This program was created using “Python 3.5.2 :: Anaconda 4.2.0 (64-bit)” downloaded from <https://www.anaconda.com/download/>.

Cloud setup steps:

1. Create a google cloud account. This will ask for a credit card
2. Create a project
3. Create a bucket in google cloud storage for the project after signing in to google cloud dashboard
4. Upload a WAV file to the bucket
   1. Make sure that the WAV file to be uploaded is Mono
5. Make the WAV file **PUBLIC** and get the link that it shows similar to this <https://storage.googleapis.com/beng/longTest.wav>. This is known as this files’ **URI.**
   1. However, this will need to be changed to “gs://beng/longTest.wav” where “beng” is the **Bucket Name** and “longTest.wav” is the **File Name**.
      1. A feature can be created to parse the file path to the correct one if wanted.
   2. This can be made more efficient than manually grabbing the files, however it seems getting individual files from the Cloud Storage would be more difficult from the interface than just manually getting the link. A transcribe all could be made though, however, it is a complicated process even for the user.
      1. This would require the User to enable the Google Cloud JSON API and download the correct credentials for each bucket. Apparently buckets on the same project do not belong to the same owners.
6. Go to the google cloud console “API’s and Services -> credentials -> create credentials -> api key
7. Go to the google cloud console “API’s and Services” -> library ->and enable Speech

Interface Usage:

1. First enter your Speech enabled API key **“python -sk <yourKeyHere>”**
   1. This will generate a “speech.key” file in the applications location and will be used for future requests.
   2. This version has not enable multiple arguments to be fulfilled at once besides getting timing of words. Enabling the key and then trying to immediately transcribe in the same line is not possible “python -sk <yourKey> -t <yourURI>” is not possible
2. The second thing will be to actually transcribe the file in question**. “python -t <yourURI>”**
   1. This will submit the request and grab the name of the operation and put it in a file called “tName.txt”. This is required since the transcription is asynchronous and needs to be called on at a later time. We do this with a “fetch”.
   2. The timing of the words is enabled by default but can be turned off by using **“python -t <yourURI> -gt False”**
3. The last thing to do is to get the transcript with **“python -f”.** There are no arguments to pass since this action looks for the “tName.txt” file and submits a request for the results of the operation. Two files will be generated, one with the transcripts in a text file and one with the timing of the words in a CSV file.
   1. The names of these files are the names of their operation along with “\_transcript.txt” and “\_wordTimes.csv”