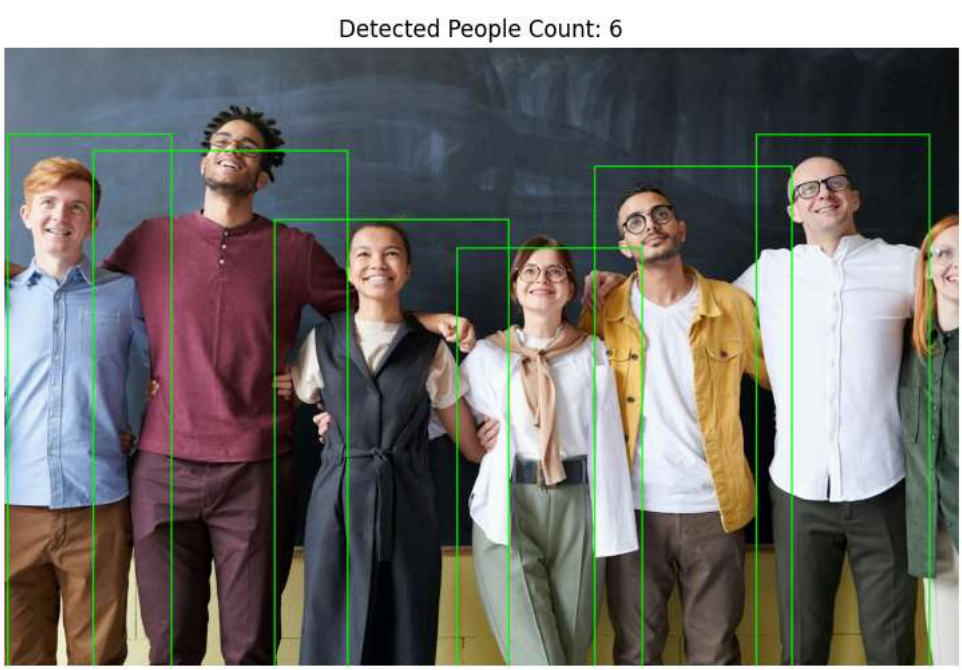
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
import urllib.request

# Load YOLOv4-tiny (lighter, faster)
YOLO_CFG = "yolov4-tiny.cfg"
YOLO_WEIGHTS = "yolov4-tiny.weights"
COCO_NAMES = "coco.names"

# Download files if not available
if not os.path.exists(YOLO_CFG):
    urllib.request.urlretrieve("https://github.com/AlexeyAB/darknet/blob/master/cfg/yolov4-tiny.cfg?raw=true", YOLO_CFG)

if not os.path.exists(YOLO_WEIGHTS):
    urllib.request.urlretrieve("https://github.com/AlexeyAB/darknet/releases/download/yolov4/yolov4-tiny.weights", YOLO_WEIGHTS)

if not os.path.exists(COCO_NAMES):
    urllib.request.urlretrieve("https://raw.githubusercontent.com/pjreddie/darknet/master/data/coco.names", COCO_NAMES)

# Load YOLO model
net = cv2.dnn.readNet(YOLO_WEIGHTS, YOLO_CFG)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

# Load class names
with open(COCO_NAMES, "r") as f:
    classes = f.read().strip().split("\n")

# Read input image
image_path = "people.jpg"  # Ensure the correct path
image = cv2.imread(image_path)
height, width = image.shape[:2]

# Prepare image for YOLO
blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416), swapRB=True, crop=False)
net.setInput(blob)
layer_names = net.getUnconnectedOutLayersNames()
outputs = net.forward(layer_names)

# Process detections
conf_threshold = 0.3  # Lowered confidence threshold to detect more objects
nms_threshold = 0.4
boxes = []
confidences = []
people_count = 0

for output in outputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]

        if confidence > conf_threshold and classes[class_id] == "person":
            center_x, center_y, w, h = (detection[:4] * np.array([width, height, width, height])).astype("int")
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            # Expand bounding box slightly for better detection in corners
            w = int(w * 1.1)
            h = int(h * 1.1)
            x = max(0, x)
            y = max(0, y)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))

# Apply Non-Maximum Suppression
indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

if len(indices) > 0:
    people_count = len(indices)

# Draw bounding boxes
for i in indices.flatten():
```

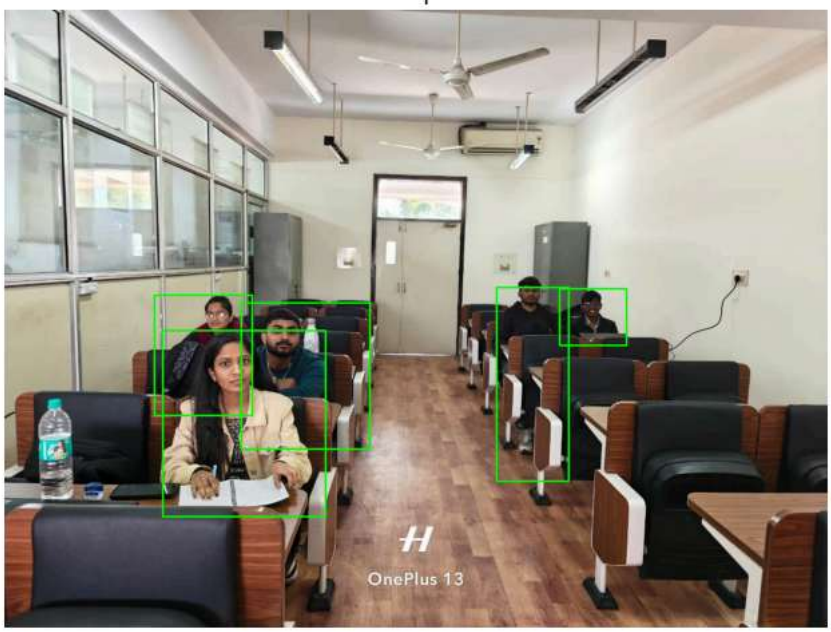
```
        x, y, w, h = boxes[i]
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Convert to RGB for plotting
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Display the image with bounding boxes
    plt.figure(figsize=(10, 6))
    plt.imshow(image_rgb)
    plt.axis("off")
    plt.title(f"Detected People Count: {people_count}")
    plt.show()

    print(f"Detected People Count: {people_count}")
```




```
Detected People Count: 6
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
import urllib.request

# Load YOLOv4-tiny (lighter, faster)
YOLO_CFG = "yolov4-tiny.cfg"
YOLO_WEIGHTS = "yolov4-tiny.weights"
COCO_NAMES = "coco.names"

# Download files if not available
if not os.path.exists(YOLO_CFG):
    urllib.request.urlretrieve("https://github.com/AlexeyAB/darknet/blob/master/cfg/yolov4-tiny.cfg?raw=true", YOLO_CFG)

if not os.path.exists(YOLO_WEIGHTS):
    urllib.request.urlretrieve("https://github.com/AlexeyAB/darknet/releases/download/yolov4/yolov4-tiny.weights", YOLO_WEIGHTS)

if not os.path.exists(COCO_NAMES):
    urllib.request.urlretrieve("https://raw.githubusercontent.com/pjreddie/darknet/master/data/coco.names", COCO_NAMES)

# Load YOLO model
net = cv2.dnn.readNet(YOLO_WEIGHTS, YOLO_CFG)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

# Load class names
with open(COCO_NAMES, "r") as f:
    classes = f.read().strip().split("\n")

# Read input image
image_path = "img1.jpeg"  # Ensure the correct path
```

```
image = cv2.imread(image_path)
height, width = image.shape[:2]

# Prepare image for YOLO
blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416), swapRB=True, crop=False)
net.setInput(blob)
layer_names = net.getUnconnectedOutLayersNames()
outputs = net.forward(layer_names)

# Process detections
conf_threshold = 0.3  # Lowered confidence threshold to detect more objects
nms_threshold = 0.4
boxes = []
confidences = []
people_count = 0

for output in outputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]

        if confidence > conf_threshold and classes[class_id] == "person":
            center_x, center_y, w, h = (detection[:4] * np.array([width, height, width, height])).astype("int")
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            # Expand bounding box slightly for better detection in corners
            w = int(w * 1.1)
            h = int(h * 1.1)
            x = max(0, x)
            y = max(0, y)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))

# Apply Non-Maximum Suppression
indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

if len(indices) > 0:
    people_count = len(indices)

# Draw bounding boxes
for i in indices.flatten():
    x, y, w, h = boxes[i]
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Convert to RGB for plotting
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the image with bounding boxes
plt.figure(figsize=(10, 6))
plt.imshow(image_rgb)
plt.axis("off")
plt.title(f"Detected People Count: {people_count}")
plt.show()

print(f"Detected People Count: {people_count}")
```




```
Detected People Count: 5
```