```python
""" personne_tracking.py | Robin Forestier | 09.03.2022

After Personne detection we want to track it.
For tracking the displacement of a mooving object, I start by calculate his centroide.
After I save his last centroide to ave 2 points by mooving object.
With this points a calculate the euclidean distance to find the nearest.
And I finishe by calculate the angle of displacement.
"""

import cv2
import math
import numpy as np

# It's importing the PersonneDetect class from the personne_detect.py file.
from personne_detect import PersonneDetect

class PersonneTracking:
    """Is used for track the trajectory of any people detected by PersonneDetect."""
    def __init__(self):
        """The function initializes the class"""
        self.img = []
        self.prev_img = []
        self.centroide = []
        self.centroide_lp = []

        self.angle = []

    def calc_centroide(self, img, rects):
        """Calculate the centroid of the bounding rect

        :param img: The image on which the contour was found
        :type img: numpy.ndarray
        :param rects: a list of tuples, where each tuple is (x, y, w, h)
        :type rects: list
        """

        self.img = img
        # save the last centroid
        self.centroide_lp = self.centroide
        self.centroide = []

        for rect in rects:
            # It's the coordinates of the point where the line is drawn.
            x = int(rect[0] + (rect[2] / 2))
            y = int(rect[1] + (rect[3] / 2))
            self.centroide.append((x,y))
            # It's drawing a circle on the image.
            cv2.circle(self.img, (x, y), 2, (0,0, 255), -1)

        # It's calculating the euclidean distance of each centroid and last centroid
        for predict the move of a person.
        self.centroide_last_pose()

    def centroide_last_pose(self):
        """Calculate the angle of displacement of each centroid and last centroid"""

        self.angle = []

        centroide_np = np.array(self.centroide_lp)

        if self.centroide_lp and self.centroide:
            for last_point in self.centroide:
                # This is the code that is used to find the index of the minimum
                value in the array.
                idx = np.array([np.linalg.norm(x + y) for (x, y) in centroide_np -
                last_point]).argmin()

                cv2.circle(self.img, last_point, 2, (255, 0, 0), -1)
                cv2.line(self.img, last_point, self.centroide_lp[idx], (255, 255,
                0), 5, cv2.LINE_AA)

                # It's calculating the angle of the line between the two points.
```

```
                y = last_point[1] - self.centroide_lp[idx][1]
                x = last_point[0] - self.centroide_lp[idx][0]
                angle = math.atan2(y, x) * 180 / math.pi

                # It's making sure that the angle is between 0 and 360 degrees.
                if angle < 0:
                    angle = 360 + angle

                # add it to the list angle
                self.angle.append(angle)

if __name__ == '__main__':
    # It's using the video file to capture the frames.
    cap = cv2.VideoCapture('video_d.mp4')

    # It's creating an object of the class PersonneDetect.
    p = PersonneDetect()
    # It's creating an instance of the class PersonneTracking.
    pt = PersonneTracking()

    while True:
        # This is a way to reset the video to the first frame if the video is
        finished.
        if cap.get(cv2.CAP_PROP_POS_FRAMES) == cap.get(cv2.CAP_PROP_FRAME_COUNT):
            cap.set(cv2.CAP_PROP_POS_FRAMES, 0)

        # It's getting the image from the video.
        _, img = cap.read()

        # It's using the PersonneDetect class to detect people in the image.
        result = p.personne_detect(img)
        # It's calculating the centroid of the bounding rect of the detected people.
        pt.calc_centroide(result, p.detected)

        # It's showing the image on the screen.
        cv2.imshow("result detect", result)

        prev = img

        # It's breaking the loop when the user press the `q` key.
        if cv2.waitKey(700) == ord('q'):
            break

    # It's closing the window and release the capture.
    cv2.destroyAllWindows()
    cap.release()
```