```
# General purpose
import os
import numpy as np
from time import sleep
import time
import digital_tube_control as dtc
import button as btn
import pi_camera as pc
import picamera as piC


# GPIO related
import RPi.GPIO as GPIO

# camera related
from picamera import PiCamera, Color

# GPIO mode: GPIO.BOARD, GPIO.BCM
GPIO.setmode(GPIO.BOARD)
mode = GPIO.getmode()
# Close GPIO warning
GPIO.setwarnings(False)

# get the project path
PRJ_PATH = os.getcwd()

def image_split_column(a)->list:
    ret = []
    startList = []
    endList = []
    height = a.shape[0]
    width = a.shape[1]
    white_max = 0
    black_max = 0

    for i in range(width):
        s = 0
        t = 0
        for j in range(height):
            if a[j][i] == 255:
                s += 1
            if a[j][i] == 0:
                t += 1
        white_max = max(white_max, s)
        black_max = max(black_max, t)
        startList.append(s)
        endList.append(t)


    n = 1
    while n < width - 2:
        n += 1
        if startList[n] > 0.005 * white_max:
            start = n
            for m in range(start + 1, width - 1):
                if endList[m] > 0.995 * black_max:
                    end = m
                    break
                else:
                    end = start+1
            n = end
```

```
            if end - start > 10:
                cut = a[:, start - 20:end + 20]
                ret.append(cut)


    return ret



def image_split_row(a)->list:
    """
    Function description: Split the image by row.
    Tips:
    1. Calculate the number of elements with a value of 255 in each row.
    2. When the number of 255 changes from zero to non-zero, it indicates the beginning of the digits area. Use startList to record the starting row index.
    3. When the number of 255 changes from non-zero to zero, it indicates the end of the digits area. Use endList to recorder the end row index.
    4. Use flag to represent the current state, outside or inside the digits area.

    :param img: input image to be splited by row.
    :return: output image after splited by row. It is a list, but its elements are np.ndarray.
    """

    ### write your codes here ###
    ##############################
    a = a.T
    ret = []
    startList = []
    endList = []
    height = a.shape[0]
    width = a.shape[1]
    white_max = 0
    black_max = 0
    for i in range(width):
        s = 0
        t = 0
        for j in range(height):
            if a[j][i] == 255:
                s += 1
            if a[j][i] == 0:
                t += 1
        white_max = max(white_max, s)
        black_max = max(black_max, t)
        startList.append(s)
        endList.append(t)


    n = 1
    while n < width - 2:
        n += 1
        if startList[n] > 0.01 * white_max:
            start = n
            for m in range(start + 1, width - 1):
                if endList[m] > 0.99 * black_max:
                    end = m
                    break
                else:
                    end = start+1
            n = end
            if end - start > 10:
                cut = a[:, start - 20:end + 20]
```

```
                ret.append(cut.T)

    return ret



def led_display(numList:list)->None:
    """
    Function description: Build a digital tube display circuit on the breadboard. Display the result with the digital tube.
    Tips:
    1.The GPIO mode we used is GPIO.BOARD.
    2.The digital tube is common anode. Use GPIO port to input high level for digital tube power pin.
    3. After the LED lamp pin of the digital tube is connected to the GPIO pin, the corresponding relationship can be confirmed by lighting the led one by one.
    4. Check "function introduction.xlsx" for GPIO functions.

    :para numList: input numbers in list to be displayed.
    :return: None
    """

    ### write your codes here ###
    ##############################87----

    # step 1:
    # Clarify the relationship between led pins and GPIO pins
    # Set the GPIO pins to GPIO.OUT mode and give them the right output


    # step 2:
    # Clarify the led composition of each number


    # step 3:
    # Display the numbers in the list one by one
    # Display every number for 1 second
    # Wait two seconds when displaying different lines

    for i in numList:
        if i == 1:
            dtc.display1()
        elif i == 2:
            dtc.display2()
        elif i == 3:
            dtc.display3()
        elif i == 4:
            dtc.display4()
        elif i == 5:
            dtc.display5()
        elif i == 6:
            dtc.display6()
        elif i == 7:
            dtc.display7()
        elif i == 8:
            dtc.display8()
        elif i == 9:
            dtc.display9()
        elif i == 0:
            dtc.display0()
```

```
    ret = None
    return ret


def take_photo()->str:
    """
    Function description: Build the camera control circuit on the breadboard. After pressing the control button, the shooting indicator(led light) lights up and the camera takes a picture.
    Tips:
    1. Use the 3.3v and GND pins on the Raspberry Pi as the power and ground of the circuit.
    2. Use the GPIO port as a signal line to sense the occurrence of key events. Set the correct GPIO mode
    3. Create a camera obj and wait for a button press to take a photo.
    4. Save the picture to /UserData/.
    5. Clean the camera.

    :para
    :return: a string which contains the picture location
    """

    ### write your codes here ###
    ##############################
    # step 1:
    #set a GPIO as an input channel for detecting
    btn.setup()

    camera = pc.setup()

    # step 2:
    # create the camera obj and wait for a button to take a photo
    # recorder the saving path
    # clear the camera
    camera.start_preview()
    print("Camera activated")
    if btn.detecting():
        pic_path = pc.shoot(camera, "./UserData/Pictures/")

    # step3:
    # return the saving path
    GPIO.cleanup()
    ret = pic_path
    return ret
```