**Report for HCI Lab1**

邓泉

软件工程学院

同济大学

中国上海，邮箱：

[1953871@tongji.edu.cn](mailto:1953871@tongji.edu.cn)

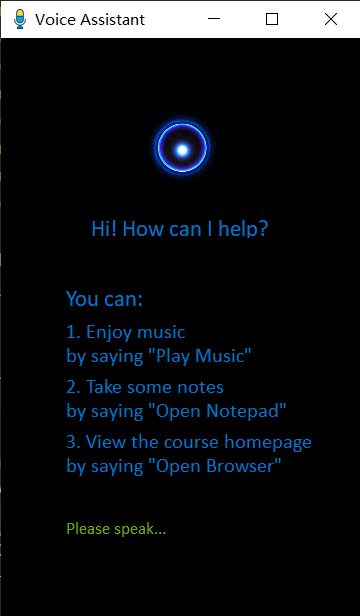
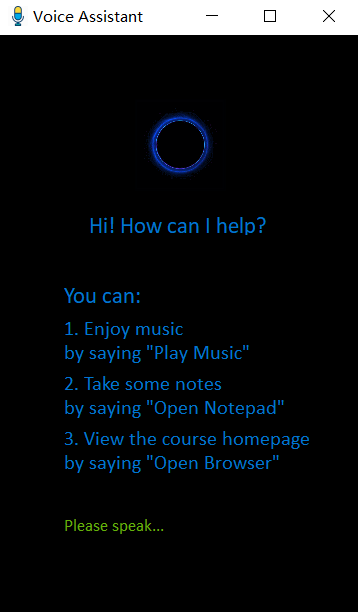
1. **GUI**

在语音助手界面上，我把语音识别的动态过程与用户交互技术相结合,提高了交互性和用户体验。程序能够实现循环识别，直到用户关闭程序窗口。程序支持三个命令，包括“play music”，“open notepad”和“open broeser”，它们将分别打开预设音频文件，txt文档和浏览器中的HCI课程主页。

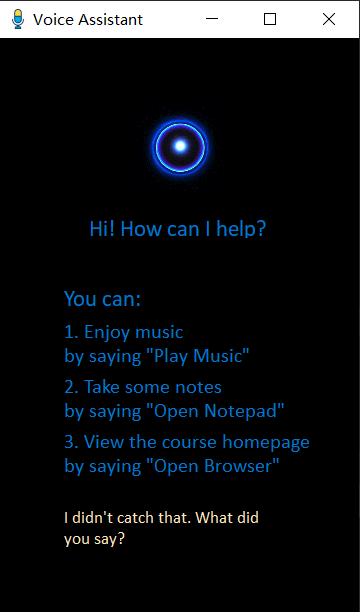
GUI主要设计:

* 设置了窗口标题”Voice Assistant”和**图标**;
* 界面中上方是**Gif动态图**,下方有颜色醒目的**文字提示栏**,两者相结合用于提示用户程序的当前运行状态;
* Gif动态图下方有文字” Hi! How can I help?”,以及告知用户可以做什么的说明文字。

1. 当程序正在接收或转化用户的语音指令时,文字提示栏显示绿色的“**Please speak...**”。此时,PyAudio模块调用设备的麦克风接收声音,程序调用recognize\_speech\_from\_mic函数将收到的麦克风声音转化成文本语言。

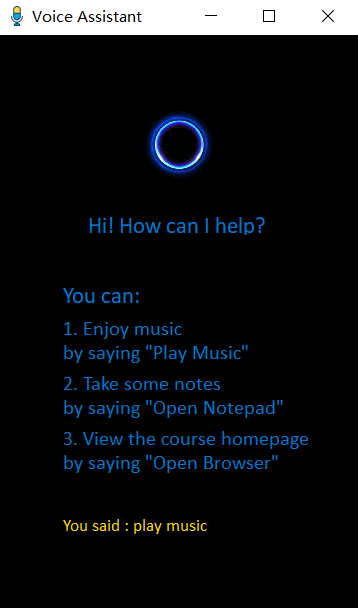


1. 当设备出现故障或用户不说话时，则程序将检测不到声音, 界面底部的提示栏将更改为**白色文本**“**I didn't catch that. What did you say?**”。

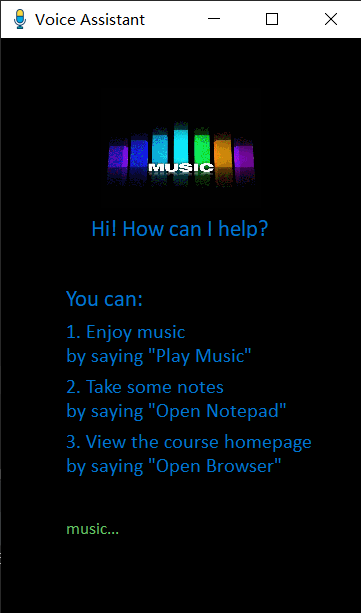
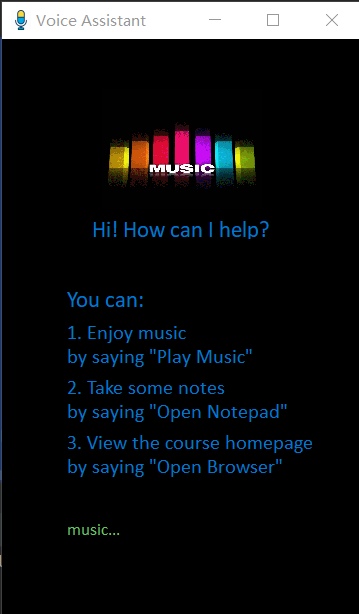


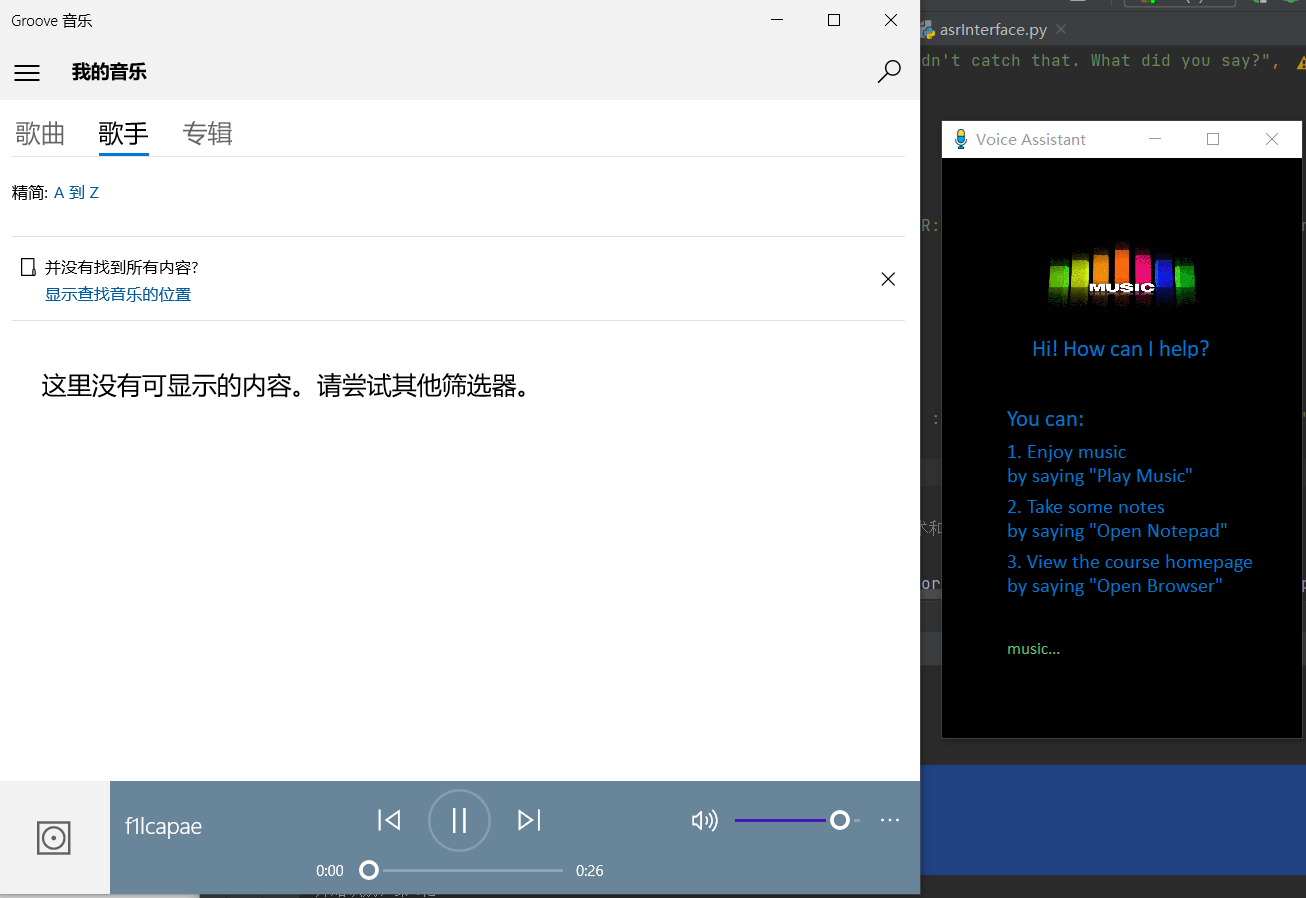
持续3s后，程序界面将回到1)状态。

1. 当用户说出指令”**Play Music**”,程序正确接收时:
   1. 文字提示栏首先用**黄色字体**显示”You said :” + 程序所识别转化的文本(play music)



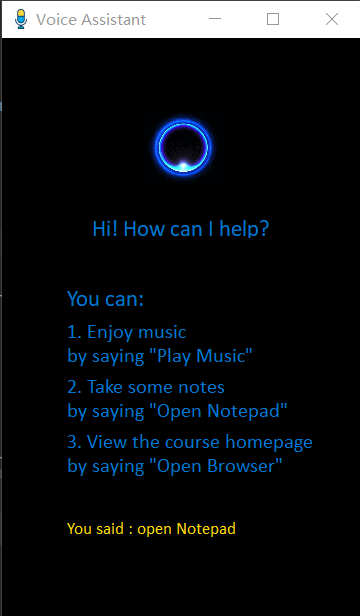
* 1. 2s后,界面上方的Gif图和文字提示栏均发生适应的改变,且音乐播放器被打开播放指定的音乐文件

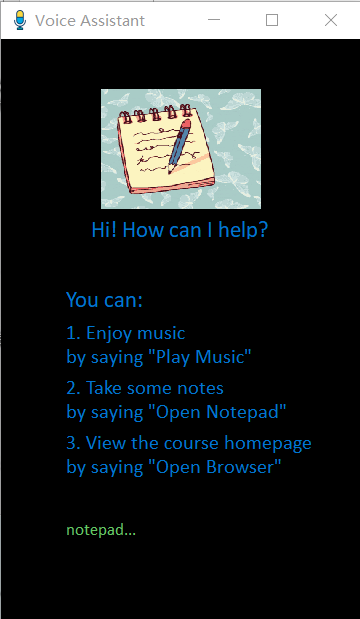
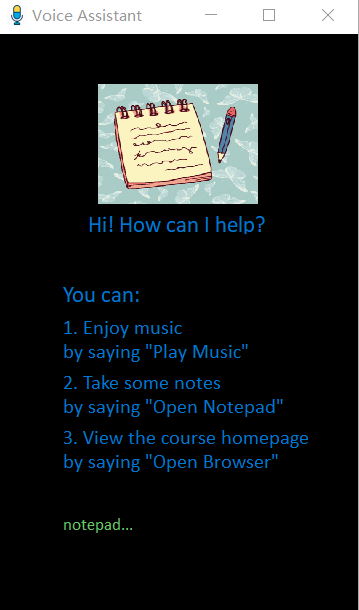


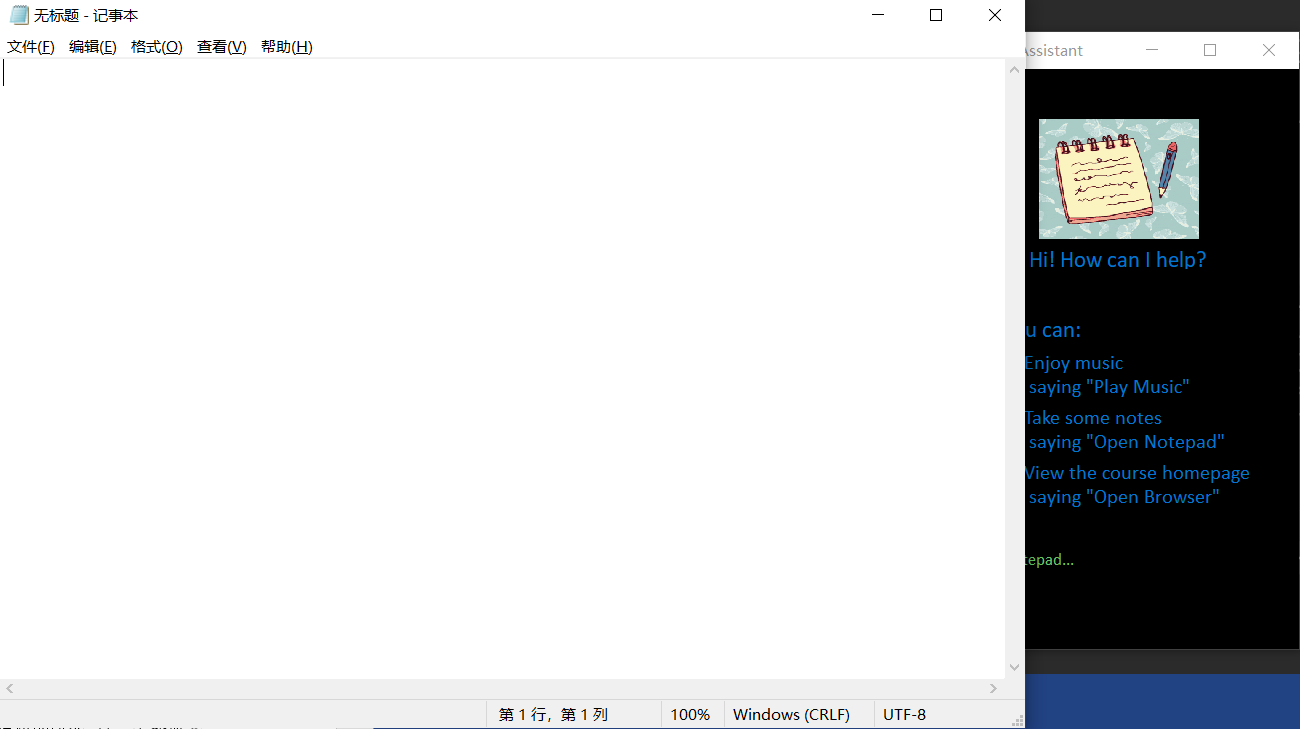
* 1. 再过3s后,程序界面回到1)状态。

1. 当用户说出指令”**Open Notepad**”,程序正确接收时:
   1. 文字提示栏首先用**黄色字体**显示”You said :” + 程序所识别转化的文本(open Notepad)



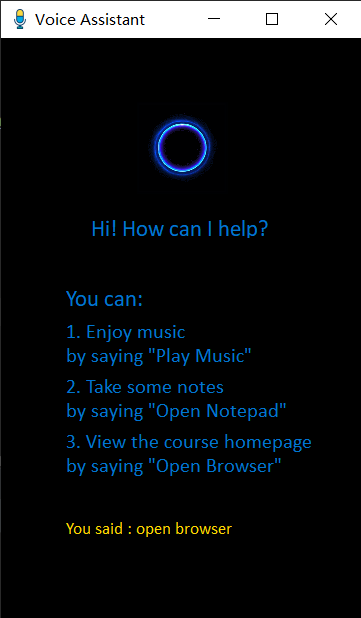
* 1. 2s后,界面上方的Gif图和文字提示栏均发生适应的改变,且记事本notepad被打开

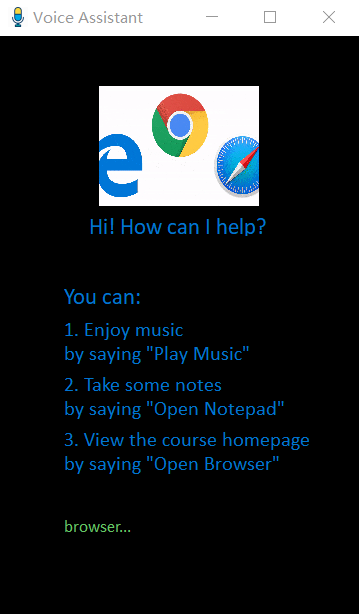
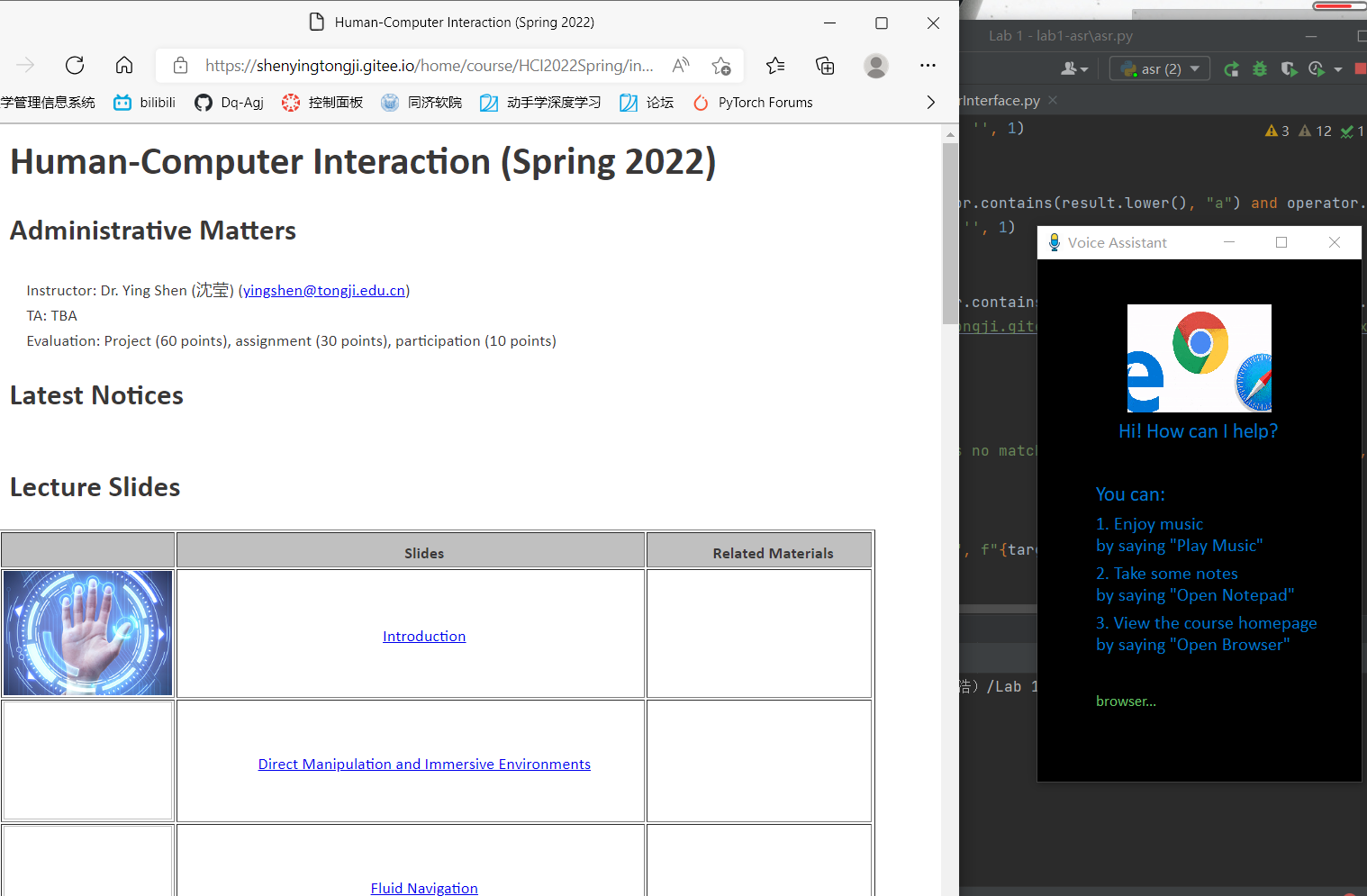
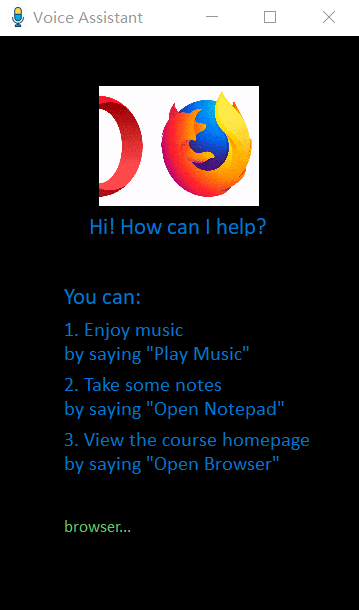


* 1. 再过3s后,程序界面回到1)状态。

1. 当用户说出指令”**Open Browser**”,程序正确接收时:
   1. 文字提示栏首先用**黄色字体**显示”You said :” + 程序所识别转化的文本(Open Browser)

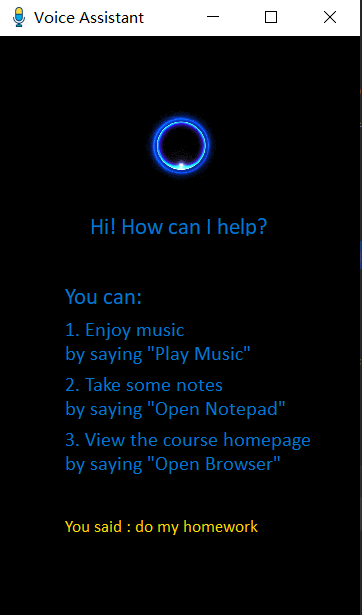


* 1. 2s后,界面上方的Gif图和文字提示栏均发生适应的改变,且HCI课程主页在默认浏览器中被打开

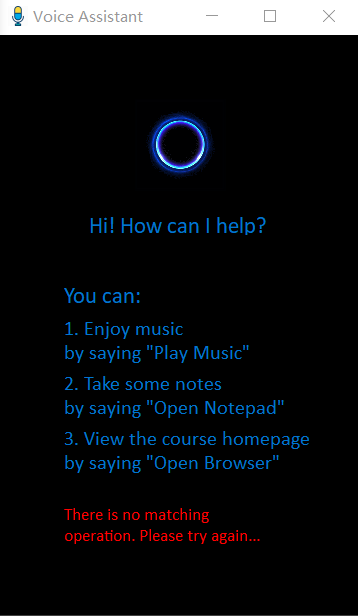
 

* 1. 再过3s后,程序界面回到1)状态。

1. 程序未正确识别或用户的指令不正确时(eg:用户指令被识别为”do my homework”):
   1. 文字提示栏同样先显示”You said :” + 程序所识别转化的文本



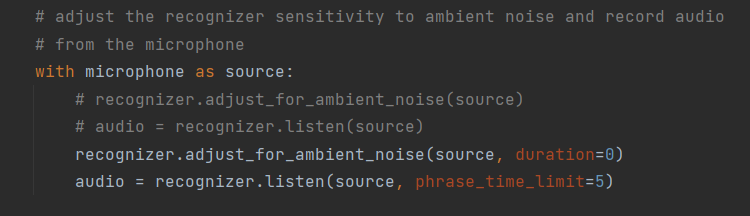
* 1. 2s后用**红色字体**显示错误信息” There is no matching operation. Please try again...”



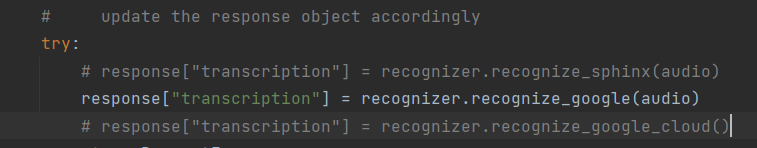
* 1. 再过3s后,程序界面回到1)状态。

1. **Accuracy of speech recognition & Code**

对于语音识别功能本身，引入guessTheWord.py中的函数recognize\_speech\_from\_mic来实现,但做了一些修改:adjust\_for\_ambient\_noise方法中的等待时间参数持续时间设置为0;并将phrase\_time\_limit in listen方法的持续时间参数设置为5，以确保在嘈杂的环境中不会受到杂音和超长识别的干扰。相关代码如下:



在自动语音识别工具包的选择上,弃用了原本的CMU Sphinx,采用识别率更高(针对本次实验)的语音Google Web Speech API。相关代码如下:



由于api的识别效果，试图让程序识别整个命令语句或单词并做出响应，有时候会出现错误。为提高程序的运行流畅度和用户体验感, 在识别整个语句失败后,程序将会继续尝试识别, 此时将单词拆分为音节进行识别，通过识别关键音节来获取命令。这样的改进大大增强了识别成功率。

例如，如果用户说“music”，只要我能识别出关键的几个重音音节，如m，u，c或其他主要音节，程序就会认为识别成功并执行相应的win32api操作。

相关代码如下:



对于识别效果的**进一步改进**:

* 通过使用更多特征音节进行匹配，可以优化音节匹配,从而提高识别准确率
* 调节语音API的相关参数,选出使用场景下的最优参数
* 尝试使用市场上其他优秀的语音识别工具包

对于根据识别结果在GUI界面上进行动态显示的功能，最初使用了signal和slot机制，但发现界面在显示切换过程中存在卡顿的现象。通过不断的反复试验，发现在此基础上使用多线程可以解决这个问题。相关代码如下:



程序中有多个线程同时操作一个对象，如果没有很好地保护该对象，会造成程序结果的不可预期.因此为了确保线程安全，程序使用了Threading模块中的Lock机制。创建一个该类对象，在线程函数执行前，“抢占”该锁，执行完成后，“释放”该锁，则确保了每次只有一个线程占有该锁。这时候对一个公共的GUI进行操作，则不会发生线程不安全的现象了。相关代码如下:

