INTERNSHIP NOTES

Created	@October 22, 2025 11:04 PM
Class	IIITH
Event	

The core concepts and functions used in the Python code blocks, focusing on the `ultralytics` library.

Task1: Basic image segmentation and classification

TASK 1: Code

Explanation of the YOLOv8 Python Code

The core of the procedure relies on the **`ultralytics`** Python package, which provides a simple, high-level API for using the powerful YOLO models.

Core Components

The two essential lines that appear in most steps are:

1. `from ultralytics import YOLO`
2. `model = YOLO("model_file.pt")`

- **`from ultralytics import YOLO`**: This imports the main **`YOLO`** **class** from the installed library. This class is the central object you'll interact with for loading models, running predictions, and accessing results.
- **`model = YOLO("yolov8n.pt")`**: This **loads the pre-trained model**.
  - **`YOLO()`**: Creates an instance of the YOLO object.

- `"yolov8n.pt"`: This is the model file. YOLOv8 comes in different sizes (nano 'n', small 's', medium 'm', etc.). The 'n' stands for **nano**, the smallest and fastest version, which is great for quick testing. The file is downloaded automatically the first time you run the script.
  - For **detection**, you use `yolov8n.pt`.
  - For **segmentation** (Step 6), you use `yolov8n-seg.pt`, which is a segmentation-specific version.

Explanation of the `model.predict()` Function

The `model.predict()` function is the single, unified way to run the model on data (images, videos, or streams). Here is a breakdown of the arguments used:

`results = model.predict(...)`

Argument	Example Value	Purpose
`source`	`"https://..."` or `r"C:\..."`	**Mandatory**: Specifies the input to process. This can be a local **file path** (image/video), a **folder path** (to process all images), or a **URL**.
`show`	`True` or `False`	If `True`, a window will pop up to display the results in real-time as the model processes the image/video. Useful for live preview.
`save`	`True` or `False`	If `True`, the annotated image (with bounding boxes or masks) is saved to the output directory.
`project`	`r"C:\Users\...\YOLO_Outputs"`	**Optional**: The **root directory** where all prediction runs are saved. If not specified, it defaults to a folder like `runs/detect` or `runs/segment` in your current working directory.
`name`	`"Paypal_Mafia_2014"`	**Optional**: The **specific folder name** for this particular run, created inside the `project` directory. This keeps your experiments organized.
`classes`	`[0]`	**Optional/Crucial for Filtering**: This is a **list of class IDs** you want the model to process or

		display. It's used to filter results.

- YOLO is trained on the **COCO dataset**, where **class ID `0` always corresponds to a "person"**.
  * By setting `classes=[0]`, you instruct the model to only output results (bounding boxes or segmentation masks) for detected persons, ignoring all other objects (cars, bikes, animals, etc.).

The `results` Variable

The line `results = model.predict(...)` captures the output.

- The `results` variable is a list of `Results` **objects**. For a single image prediction, it usually contains one `Results` object.
- This object contains all the detailed, low-level information about the prediction, such as:
  - The coordinates of all bounding boxes.
  - The confidence score for each detection.
  - The segmentation masks (for segmentation models).
  - The detected class IDs.

The Key Difference: Detection vs. Segmentation

The main difference between Step 5 and Step 6 is which model is loaded:

Feature	Step 5: Object Detection	Step 6: Instance Segmentation
Goal	Find objects and draw a **bounding box** around them.	Find objects and generate a pixel-level **mask** for the exact shape of the object.
Model File	`yolov8n.pt`	`yolov8n-seg.pt`
Output	Rectangles/Boxes around people.	**Pixel-perfect shapes** (masks) highlighting the entire area of each person.

By using the `yolov8n-seg.pt` model and the `classes=[0]` filter, you can run an , person-only instance segmentation task on your local image.