```python
"""door_detect.py | Robin forestier | 08.03.2022

This code is used for finding the door with the existing template create by <door_select.py >.
"""

import glob
import cv2

# It's used to capture the video from the camera.
cap = cv2.VideoCapture(0)


def load_images_from_folder():
    """Load all the template images from the current directory

    :return: A list of images.
    :rtype: list
    """

    # It's a function that return a list of all the files with the extension .png in the current directory.
    filenames = glob.glob("*.png")
    # Sort it by name
    filenames.sort()
    images = []

    # It's a loop for loading all the images in the current directory.
    for img in filenames:
        n = cv2.imread(img)
        if n is not None:
            print("[INFO] Door template loaded.")
            images.append(n)
        else:
            print("[Error]  " + img + " Not load.")

    return images


def detectDors(img, template):
    """We use template matching to detect the doors

    :param img: The image we want to detect the template on
    :type img: numpy.ndarray
    :param template: the template image
    :type template: numpy.ndarray
    :return: the image with the rectangles around the detected doors, the max location of the template and the width and
    height of the template.
    :rtype: numpy.ndarray, tuple, tuple
    """

    # It's converting the image from BGR to gray.
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGRA2GRAY)
    # It's making a copy of the image to draw the rectangles on it.
    copy = img.copy()

    # for each template
    for tmp in template:
        # It's converting the template from BGR to gray.
        tmp = cv2.cvtColor(tmp, cv2.COLOR_BGRA2GRAY)
        # It's getting the width and the height of the template.
        w, h = tmp.shape[::-1]
        # It's matching the template to the image.
        res = cv2.matchTemplate(gray_img, tmp, cv2.TM_CCOEFF_NORMED)
        # Normalize result
        cv2.normalize(res, res, 0, 1, cv2.NORM_MINMAX, -1)
        # Detect the max location.
        (_, max_val, _, max_loc) = cv2.minMaxLoc(res)

        # Draw the rect around the detected template
        cv2.rectangle(copy, max_loc, (max_loc[0] + w, max_loc[1] + h), (255, 0, 0), 2)
```

```
70          cv2.rectangle(copy, (max_loc[0] + 1, max_loc[1] + 1), (max_loc[0] + w,
            max_loc[1] + int(h / 2)), (255,255,0), -1)
71          cv2.rectangle(copy, (max_loc[0] + 1, max_loc[1] + h - 1), (max_loc[0] + w,
            max_loc[1] + int(h / 2) + 1), (0, 255, 255), -1)
72
73      return copy, max_loc, w, h
74
75
76  if __name__ == '__main__':
77      # It's loading all the template images from the current directory.
78      template = load_images_from_folder()
79
80      while True:
81          # It's getting the image from the camera and storing it in the variable `img`.
82          _, img = cap.read()
83
84          # resize image form (2592, 1944) -> (640, 480)
85          img = cv2.resize(img, (640, 480), interpolation=cv2.INTER_AREA)
86          result = detectDors(img, template)
87          # It's showing the image with the rectangles around the detected template.
88          cv2.imshow("Result", result)
89
90          # It's waiting for the user to press the key `q` to quit the program.
91          if cv2.waitKey(1) == ord("q"):
92              break
93
94      # It's closing the camera and the windows.
95      cv2.destroyAllWindows()
96      cap.release()
97
```