Object Detection Code explanation:

This imports the OpenCV library, which is essential for image and video processing tasks.

Python code

```
thres = 0.45  # Threshold to detect object
```

Sets a threshold value for object detection. Objects with a detection confidence below this threshold will be ignored.

Python code

```
cap = cv2.VideoCapture(0)  # Change camera index if needed
```

Initializes video capture from the default camera (index 0). If you have multiple cameras, you can change the index to select a different one.

```
cap.set(3, 1280)
cap.set(4, 720)
cap.set(10, 70)
```

Configures the video capture settings:

- `cap.set(3, 1280)`: Sets the width of the video frame to 1280 pixels.
- `cap.set(4, 720)`: Sets the height of the video frame to 720 pixels.
- `cap.set(10, 70)`: Sets the brightness of the video capture (70 is an arbitrary value).

Python code

```
classNames = []
classFile = 'coco.names'
with open(classFile, 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')
```

Loads the class names from the 'coco.names' file, which contains names of the object categories. The file is read line by line, and each line is stored in the `classNames` list.

```
configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'
```

Specifies the paths to the configuration file (`.pbtxt`) and the pre-trained model weights (`.pb`). These files are necessary for the object detection model.

Python code

```
net = cv2.dnn_DetectionModel(weightsPath, configPath)
```

Creates a detection model using the specified configuration and weights files.

Python code

```
net.setInputSize(320, 320)
net.setInputScale(1.0 / 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

Configures the input parameters for the detection model:

- `net.setInputSize(320, 320)`: Sets the input size of the model to 320x320 pixels.
- `net.setInputScale(1.0 / 127.5)`: Normalizes the input by scaling the pixel values.
- `net.setInputMean((127.5, 127.5, 127.5))`: Sets the mean subtraction values for preprocessing.
- `net.setInputSwapRB(True)`: Swaps the red and blue channels, as the model expects the input in RGB format.

```
while True:
    success, img = cap.read()
```

Starts a loop to continuously capture frames from the camera. `cap.read()` reads a frame from the camera, returning a boolean (`success`) and the frame itself (`img`).

Python code

```
    classIds, confs, bbox = net.detect(img, confThreshold=thres)
    print(classIds, bbox)
```

Performs object detection on the captured frame (`img`). The method `net.detect` returns:

- `classIds`: List of detected object class IDs.
- `confs`: List of confidence scores for each detected object.
- `bbox`: List of bounding boxes for each detected object. The detected class IDs and bounding boxes are printed.

Python code

```
    if len(classIds) != 0:
        for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
```

Checks if any objects were detected (`len(classIds) != 0`). If so, it iterates through each detected object, flattening the arrays for easier iteration.

Python code

```
            cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)
            cv2.putText(img, classNames[classId - 1].upper(), (box[0] + 10, box[1] + 30),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
            cv2.putText(img, str(round(confidence * 100, 2)), (box[0] + 200, box[1] +
30),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
```

For each detected object:

- Draws a bounding box around the detected object using `cv2.rectangle`.
- Puts the class name (label) of the detected object above the bounding box using `cv2.putText`.
- Puts the confidence score of the detected object next to the label.

Python code

```
    cv2.imshow("Output", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Displays the processed frame with detections in a window named "Output". If the 'q' key is pressed, the loop breaks, stopping the video capture.

Python code

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

Releases the camera and closes all OpenCV windows.

Conclusion

This code sets up a real-time object detection system using OpenCV, a pre-trained neural network model, and the COCO dataset. The model captures frames from the camera, processes them to detect objects, and displays the results with bounding boxes and labels.