**Problem 1:**

Use the YOLOv8 nano model to perform real-time object detection using your webcam.

**Tasks:**

1. Load the yolov8n.pt model.
2. Capture a live feed from your webcam using OpenCV.
3. Run real-time detection frame-by-frame and show the results live with bounding boxes.
4. Press q to quit the window.

*Hint:* Use cv2.VideoCapture(0) and model.predict() inside a loop.

**Problem 2:**

Choose a real-world object (e.g., your phone, a stapler, a plant) and build a mini dataset to train a detector.

**Tasks:**

1. Capture or collect at least 10 images of the object from different angles.
2. Annotate these images using any tool (e.g., [makesense.ai](https://www.makesense.ai/)) and export in YOLO format.
3. Split them into training and validation folders.
4. Organize the directory as:

/custom\_data

/images

/train

/val

/labels

/train

/val

1. Document the annotation tool used and challenges faced (1-2 sentences).

**Problem 3: Train a YOLOv8 Model on Your Custom Object**

Using the dataset from Problem 2, train a YOLOv8 model.

**Tasks:**

1. Create a **data.yaml** file with correct paths and class info.
2. Train the model using:

model = YOLO("yolov8n.pt")

model.train(data="data.yaml", epochs=10)

1. After training, describe the training performance:
   * Loss trends
   * Any issues (overfitting, underfitting)
   * Training time

**Problem 4:**

Test your custom-trained model on **3 new images** of your chosen object.

**Tasks:**

1. Load the trained model (best.pt) from the runs/detect/train/weights/ folder.
2. Run predictions on test images.
3. Display and save the results.
4. Write a short evaluation:
   * Was the object detected reliably?
   * Any false detections or missed ones?
   * Confidence levels?

**Problem 5:**

Use the Ultralytics YOLOv8 prediction results to extract and visualize detailed information.

**Tasks:**

1. For a single prediction image, extract the following:
   * Bounding box coordinates
   * Class label
   * Confidence score
2. Print the prediction information.
3. Draw custom bounding boxes and labels on the image using OpenCV (without results.show()).

📌 *Hint:* Use results[0].boxes.xyxy, results[0].boxes.conf, and results[0].boxes.cls.