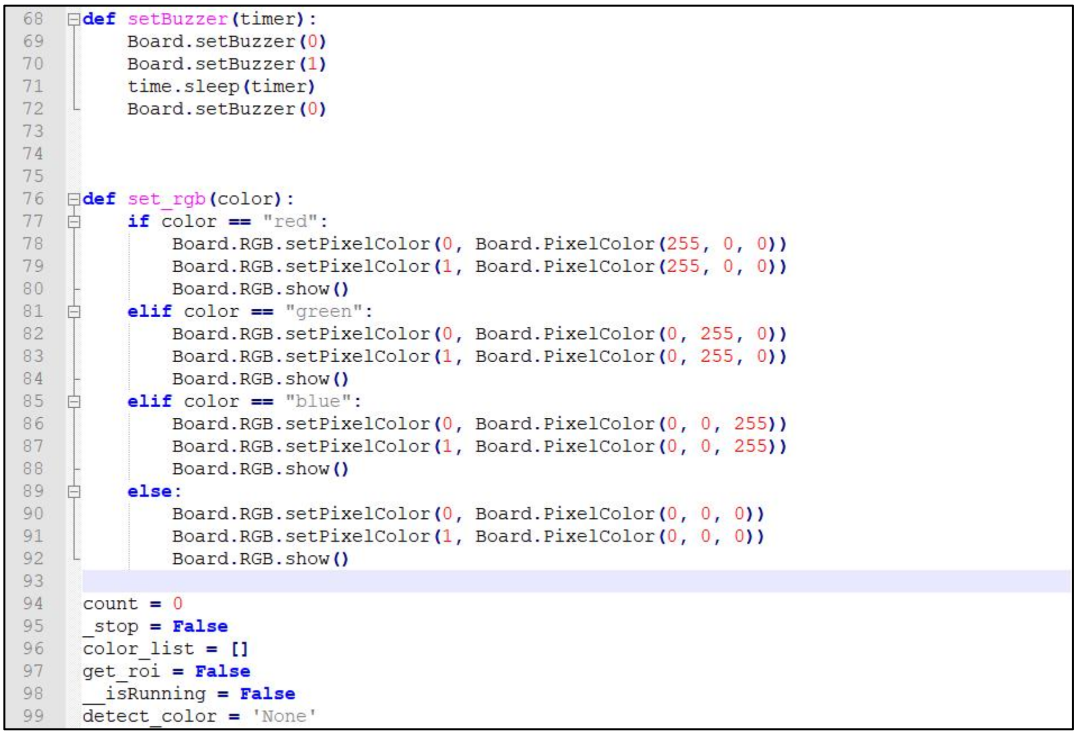
Lesson 2 Color Sorting

1. Working Principle

For the color recognition part, the color of the object is converted through LAB space, and then frame the outline of the target after processing the image.

After recognizing, robot arm will lift up and transport the different colored block to the corresponding position through the following two steps. The first step: after gripping block, control the impulse of ID6 servo to transport the colored block to close to the position of placement coordination. The second step: through inverse kinematics, directly control robotic arm to move to corresponding coordinate position.

The source code of program is located in: **/home/pi/MasterPi/Functions/ColorSorting.py**



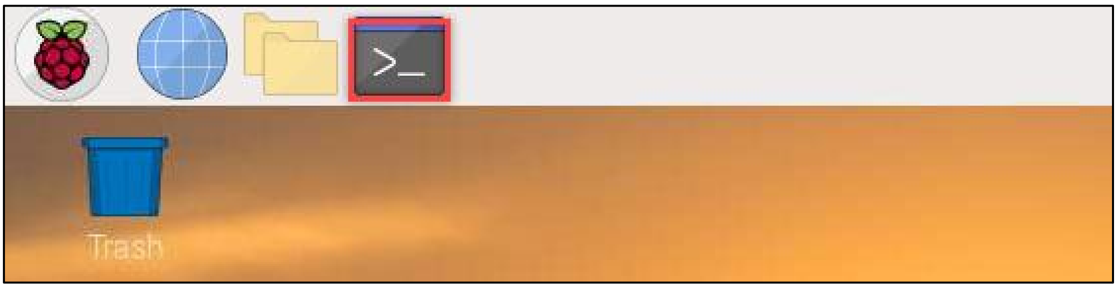
```
def setBuzzer(timer):
    Board.setBuzzer(0)
    Board.setBuzzer(1)
    time.sleep(timer)
    Board.setBuzzer(0)



def set_rgb(color):
    if color == "red":
        Board.RGB.setPixelColor(0, Board.PixelColor(255, 0, 0))
        Board.RGB.setPixelColor(1, Board.PixelColor(255, 0, 0))
        Board.RGB.show()
    elif color == "green":
        Board.RGB.setPixelColor(0, Board.PixelColor(0, 255, 0))
        Board.RGB.setPixelColor(1, Board.PixelColor(0, 255, 0))
        Board.RGB.show()
    elif color == "blue":
        Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 255))
        Board.RGB.setPixelColor(1, Board.PixelColor(0, 0, 255))
        Board.RGB.show()
    else:
        Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 0))
        Board.RGB.setPixelColor(1, Board.PixelColor(0, 0, 0))
        Board.RGB.show()

count = 0
_stop = False
color_list = []
get_roi = False
__isRunning = False
detect_color = 'None'
```

2. Operation Steps

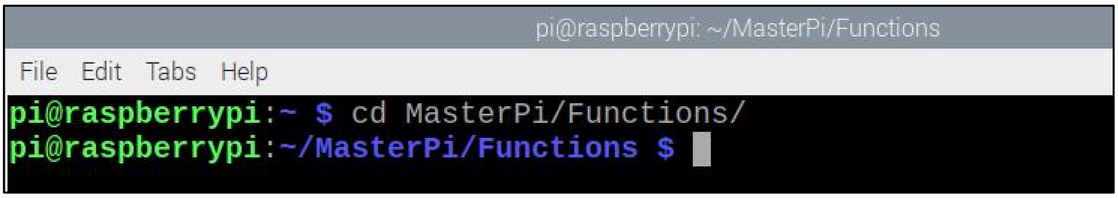
The entered command should be case sensitive.

Step 1: Turn on MaserPi, then connect to Raspberry Pi system desktop through VNC.

Step 2: Click >_ or press "Ctrl+Alt+T" to enter LX terminal.

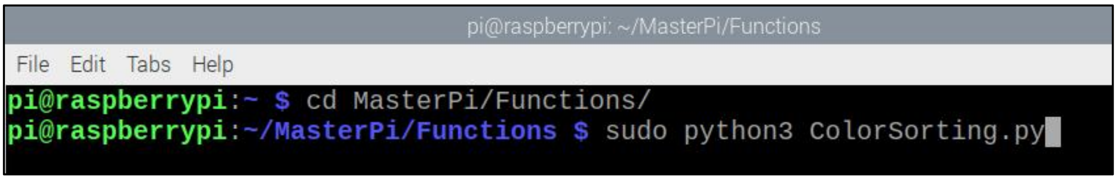


Step 3: Enter "cd MasterPi/Functions/" command, and then press "Enter" to come to the directory of games programmings.



```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ 
```

Step 4: Enter "sudo python3 ColorSorting.py", then press "Enter" to start the game.



```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ sudo python3 ColorSorting.py
```

Step 5: If you want to exit the game programming, press "Ctrl+C" in the LX terminal interface. If fail to exit, please try it few more times.

3. Project Outcome

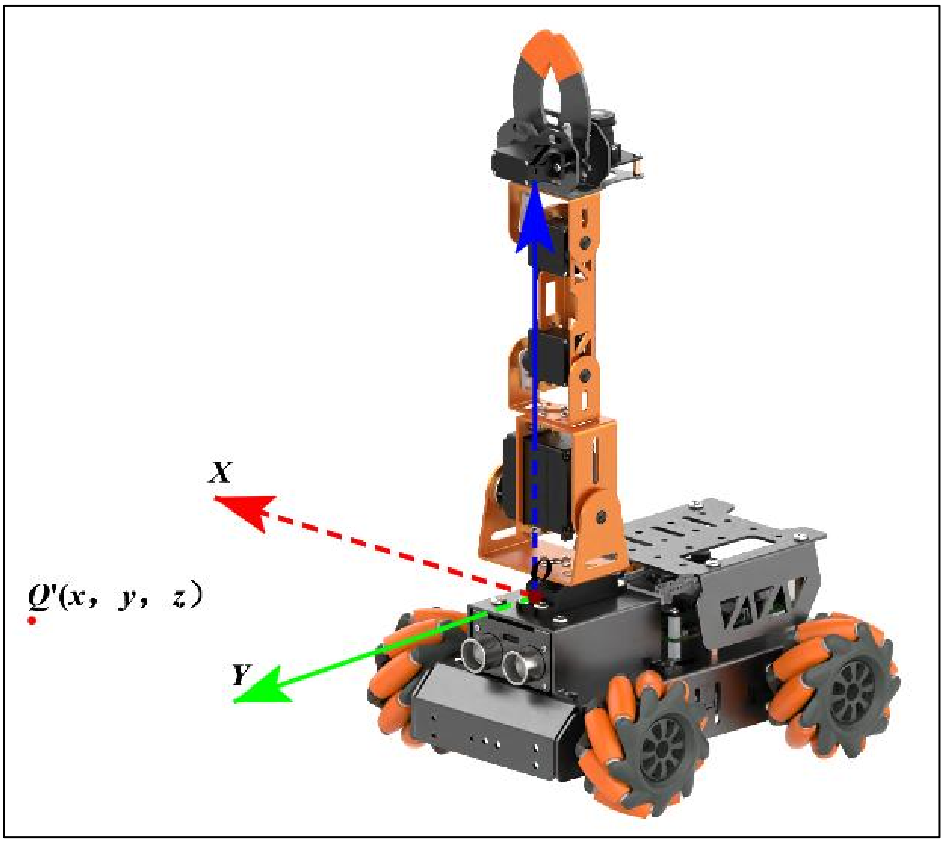
Place the red, green and blue blocks in a smooth and flat surface. The robot arm will beep once when the block is recognized. Then take the recognized colored block with your hand and place it in front of the gripper. Robot arm will grip the recognized block and carry them to the left corresponding position in turn.

4. Function Extension

4.1 Change placement position

The default effect of the color sorting program: grip the recognized red, green and blue blocks and place them to the corresponding position on the left side.

The robot coordinate system is as follow:



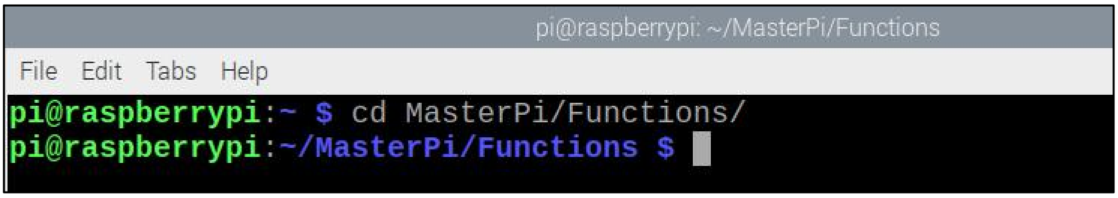
The corresponding relationship between the position of the color block and the coordinate parameters are shown in the following table:

The change of the coordinate parameter	The change of block position
x increase	The color block moves to the right along the x-axis
x decrease	The color block moves to the left along the x-axis
y increase	The color block moves forward along the y-axis
y decrease	The color block moves backward along the y-axis
z increase	The color block moves upward along the z-axis
z decrease	The color block moves downward along the z-axis

Note: when the parameter y is a negative value, it is invalid; the value of parameter z cannot be less than -3.

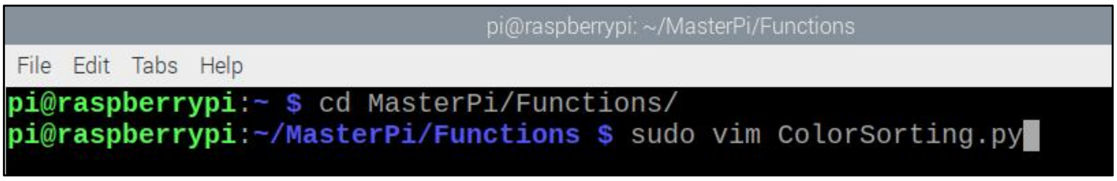
We will modify the placement position of red block to the front of robot as example. The modification method is same to other colored blocks. The steps is as follow:

Step 1: Enter “cd MasterPi/Functions//” command, and then press “Enter” to come to the category of game programmings.



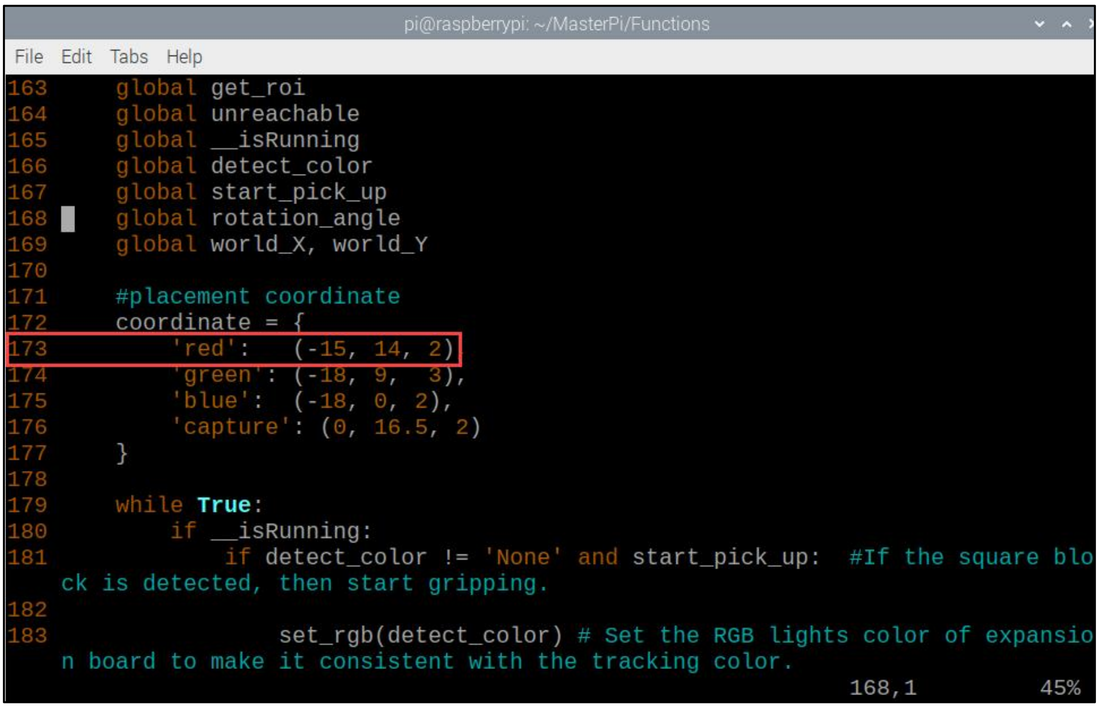
```
pi@raspberrypi: ~/MasterPi/Functions
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ 
```

Step 2: Enter “sudo vim ColorSorting.py” command, and then press “Enter” to open program file.



```
pi@raspberrypi: ~/MasterPi/Functions
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ sudo vim ColorSorting.py
```

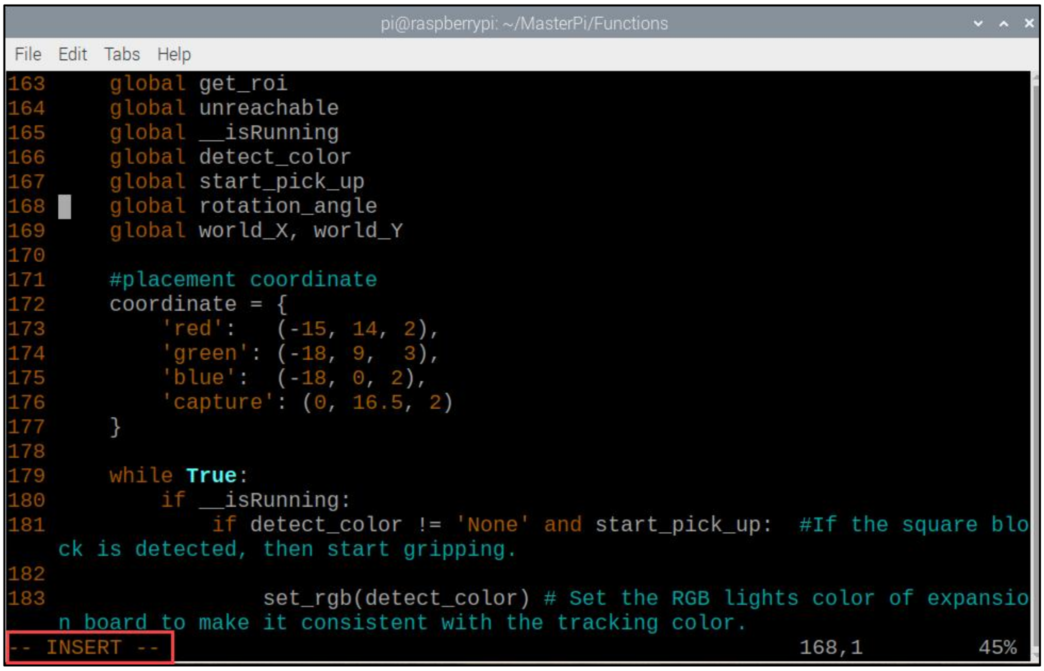
Step 3: Find the code shown in the following red box:



```
pi@raspberrypi: ~/MasterPi/Functions
File  Edit  Tabs  Help
163     global get_roi
164     global unreachable
165     global __isRunning
166     global detect_color
167     global start_pick_up
168     global rotation_angle
169     global world_X, world_Y
170
171     #placement coordinate
172     coordinate = {
173         'red':    (-15, 14, 2),
174         'green':  (-18, 9,   3),
175         'blue':   (-18, 0, 2),
176         'capture': (0, 16.5, 2)
177     }
178
179     while True:
180         if __isRunning:
181             if detect_color != 'None' and start_pick_up:  #If the square blo
    ck is detected, then start gripping.
182
183                 set_rgb(detect_color) # Set the RGB lights color of expansio
    n board to make it consistent with the tracking color.
                                                              168,1         45%
```

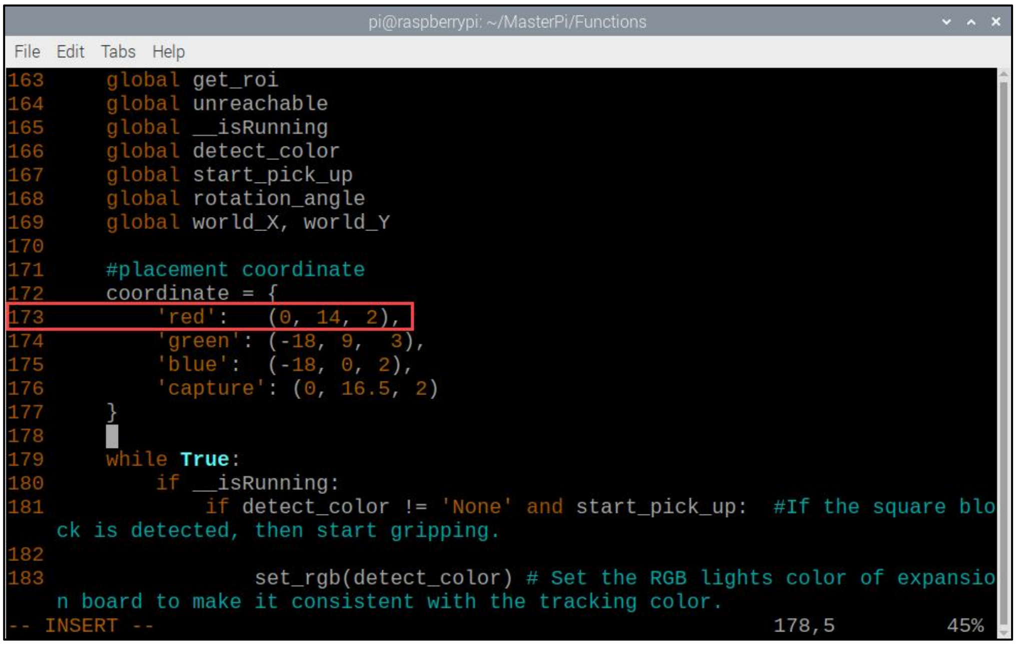
Note: After entering the position number of code, press “Shift+G” to jump to the corresponding position. (The position number of the code in figure is for reference only.)

Step 4: Press “i” to enter the editing mode.



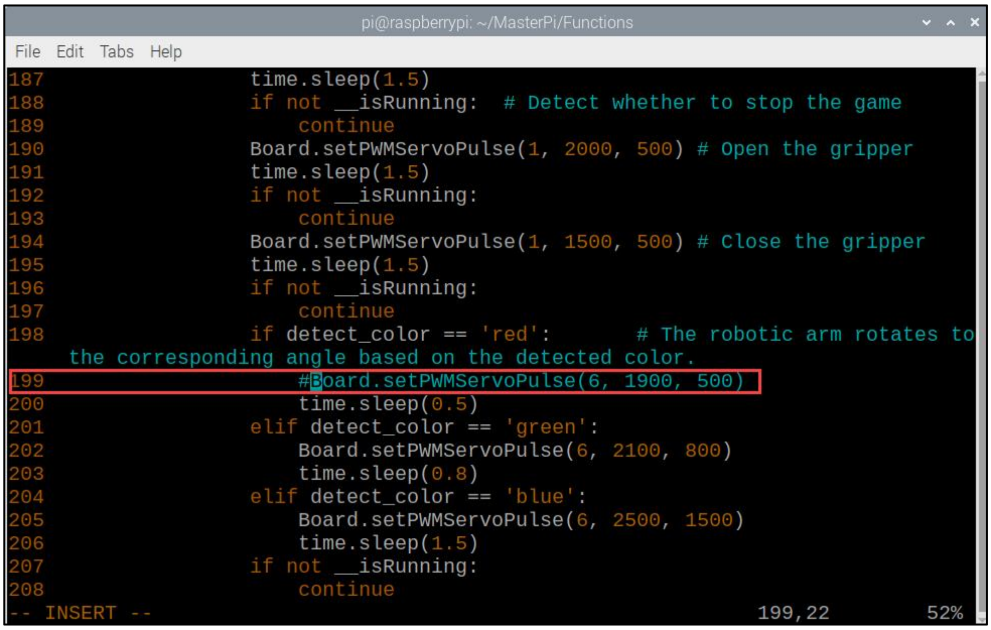
```
    global get_roi
    global unreachable
    global __isRunning
    global detect_color
    global start_pick_up
    global rotation_angle
    global world_X, world_Y

    #placement coordinate
    coordinate = {
        'red':   (-15, 14, 2),
        'green': (-18, 9,  3),
        'blue':  (-18, 0, 2),
        'capture': (0, 16.5, 2)
    }

    while True:
        if __isRunning:
            if detect_color != 'None' and start_pick_up:  #If the square block is detected, then start gripping.

                set_rgb(detect_color) # Set the RGB lights color of expansion board to make it consistent with the tracking color.
-- INSERT --                                                   168,1         45%
```

Step 5: In 'red': (-15,14,2), “-15” is the x-axis parameter, “14” is the Y-axis parameter and “2” is the parameter of z-axis. We will modify “15” to “0” , keep the parameters of y-axis and z-axis unchanged and place the color block to the front. The modification method is as follow:



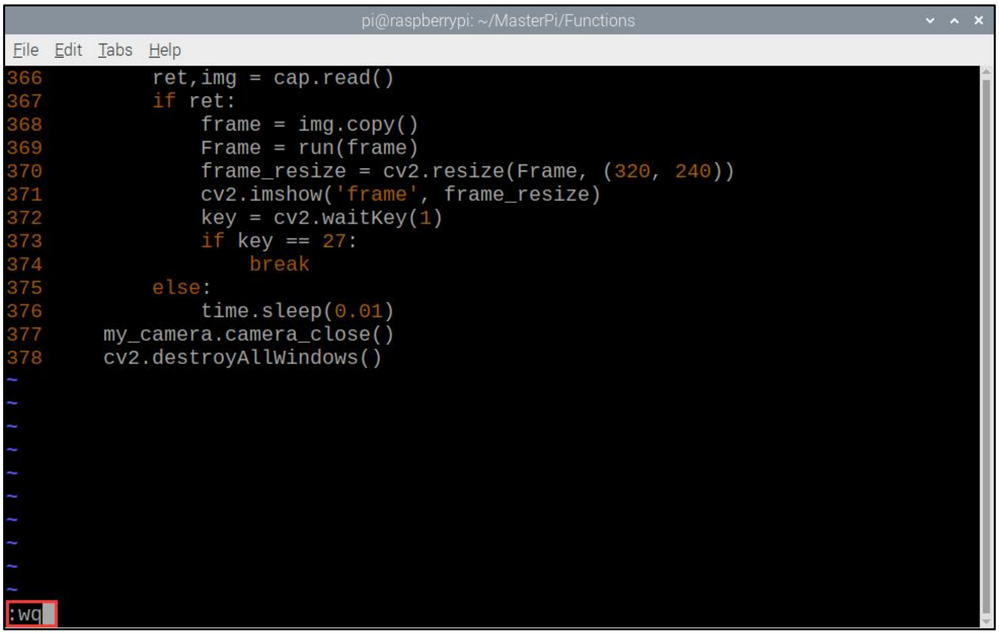
```
    global get_roi
    global unreachable
    global __isRunning
    global detect_color
    global start_pick_up
    global rotation_angle
    global world_X, world_Y

    #placement coordinate
    coordinate = {
        'red':   (0, 14, 2),
        'green': (-18, 9,  3),
        'blue':  (-18, 0, 2),
        'capture': (0, 16.5, 2)
    }

    while True:
        if __isRunning:
            if detect_color != 'None' and start_pick_up:  #If the square block is detected, then start gripping.

                set_rgb(detect_color) # Set the RGB lights color of expansion board to make it consistent with the tracking color.
-- INSERT --                                                   178,5         45%
```

Step 6: Find the code in red box, and then add “#” in front of it.



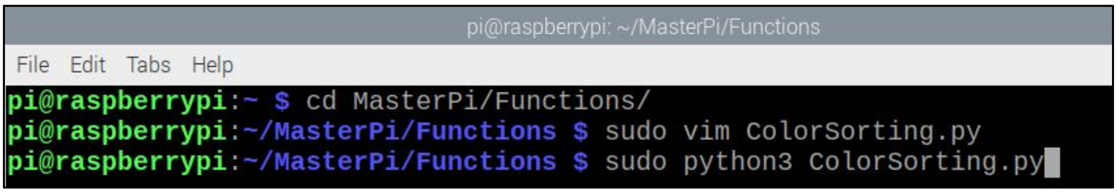
```
                time.sleep(1.5)
                if not __isRunning:  # Detect whether to stop the game
                    continue
                Board.setPWMServoPulse(1, 2000, 500) # Open the gripper
                time.sleep(1.5)
                if not __isRunning:
                    continue
                Board.setPWMServoPulse(1, 1500, 500) # Close the gripper
                time.sleep(1.5)
                if not __isRunning:
                    continue
                if detect_color == 'red':       # The robotic arm rotates to the corresponding angle based on the detected color.
                    #Board.setPWMServoPulse(6, 1900, 500)
                    time.sleep(0.5)
                elif detect_color == 'green':
                    Board.setPWMServoPulse(6, 2100, 800)
                    time.sleep(0.8)
                elif detect_color == 'blue':
                    Board.setPWMServoPulse(6, 2500, 1500)
                    time.sleep(1.5)
                if not __isRunning:
                    continue
-- INSERT --                                             199,22          52%
```

Step 7: Save the modified content. Press “Esc” and then enter “:wq” to save and exit.



```
        ret,img = cap.read()
        if ret:
            frame = img.copy()
            Frame = run(frame)
            frame_resize = cv2.resize(Frame, (320, 240))
            cv2.imshow('frame', frame_resize)
            key = cv2.waitKey(1)
            if key == 27:
                break
        else:
            time.sleep(0.01)
    my_camera.camera_close()
    cv2.destroyAllWindows()
~
~
~
~
~
~
~
~
~
~
:wq
```

Step 8: Enter “sudo python3 ColorSorting.py” command again and then press “Enter” to start the game.



```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ sudo vim ColorSorting.py
pi@raspberrypi:~/MasterPi/Functions $ sudo python3 ColorSorting.py
```