```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

heading to last

```
# Counting the total number of the items in the images



import torch
import torchvision
import cv2
import numpy as np
from torchvision.transforms import functional as F
from google.colab.patches import cv2_imshow
from collections import Counter

model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

coco_cat_names = ['__background__', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
                  'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign',
                  'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
                  'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag',
                  'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite',
```

```
                'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket',
                'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana',
                'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
                'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table',
                'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
                'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock',
                'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']


def detect_obj(path, threshold=0.5, specific_item=None):
    img = cv2.imread(path)
    if img is None:
        print(f'Error: Could not load image from {path}. Please check the file path and ensure the
        return None, None

    original_img = img.copy()
    img_tensor = F.to_tensor(img)

    with torch.no_grad():
        pred = model([img_tensor])

    boxes = pred[0]['boxes'].cpu().numpy()
    labels = pred[0]['labels'].cpu().numpy()
    scores = pred[0]['scores'].cpu().numpy()

    object_counts = Counter()  # To store count of detected objects

    for i, box in enumerate(boxes):
        if scores[i] >= threshold:
            label_index = labels[i]
            if 0 <= label_index < len(coco_cat_names):
```

```
                label = coco_cat_names[label_index]
            else:
                label = f"Unknown Label ({label_index})"
                print(f"Warning: Encountered unknown label index: {label_index}")

            score = scores[i]

            start = (int(box[0]), int(box[1]))
            end = (int(box[2]), int(box[3]))
            cv2.rectangle(original_img, start, end, (0, 255, 0), 2)
            cv2.putText(original_img, f"{label}: {score:.2f}", start, cv2.FONT_HERSHEY_SIMPLEX, 0.5

            # Count the detected object
            object_counts[label] += 1

    # If a specific item is requested, filter the count for that item
    if specific_item:
        specific_item_count = object_counts.get(specific_item, 0)
        print(f"Count of '{specific_item}': {specific_item_count}")

    # Show the count of all detected objects
    print("\nObject counts:")
    for label, count in object_counts.items():
        print(f"{label}: {count}")

    return original_img, object_counts

if __name__ == '__main__':
    path = '/content/traffic.webp'  # Make sure this path is correct
    detected_image, object_counts = detect_obj(path)
```

```
    if detected_image is not None:
        cv2_imshow(detected_image)  # Display image in Colab

    # Example: Get the count of a specific item (e.g., 'cat')
    # detect_obj(path, specific_item="cat")
```

Object counts:
car: 56
truck: 5
bus: 2
traffic light: 1