YOLOv8 Person Counter — Line-by-Line Detailed Explanation

```
from ultralytics import YOLO
```

Imports the **YOLO** class from the **ultralytics** package, which is used to load and run the YOLOv8 model for **object detection**.

```
import cv2
```

Imports **OpenCV**, a computer vision library used for capturing, processing, and displaying video frames.

```
import cvzone
```

Imports **cvzone**, a helper library that makes it easy to draw styled **bounding boxes** and **labels** on frames.

```
import math
```

Imports the built-in **math** module, here used to format and round **confidence scores**.

```
feed = 'rtsp://admin:Admin@123@192.168.16.28/Streaming/Channels/101'
```

Defines the **RTSP URL** for the IP camera feed. This is the live video source.

```
cap = cv2.VideoCapture(feed)
```

Creates a **VideoCapture** object to connect to the RTSP stream and read frames using **cap.read()**.

```
model = YOLO('.../yolov8n.pt')
```

Loads the **YOLOv8n** model weights from the local path — initializing the object detector.

```
classNames = [...]
```

Defines all **COCO dataset** class names. YOLO returns numerical IDs that map to these labels (e.g., 0 = person).

```
while True:
```

Starts the **main loop** that continuously reads and processes frames from the camera until manually stopped.

```
success, img = cap.read()
```

Reads a single frame from the stream. **success** indicates if it succeeded; **img** holds the frame data.

```
if not success: ... break
```

If reading fails, it prints an error message and **breaks** the loop to safely exit.

```
results = model(img, stream=True)
```

Runs **YOLO detection** on the frame. The **stream=True** option makes it memory-efficient for video processing.

```
person_count = 0
```

Initializes a **counter** to count the number of people detected in the current frame.

```
for r in results: boxes = r.boxes
```

Iterates through each detection result and extracts the list of detected **bounding boxes**.

```
for box in boxes: ...
```

Processes each detected object individually.

```
cls = int(box.cls[0])
```

Gets the detected object's **class index** (e.g., 0 for 'person').

```
conf = math.ceil((box.conf[0] * 100)) / 100
```

Calculates the **confidence** value for the detection and rounds it to two decimal places.

```
if classNames[cls] == 'person' and conf > 0.5:
```

Filters detections — counts only when the detected class is **person** and **confidence > 0.5**.

```
x1, y1, x2, y2 = map(int, box.xyxy[0])
```

Extracts integer **bounding box coordinates** (top-left and bottom-right corners).

```
cvzone.cornerRect(img, (x1, y1, w, h))
```

Draws a stylish **corner rectangle** around the detected person using **cvzone**.

```
cvzone.putTextRect(img, f'Person {conf}', ...)
```

Displays a label showing **'Person'** and its confidence score on the video frame.

```
cv2.putText(img, f'Person Count: {person_count}', ...)
```

Draws the **total person count** on the top-left corner of the frame in green text.

```
cv2.imshow('RTSP Feed - YOLOv8 Person Counter', img)
```

Displays the live video feed with detections in an OpenCV window.

```
if cv2.waitKey(1) & 0xFF == ord('q'): break
```

Waits for keyboard input. Pressing **'q'** exits the loop and closes the feed.

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

Releases the video stream and **closes all OpenCV windows** — ensuring a clean exit.