g

Your single-letter query "g" is ambiguous and could refer to several things. Based on the search results, here are the most likely interpretations:

G is the seventh letter of the Latin alphabet, used in modern English and other Western European languages. Its name in English is "gee" (pronounced /ˈdʒiː/), plural "gees".[1]

Letter History

The letter G evolved from the Greek letter gamma (Γ), which was adopted by the Etruscan language and then borrowed by Latin as the rounded form C. The letter G was introduced in the Old Latin period as a variant of C to distinguish the voiced sound /g/ from the voiceless sound /k/. The recorded originator of G is freedman Spurius Carvilius Ruga, who added the letter to the Roman alphabet during the 3rd century BCE when he opened the first fee-paying school around 230 BCE.[1]

Typographic Forms

The lowercase "g" has two typographic variants: the single-storey (opentail) form and the double-storey (looptail) form. The single-storey form is commonly used in handwriting and fonts intended for children, while the double-storey form with a loop is traditional in serif typefaces. Most sans-serif typefaces use the opentail form.[1]

Usage in English

G is the tenth least frequently used letter in English, with a frequency of about 2.02% in words. In English, the letter G can represent different sounds depending on context, including /g/ (hard g) as in "go" and /d͡ʒ/ (soft g) as in "gem".[1]

⁂

give me more project ideas. (for simple projects).

the project idea should be rare.

Here are some rare and unconventional simple computer vision project ideas that you won't commonly see:

Unique Detection Projects

Plumbing Leak Detection monitors water pipes and surfaces for moisture accumulation by analyzing color changes and wet patches in real-time camera feeds. This system can alert homeowners or building managers before minor leaks escalate into costly water damage, making it valuable for property maintenance automation.[8]

Plant Growth Time-Lapse Tracker captures periodic images of plants and uses computer vision to measure growth rates by tracking leaf expansion, stem elongation, and flowering stages. Gardeners and researchers can visualize development patterns without manual measurement, making botanical studies more accessible.[9]

Refrigerator Inventory Monitor uses a camera inside your fridge to catalog food items, track expiration dates through label recognition, and suggest recipes based on available ingredients. This reduces food waste by alerting you when items are approaching spoilage and helps with automated grocery list generation.[10]

Posture Correction Assistant analyzes your sitting or standing position through webcam feeds and provides real-time alerts when you slouch or maintain poor ergonomic positions. This helps remote workers and students develop healthier habits during extended computer sessions.[11]

Pet Activity Monitor tracks your pet's movements throughout the day to identify unusual behavior patterns that might indicate health issues. The system learns normal activity levels and alerts owners when cats or dogs become unusually lethargic or hyperactive, serving as an early warning system for veterinary concerns.[8]

Unusual Recognition Systems

Signature Verification System compares handwritten signatures in real-time against stored references to detect forgeries on documents or digital signature pads. Banks and notary services can use this for authentication without requiring biometric hardware.[9]

Book Spine Reader scans your bookshelf to automatically catalog titles by recognizing text on book spines, creating a digital library inventory without manual data entry. This helps organize personal collections and can suggest reading recommendations based on identified genres.[10]

Coin Sorter and Counter identifies different coin denominations through size, color, and pattern recognition, automatically calculating total values from images or video feeds. This project teaches classification techniques while creating a practical tool for cash handling or numismatic cataloging.[11]

Recipe Ingredient Identifier analyzes photos of raw ingredients on your kitchen counter and suggests possible recipes you can prepare with those items. This combines object detection with database matching to reduce cooking planning time.[12]

Emotion-Based Music Controller detects facial expressions to determine your emotional state and automatically adjusts music playlists to match or elevate your mood. This creates a responsive entertainment system that adapts without manual playlist curation.[9]

Creative Monitoring Applications

Desk Organization Tracker monitors your workspace throughout the day and creates heatmaps showing which areas accumulate the most clutter. This helps identify organization patterns and suggests optimal desk arrangements based on item usage frequency.[8]

Mail and Package Detection System monitors your doorstep for deliveries and sends notifications when packages arrive, distinguishing between mail, boxes, and visitors. This prevents package theft by enabling immediate retrieval alerts.[12]

These projects offer practical solutions to everyday problems while remaining uncommon in project portfolios, making them excellent conversation starters during interviews or portfolio reviews.[11] [9]

⁂

i found a project to do. below i have attached the youtube video link. i am going to do that.

https://youtu.be/tbscP_d11Zw?si=JuOlCtvCEb1xWBH6

I can see you've chosen to work on a **YOLOv8 People Counting Project** - an excellent real-time computer vision project that tracks and counts people entering and exiting a location (like a hotel).[18]

Project Overview

This project combines multiple computer vision techniques to create a comprehensive people counting system using YOLOv8 for object detection, object tracking, and region-of-interest (ROI) based counting.[18]

Key Components

Object Detection uses the pre-trained YOLOv8s.pt model to detect people in video frames, filtering specifically for the "person" class from the COCO dataset. The system draws bounding boxes around detected individuals with class labels.[18]

Region of Interest (ROI) Definition involves creating two specific detection zones in front of a sliding door using CV2 polylines. You'll use a mouse function to capture X,Y coordinates and define polygonal areas where counting should occur.[18]

Object Tracking implements a custom tracker that assigns unique IDs to each person, maintaining identity across frames using the tracker.py file included in the repository. This ensures the same person isn't counted multiple times.[18]

Directional Counting Logic determines entry versus exit by tracking which ROI zone people pass through first. Entry requires passing through Area 2 then Area 1, while exit requires Area 1 then Area 2.[18]

Technical Implementation

The project uses **CV2.pointPolygonTest()** to detect when a person's bounding box bottom-right corner (X4, Y4) enters specific ROI zones. Dictionary storage maintains person IDs and positions as they move between zones, with set data structures preventing duplicate counts.[18]

Visual feedback differentiates entry (green rectangles) from exit (red rectangles), with real-time counters displaying total people entering and exiting.[18]

Prerequisites

You'll need to install OpenCV, PyTorch, and YOLOv8 on Windows, with the instructor providing setup tutorials referenced in the video. The project repository includes the tracker.py file, coco.txt class definitions, and sample video footage of a hotel entrance.[18]

This is a practical project that teaches object detection, tracking, spatial analysis with polygons, and state management - all valuable skills for computer vision applications in retail analytics, security systems, and facility management.[18]

⁂

i need to do that project with your help. tell me each and every steps to do. consider i am a beginner for these type of things. so you need to give me every steps.

Perfect! I'll guide you through every single step to complete this YOLOv8 People Counting project. Let's start from the very beginning.

Step 1: Install Python

Check if Python is already installed:

- Press `Windows + R` keys together
- Type `cmd` and press Enter
- In the black window that opens, type: `python --version` and press Enter

If you see a version number (like Python 3.10.x), Python is installed. If not, follow these steps:[19]

- Go to https://www.python.org/downloads/
- Download Python (version 3.9 to 3.11 recommended)
- Run the installer
- **IMPORTANT:** Check the box "Add Python to PATH" before clicking Install

Step 2: Install Required Libraries

Open Command Prompt (press `Windows + R`, type `cmd`, press Enter) and run these commands one by one:[20] [19]

```
pip install opencv-python
```

Wait for it to complete, then:

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu1
```

Wait for it to complete, then:

```
pip install ultralytics
```

This last command installs YOLOv8.[20]

Step 3: Download the Project Repository

- Open your web browser
- Go to the GitHub link mentioned in the video description
- Click the green "Code" button
- Click "Download ZIP"
- Once downloaded, right-click the ZIP file and select "Extract All"
- Choose a location (like Desktop or Documents) and extract it

Step 4: Download the Video File

According to the video, the instructor provides a video file link in the repository:[21]

- Look inside the extracted folder for a text file with a mega.nz link
- Copy that link and paste it in your browser
- Download the video file
- Extract it if it's in a ZIP file
- Place the video file in the main project folder (people-counter-yolov8-main)

Step 5: Set Up Your Code Editor

Install Visual Studio Code (recommended for beginners):

- Go to https://code.visualstudio.com/
- Download and install it
- Open Visual Studio Code

- Click "File" → "Open Folder"
- Navigate to your extracted project folder and select it

Step 6: Create the Main Python File

Inside your project folder, you should see:

- `main.py` (the main code file)
- `tracker.py` (the tracking helper file)
- `coco.txt` (list of object classes)
- Your video file

Step 7: Understanding the Code Structure

The project has **four main parts**:[21]

Part 1: Object Detection - Detects people using YOLOv8
Part 2: Draw Region of Interest (ROI) - Creates two zones to monitor
Part 3: Object Tracking - Assigns unique IDs to each person
Part 4: Counting Logic - Counts people entering and exiting

Step 8: Run the Basic Detection Code

First, create a simple test file to ensure everything works:

Create a new file called `test.py` in your project folder with this code:

```
from ultralytics import YOLO
import cv2

# Load the model
model = YOLO('yolov8s.pt')

# Test with webcam
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Detect objects
    results = model(frame)

    # Show results
    annotated_frame = results[^4_0].plot()
    cv2.imshow('YOLOv8 Test', annotated_frame)

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

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

To run this:

- Open Command Prompt
- Navigate to your project folder: `cd C:\path\to\your\project\folder`
- Run: `python test.py`
- Press 'q' to quit

Step 9: Implement the Mouse Callback Function

This helps you get coordinates for drawing ROI zones. Add this to your `main.py`:[21]

```
import cv2
import numpy as np

def RGB(event, x, y, flags, param):
    if event == cv2.EVENT_MOUSEMOVE:
        print(f"X: {x}, Y: {y}")

cv2.namedWindow('Frame')
cv2.setMouseCallback('Frame', RGB)
```

Step 10: Get the Full Code Template

Here's the complete structure you need to build in `main.py`:

```
from ultralytics import YOLO
import cv2
import numpy as np
from tracker import Tracker

# Load COCO classes
with open('coco.txt', 'r') as f:
    class_list = f.read().split('\n')

# Initialize model and tracker
model = YOLO('yolov8s.pt')
tracker = Tracker()

# Define ROI areas (you'll fill these with your coordinates)
area1 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])
area2 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])

# Tracking sets
entering = set()
exiting = set()

# Dictionary to store people positions
people_entering = {}
```

```
people_exiting = {}

# Mouse callback for getting coordinates
def RGB(event, x, y, flags, param):
    if event == cv2.EVENT_MOUSEMOVE:
        print(f"X: {x}, Y: {y}")

# Load video
cap = cv2.VideoCapture('your_video.mp4')

cv2.namedWindow('Frame')
cv2.setMouseCallback('Frame', RGB)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Detect objects
    results = model(frame)

    # Process detections
    detections = []
    for r in results:
        boxes = r.boxes
        for box in boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[^4_0])
            conf = box.conf[^4_0]
            cls = int(box.cls[^4_0])
            class_name = class_list[cls]

            if 'person' in class_name:
                detections.append([x1, y1, x2, y2])

    # Update tracker
    bbox_id = tracker.update(detections)

    # Draw ROI zones
    cv2.polylines(frame, [area1], True, (0, 255, 0), 2)
    cv2.polylines(frame, [area2], True, (0, 255, 0), 2)
    cv2.putText(frame, '1', (510, 465), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
    cv2.putText(frame, '2', (510, 555), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)

    # Process tracked objects
    for bbox in bbox_id:
        x3, y3, x4, y4, id = bbox

        # Check if in Area 2 (entry detection)
        result2 = cv2.pointPolygonTest(np.array(area2, np.int32), (x4, y4), False)
        if result2 >= 0:
            people_entering[id] = (x4, y4)

        # Check if in Area 1 (and was in Area 2)
        if id in people_entering:
            result1 = cv2.pointPolygonTest(np.array(area1, np.int32), (x4, y4), False)
            if result1 >= 0:
```

```
                cv2.rectangle(frame, (x3, y3), (x4, y4), (0, 255, 0), 2)
                cv2.circle(frame, (x4, y4), 4, (255, 0, 255), -1)
                cv2.putText(frame, str(id), (x3, y3), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0,
                entering.add(id)

    # Display counters
    cv2.putText(frame, f'Entering: {len(entering)}', (50, 50), cv2.FONT_HERSHEY_SIMPLEX,

    cv2.imshow('Frame', frame)

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

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

Step 11: Get Coordinates for Your ROI

Run the code and move your mouse to these positions to get coordinates:[21]

- Top-left corner of Area 1
- Top-right corner of Area 1
- Bottom-right corner of Area 1
- Bottom-left corner of Area 1
- Repeat for Area 2

Write down these coordinates and update them in the `area1` and `area2` arrays.

Step 12: Add the Exit Counting Logic

After the entry counting code, add similar logic for exits:[21]

```
# Check if in Area 1 (exit detection)
result1_exit = cv2.pointPolygonTest(np.array(area1, np.int32), (x4, y4), False)
if result1_exit >= 0:
    people_exiting[id] = (x4, y4)

# Check if in Area 2 (and was in Area 1)
if id in people_exiting:
    result2_exit = cv2.pointPolygonTest(np.array(area2, np.int32), (x4, y4), False)
    if result2_exit >= 0:
        cv2.rectangle(frame, (x3, y3), (x4, y4), (0, 0, 255), 2)
        exiting.add(id)

# Display exit counter
cv2.putText(frame, f'Exiting: {len(exiting)}', (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0
```

Step 13: Test and Debug

Run your code: `python main.py`

Common issues and fixes:

- **"No module named 'ultralytics'"**: Run `pip install ultralytics` again
- **"Video file not found"**: Check the video filename matches exactly
- **Model not downloading**: Make sure you have internet connection
- **Press 'q' doesn't work**: Click on the video window first, then press 'q'

Step 14: Fine-tune Your Project

Adjust these parameters for better results:[21]

- ROI area coordinates for better coverage
- Confidence threshold for detections
- Rectangle colors for better visibility
- Text positions and sizes

Would you like me to help you with any specific step in more detail?

⁂

PS D:\People-Detection-Counting> python main.py

Traceback (most recent call last):
File "D:\People-Detection-Counting\main.py", line 4, in <module>
from tracker import Tracker
ImportError: cannot import name 'Tracker' from 'tracker' (D:\People-Detection-Counting\tracker.py)
PS D:\People-Detection-Counting>

The error shows that the `Tracker` class isn't properly defined in your `tracker.py` file. Let me help you fix this.

Solution: Create the Tracker Class

You need to create or fix the `tracker.py` file. Here's the complete code for `tracker.py`:

Open or create `tracker.py` in your project folder and paste this code:

```
import math

class Tracker:
    def __init__(self):
```

```
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by one
        self.id_count = 0

    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x1, y1, x2, y2 = rect
            cx = (x1 + x2) // 2
            cy = (y1 + y2) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[^5_0], cy - pt[^5_1])

                if dist < 35:
                    self.center_points[id] = (cx, cy)
                    objects_bbs_ids.append([x1, y1, x2, y2, id])
                    same_object_detected = True
                    break

            # New object is detected we assign the ID to that object
            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x1, y1, x2, y2, self.id_count])
                self.id_count += 1

        # Clean the dictionary by center points to remove IDS not used anymore
        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        # Update dictionary with IDs not used removed
        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```

Verify the Fix

After creating/updating `tracker.py`, run this command to verify:

```
python -c "from tracker import Tracker; print('Success!')"
```

If you see "Success!" then the tracker is working correctly.

Now Run Your Main Code

```
python main.py
```

Additional Check

Make sure your file structure looks like this:

```
D:\People-Detection-Counting\
├── main.py
├── tracker.py          ← Make sure this file exists with the code above
├── coco.txt
└── your_video.mp4
```

If you still get an error, please share:

1. What's currently inside your `tracker.py` file?
2. The exact error message you receive

This should resolve your import error and allow the tracking functionality to work properly.[30]

⁂

PS D:\People-Detection-Counting> python main.py

Traceback (most recent call last):
File "D:\People-Detection-Counting\main.py", line 15, in <module>
area1 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])
^^
NameError: name 'x1' is not defined

The error shows that you're trying to use `x1, y1, x2, y2, x3, y3, x4, y4` variables that don't exist yet. You need to replace these with actual coordinate numbers.

Solution: Get Coordinates First

You have two options:

Option 1: Use Temporary Coordinates (Quick Start)

Replace these lines in `main.py`:

```
area1 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])
area2 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])
```

With temporary placeholder coordinates:

```
area1 = np.array([[312, 388], [289, 390], [474, 469], [497, 462]])
area2 = np.array([[279, 392], [250, 397], [283, 481], [304, 473]])
```

These are example coordinates - you'll adjust them later to match your video.[31]

Option 2: Get Exact Coordinates for Your Video (Recommended)

Create a separate file called `get_coordinates.py` to find your exact ROI coordinates:

```
import cv2

# Global list to store coordinates
coordinates = []

def mouse_callback(event, x, y, flags, param):
    global coordinates

    if event == cv2.EVENT_LBUTTONDOWN:
        # Left click to add coordinate
        coordinates.append([x, y])
        print(f"Coordinate {len(coordinates)}: X={x}, Y={y}")

        # Draw a circle at clicked point
        cv2.circle(frame, (x, y), 5, (0, 255, 0), -1)
        cv2.imshow('Get Coordinates', frame)

    elif event == cv2.EVENT_MOUSEMOVE:
        # Show current mouse position
        temp_frame = frame.copy()
        cv2.putText(temp_frame, f'X: {x}, Y: {y}', (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
        cv2.imshow('Get Coordinates', temp_frame)

# Load your video
cap = cv2.VideoCapture('your_video.mp4')  # Replace with your video filename

# Read first frame
ret, frame = cap.read()
if not ret:
    print("Error: Cannot read video file!")
    exit()

# Create window and set mouse callback
cv2.namedWindow('Get Coordinates')
cv2.setMouseCallback('Get Coordinates', mouse_callback)

print("\n=== INSTRUCTIONS ===")
print("1. Click 4 points for Area 1 (in order: top-left, top-right, bottom-right, bottom-
print("2. Click 4 points for Area 2 (same order)")
print("3. Press 'r' to reset coordinates")
print("4. Press 'q' when done")
print("====================\n")

cv2.imshow('Get Coordinates', frame)
```

```
while True:
    key = cv2.waitKey(1) & 0xFF

    if key == ord('q'):
        break
    elif key == ord('r'):
        # Reset coordinates
        coordinates = []
        ret, frame = cap.read()
        cv2.imshow('Get Coordinates', frame)
        print("Coordinates reset!")

cap.release()
cv2.destroyAllWindows()

# Print the final coordinates
print("\n=== YOUR COORDINATES ===")
if len(coordinates) >= 4:
    print(f"\narea1 = np.array([{coordinates[^6_0]}, {coordinates[^6_1]}, {coordinates[^6
if len(coordinates) >= 8:
    print(f"area2 = np.array([{coordinates[^6_4]}, {coordinates[^6_5]}, {coordinates[^6_6
print("\nCopy these lines into your main.py file!")
```

Run this file:

```
python get_coordinates.py
```

How to use it:

1. The video's first frame will appear
2. Click 4 points around the first area you want to monitor (Area 1) - click in order: top-left, top-right, bottom-right, bottom-left
3. Click 4 points for Area 2 (same order)
4. Press 'q' when done
5. Copy the printed `area1` and `area2` lines into your `main.py`

Quick Fix to Run Now

If you just want to test that everything works, update your `main.py`:

Find these lines:

```
area1 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])
area2 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])
```

Replace with:

```
# Temporary coordinates - adjust these based on your video
area1 = np.array([[312, 388], [289, 390], [474, 469], [497, 462]])
area2 = np.array([[279, 392], [250, 397], [283, 481], [304, 473]])
```

Also make sure your video filename is correct:

```
cap = cv2.VideoCapture('your_video.mp4')  # Change 'your_video.mp4' to your actual video
```

Now try running:

```
python main.py
```

This should work! Once it runs, you can use the coordinate tool to get the exact positions for your specific video.[31]

⁂

0: 384×640 21 persons, 1 bus, 5 handbags, 4 potted plants, 152.4ms

Speed: 16.7ms preprocess, 152.4ms inference, 8.0ms postprocess per image at shape (1, 3, 384, 640)
Traceback (most recent call last):
File "D:\People-Detection-Counting\main.py", line 53, in <module>
class_name = class_list[cls]
~~~~~~~~~~^^^^^
IndexError: list index out of range

The error shows that the `coco.txt` file isn't loading correctly, so the `class_list` is empty or too short. Let me help you fix this.

Solution: Create/Fix the coco.txt File

Create a new file called `coco.txt` in your project folder (`D:\People-Detection-Counting\`) and paste this content:

```
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
```

Make sure:

- Each class name is on a **separate line**
- There are **no empty lines** at the beginning
- Save the file as `coco.txt` (not `coco.txt.txt`)

Alternative: Fix the Code to Handle Missing Classes

If you want to make your code more robust, update the detection processing section in `main.py`:

Find this section:

```
for r in results:
    boxes = r.boxes
    for box in boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[^7_0])
        conf = box.conf[^7_0]
        cls = int(box.cls[^7_0])
        class_name = class_list[cls]

        if 'person' in class_name:
            detections.append([x1, y1, x2, y2])
```

Replace with this safer version:

```
for r in results:
    boxes = r.boxes
    for box in boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[^7_0])
        conf = box.conf[^7_0]
        cls = int(box.cls[^7_0])

        # Check if class index is valid
        if cls < len(class_list):
            class_name = class_list[cls]
        else:
            class_name = f"class_{cls}"  # Fallback name
```

```
        if 'person' in class_name:
            detections.append([x1, y1, x2, y2])
```

Verify the Fix

Check if your `coco.txt` file loads correctly:

Run this test:

```
python -c "with open('coco.txt', 'r') as f: classes = f.read().split('\n'); print(f'Loade
```

You should see:

```
Loaded 80 classes
First class: person
Last class: toothbrush
```

Now Run Your Main Code

```
python main.py
```

The error should be resolved! The detection message you saw (`0: 384x640 21 persons, 1 bus, 5 handbags, 4 potted plants, 152.4ms`) shows that YOLOv8 is working perfectly - it detected 21 people in the frame.[32]

If you still get an error, please share:

1. The contents of your `coco.txt` file (first 5 lines)
2. The output of the verification command above

⁂

1. https://en.wikipedia.org/wiki/G
2. https://www.google.com
3. https://www.gadventures.com
4. https://gshock.casio.com/intl/
5. https://blog.google/inside-google/company-announcements/gradient-g-logo-design/
6. https://developer.mozilla.org/en-US/docs/Web/SVG/Reference/Element/g
7. https://www.instagram.com/gflip/
8. https://www.reddit.com/r/computervision/comments/va85ll/any_ideas_for_a_computer_vision_project/
9. https://www.datacamp.com/blog/computer-vision-projects
10. https://www.sabrepc.com/blog/Deep-Learning-and-AI/10-Computer-Vision-Applications-for-Todays-World
11. https://neptune.ai/blog/15-computer-visions-projects

12. https://www.projectpro.io/article/computer-vision-projects/437
13. https://www.reddit.com/r/computervision/comments/1gshjte/what_was_the_strangest_computer_vision_project/
14. https://opencv.org/blog/top-computer-vision-projects/
15. https://milvus.io/ai-quick-reference/what-are-some-good-computer-vision-projects
16. https://www.youtube.com/watch?v=eCDTp5h59hg
17. https://www.reddit.com/r/computervision/comments/60oupx/what_are_some_of_the_most_exciting_applications/
18. https://www.youtube.com/watch?v=tbscP_d11Zw
19. https://www.geeksforgeeks.org/python/how-to-install-opencv-for-python-in-windows/
20. https://yolov8.org/how-to-install-yolov8/
21. https://www.youtube.com/watch?v=tbscP_d11Zw
22. https://www.youtube.com/watch?v=w-zHT0Wvdt8
23. https://stackoverflow.com/questions/47754749/how-to-install-pytorch-in-windows
24. https://www.exxactcorp.com/blog/deep-learning/yolov8-setup-tutorial-for-object-detection
25. https://hackmd.io/@Yucheng208/YOLOv8-Tutorial
26. https://www.dfrobot.com/blog-13903.html
27. https://yolov8.com
28. https://breakthrough.github.io/Installing-OpenCV/
29. https://pytorch.org/get-started/locally/
30. https://www.youtube.com/watch?v=tbscP_d11Zw
31. https://www.youtube.com/watch?v=tbscP_d11Zw
32. https://www.youtube.com/watch?v=tbscP_d11Zw