```python
from shapely.geometry import Polygon


# Identification.py
# Authors: Diogo RosÃ¡rio, JoÃ£o Raposo
# Description: This file defines the Identification object used in main.py. It aims to represent an identification with the following attributes:
# - x: Float representing the starting width in pixels of the identification
# - y: Float representing the starting height in pixels of the identification
# - width: Width of the identification
# - height: Height of the identification
# - confidence: Confidence value indicating the drone's belief in the identification as a Car, House, or Tree
# - class_type: Type of identification, either "Car," "House," or "Tree"
# - drone_confidence: Confidence/reputation value specific to each drone; a static value
class Identification():

    def __init__(self, x, y, width, height, confidence, class_type, drone, drone_confidence):
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.confidence = confidence
        self.class_type = class_type
        self.drone = drone
        self.drone_confidence= drone_confidence

    # Creates two polygons that correspond to two identifications(rectangle)
    # Checks if to identifications collide with each other
    def checkCollision(self, other):
        # Create polygons for each rectangle
        poly1 = Polygon([(self.x , self.y),
                         (self.x + self.width , self.y),
                         (self.x + self.width , self.y + self.height),
                         (self.x , self.y + self.height)])

        poly2 = Polygon([(other.x , other.y),
                         (other.x + other.width , other.y),
                         (other.x + other.width , other.y + other.height),
                         (other.x , other.y + other.height)])

        # Check if the polygons (rectangles) intersect
        return poly1.intersects(poly2) or poly2.intersects(poly1)

    # Function that compares two identifications.
    # Used to sort a list of identifications
    def comparator(this, other):
        confidence_this = this.confidence * this.drone_confidence
        confidence_other = other.confidence * other.drone_confidence

        if(confidence_this > confidence_other):
            return 1
        elif(confidence_this < confidence_other):
            return -1
        else:
            if(this.drone_confidence > other.drone_confidence):
                return 1
            elif(this.drone_confidence < other.drone_confidence):
                return -1
            else:
                return 0


    # String representation of this object (Identification)
    def __str__(self):
        return "[\nClass: " + self.class_type + "\n" + "Confidence: " + str(self.confidence) + "\n" + "x: " + str(self.x) + "\n" + "y: " + str(self.y)

```