# 自动语音识别系统优化报告

现如今，语音识别技术已经有了迅猛的发展，并且在很多领域都有了应用，其被广泛应用于语音助手、智能家居、语音翻译、电话客服、语音搜索等领域。通过语音识别技术，人们可以通过自然语言与设备进行交互，实现更加智能化、人性化的操作，是人机交互技术的一大重大突破。

本次作业，我们通过搭建一个简单的语音识别系统初步认识了语音识别系统的实现过程，与实现方法，并学习了PyQT5，speech\_recognition等python包的使用。下面是此次作业中，对原有GUI与功能的修改。

#### 1.运行环境

·VScode代码编辑器

·Python3.7.16

**用到的python库：**

·图形界面库：PyQt5

·语音识别库：speech\_recognition

·音乐播放：pygame

·打开计算机应用程序：subprocess

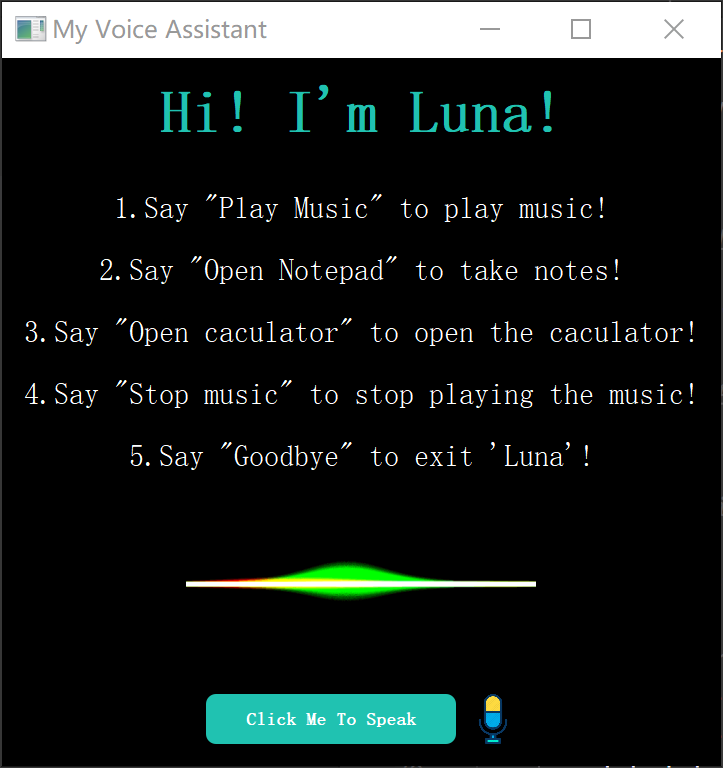
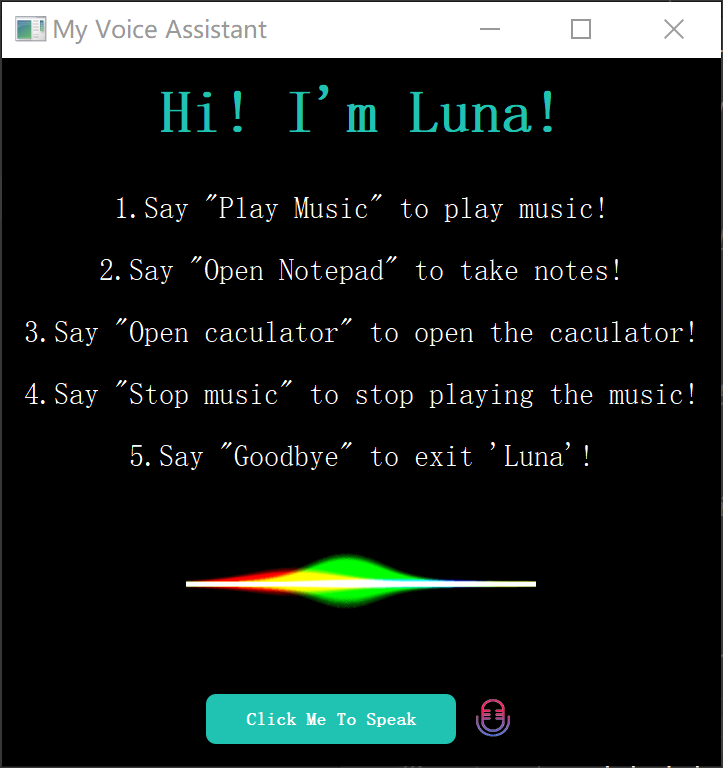
·比较相似度：difflib

### 2.Modifications

通过对原有界面asr.py/asrInterface.py的分析与学习，以及通过各大博客和官网对PyQt5的学习，我将原来的界面进行了大幅度的修改，同时通过查阅资料给语音识别系统的功能增加了“打开计算器”以及“停止播放音乐”的功能。

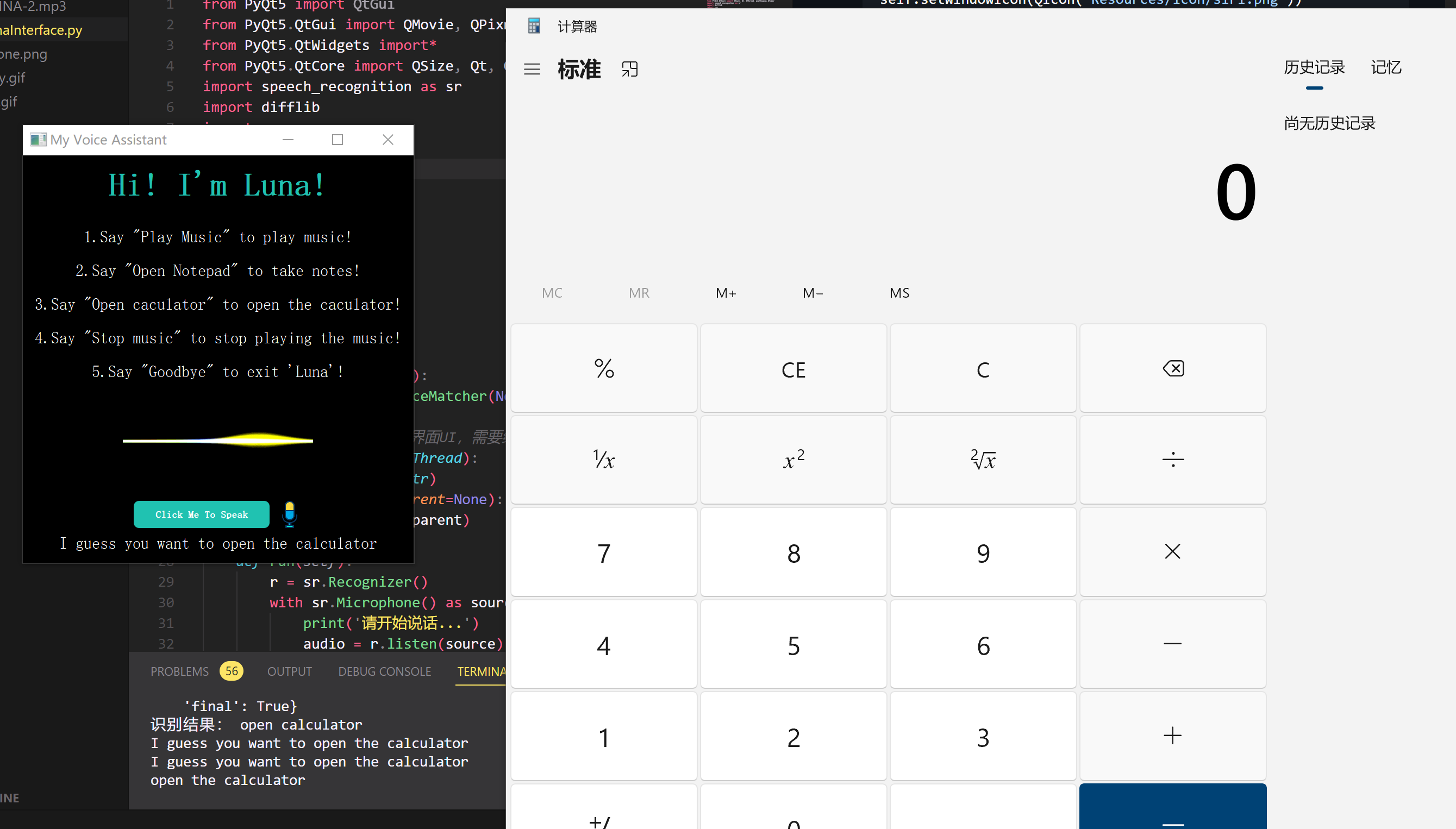
2.1Modifications To GUI

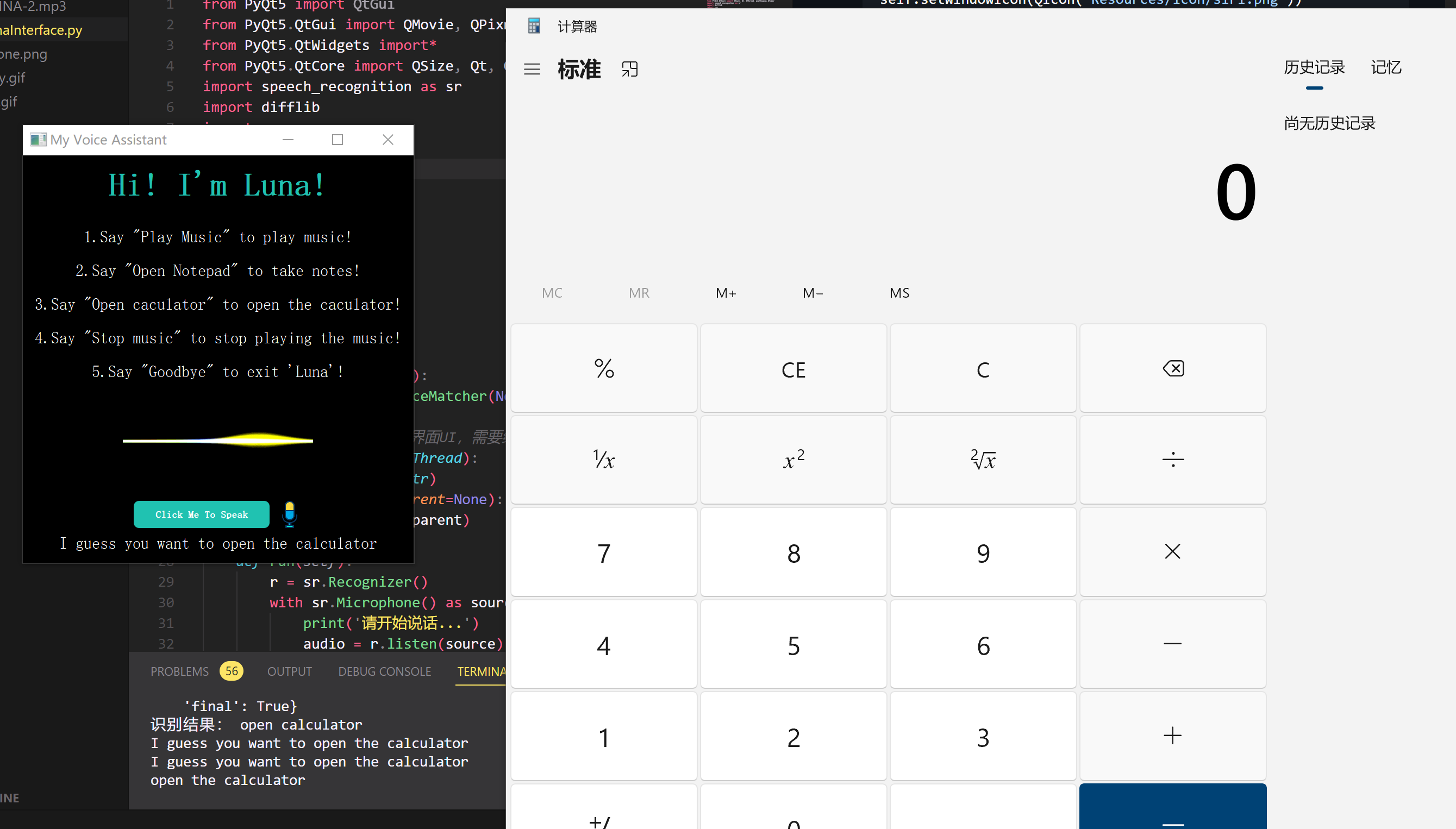
1. 将窗口的名称改成“My Voice Assistant”，简洁明了的指出系统的功能。
2. 给该语音识别系统取名为“Luna”，该名字叫起来很亲切，使得其更容易被人们记住。
3. 下面罗列了该系统的功能，以及操作方法，简单易读，便于用户上手操作。
4. 加上了一个动态的“彩色音频条”，使得界面呈现动态，不那么单调。
5. 加上了一个按钮，当用户想要说话的时候点击按钮开始语音识别，其他时候关闭语音识别功能，避免了在不想识别的时候，因为环境杂音识别到不想要的信息。用户点击按钮的时候，按钮会变成深蓝色，同时右边的话筒图标变成动态的，从而给予用户一个“系统已经被唤醒，正在进行语音识别”的反馈。



1. 当用户按照上面的指示说了自己想要执行的指令之后，系统会对用户读入的内容进行语音识别，并将识别结果反馈在界面的下方。然后执行相应的操作。

eg.用户对Luna说“Open calculator”，Luna先将识别结果显示在界面下方，然后打开系统的计算器。





#### 2.2Modifications To Functions

##### 2.2.1Speech Recognition

首先，我给系统添加了语音识别的方法。因为我的系统当点击下方按钮的时候要同时使得右边的话筒变成动图，并且进行语音识别，这两个操作得同时实现，所以我需要给语音识别利用QThread创建一个新的线程，使得这两个工作同时实现。

因此，我写了一个继承自Qthread类的语音识别的类：

*#为了在语音识别的过程中更新界面UI，需要给其建立一个新的线程*

*class* RecognitionThread(*QThread*):

    signal = pyqtSignal(*str*)

*def* \_\_init\_\_(*self*, *parent*=None):

*super*().\_\_init\_\_(parent)

*def* run(*self*):

        r = sr.Recognizer()

        with sr.Microphone() as source:

            print('请开始说话...')

            audio = r.listen(source)

        try:

            text = r.recognize\_google(audio, *language*='en-US')

*self*.signal.emit(text)

        except sr.UnknownValueError:

            print('无法识别输入音频')

        except sr.RequestError as e:

            print('无法从 Google Web 语音 API 中获取识别结果，错误原因：', e)

*self*.signal.emit('')

            print('无法识别输入音频')

        except sr.RequestError as e:

            print('无法从 Google Web 语音 API 中获取识别结果，错误原因：', e)

*self*.signal.emit('')

这里需要注意，为了能够更加精准的进行语音识别，我没有使用PocketSphinx进行语音识别，而是使用了SpeechRecognition库中进度更高的recognize\_google函数进行语音识别，但是该函数需要调用Google的API，因此，如果想要让程序正常进行识别，在国内需要开代理。

下面是我应用我写的类来实现语音识别的函数：

*#点击按钮开始进行语音识别*

*def* start\_recognition(*self*):

*# 将右下角的phone图标变成动图*

        phone\_dy = QMovie(*self*.PhoneGif)

*# 调整动图大小*

        scaled\_size = QSize(50, 50)

        phone\_dy.setScaledSize(scaled\_size)

*#QLabel控件是 Qt 中用于显示文本和图像的控件*

*self*.PhoneLabel.setMovie(phone\_dy)  *#将动图放入QLabel中*

*self*.PhoneLabel.setAlignment(Qt.AlignCenter)

*#使得动图动起来的启动函数*

        phone\_dy.start()*#如果没有调用 movie.start() 方法，动图就不会动起来*

        print('phone该动动了吧')

*#这边需要给语音识别创建一个新的线程才可以使得上面的动起来*

*# 创建一个语音识别线程，用于在后台进行语音识别*

*self*.thread = RecognitionThread(*self*)

*self*.thread.signal.connect(*self*.show\_recognition\_result)

*self*.thread.start()

这个函数是按钮绑定的函数，一但按下按钮，就开始执行该函数。这个函数中首先将右下角的话筒图片换成GIF动图，然后创建一个thread线程，进行语音识别，并将其与槽函数show\_recognition\_result相连接。

下面给出处理识别结果，并将结果显示在界面上的函数：

*# 显示语音识别的结果*

*def* show\_recognition\_result(*self*, *text*):

*# 将右下角的 phone.png 图标显示在 QLabel 控件中*

        phone\_icon = QPixmap(os.path.abspath('phone.png'))

        scaled\_size = QSize(50, 50)

        phone\_icon = phone\_icon.scaled(scaled\_size, Qt.KeepAspectRatio)

*self*.PhoneLabel.setPixmap(phone\_icon)

*self*.PhoneLabel.setAlignment(Qt.AlignCenter)

        print('识别结果：',text)

*#接下来需要将识别的结果进行相似度分析，选择相似度最高的指令执行相应的操作*

        commend=["play music","open notepad","open the calculator","stop music","Goodbye"]

*list* = [string\_similar(text, "play music"),

                string\_similar(text, "open notepad"),

                string\_similar(text, "open the calculator"),

                string\_similar(text, "stop music"),

                string\_similar(text, "Goodbye")]

        max\_value = max(*list*)           *# 获得这中间的最大值*

        max\_index = *list*.index(max\_value)     *# 最大值的索引*

        print("I guess you want to",commend[max\_index])

*#在屏幕上打印出*

*self*.show\_text("I guess you want to "+commend[max\_index])

*#打开对应的功能*

*self*.execution(commend[max\_index])

*#接下来需要将识别的结果进行相似度分析，选择相似度最高的指令执行相应的操作*

        commend=["play music","open notepad","open the calculator","stop music","Goodbye"]

*list* = [string\_similar(text, "play music"),

                string\_similar(text, "open notepad"),

                string\_similar(text, "open the calculator"),

                string\_similar(text, "stop music"),

                string\_similar(text, "Goodbye")]

        max\_value = max(*list*)           *# 获得这中间的最大值*

        max\_index = *list*.index(max\_value)     *# 最大值的索引*

        print("I guess you want to",commend[max\_index])

*#在屏幕上打印出*

*self*.show\_text("I guess you want to "+commend[max\_index])

*#打开对应的功能*

*self*.execution(commend[max\_index])

该函数给出了语音识别的结果并将其打印在控制台中，同时对其识别的结果进行处理，将识别出的指令利用show\_text函数显示在界面的下方，并在execution函数中执行相应的操作。

#### 2.2.2 Add More Functions

我给该语音识别系统增加了以下功能：

·用户说“Open The Calculator”来打开计算器。

·用户说“Stop Music”来在音乐播放的情况下停止音乐的播放。

·用户说“Goodbye”来退出Luna。（因为原来的那个系统好像没有退出系统的方法）

具体功能的实现在函数execution中：

*def* execution(*self*,*command*):

        print(command)

        if command=="play music":

*self*.playing=True

*#播放音乐*

            pygame.mixer.init()

            pygame.mixer.music.load(*self*.Music\_CHINA)

            pygame.mixer.music.play()

        if command=="open the calculator":

            subprocess.Popen('calc.exe')

        if command=="open notepad":

            subprocess.Popen('notepad.exe')

        if command=="stop music":

            if *self*.playing:

                pygame.mixer.music.stop()

*self*.playing=False

        if command=="Goodbye":

*# 设置 QLabel 的文本并居中显示*

*self*.show\_text("See You Next Time!")

*# time.sleep(5000)*

            sys.exit()

*self*.playing=False

        if command=="Goodbye":

*# 设置 QLabel 的文本并居中显示*

*self*.show\_text("See You Next Time!")

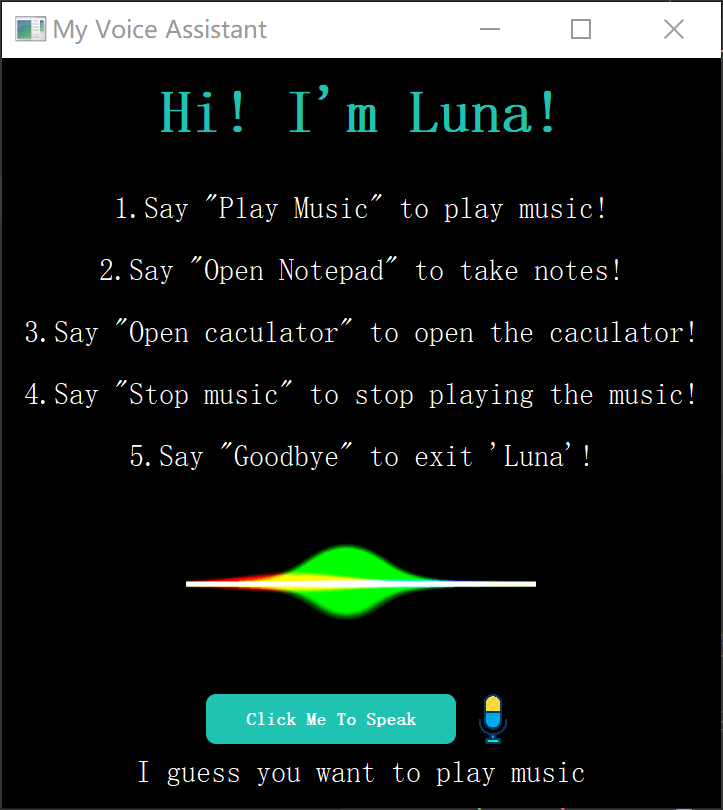
*# time.sleep(5000)*

            sys.exit()

在这边播放音乐需要下载pygame库来实现。

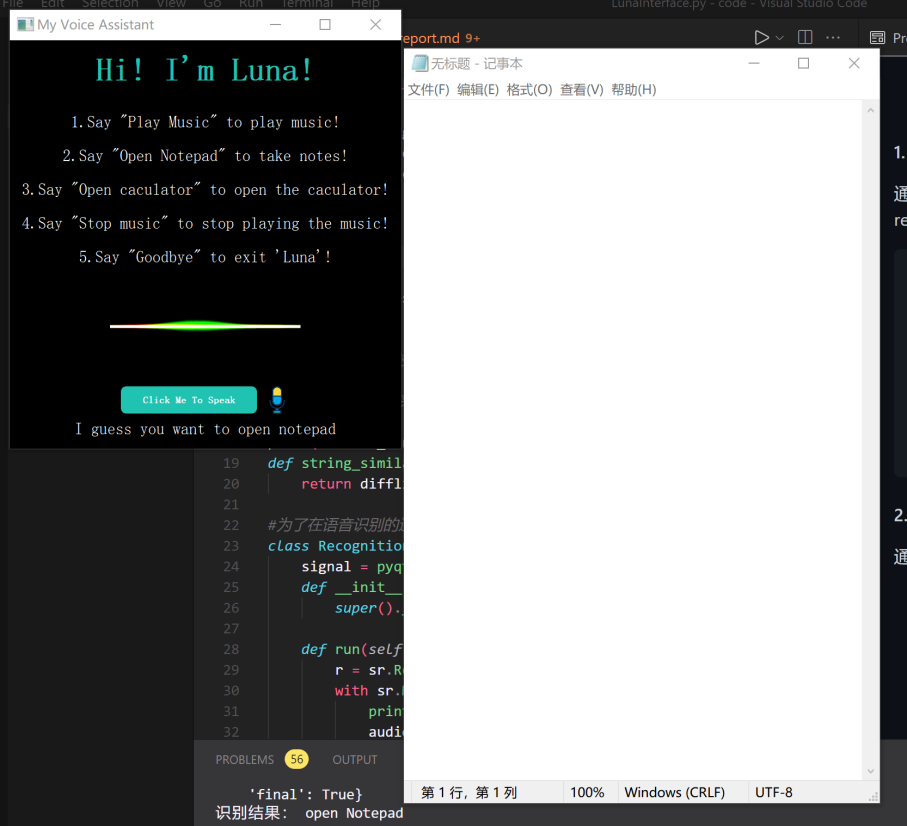
**功能展示：**

1. play music



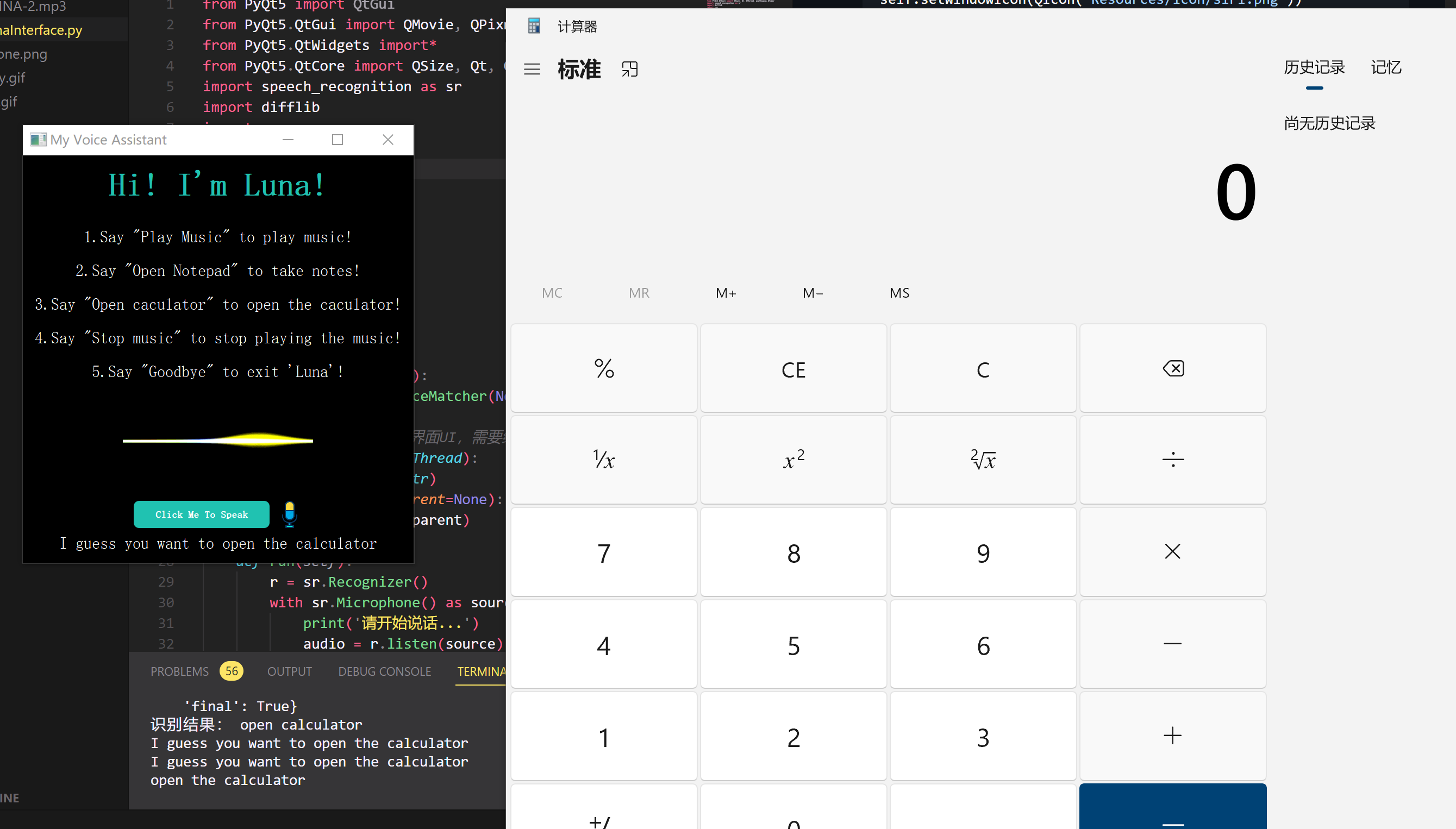
识别结果展示在按钮下方，并且可以听到正在播放mp3音乐。播放音乐之后，如果说“Stop Music”就可以停止播放音乐。

1. Open Notepad



当说出“Open Notepad”的时候，可以正常打开记事本。

1. Open calculator



当说出“Open calculator”的时候，可以正常打开计算器。

**3.语音识别的准确性与准确性提高方法**

在制作该套语音识别系统的时候，我发现有的时候我说的话不一定会被百分之一百识别正确，如果直接运用recognize\_google的识别结果作为指令，极有可能会发现没有匹配的指令可以执行。所以我们需要对recognize\_google的识别结果进行一些处理。

我引进了一个函数string\_similar来比较两个字符串的相似程度：

*def* string\_similar(*s1*, *s2*):

    return difflib.SequenceMatcher(None, s1, s2).quick\_ratio()

其返回的值在0~1之间，越接近1表示两个字符串的相似度越大，越接近0则越不相似。

        commend=["play music","open notepad","open the calculator","stop music","Goodbye"]

*list* = [string\_similar(text, "play music"),

                string\_similar(text, "open notepad"),

                string\_similar(text, "open the calculator"),

                string\_similar(text, "stop music"),

                string\_similar(text, "Goodbye")]

        max\_value = max(*list*)           *# 获得这中间的最大值*

        max\_index = *list*.index(max\_value)     *# 最大值的索引*

然后我们将我们会用到的用户的指令存放在一个数组之中，再将识别结果text与每一个指令用string\_similar作比较，得到相似度存在list之中，最相似的那一个就是用户想要执行的指令，从而大大减少了因为识别有一些些失误，而让系统不能执行指令。

另一方面，语音识别的准确性也与我们调用的库函数有关系。PocketSinphinx识别准确性特别差，因此我在制作语音识别系统的时候用了recognize\_google函数，该函数语音识别效果较好，但是缺点是recognize\_google需要连接代理才可以使用，直接使用国内wifi则会调用API失败。