**华中科技大学**

**《Python 程序设计与软件工程基础》**

**个人实验报告**

|  |  |
| --- | --- |
| 实验名称： | 二维码编解码 |
| 院（系）： | 电子信息与通信学院 |
| 专业班级： | 提高2001班 |
| 姓名： | 周楚骎 |
| 学号： | U202013858 |
| 时间： | 2022.9---2022.11 |
| 指导教师： | 陈建文 |

一、实验背景

二维码又称二维条码，常见的二维码为QR Code，QR全称Quick Response，是一种编码方式。它比传统的Bar Code条形码能存更多的信息，也能表示更多的数据类型。二维条码/二维码（2-dimensional bar code）是用某种特定的几何图形按一定规律在平面（二维方向上）分布的、黑白相间的、记录数据符号信息的图形；在代码编制上巧妙地利用构成计算机内部逻辑基础的“0”、“1”比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图象输入设备或光电扫描设备自动识读以实现信息自动处理：它具有条码技术的一些共性：每种码制有其特定的字符集；每个字符占有一定的宽度；具有一定的校验功能等。同时还具有对不同行的信息自动识别功能、及处理图形旋转变化点。

而二维码以及其编解码技术已经用于了我们生活的方方面面，无论是付款时需要扫的付款码的编码与解码，还是我们每天都离不开的健康码的编码与解码，都体现出来二维码以及其编解码技术的重要性，因而做出一个二维码编解器是很有必要的。

二、需求设计

1、二维码编码器

2、二维码解码器

3、编解码的图形用户界面GUI

三、思路与实现

我在这个项目中导入了cv2、qrcode、MyQR、pyzbar、validators、webbrowser、tkinter、os、PIL等库用于编写这个编解码器。其中cv2、PIL用于读取、操作和保存图片；qrcode和MyQR用于编码创建新的二维码；pyzbar用于解码识别二维码内容；validators用于识别二维码内容是否是网址；webbrowser用于在浏览器中打开网址；os用于读取和打开文件。

首先定义一个名为GUI的子类用来创建初始的二维码编解码器操作界面。GUI从父类tkinter.Tk中继承其所有的特性，于是在初始化时需要super( ).\_\_init\_\_( )来进行继承。然后设置初始界面大小为500\*500，并给予程序名称“周楚骎的编解码器”。并通过tkinter的设置程序图标的功能，将图标设置为我们指定的“Qin.jpg”。再在界面中创建一个标签“周楚骎的编解码器”作为这个界面的标题，在画面的左右两端创建两个按钮，一个为“编码器”，若点击则创建一个新类CoderGUI；另一个为“解码器”，若点击则创建一个新类DecoderGUI。最后执行mainloop方法，将标签显示在屏幕，进入等待状态，准备响应用户发起的[GUI](https://so.csdn.net/so/search?q=GUI&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/San__Qi/article/details/_blank)事件。在mainloop函数中，tkinter内部会监控这些事件，如键盘活动，鼠标单击等。运行结果如下图：



类GUI的\_\_init\_\_代码如下：

class GUI(tkinter.Tk):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.geometry("500x500")  
 self.title('周楚骎的编解码器')  
 self.iconphoto(True, tkinter.PhotoImage(file='Qin.png'))  
 l1 = tkinter.Label(self, text="周楚骎的二维码编解码器", font=("Microsoft Sans Serif", 24))  
 # l1.place(x=50, y=50)  
 l1.pack()  
 button1 = tkinter.Button(self, text="编码器", command=CoderGUI, height=10, width=20, font=20, bd=5)  
 button1.pack(side=tkinter.LEFT)  
 button2 = tkinter.Button(self, text="解码器", command=DecoderGUI, height=10, width=20, font=20, bd=5)  
 button2.pack(side=tkinter.RIGHT)  
 self.mainloop()

<1> 二维码生成器

当点击主界面的编码器时会创建一个新类CoderGUI用来创建初始的二维码编码器操作界面。



CoderGUI从父类tkinter.Toplevel中继承其所有的特性，于是在初始化时需要super( ).\_\_init\_\_( )来进行继承。然后设置初始界面大小为412\*500，并给予程序名称“周楚骎的编码器”。定义self的属性path为tkinter的字符串变量，第一排，一个标签内容为“二维码内容”放在左边，中间为一个tkinter的Entry作为输入框，作为二维码需要编码的内容。第二排，一个Label标签内容为“二维码背景文件”放置在最左边，中间为一个tkinter的Entry作为输入框，用来输入背景文件存储的地址也可以通过选择文件来进行自动输入，最右边为一个“路径选择”按钮，按下后会调用CodeGUI这个类的一个selectPath函数

def selectPath(self):  
 # 选择文件path\_接收文件地址  
 path\_ = filedialog.askopenfilename()  
 # 通过replace函数替换绝对文件地址中的/来使文件可被程序读取  
 # \\转义后为\，所以\\\\转义后为\\  
 path\_ = path\_.replace("/", "\\\\")  
 # path设置path\_的值  
 self.path.set(path\_)

selectPath函数会调用tkinter的filedialog.py中的askopenfilename函数，并将该地址文件字符串转化为程序可以读取的形式。第三排，一个Label标签内容为“二维码储存文件名”放置在最左边，中间为tkinter的Entry用于输入储存的文件名，注意要用.jpg或.png后缀，不然后续会无法正常打开。最下面为一个“生成二维码”的tkinter的Button按钮，按下后会调用create\_qpcode函数

def create\_qpcode(self):  
 background = self.e3.get()  
 loc = self.e2.get()  
 if loc == '':  
 return  
 if os.path.exists(background):  
 b2 = tkinter.Button(self, text='将图片作为背景', width=20, height=5, command=self.create\_background)  
 b2.grid(row=4, column=0)  
 b2 = tkinter.Button(self, text='将图片放在中心', width=20, height=5, command=self.create\_center)  
 b2.grid(row=4, column=2)  
 else:  
 self.qpcode\_create()

首先会从两个输入框中获取背景文件地址以及存储地址。如果存储地址为空则返回，否则通过os库的path.exits判断背景文件是否存在，如果背景文件存在，则弹出两个按钮，分别为“将图片作为背景”和“将图片放在中心”。如果选择“将图片作为背景”，则调用此类中的create\_background函数，如果选择“将图片放在中心”，则调用此类中的create\_center函数，这两个函数较类似，都是先获得所需编码的内容，背景文件地址以及后续存储的文件名然后分别调用qrcode\_create\_background和qrcode\_create\_center函数，来创建二维码。如果背景文件不存在就会调用此类中的qpcode\_create函数，这个函数会获得所需编码的内容以及后续存储的文件名，然后调用独立的qpcode\_create函数。

1. 生成带背景二维码文件qrcode\_create\_background函数

def qrcode\_create\_background(data\_str, back\_str, save\_name):  
 flag = 0  
 save\_name = save\_name.split('.')  
 if save\_name[-1] == 'jpg':  
 flag = 1  
 save\_name[-1] = 'png'  
 save\_name = str.join(".", save\_name)  
 myqr.run(words=data\_str, picture=back\_str, colorized=True, save\_name=save\_name)  
 img = cv2.imread(save\_name)  
 cv2.imshow("IMG", img)  
 cv2.waitKey()  
 cv2.destroyAllWindows()  
 img = Image.open(save\_name)  
 save\_name = save\_name.split(".")  
 if flag == 1:  
 save\_name[-1] = 'jpg'  
 save\_name = str.join(".", save\_name)  
 r, g, b, a = img.split()  
 img = Image.merge("RGB", (r, g, b))  
 img.save(save\_name)  
 save\_name = save\_name.split('.')  
 save\_name[-1] = 'png'  
 save\_name = str.join(".", save\_name)  
 os.remove(save\_name)

首先由于MyQR这个库在生成带背景的二维码文件的时候只能生成.png编码格式的图片，所以首先需要将想要保存的文件名.jpg后缀改为.png，然后就可以调用MyQR中的函数myqr.run(words=data\_str, picture=back\_str, colorized=True, save\_name=save\_name)  
生成想要的编码后的文件，生成完后，我通过cv2的imread函数将该图片读入，并通过imshow函数将该图片展示出来，当点击键盘上任意按钮的时候可以关闭展示的界面。然后需要将生成的.png文件改为目标.jpg格式，通过Image库的open函数打开该png格式图片，然后将通过Image的merge函数该图片的RGBA四通道的改为RGB三通道的形式，然后储存为.jpg格式。同时我们删除原先生成的.png文件。但要注意因为MyQR库只能编码英文字符，所以无法讲中文编码到二维码里，在生成带背景的二维码的时候只能编码英文内容。

生成的效果图如下：



1. 生成带中心图片的二维码文件qrcode\_create\_center函数

def qrcode\_create\_center(data\_str: str, back\_str: str, save\_name: str):  
 img = qrcode.make(data\_str)

img = img.convert("CMYK") # RGBA  
 # 添加logo  
 icon = Image.open(back\_str).convert("RGBA")  
 # 获取图片的宽高  
 img\_w, img\_h = img.size  
 factor = 6  
 size\_w = int(img\_w / factor)  
 size\_h = int(img\_h / factor)  
 icon\_w, icon\_h = icon.size  
 if icon\_w > size\_w:  
 icon\_w = size\_w  
 if icon\_h > size\_h:  
 icon\_h = size\_h  
 # 重新设置logo的尺寸  
 icon = icon.resize((icon\_w, icon\_h), Image.ANTIALIAS)  
 w = int((img\_w - icon\_w) / 2)  
 h = int((img\_h - icon\_h) / 2)  
 img.paste(icon, (w, h), icon)  
 # 显示图片  
 img.save(save\_name)  
 img = cv2.imread(save\_name)  
 cv2.imshow("IMG", img)  
 cv2.waitKey()  
 cv2.destroyAllWindows()

首先利用qrcode的make函数生成编码了待编码内容的二维码，将生成的图片换为CMYK或者RGBA四通道。然后通过Image库读取想要作为背景的图片文件转化为RGBA四通道。读取生成的二维码图片的长宽，除以6作为待插入的中心图片的大小，如果待插入图片大小大于该大小，则利用resize函数将该图片变成想要的长与宽，再将改变了长宽的中心图片用paste函数将中心图片与二维码图片进行组合，利用save函数存储组合后的图片，通过cv2读取并展示该图片，按下键盘上任意键后关闭展示窗口。

生成的效果图如下：



1. 生成普通的二维码文件qpcode\_create函数

def qrcode\_create(data\_str: str, save\_name: str):  
 img = qrcode.make(data\_str)  
 img.save(save\_name)  
 img = cv2.imread(save\_name, 1)  
 cv2.imshow("IMG", img)  
 cv2.waitKey()  
 cv2.destroyAllWindows()

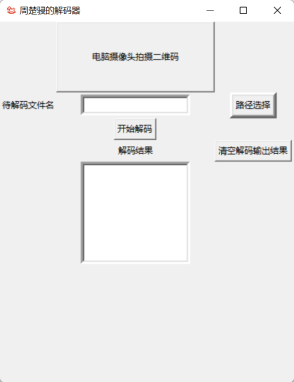
利用qrcode的make函数生成编码了待编码内容的二维码，利用save函数储存该图片，再利用cv2读取该图片并展示，按下键盘上任意键后关闭展示窗口。

生成的效果图如下：



<2> 二维码读取器

当点击主界面的解码器时会创建一个新类DecoderGUI用来创建初始的二维码解码器操作界面。



DecoderGUI从父类tkinter.Toplevel中继承其所有的特性，于是在初始化时需要super( ).\_\_init\_\_( )来进行继承。然后设置初始界面大小为408\*500，并给予程序名称“周楚骎的解码器”。定义self的属性path为tkinter的字符串变量，第一排，一个tkinter的Button按钮内容为“电脑摄像头拍摄二维码”，按下后会调用camera\_decoder函数

def camera\_decoder(self):  
 msg = cameraDecoder()  
 self.t1.insert('insert', msg)

该函数会调用独立的cameraDecoder函数打开电脑摄像头并开始检测二维码，并将检测结果返回，输出到解码结果的输出框内。第二排，一个Label标签内容为“待解码文件名”放置在最左边，中间为一个tkinter的Entry作为输入框，用来输入带解码文件存储的地址也可以通过选择文件来进行自动输入，最右边为一个“路径选择”按钮，按下后会调用DecoderGUI这个类的一个selectPath函数，这个函数与CoderGUI中的selectPath函数相同，selectPath函数会调用tkinter的filedialog.py中的askopenfilename函数，并将该地址文件字符串转化为程序可以读取的形式。第三排，有一个tkinter的Button“开始解码”按钮，按下后会调用此类中select\_decoder函数来进行解码。

def select\_decoder(self):  
 msg = self.e1.get()  
 res = qrcode\_decode\_local(msg)  
 self.t1.insert('insert', res)

select\_decoder函数会读取待解码文件名，将其作为变量传输到qrcode\_decode\_local这个函数中进行解码，并返回解码结果，输出到解码结果的输出框内。第四排中间为一个tkinter的Label内容为“解码结果”，右边为一个tkinter的Button“清空解码输出结果”的按钮，按下后会调用此类中的clear\_box函数来清空解码结果输出的输出框。

def clear\_box(self):  
 self.t1.delete("1.0", "end")

clear\_box函数会用tkinter的Text的delete函数删除输出文本的全部内容。第五排中间为tkinter的Text用于作为解码结果的输出框，输出框内可输出多次的解码结果，也可通过按下清空键清空。

1. 调用电脑摄像机进行解码 cameraDecoder函数

def cameraDecoder():  
 capture = cv2.VideoCapture(0) # 0是代表摄像头编号，只有一个的话默认为0  
 data = ''  
 while True:  
 # 调用摄像机  
 ref, capframe = capture.read()  
 if ref is True:  
 # 二维码识别  
 data = qrcode\_decode\_camera(capframe)  
 cv2.imshow('frame', capframe)  
 if cv2.waitKey(15) == ord('q') or data:  
 # 当输入 q 或截取到有用信息时关闭窗口  
 break  
 capture.release()  
 cv2.destroyAllWindows()  
 return data

首先利用opencv2的VideoCapture函数打开电脑摄像头，获取摄像头的画面，如果获得了画面就进行二维码识别，调用qrcode\_decode\_camera函数识别获取的画面中的二维码，并展示摄像头获取的画面，当键盘键入’q’或者识别到了有用的信息时关闭摄像头，并返回识别到的解码内容。而qrcode\_decode\_camera函数的解释如下：

def qrcode\_decode\_camera(src\_img):  
 data = False  
 gray\_img = cv2.cvtColor(src\_img, cv2.COLOR\_BGR2GRAY)  
 barcode\_objects = decode(gray\_img)  
 if len(barcode\_objects) > 0:  
 for obj in barcode\_objects:  
 data = obj.data  
 data = data.decode('utf-8')  
 # print(data)  
 if validators.url(data):  
 webbrowser.open(data, new=0, autoraise=True)  
 return data

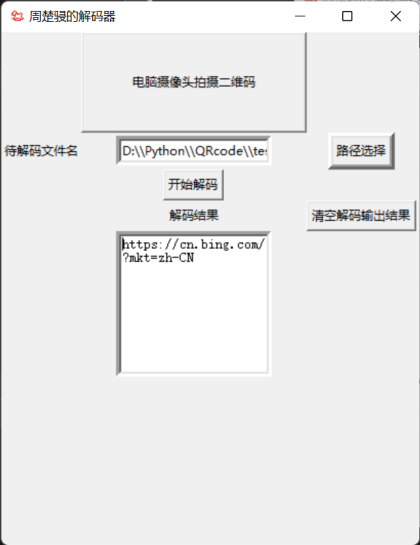
qrcode\_decode\_camera中将获取的画面图片转化为黑白单通道，然后利用pyzbar库的decode函数识别图中二维码，返回识别到的内容，当识别到的内容不为空时，获取识别到的内容中的data数据，用“utf-8”编码格式进行解码，用validators库的url函数判断该数据是否为网址，如果解码结果为一个网址，就利用webbrowser库的open函数在本地浏览器打开该网址，然后返回解码结果。

1. 解码本地文件 qrcode\_decoder\_local函数

def qrcode\_decode\_local(src\_img):  
 data = '解码结果'  
 if not os.path.exists(src\_img):  
 data = '不存在该文件'  
 # print(data)  
 return data  
 gray\_img = cv2.imread(src\_img, cv2.IMREAD\_GRAYSCALE)  
 barcode\_objects = decode(gray\_img)  
 if len(barcode\_objects) > 0:  
 for obj in barcode\_objects:  
 data = obj.data  
 data = data.decode('utf-8')  
 # print(data)  
 if validators.url(data):  
 webbrowser.open(data, new=0, autoraise=True)  
 return data

用os库的path.exists函数判断待解码文件是否存在，如果不存在，就返回解码结果为“不存在该文件”，如果存在就继续下面的操作，将获取的画面图片转化为黑白单通道，然后利用pyzbar库的decode函数识别图中二维码，返回识别到的内容，当识别到的内容不为空时，获取识别到的内容中的data数据，用“utf-8”编码格式进行解码，用validators库的url函数判断该数据是否为网址，如果解码结果为一个网址，就利用webbrowser库的open函数在本地浏览器打开该网址，然后返回解码结果。

解码效果图如下：在浏览器打开了该网址，并输出了解码结果在解码结果输出框内。



四、实验总结

此次实验，我学习并实现了python编解码器的设计，一方面学习到了python中的很多库的使用以及其中的种种函数，同时我也更加熟悉通过tkinter设计GUI界面。同时也学会了编程的模块化设计以及调试。此次python作业，让我学习到了很多，提升了我的动手能力。虽然这只是一个简单的编解码器的设计，但是python能实现的功能远不止如此，我相信掌握了python的使用，我在之后的科研以及编程等方面上也有所优势。