실습 1. 아래의 코드에서 빈칸을 채워서 웹캠을 통해 실시간으로 얼굴을 감지하고, 감지된 얼굴 주위에 Bounding Box를 그리는 프로그램을 완성해보세요. a_od.py

```
import cv2

# XML 파일을 로드하여 Haar Cascade 분류기 객체를 생성합니다.
face_cascade = cv2.CascadeClassifier(____)

# 웹캠에서 영상을 캡처합니다.
cap = cv2.____()

while True:
    # 웹캠의 현재 프레임을 가져옵니다.
    ret, frame = cap.____

    # 현재 프레임을 그레이스케일로 변환합니다.
    gray = cv2.cvtColor(frame, cv2.____)

    # 분류기를 사용하여 얼굴을 감지합니다.
    faces = face_cascade.____(gray, scaleFactor=1.1, minNeighbors=5)

    # 감지된 각 얼굴에 대하여 Bounding Box를 그립니다.
    for (x, y, w, h) in faces:
        cv2.____(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # 현재 프레임에 감지된 얼굴과 Bounding Box를 출력합니다.
    cv2.____('Face Detection', frame)

    # 'q' 키를 누르면 루프에서 빠져나옵니다.
    if cv2.____(1) & 0xFF == ord('q'):
        break

# 웹캠을 해제하고 모든 창을 닫습니다.
cap.____()
cv2.____()
```

2. 아래의 코드에서 빈칸을 채워서 이미지에서 도형을 감지하고, 감지된 도형 주위에 Bounding Box를 그리는 프로그램을 완성해보세요. b_od.py

```
# 필요한 라이브러리를 import합니다.
import cv2
import numpy as np

# 이미지 파일을 불러옵니다.
img = cv2.imread(____)

# 이미지를 그레이스케일로 변환합니다.
gray = cv2.cvtColor(img, cv2.____)

# Canny 알고리즘을 사용하여 엣지를 검출합니다.
edges = cv2.____(gray, 50, 150)

# 외곽선을 찾습니다.
contours, _ = cv2.____(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# 찾은 외곽선을 화면에 그립니다.
for cnt in contours:
    x, y, w, h = cv2.____(cnt)
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

# 결과를 화면에 출력합니다.
cv2.imshow('Shape Detection', img)
cv2.____(0)
cv2.destroyAllWindows()
```

3. Object detection을 위해 라벨링을 하는 과정에서 각 객체의 윤곽선이나 경계 상자 (bounding box) 정보를 사용하게 됩니다. 이 정보를 이용해서 객체별로 특징점을 추출하고 라벨링할 수 있습니다. 아래 키워드 힌트를 사용해서 코드를 완성하세요
이미지 파일은 그림판에서 0~ 9 까지 무순위로 작성한 그림을 사용합니다.

[키워드 힌트]c_od.py

'image.jpg' cvtColor threshold
findContours SIFT_create boundingRect y:y+h, x:x+w
sift pt putText imshow waitKey destroyAllWindows

```
import cv2
import numpy as np

# 이미지 불러오기
```

```
image = cv2.imread('__________')

# 그레이스케일로 변환
gray = cv2.__________(image, cv2.COLOR_BGR2GRAY)

# 이진화
_, binary_img = cv2.__________(gray, 128, 255, cv2.THRESH_BINARY_INV)

# 윤곽선 찾기
contours, _ = cv2.__________(binary_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# SIFT 객체 생성
sift = cv2.__________()

# 객체를 탐지하고 라벨링
for i, contour in enumerate(contours):
    x, y, w, h = cv2.__________(contour)
    roi = gray[_______:_______, _______:_______]

    # 특징점 추출
    keypoints, descriptors = _________.detectAndCompute(roi, None)

    # 원래 이미지에 특징점 표시 (라벨링을 위한 예)
    for point in keypoints:
        transformed_x, transformed_y = int(x + point.____[0]), int(y + point.____[1])
        cv2.circle(image, (transformed_x, transformed_y), 3, (0, 255, 0), -1)

    # 라벨 표시 (여기서는 i를 라벨로 사용)
    cv2.__________(image, str(i), (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)

# 결과 이미지 표시
cv2.__________('Labeled Objects with Keypoints', image)
cv2.__________(0)
cv2.__________()
```