Step 1 : Install the libraries

```
!pip install tensorflow tensorflow-hub opencv-python numpy
```

Step 2 : Copy this code and paste it in your Compiler

```
import cv2
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from time import time

class TFObjectDetection:
    def __init__(self, model_url='https://tfhub.dev/tensorflow/ssd_mobilenet_v2/2'):
        """
        Initialize the object detector with TensorFlow Hub model
        :param model_url: TF Hub model URL
        """
        # Load model from TF Hub
        self.model = hub.load(model_url)
        self.classes = {
            1: 'person', 2: 'bicycle', 3: 'car', 4: 'motorcycle', 5: 'airplane',
    6: 'bus', 7: 'train', 8: 'truck', 9: 'boat', 10: 'traffic light',
    11: 'fire hydrant', 13: 'stop sign', 14: 'parking meter', 15: 'bench',
    16: 'bird', 17: 'cat', 18: 'dog', 19: 'horse', 20: 'sheep',
    21: 'cow', 22: 'elephant', 23: 'bear', 24: 'zebra', 25: 'giraffe',
    27: 'backpack', 28: 'umbrella', 31: 'handbag', 32: 'tie', 33: 'suitcase',
    34: 'frisbee', 35: 'skis', 36: 'snowboard', 37: 'sports ball', 38: 'kite',
    39: 'baseball bat', 40: 'baseball glove', 41: 'skateboard', 42: 'surfboard', 43: 'tennis racket',
    44: 'bottle', 46: 'wine glass', 47: 'cup', 48: 'fork', 49: 'knife', 50: 'spoon',
    51: 'bowl', 52: 'banana', 53: 'apple', 54: 'sandwich', 55: 'orange', 56: 'broccoli',
    57: 'carrot', 58: 'hot dog', 59: 'pizza', 60: 'donut', 61: 'cake',
    62: 'chair', 63: 'couch', 64: 'potted plant', 65: 'bed', 67: 'dining table',
    70: 'toilet', 72: 'tv', 73: 'laptop', 74: 'mouse', 75: 'remote', 76: 'keyboard',
    77: 'cell phone', 78: 'microwave', 79: 'oven', 80: 'toaster', 81: 'sink', 82: 'refrigerator',
    84: 'book', 85: 'clock', 86: 'vase', 87: 'scissors', 88: 'teddy bear', 89: 'hair drier',
    90: 'toothbrush'
        }

        # Initialize webcam
        self.cap = cv2.VideoCapture(0)
        if not self.cap.isOpened():
            raise IOError("Cannot open webcam")

        # Frame counter and FPS calculation
        self.frame_count = 0
```

```
        self.fps = 0
        self.start_time = time()

    def process_frame(self, frame):
        """Process a single frame for object detection"""
        # Convert frame to tensor
        input_tensor = tf.convert_to_tensor(frame)
        input_tensor = input_tensor[tf.newaxis, ...]

        # Perform inference
        detections = self.model(input_tensor)

        # Parse detections
        boxes = detections['detection_boxes'][0].numpy()
        scores = detections['detection_scores'][0].numpy()
        classes = detections['detection_classes'][0].numpy().astype(np.int32)

        # Get frame dimensions
        height, width = frame.shape[:2]

        # Draw bounding boxes and labels
        for i in range(len(scores)):
            if scores[i] > 0.5:  # Confidence threshold
                class_id = classes[i]
                if class_id in self.classes:
                    class_name = self.classes[class_id]
                else:
                    class_name = 'unknown'

                # Convert box coordinates from normalized to pixel values
                ymin, xmin, ymax, xmax = boxes[i]
                x1, y1 = int(xmin * width), int(ymin * height)
                x2, y2 = int(xmax * width), int(ymax * height)

                # Draw rectangle
                color = (0, 255, 0)  # Green
                cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)

                # Display label and confidence
                label = f"{class_name}: {scores[i]:.2f}"
                cv2.putText(frame, label, (x1, y1 - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

        return frame

    def calculate_fps(self):
        """Calculate and display FPS"""
```

```
        self.frame_count += 1
        if self.frame_count >= 10:
            end_time = time()
            self.fps = self.frame_count / (end_time - self.start_time)
            self.frame_count = 0
            self.start_time = end_time

        # Display FPS on frame
        return self.fps

    def run(self):
        """Main loop for real-time detection"""
        try:
            while True:
                ret, frame = self.cap.read()
                if not ret:
                    break

                # Process frame
                processed_frame = self.process_frame(frame)

                # Calculate and display FPS
                fps = self.calculate_fps()
                cv2.putText(processed_frame, f"FPS: {fps:.2f}", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

                # Display result
                cv2.imshow('Real-Time Object Detection (TF)', processed_frame)

                # Exit on 'q' key
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break

        finally:
            self.cap.release()
            cv2.destroyAllWindows()

if __name__ == "__main__":
    detector = TFObjectDetection()
    detector.run()
```