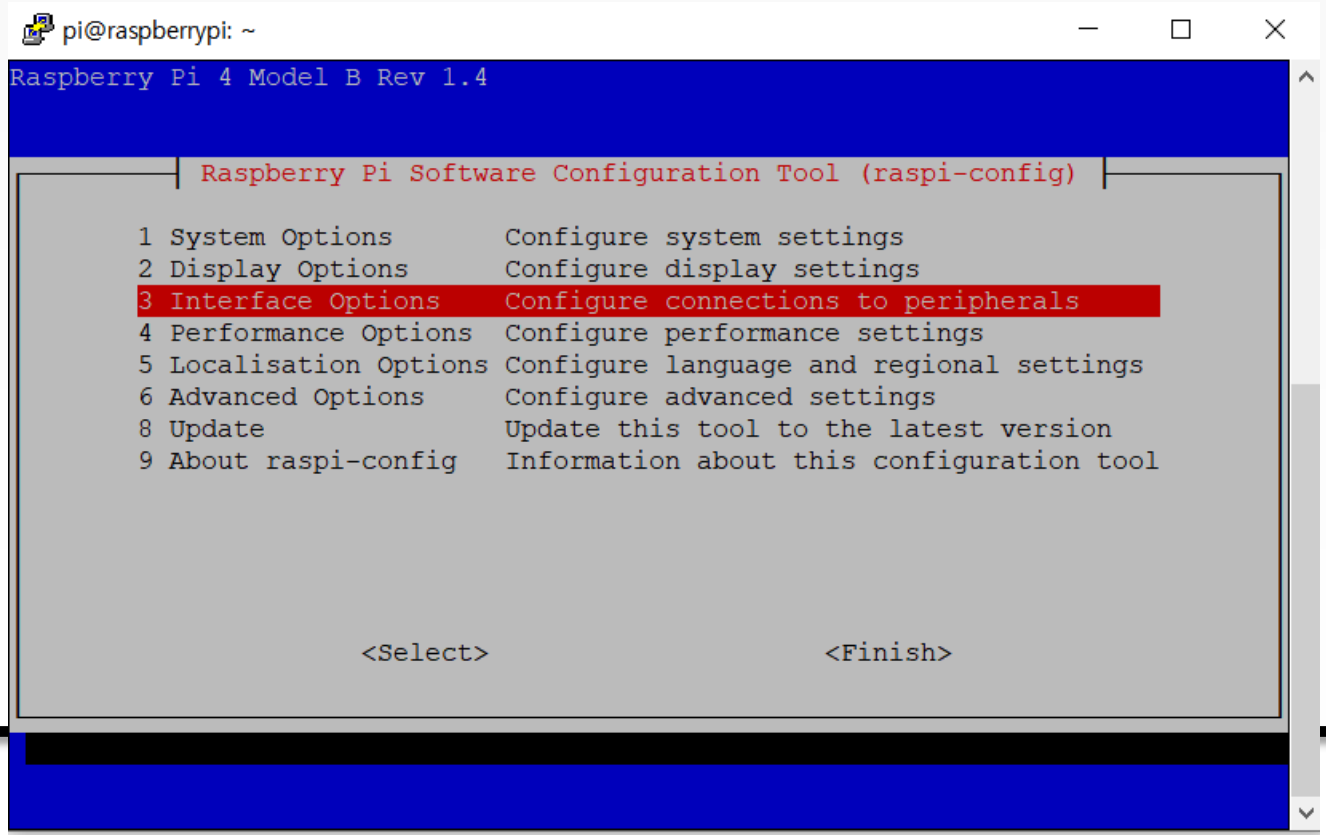


# 카메라 인터페이스 활성화

- **Interface Options**
  - 작업 표시줄의 터미널 프로그램 아이콘을 실행한 다음

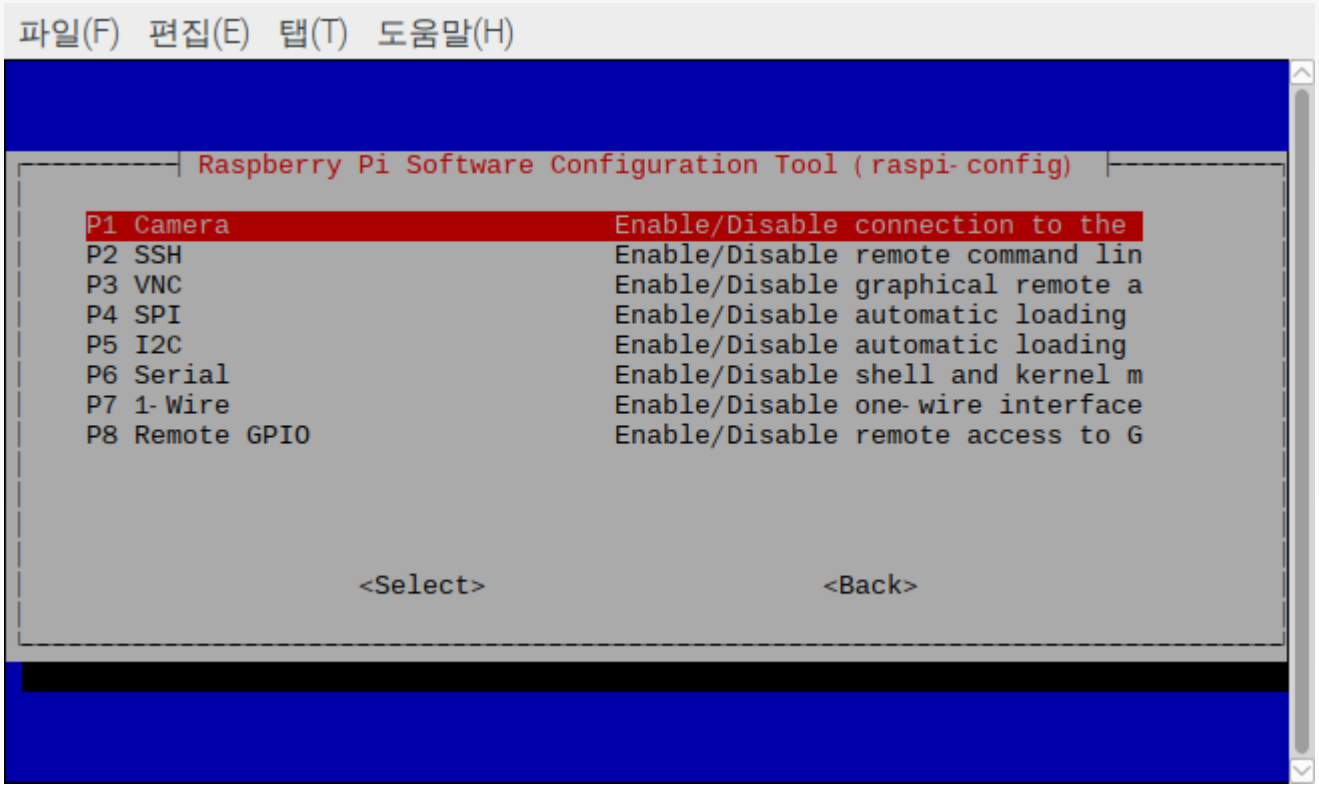



  - **sudo raspi-config** 입력 후 3.Interface Options 선택



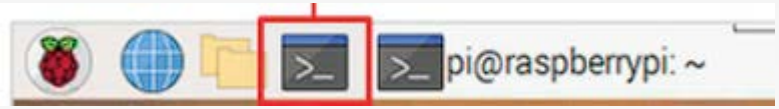
# 카메라 인터페이스 활성화

- **Interface Options**
  - P1 Camera를 선택하고 엔터 키를 눌러 Enable 로 변경한 다음 reboot

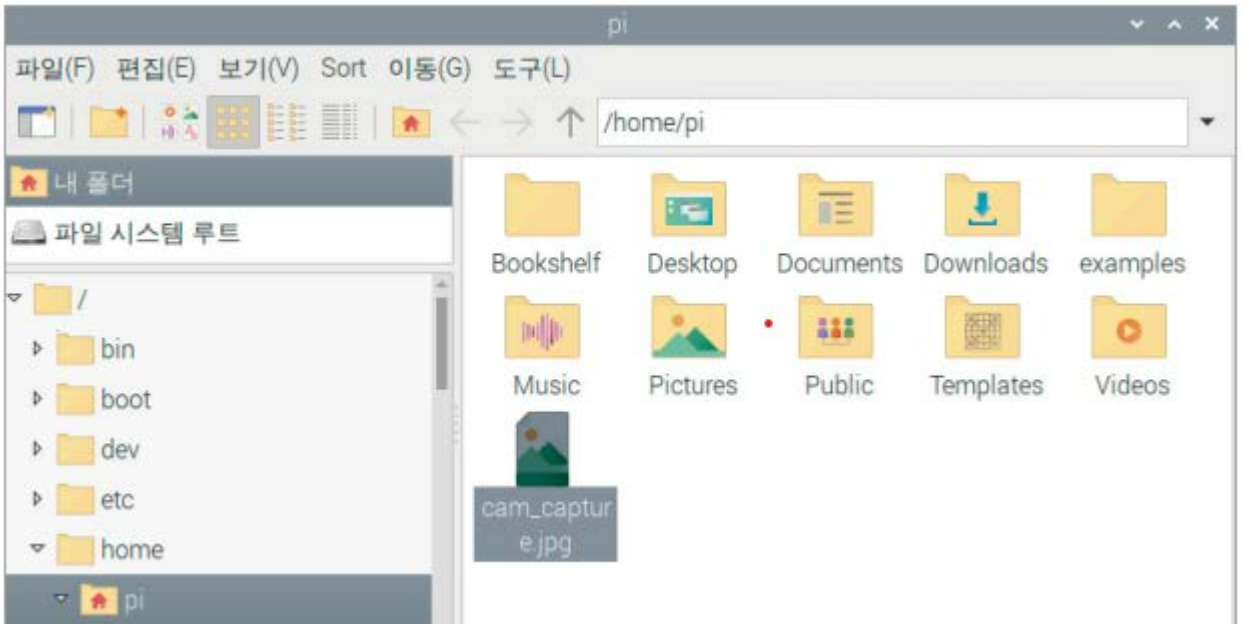


# 카메라 테스트

- **still cut**
  - 작업 표시줄의 터미널 프로그램 아이콘을 실행한 다음




  - **raspistill –o cam_capture.jpg** 입력하면 잠시 뒤에(2~3초 뒤) 스틸 컷 촬영이 되고 파일 저장

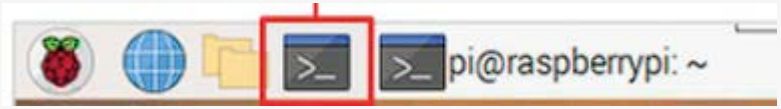


# 카메라 테스트

- **still cut**
  - **raspistill –o cam_capture_0.jpg –t 10000** 입력하면 10초 뒤에 스틸 컷 촬영이 되고 지정된 파일명으로 저장
  - 파일명을 이미 존재하는 파일명으로 지정하면 덮어쓰기함
  - 아래 명령어를 입력해서 스틸 컷을 찍어보시오.
  - **raspistill -vf -o cam_capture_v.jpg –t 10000**
  - **raspistill -hf -o cam_capture_h.jpg –t 10000**
  - **raspistill -vf -hf -o cam_capture_v_h.jpg –t 10000**
  - -o : Output, -vf : Vertically Flipped, -hf : Horizontally Flipped

# 카메라 테스트

- **video**
  - 작업 표시줄의 터미널 프로그램 아이콘을 실행한 다음



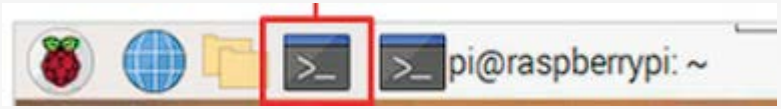

  - **raspivid –o video.h264 –t 30000** 입력하면 동영상 촬영이 되고 h264 확장자를 가지는 파일로 저장
  - 라즈베리 파이 카메라는 기본적으로 동영상을 H264 확장자(블루레이, 캠코더, CCTV 등에서 사용되는) 로 저장

  - MP4로 변환하기 위해 터미널 프로그램에서 gpac 패키지 설치
  - **sudo apt-get install –y gpac**

  - **raspivid –t 30000 –w 640 –h 480 –fps 25 –b 1200000 –p 0,0,640,480 –o video00.h264**
    촬영시간, 너비, 높이, FPS, bitrate, pixel, output
  - **MP4Box -add video00.h264 video00.mp4**
    MP4 파일로 변환

# 라즈베리 파이 카메라 모듈 테스트

- **picamera 모듈 설치**
  - 작업 표시줄의 터미널 프로그램 아이콘을 실행한 다음




  - **sudo apt-get install python3-picamera** 입력하면 설치됨

  - 앞으로 작성할 소스 코드들은 영상 데이터를 전송하므로 전송량이 많아져서 VNC Viewer 화면으로는 출력이 되지 않을 수 있으며, 라즈베리 파이에 모니터를 연결해서 확인 가능

# 라즈베리 파이 카메라 모듈 테스트

- **preview 함수**

```
import time
import picamera  # 라즈베리파이 카메라 모듈

picam = picamera.PiCamera()      # 카메라 Open
try:
    picam.start_preview()        # 화면에 Preview 영상 보여주기
    time.sleep(5)                # 5초 뒤
    picam.stop_preview()         # Preview 영상 Stop
finally:
    picam.close()                # 카메라 Close
```

# 라즈베리 파이 카메라 모듈 테스트

- **화면 회전 및 투명도 설정**
  - 다음과 같이 수정해서 카메라를 테스트
  - 180도 회전, 50% 투명도 설정
  - 회전 각도는 0, 90, 180, 270 중 선택 가능

```
import time
import picamera

picam = picamera.PiCamera()
try:
    picam.rotation = 180                    # 화면을 180도 회전
    picam.start_preview(alpha =127)         # 화면의 투명도 설정 (0~255)
    time.sleep(5)
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **화면 회전 및 투명도 설정**
  - 아래 코드를 화면의 투명도가 0%에서 시작해서 100%까지 증가하도록 수정해 보시오.

```
import time
import picamera

picam = picamera.PiCamera()
try:
    picam.rotation = 180                    # 화면을 180도 회전
    picam.start_preview(alpha =127)         # 화면의 투명도 설정 (0~255)
    time.sleep(5)
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **카메라 노출 모드**
  - 다음과 같이 수정해서 카메라를 테스트
  - exposure_mode 값은 string 타입이며 디폴트 값은 auto
  - off, auto, night, nightpreview, backlight, spotlight, sports, snow, beach, verylong, fixedfps, antishake, fireworks

```
import time
import picamera

picam = picamera.PiCamera()

try:
    picam.rotation =180
    picam.start_preview(alpha =127)
    picam.exposure_mode = 'night'    # 카메라 노출 모드
    time.sleep(5)
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **카메라 노출 모드**
  - exposure_mode 값은 모두 string 타입이며 tuple 형식으로 저장되어 있는데, 아래 코드를 참고해서 3초 간격으로 카메라 노출 모드의 모든 값들이 순차적으로 설정되어 디스플레이될 수 있도록 수정해 보시오.

  - off → auto → night → … → antishake → fireworks

```
import time
tup = ('a','bc','def','xyz')
for n in tup:
    print(n)
    time.sleep(3)
```

```
a
bc
def
xyz
```

- **3초 간격으로 출력**

# 라즈베리 파이 카메라 모듈 테스트

- **카메라 이미지 효과**
  - 다음과 같이 수정해서 카메라를 테스트
  - image_effect 값은 string 타입이며 디폴트는 none
  - none, negative, solarize, sketch, denoise, emboss, oilpaint, hatch, gpen, pastel, watercolor, film, blur, saturation, colorswap, washedout, posterise, colorpoint, colorbalance, cartoon, deinterlace1, deinterlace2
  - 3초 간격으로 카메라 이미지 효과의 모든 값들이 순차적으로 설정되어 디스플레이될 수 있도록 수정해 보시오.

```
import time
import picamera

picam = picamera.PiCamera()

try:
    picam.rotation =180
    picam.start_preview(alpha =127)
    picam.exposure_mode = 'night'
    picam.image_effect = 'blur'     # 이미지 효과
    time.sleep(5)
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **AWB(Auto White Balance)**
  - 다음과 같이 수정해서 카메라를 테스트
  - awb_mode 값은 string 타입이며 기본값은 auto
  - off, auto, sunlight, cloudy, shade, tungsten, fluorescent, incandescent, flash, horizon
  - 3초 간격으로 AWB의 모든 값들이 순차적으로 설정되어 디스플레이될 수 있도록 수정해 보시오.

```
import time
import picamera

picam = picamera.PiCamera()

try:
    picam.rotation =180
    picam.start_preview(alpha =127)
    picam.exposure_mode = 'night'
    picam.image_effect = 'blur'
    picam.awb_mode = 'sunlight'      # AWB
    time.sleep(5)
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **그 외 설정**
  - 다음과 같은 속성 설정이 가능
  - resolution
  - brightness
  - contrast
  - saturation
  - sharpness

```
import time
import picamera

picam = picamera.PiCamera()
picam.resolution = (1024,768)          # 해상도 설정

try:
    picam.rotation =180
    picam.start_preview(alpha =127)
    picam.exposure_mode = 'night'
    picam.image_effect = 'blur'
    picam.awb_mode = 'sunlight'
    picam.brightness = 50              # 밝기, 0 ~ 100
    picam.contrast = 50                # 대비, -100 ~ 100
    picam.saturation = 50              # 채도, -100 ~ 100
    picam.sharpness = 50               # 선예도, -100 ~ 100
    time.sleep(5)
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **still cut 파일 저장**
  - capture 함수에 의해 still cut 파일 저장 가능

```
import time
import picamera

picam = picamera.PiCamera()
picam.resolution = (1024,768)

try:
    filename = input('File name : ')     # shell에서 파일명 입력
    picam.rotation =180
    picam.start_preview(alpha =127)
    picam.exposure_mode = 'night'
    picam.image_effect = 'blur'
    picam.awb_mode = 'sunlight'
    picam.brightness = 50
    picam.contrast = 50
    picam.saturation = 50
    picam.sharpness = 50
    time.sleep(5)
    picam.capture(filename+'.jpg')       #입력받은 파일명.jpg 로 파일 저장
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **텍스트 추가**
  - 화면 상단에 텍스트를 출력. annotate_text

```
import time
import picamera, Color                  # Color 모듈 추가

picam = picamera.PiCamera()
picam.resolution = (1024,768)

try:
    filename = input('File name : ')
    picam.rotation =180
    picam.start_preview(alpha =127)
    picam.exposure_mode = 'night'
    picam.image_effect = 'blur'
    picam.awb_mode = 'sunlight'
    picam.brightness = 50
    picam.contrast = 50
    picam.saturation = 50
    picam.sharpness = 50

    picam.annotate_text_size = 50                # 출력될 텍스트 크기
    picam.annotate_background = Color('black')   # 배경 색상
    picam.annotate_foreground = Color('yellow')  # 출력될 텍스트
    picam.annotate_text = time.ctime()           # 현재 시각을 텍스트로 출력

    time.sleep(5)
    picam.capture(filename+'.jpg')
    picam.stop_preview()
finally:
    picam.close()
```

# 라즈베리 파이 카메라 모듈 테스트

- **동영상 촬영**
  - start_recording
  - wait_recording
  - stop_recording

```
import time
import picamera, Color

picam = picamera.PiCamera()
picam.resolution = (1024,768)
picam.framerate = 15    # framerate 변경

try:
    filename = input('File name : ')
    picam.rotation =180
    picam.start_preview(alpha =127)
    picam.exposure_mode = 'night'
    picam.image_effect = 'blur'
    picam.awb_mode = 'sunlight'
    picam.brightness = 50
    picam.contrast = 50
    picam.saturation = 50
    picam.sharpness = 50

    picam.annotate_text_size = 50
    picam.annotate_background = Color('black')
    picam.annotate_foreground = Color('yellow')
    picam.annotate_text = time.ctime()

    picam.start_recording(filename + '.h264')    # 촬영 시작
    picam.wait_recording(5)    # 5초 동안 촬영
    picam.stop_recording()     # 촬영 종료

    picam.stop_preview()
finally:
    picam.close()
```

# OpenCV

- **OpenCV**
  - 2000년 처음 배포된 Computer Vision 프로젝트
  - Windows, Linux, OS X, 안드로이드, iOS 등 다양한 운영체제와 C++, Java, Python 등 다양한 프로그래밍 언어를 지원
  - C++로 구현되어 있음
  - 필터링, 색 변환, 이미지 모핑, 카메라 보정 등 가능
  - AI 기술과 결합하여 이미지/동영상에서 객체 검출, 추적, 특징 추출, 패턴 인식, 사람이나 동물의 얼굴 인식 등의 작업 가능
- **python3-opencv**
  - C++로 구현되어 있는 OpenCV 라이브러리를 파이썬으로 wrapping해서 모듈화
  - 설치 시 numpy 모듈이 함께 설치됨
  - numpy는 각종 수학 연산을 위한 라이브러리를 제공하며, 이미지/동영상 처리에 필요한 행렬 연산을 손쉽게 하도록 함
  - 모든 OpenCV 배열/행렬 구조는 numpy 배열/행렬 구조로 변환되어 처리

# Ch.02 리뷰 – 파이썬 패키지

- **모듈들의 집합**
  - 라즈베리 파이에서는 프롬프트에서 아래와 같이 입력해서 설치

  - **sudo apt update**
  - **sudo apt full-upgrade**
  - **sudo apt install python3-opencv**

  - numpy와 함께 설치됨

# Ch.02 리뷰 – 파이썬 패키지

## •OpenCV 패키지 사용해보기

- photo.jpg 파일을 .py 파일과 같은 폴더(디렉토리)에 저장한 다음 아래 코드를 실행. 반드시 같은 폴더에 저장해야 함.
- **/lib/python3.9 에 저장**

```
import cv2

img = cv2.imread('photo.jpg')    # image 파일 읽기

cv2.imshow('photo', img)         # image 파일 출력
cv2.waitKey(0)                   # 키보드 입력 대기
cv2.destroyAllWindows()          # 키보드 입력이 되면 창 닫기
```

# Ch.02 리뷰 – 파이썬 패키지

- **OpenCV 패키지 사용해보기**
  - 회색조로 변환

```
import cv2

img = cv2.imread('photo.bmp')    # image 파일 읽기
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                                 # color을 BGR에서 회색조로 convert
cv2.imshow('photo', gray)        # image 파일 출력
cv2.waitKey(0)                   # 키보드 입력 대기
cv2.destroyAllWindows()          # 키보드 입력이 되면 창 닫기
```

# Ch.02 리뷰 – 파이썬 패키지

## •OpenCV 패키지 사용해보기

- 얼굴 인식
- haarcascade_frontalface_default.xml 파일을 .py 파일과 같은 폴더(디렉토리)에 저장한 다음 아래 코드를 실행.

```
import cv2

img = cv2.imread('photo.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
        # CascadeClassifier(다단계 분류기) 객체를 생성한 후
        # 변수 face_cascade가 가리키도록 함
        # haarcascade_frontalface_default.xml 파일은
        # 얼굴의 앞면을 검출하기 위해
        # 머신 러닝으로 미리 학습시켜 놓은 분류기 파일
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        # detectMultiScale 함수는
        # 지정된 그림 파일(gray)을
        # 1.3배만큼씩 축소, 즉 30%씩 축소해가며 얼굴을 검출
        # 그 과정에서 5회 검출이 되어야 얼굴로 최종 인식
for (x,y,w,h) in faces:
        img = cv2.rectangle(img,(x,y),(x +w,y +h),(255,0,0),2)
        # 검출된 얼굴의 좌표 : x, y
        # 검출된 얼굴의 크기 : w, h
        # rectangle 형태를 이미지 파일에 추가

cv2.imshow('photo', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# Ch.02 리뷰 – 파이썬 패키지

## •OpenCV 패키지 사용해보기

- 눈 인식
- haarcascade_eye.xml 파일을 .py 파일과 같은 폴더(디렉토리)에 저장한 다음 아래 코드를 실행.

```
import cv2

img = cv2.imread('photo.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
        # haarcascade_eye.xml는 눈을 검출하기 위한 분류기 파일

faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
        img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        # gray 변수가 가리키는 이미지 내의 얼굴 영역에 대한 객체를 roi_gray로
        # roi : region of interest
        roi_color = img[y:y+h, x:x+w]
        # img 변수가 가리키는 이미지 내의 얼굴 영역에 대한 객체를 roi_color로
        eyes = eye_cascade.detectMultiScale(roi_gray)
        # roi_gray 변수가 가리키는 회색의 얼굴 영역에서 눈을 검출
        for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
                # 검출된 눈의 좌표 : ex, ey
                # 검출된 눈의 크기 : ew, eh
                # rectangel 형태를 2개 이미지 파일에 추가

cv2.imshow('photo', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# OpenCV 모듈을 활용한 영상 처리

- OpenCV 모듈을 이용해 카메라 영상 읽고 출력하기
  - 아래 코드는 파일을 따로 저장해 두세요

```
import cv2

cap = cv2.VideoCapture(0)       # 영상 입력 기능을 갖는 VideoCapture 객체를 생성
                                # 0 은 카메라의 번호
if cap.isOpened():      # isOpened()는 영상 입력 기능이 정상적으로 실행되었다면 1
                        #                                 아니라면 0
        # 영상의 현재 가로, 세로, FPS 값을 출력
        print('width:', cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        print('height:', cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        print('fps:', cap.get(cv2.CAP_PROP_FPS))

while cap.isOpened():   # 영상 입력 기능이 정상적으로 실행되었다면 반복 실행
        ret,img = cap.read()    # read()는 비디오 프레임을 읽어 2개의 값을 반환
                                # ret : 비디오 프레임이 정상적으로 읽혔다면 1
                                #       아니라면 0
                                # img : matrix image 객체
        if ret:         # 정상적으로 읽혔다면
                        # matrix image 객체를 읽어 show. 영상 제목을 HI로 설정
                cv2.imshow('HI', img)

                key = cv2.waitKey(1) & 0xFF # 키보드 입력을 1ms wait
                                            # 64비트 OS라면 0xFF를 AND 연산
                if key ==27:                # ESC 키가 입력되면 (ASCII 코드 값)
                # if key ==ord('q')
                        break

cap.release()           # isOpened()에 의해 실행되었던 영상 입력 기능을 종료
cv2.destroyAllWindows() # 모든 창을 닫음
```

# OpenCV 모듈을 활용한 영상 처리

- **OpenCV 모듈을 이용해 카메라 영상 저장하기**

```
import cv2

cap = cv2.VideoCapture(0)
if cap.isOpened():
        w =int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))       # 문자열을 정수로 형 변환
        h =int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps =int(cap.get(cv2.CAP_PROP_FPS))

        # VideoWriter_fourcc 함수에 의해 FourCC 설정(코덱)
        # Four Character Code : 4 바이트 문자열로 데이터 형식을 구분
        #                       DIVX, H264, MJPG 등
        # 파라미터를 *'DIVX' 로 지정하면 'D', 'I', 'V', 'X' 한 글자씩 넘겨짐
        fourcc = cv2.VideoWriter_fourcc(*'DIVX')

        # VideoWriter('파일명', FourCC, FPS, 해상도(가로, 세로))
        out = cv2.VideoWriter('output.avi', fourcc, fps, (w,h))

while cap.isOpened():
        ret,img = cap.read()

        if ret:
                cv2.imshow('HI2', img)

                out.write(img)  # matrix image 객체를 out 변수가 가리키는 파일에 저장

                k = cv2.waitKey(1) & 0xFF
                if k ==27:
                        break

out.release()   # 영상 저장 기능을 종료
cap.release()
cv2.destroyAllWindows()
```

# OpenCV 모듈을 활용한 영상 처리

- OpenCV 모듈을 이용해 영상 파일 읽고 출력하기

```
import cv2

cap = cv2.VideoCapture('output.avi')    # 카메라가 아니라 파일을 입력으로 받음
if cap.isOpened():
        print('width:', cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        print('height:', cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        print('fps:', cap.get(cv2.CAP_PROP_FPS))

while cap.isOpened():
        ret,img = cap.read()

        if ret:
                cv2.imshow('Video Capture', img)

                k = cv2.waitKey(30) & 0xFF # 30ms wait
                if k ==27:
                        break
        else: break

cap.release()
cv2.destroyAllWindows()
```

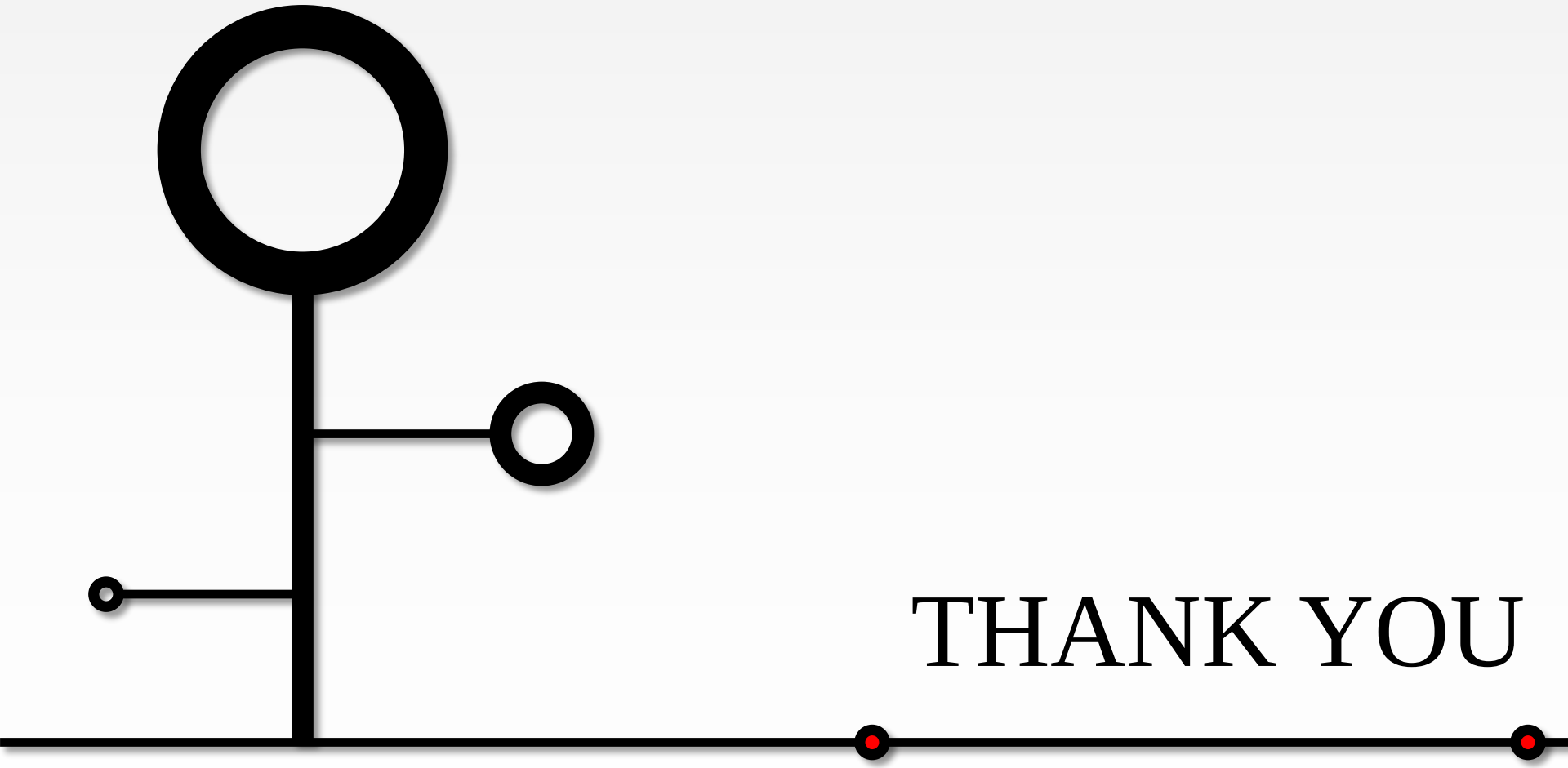
# 인공지능 라이브러리 활용하기

- **OpenCV 모듈/머신 러닝 필터를 이용해 영상 얼굴 인식하기**

```
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

while cap.isOpened():
        ret,img = cap.read()

        if ret:
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                faces = face_cascade.detectMultiScale(gray, 1.3, 5)
                for (x,y,w,h) in faces:
                        img = cv2.rectangle(img,(x,y),(x +w,y +h),(255,0,0),2)

                cv2.imshow('Video Capture', img)

                key = cv2.waitKey(1) & 0xFF
                if key == 27:
                        break

cap.release()
cv2.destroyAllWindows()
```

# 인공지능 라이브러리 활용하기

- **OpenCV 모듈/머신 러닝 필터를 이용해 영상 얼굴/눈 인식하기**
  - 31페이지 이미지 파일의 눈 인식 소스 코드를 참고해서 작성해 보세요



# THANK YOU