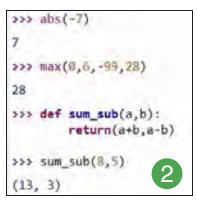
树莓派自定义“灯带函数”

文/山东省招远第一中学 牟晓东

众所周知,函数是程序设计语言的“基石”,即“组织好的、可重复使用的、用来实现单一或相关功能的代码段”。Python本身提供了功能丰富的“内置函数”,可以在命令行模式中输入“dir(__builtins__)”命令进行查看,例如求绝对值函数、求最大值和最小值函数,还包括input输入和print输出函数等(如图1)。

此时,可以直接在命令行交互模式下使用内置函数,比如输入“abs(-7)”来求解-7的绝对值,回车后就会返回数值7;输入“max(0,6,-99,28)”求解四个数中的最大值,就会返回数值28。另外,Python还支持用户根据自己的不同需求进行自定义函数操作,比如编写一个能够同时求解二数之和、之差的函数sum_sub():

```
def sum_sub(a,b):
return(a+b,a-b)
```



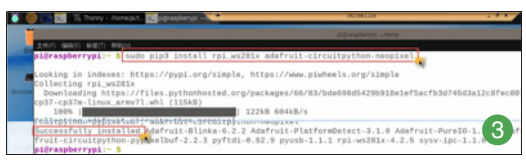
输入“sum_sub(8,5)”进行测试,回车后就会返回两个结果:13和3(如图2)。

在开源硬件编程中使用Python可以灵活地控制各种周边硬件,从而实现更为丰富的功能。我们在树莓派中使用Python进行函数的自定义编写,控制灯带模仿现实生活中十字路口的红绿灯,分别是单函数“带参”的红绿灯带和双函数“无参”的红绿灯带。

1.准备工作

将可编程ws281x灯带通过古德微扩展板的18号接口与树莓派连接,注意灯带的三根引线分别标注+5V、GND和Din,不要接反。

在树莓派中通过Python编程控制灯带需要安装rpi_ws281x库模块,因此需要通过“Windows的远程桌面”连接树莓派。在控制终端命令行模式中输入命令:“sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel”,回车后等待进度条到达100%后会有“Successfully installed”的提示(如图3)。



2.Python单函数“带参”控制红绿灯带

(1)首先,以“ws”为别名导入rpi_ws281x库:“import rpi_ws281x as ws”,再导入time库中的sleep函数:“from time import sleep”;接着,设置灯带中激活的“灯珠”数量LED_COUNT值为60:“LED_COUNT = 60”,灯带接入的端口号LED_PIN是18号:“LED_PIN = 18”,并且创建灯带对象strip,实例化PixelStrip,参数为LED_COUNT和LED_PIN:“strip = ws.PixelStrip(LED_COUNT,LED_PIN)”,语句“strip.begin()”的作用是对灯带进行初始化。

(2)接下来自定义“带参”(参数为color)的lights()函数:“def lights(color):”,注意后面有英文冒号。

(3)建立四分支选择结构,特别要注意Python的“四空格”缩进。

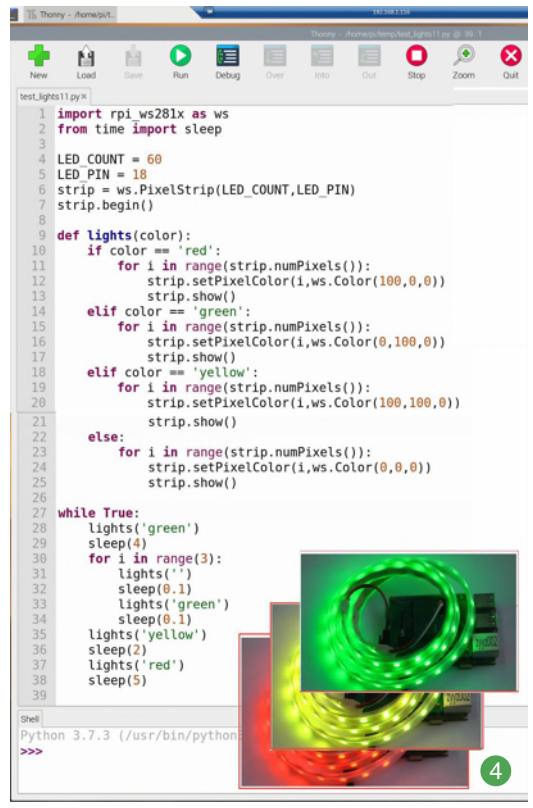
第一分支为“if color == 'red':”,即函数参数为“red”时,使用循环结构控制60个灯珠全部发红光,RGB值为(100,0,0);不要忘记对灯带进行刷新操作:“strip.show()”。

第二和第三分支分别为“elif color == 'green':”和“elif color == 'yellow':”,即函数参数为“green”和“yellow”时,使用循环结构控制60个灯珠全部发绿光、黄光,RGB值分别为(0,100,0)和(100,100,0);也要对灯带进行刷新操作:“strip.show()”。

第四分支为“else:”,即函数参数为空字符串,控制60个灯珠全部熄灭,RGB值为(0,0,0)。

(4)主程序是一个“while True:”循环结构,通过传递不同的参数对lights()函数进行调用。先传递“green”参数亮绿灯:“lights('green')”,持续4秒钟:“sleep(4)”;再建立执行三次的循环结构:“for i in range(3):”,传递参数为空字符串,关闭灯带:“lights('')”,0.1秒后再次亮起绿灯:“lights('green')”,持续0.1秒后传递“yellow”参数亮黄灯:“lights('yellow')”;持续2秒后传递“red”参数亮红灯:“lights('red')”,再持续5秒后结束本次循环;进入下一次循环:亮绿灯……

(5)保存程序为test_lights11.py,在命令行窗口中输入命令“sudo python3 test_lights11.py”运行测试,一个灯带式的“红绿灯”开始工作起来,这就是Python单函数“带参”红绿灯带(如图4)。



3.Python双函数“无参”控制红绿灯带

(1)第一部分库模块的导入和对灯带进行初始化的代码与刚才一致,直接复制和粘贴。整条灯带的60个灯珠编号为0-59,将其均分为四组:0-14、15-29、30-44和45-59,其中的0-14和30-44对应“十字路口”的X方向,而15-29和45-59则对应Y方向。

(2)定义X方向上的x_lights()函数:“def x_lights():”:

第一部分的“for i in range(15)”“亮绿灯”(0,100,0)代码,包括0-14、30-44两组灯珠,注意30-44的表示方法是“i+30”,也就是同时控制X方向马路正面和对面;不要忘记添加灯带刷新代码“strip.show()”,绿灯保持亮5秒:“sleep(5)”,省略了绿灯闪烁过程(可自行添加内循环来实现);

第二部分“亮黄灯”(100,100,0)代码,复制粘贴后,只须改RGB颜色值即可,黄灯保持亮2秒:“sleep(2)”;

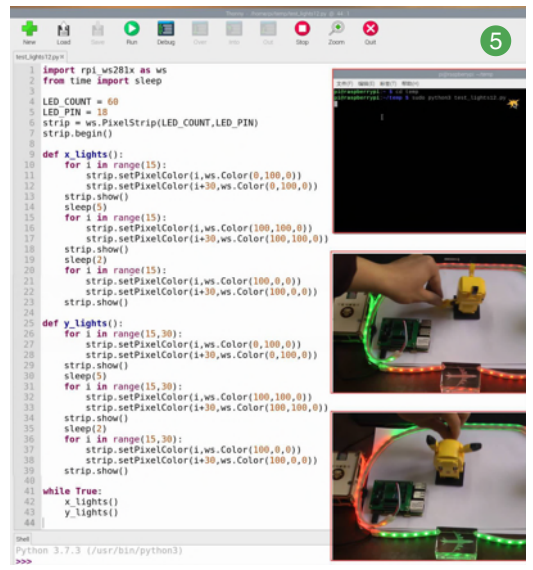
第三部分“亮红灯”代码,也是粘贴操作,改RGB值为(100,0,0),注意最后不必使用sleep()控制红灯持续亮的时间。

(3)定义Y方向上的y_lights()函数:“def y_lights():”:

复制x_lights()函数粘贴为y_lights()函数,只修改三个循环结构中range()的参数为“15,30”即可,因为Y方向控制的是15-29、45-59两组灯珠,同样也是使用“i+30”来表示45-59。

两个函数代码编写完毕,最后写主程序的“while True:”循环结构,直接调用x_lights()和y_lights()即可。

保存程序为test_lights12.py,在命令行窗口中输入命令“sudo python3 test_lights12.py”运行测试,模拟十字路口的四个方向红绿灯工作过程:红灯停、绿灯行,黄灯亮了等一等……这就是Python双函数“无参”红绿灯带(如图5)。



趣味数学——找规律

文/陈新龙

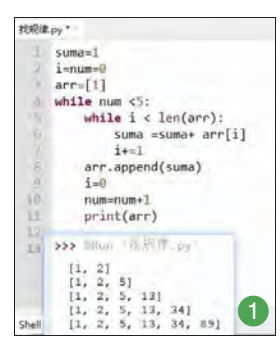
找规律是常见的数学题型。目的是让学生发现图形和数字的排列规律,从而理解并掌握找规律的方法,培养学生的观察及推理能力。例如数列1,2,4,7,11,16,(X),(Y)……其中X和Y的值分别为多少呢?数列中两数之间的差成一个等差数列关系,相差为:1,2,3,4,5,6……因此X的值为22,Y的值为29。

今天我们用Python解决一道奥数找规律问题:现有一组数列为1,2,5,13,34,(z)。请问z的值为多少呢?

这个数列的规律你发现了吗?第N个数字等于它前一个数字加前面所有数字的和,例如5=2+2+1,13=5+5+2+1,34=13+13+5+2+1。那么如何用Python表示出这个数列呢?并求出需要的答案呢?

首先设变量a=1表示第一个数字。设置两个循环变量i和num用来统计元素是否达到规定的个数,这种变量也称为计数器,并且定义列表arr用于存放输出的结果。

控制while循环设置计数器和产生元素的个数,在每次循环结束前加1,在循环过程中,数值不断累加,计数器也加1,第i次循环加上一次的结果,计算出结果后将计算出的新元素添加到arr列表中,并且把元素i清零,这样下次循环又是从第一个数字开始累加,依次循环,直到全部结束为止,每循环一次,打印显示列表值,用于观察(图1)。



通过分析,我们可以用Python计算出有规律的数列,并且有序将每次的列表中的值输出,那么现在增加难度:如果我们想求出该数列中第100个数的值是多少?前100个数值之和为多少呢?大家赶紧动手编写代码吧。