```
"""goals5faces.py

  Run the face (and eye) detectors and show the results.
"""

# Import OpenCV
import cv2

# Set up video capture device (camera).  Note 0 is the camera number.
# If things don't work, you may need to use 1 or 2?
camera = cv2.VideoCapture(0, cv2.CAP_V4L2)
if not camera.isOpened():
    raise Exception("Could not open video device: Maybe change the cam number?")

# Change the frame size and rate.  Note only combinations of
# widthxheight and rate are allowed.  In particular, 1920x1080 only
# reads at 5 FPS.  To get 30FPS we downsize to 640x480.
camera.set(cv2.CAP_PROP_FRAME_WIDTH,  640)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
camera.set(cv2.CAP_PROP_FPS,           30)

# Get the face/eye detector models from XML files.  Instantiate detectors.
faceXML = "haarcascade_frontalface_default.xml"
eyeXML1 = "haarcascade_eye.xml"
eyeXML2 = "haarcascade_eye_tree_eyeglasses.xml"

faceDetector = cv2.CascadeClassifier(faceXML)
eyeDetector  = cv2.CascadeClassifier(eyeXML1)


# Keep scanning, until 'q' hit IN IMAGE WINDOW.
while True:
    # Grab an image from the camera.  Often called a frame (part of sequence).
    ret, frame = camera.read()


    # Convert the image to gray scale.
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Grab the faces - the cascade detector returns a list faces.
    faces = faceDetector.detectMultiScale(gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (30,30),
        flags = cv2.CASCADE_SCALE_IMAGE)

    # Process the faces: Each face is a bounding box of (x,y,w,h)
    # coordinates.  Draw the bounding box ON THE ORIGINAL IMAGE.
    if len(faces) > 0:
        # Grab the first face.
        face = faces[0]

        # Grab the face coodinates.
        (x, y, w, h) = face

        # Draw the bounding box on the original color frame.
        cv2.rectangle(frame, (x, y), (x+w-1, y+h-1), (0, 255, 0), 3)
        # orig_box_colors = frame[y:y+h-1, x:x+w-1, 0:3]
        # new_box = (orig_box_colors[:,:,2], orig_box_colors[:,:,1], orig_box_colors[:,:,0])
        half = w//2
        if w % 2 == 0:
            temp = frame[y:y+h, x:x+half, :].copy()
            frame[y:y+h, x:x+half, :] = frame[y:y+h, x+w-1:x+half-1:-1, :]
            frame[y:y+h, x+w-1:x+half-1:-1, :] = temp
        else:
            temp = frame[y:y+h, x:x+half, :].copy()
            frame[y:y+h, x:x+half, :] = frame[y:y+h, x+w-1:x+half:-1, :]
            frame[y:y+h, x+w-1:x+half:-1, :] = temp

        # Also look for eyes - only within the region of the face!
        # This similarly a list of eyes relative to this region.
        eyes = eyeDetector.detectMultiScale(gray[y:y+h,x:x+w])

        # Process the eyes: As before, eyes is a list of bounding
        # boxes (x,y,w,h) relative to the processed region.
        for (xe,ye,we,he) in eyes:
            # Can you draw circles around the eyes?  Consider the function:
            # cv2.circle(frame, (xc, yc), radius, (b,g,r), linewidth)
            pass # replace this.

```

```
81      # frame[480-150:480-50, 50:150, 0:3] = (0, 70, 255)
82
83      # Show the processed image with the given title.  Note this won't
84      # actually appear (draw on screen) until the waitKey(1) below.
85      cv2.imshow('Processed Image', frame)
86
87      # Check for a key press IN THE IMAGE WINDOW: waitKey(0) blocks
88      # indefinitely, waitkey(1) blocks for at most 1ms.  If 'q' break.
89      # This also flushes the windows and causes it to actually appear.
90      if (cv2.waitKey(1) & 0xFF) == ord('q'):
91          break
92
93  # Close everything up.
94  camera.release()
95  cv2.destroyAllWindows()
```