**Python 小组项目**

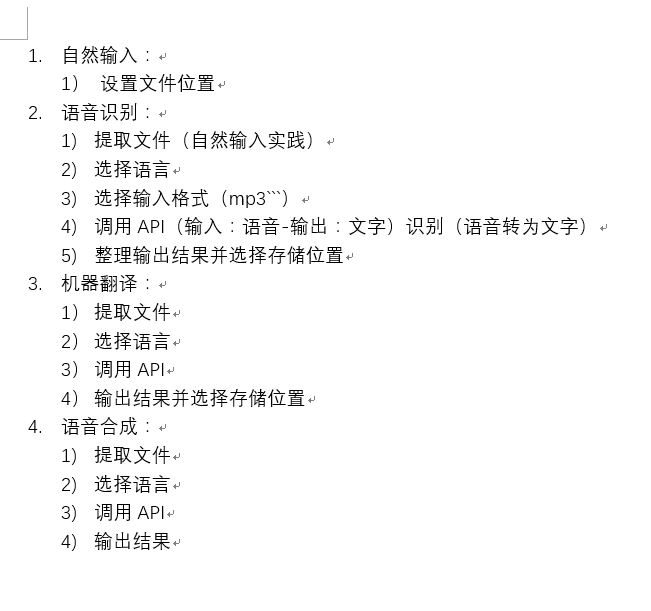
项目周期：5 weeks

小组成员：苏珊、杨若瑶、朱莹、徐林源

* **Project II 开发一个多语种自动语音翻译（问答）系统 ---- 综合使用 ASR, MT, TTS,** **QA （KG) & Cralwer**

Project II

**一、项目整体流程：**



问题补充：类的封装、多个API备用、语音分割（固定输入时长）、选择语言（API是否兼容）、判断是否需要分词断句（中文需要）、建立语言模型（API内部）

**二、语音识别流程**

＃1 从麦克风获取音频，将音频写入RAW/WAV/AIFF/FLAC文件

获取音频，将音频转换为所需格式

<https://github.com/Uberi/speech_recognition/blob/master/examples/write_audio.py>

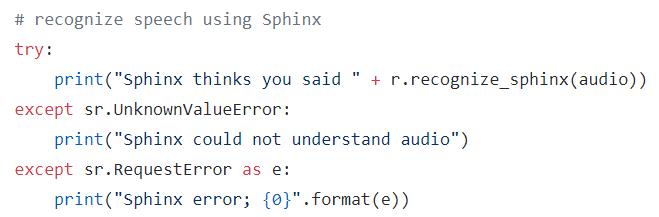
＃2消除噪声对语音识别的影响，默认将文件流的第一秒识别为音频的噪声级别<https://github.com/Uberi/speech_recognition/blob/master/examples/calibrate_energy_threshold.py>

＃3 选择语言

＃4使用特定路径下的音频文件作为音频源， 使用Sphinx/Google语音识别功能/Google Cloud Speech/Wit.ai/Microsoft Bing Voice Recognition…进行语音识别

<https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py>

<https://github.com/Uberi/speech_recognition/blob/master/examples/extended_results.py> （多个返回结果，第一个为最佳结果）



＃5 将语音识别出的信息存储在相应位置，统一转换为机器翻译需要的格式

运行注意事项：搭建环境、库的官方文档（其他功能介绍）

**三、机器翻译流程：**

＃1 登录

＃2 读取文件

＃3 根据调用API接受的最大译文长度，判断是否需要分段

分句（根据停顿间隔、语义？语音识别后无标点符号？）

＃4 选择语言

＃5 加密用户名密码信息

＃6 获取API返回翻译结果

＃7 保存至结果文件，转换成下一接口接受格式输出

**四、注意事项：**

代码包括：两个流程封装成两个类、测试资料与测试代码准备（测试结果展示）

编码文档：代码中的标注（文件级别、类级别、函数级别（参数、返回值、重点语句解释））

开发文档：老师提供模板

项目实现：前端UI和各程序间的连接（老师、感兴趣同学）、注意留有接口（测试时可应用假数据）

**五、例子代码阅读：**

**语音识别：**

<https://github.com/Uberi/speech_recognition/blob/master/speech_recognition/__init__.py>

**5-34 引入库、环境配置（Python 2 3）**

Urllib是python内置的HTTP请求库 <https://www.cnblogs.com/zhaof/p/6910871.html>

**37-54 异常错误检查**

**46-206 麦克风输入**

class AudioSource(object):

class Microphone(AudioSource):

**209-322 读取音频文件**

class AudioFile(AudioSource):

def \_\_init\_\_(self, filename\_or\_fileobject):

def \_\_enter\_\_(self): 尝试以各种格式读取音频文件

def \_\_exit\_\_(self, exc\_type, exc\_value, traceback):

class AudioFileStream(object):音频格式转换

“””

创建一个新的``AudioFile``实例，给定一个WAV / AIFF / FLAC音频文件``filename\_or\_fileobject``。AudioSource的子类。

如果``filename\_or\_fileobject``是一个字符串，则它被解释为文件系统上音频文件的路径。否则，``filename\_or\_fileobject``应该是类似文件的对象，比如``io.BytesIO``或类似的文件。

请注意，从音频中读取的函数（例如``recognizer\_instance.record``或``recognizer\_instance.listen``）将在流中向前移动。例如，如果您执行``recognizer\_instance.record（audiofile\_instance，duration = 10）``两次，第一次它将返回前10秒的音频，第二次它将返回之后10秒的音频。当输入一个AudioFile context时，它总是重置为开始。

WAV文件必须是PCM / LPCM格式; WAVE\_FORMAT\_EXTENSIBLE和压缩的WAV不受支持，并可能导致未定义的行为。

支持AIFF和AIFF-C（压缩AIFF）格式。

FLAC文件必须采用本地FLAC格式; OGG-FLAC不受支持，可能导致未定义的行为。

“””

# hasattr() 函数用于判断对象是否包含对应的属性。

**325-495 写入音频文件**

class AudioData(object):

**499-1262 语音识别**

class Recognizer(AudioSource):

def \_\_init\_\_(self):

def record(self, source, duration=None, offset=None): 语音识别、开始、结束

def adjust\_for\_ambient\_noise(self, source, duration=1): 环境噪声

def snowboy\_wait\_for\_hot\_word(self, snowboy\_location, snowboy\_hot\_word\_files, source, timeout=None): 等待热词

def listen(self, source, timeout=None, phrase\_time\_limit=None, snowboy\_configuration=None): 热词检测（包括断句相关内容）

def listen\_in\_background(self, source, callback, phrase\_time\_limit=None): 后台监听器

**746** def recognize\_sphinx(self, audio\_data, language="en-US", keyword\_entries=None, grammar=None, show\_all=False):

在``audio\_data``（AudioData`实例）上使用CMU Sphinx执行语音识别。

# import the PocketSphinx speech recognition module

# create decoder object

# obtain audio data

# obtain recognition results (keyword\_entries, grammar)

# return results

**845** def recognize\_google(self, audio\_data, key=None, language="en-US", show\_all=False):

# obtain audio transcription results

# ignore any blank blocks

# return results

**906** defrecognize\_google\_cloud(self, audio\_data, credentials\_json=None, language="en-US", preferred\_phrases=None, show\_all=False):

**981** def recognize\_wit(self, audio\_data, key, show\_all=False):

**1018** def recognize\_bing(self, audio\_data, key, language="en-US", show\_all=False):

**1111** def recognize\_houndify(self, audio\_data, client\_id, client\_key, show\_all=False):

**1164** def recognize\_ibm(self, audio\_data, username, password, language="en-US", show\_all=False): 有中文

def recognize\_tensorflow(self, audio\_data, tensor\_graph='tensorflow-data/conv\_actions\_frozen.pb', tensor\_label='tensorflow-data/conv\_actions\_labels.txt'): 引入tensorflow数据画图

**1265-1295 FLAC转换器**def get\_flac\_converter():

**1298-1304 python2 兼容性** def shutil\_which(pgm):

**1307-1333 文件可同时重复打开** class PortableNamedTemporaryFile(object):

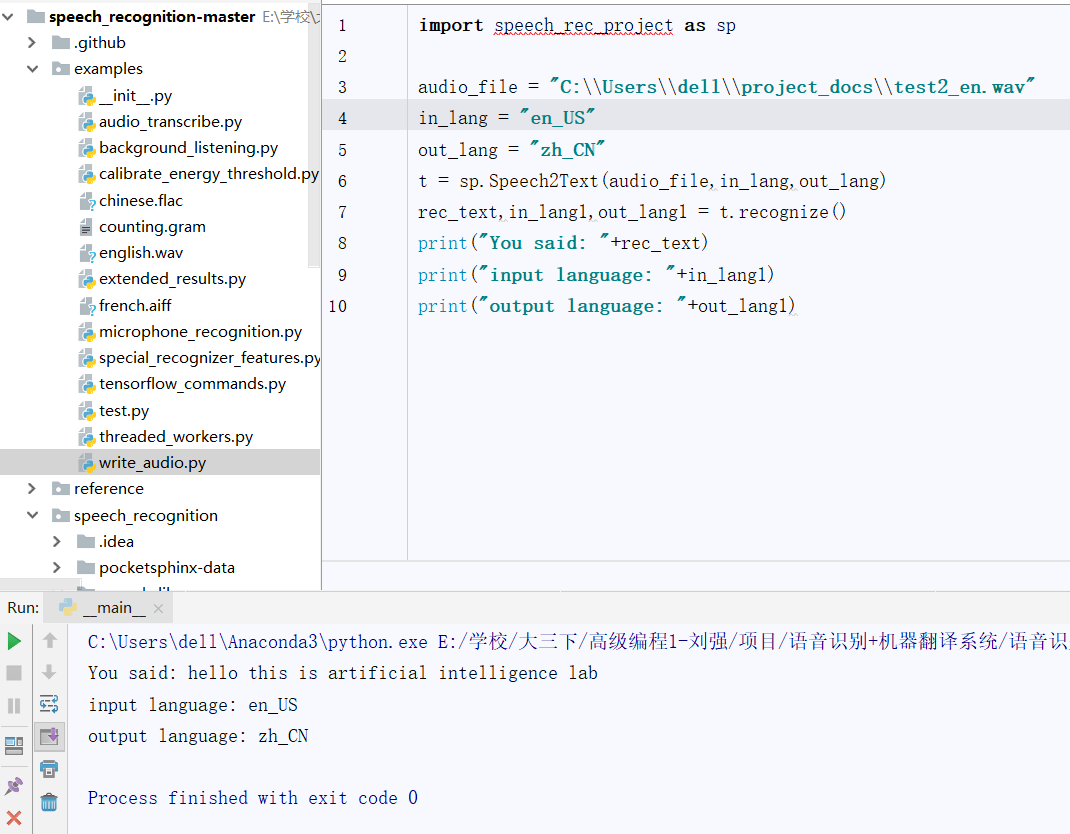
**1343-1360 另一个API语音识别，目前不适用python3，收费项目**

def recognize\_api(self, audio\_data, client\_access\_token, language="en", session\_id=None, show\_all=False):

**六、接口统一问题**

**前端 > 语音识别：** pcm或wav格式音频，具体传送形式（路径？）

**语音识别 > 机器翻译：**返回三个值，均为字符串形式



（1）rec\_text语音识别结果

机器翻译用的时候需要判断一下，如果是空字符串（rec\_text=””）就不要进行机器翻译，等待用户重新说话。

str=""

if str.strip()=="":

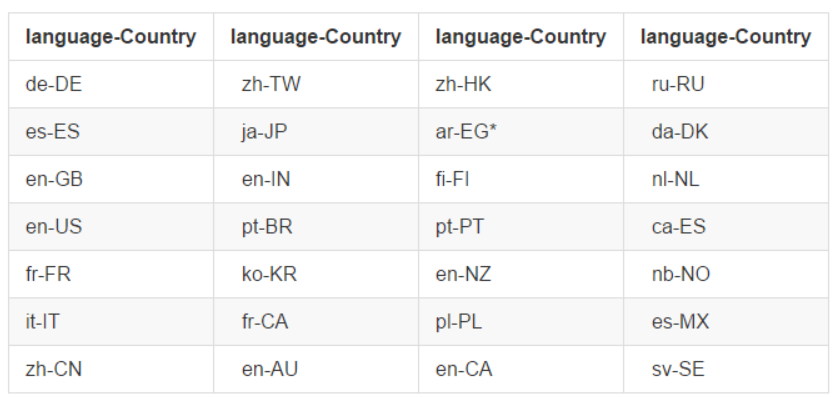
print("str is null")

if not str.strip():

print("str is null")

（2）in\_lang1 源语言

（3）out\_lang1 目标语言

（语音识别支持语言）

（机器翻译支持语言）

**七、未实现功能**

**语音识别：**支持更多语言识别、支持更多API调用

**机器翻译：**

串通语音识别和机器翻译的代码，完成测试代码

整理编码文档、撰写开发文档