CONTENTS:

CHAPTER

ONE

SOME CONTEXT

When using your phone to call someone, it is necessary for the system to transmit, in addition to the actual data (i.e. your voice), some important metadata, such as whether the phone was picked up on the other side, or simply the number you're trying to reach. Transmission of such metadata is called *signaling*, and has evolved alongside the telephone for many years.

Back in the early days of telephony, register signaling (that is, indicating who you are trying to contact) was done manually, by talking to an operator working at the telephone exchange. This is an example of *in-band signaling* the metadata is sent using the same medium as the data, rather than on a separate channel.

While most of the signaling in modern systems is out-of-band to avoid data being falsely interpreted as metadata (look up "2600Hz" on Wikipedia for a fun read), landline telephony still uses in-band for register signaling. That signaling system is DTMF, which is the subject of this project.

Fun Fact

Before DTMF and even before computers, register signaling was first automated with Strowger switches, mechanical switches invented by Almon Brown Strowger. Strowger, an undertaker, was convinced that the wife of his competitor, who worked as a telephone operator, redirected all calls "to the undertaker" to her husband.

CHAPTER

TWO

WHAT IS DTMF?

DTMF, or “Dual-Tone Multi-Frequency” is a in-band register signaling system. In other words, it is a way for a person or a computer to transmit the number of the desired recipient to the telephone exchange, using the voice frequency band that will be used for the actual communication.

In yet another simpler set of words: it sends audible beeps through your telephone to tell the central exchange who you’re trying to contact. A computer on the other end decodes those beeps into numbers and connects your call appropriately.

As the name suggests, each beep is actually made up of two sine waves rather than a single one. The following matrix show all possible signals and the frequency combination to represent them:

DTMF keypad frequencies (with sound clips)

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Fig. 1: Source : Wikipedia

For example, pressing the “5” key on your landline phone will generate a sound made up of two sine waves, one at 770 Hz, and the other one at 1336 Hz.

Why two frequencies instead of one? There are two main reasons:

- First, it is much less likely for real data to be mistaken for a signal, whereas a good enough whistling could easily trick a single frequency system.
- Secondly, dual tones means we need only 8 frequencies to represent 16 different signals, and as such they can be placed much further apart from each other inside the voice frequency band, making it easier for decoders to do their jobs.

Before you go further, it is advised to play a bit with DTMF. Pick up your phone and press some keys to hear the beeps, or go to http://onlinetonegenerator.com/dtmf.html to generate them through your browser. You can even put your phone next to your computer speakers and dial a number this way! This will only work on landlines, though.

Fun Fact

Cell phones actually do their signaling out-of-band, but still play the sound of the beeps to the user even if they are not transmitted. This is a example of skeuomorphism.

CHAPTER

THREE

SO. . . WHAT DO I HAVE TO DO?

The goal of this project is to write a DTMF decoder: your program will take as input a sound file representing a phone "conversation", analyze it, and print on the standard output the recognized signals, one per line. An example invocation would look like this:

```
$ python decoder.py jenny.wav
8
6
7
5
3
0
9
```

Real DTMF decoders use the Goertzel algorithm for this task, which is basically a variation of the discrete fourier transform, but you are free to use a regular DFT if you so desire.

3.1 Parsing the sound files

The sound files that will be given are WAV files with a format made to resemble telephony : 8Khz sampling rate, mono channel, 8 bits PCM. It is highly suggested that you make use of scipy's wavfile.read function to help you parse them.

Fun Fact

The actual encoding used in telephony is not a linear PCM but a specific format called mu-law (or its variant a-law in Europe and Asia), which would require some magic on your part to parse the files or even play them outside of a competent player like VLC. The 8Khz and 8 bits per sample parts are the same, though!

3.2 DTMF tone duration

Unfortunately, there is no single standard regarding the minimum length of a tone and the blank between two tones, as the tones were originally generated by humans pressing the keys on their phones. For this project, we will stick to a lenient 100ms tone duration.

Be sure to debounce your signals! Two consecutive "one" signals separated by only 30ms should be interpreted as one long signal, but if they're separated by more than 100ms, then it should be two consecutive numbers. A very common mistake is to register the same signal multiple times when it fires.

3.3 Additional libraries

If you need to use libraries in addition to the standard lib, such as scipy, please indicate them in the *requirements.txt* file as per the output of *pip freeze*.

3.4 The write-up

In addtion to the program, you are asked to provide a README describing the process you went through to implement the project, in the ever-classical markdown format. One page is enough, but don't forget it, it's part of your grade!

3.5 Format of your submission

You must upload on the platform an archive named *<LOGIN>-jenny.tar.bz2*, containing the following:

```
<LOGIN>-jenny/
  |- decoder.py
  |- README.md
  |- requirements.txt (if needed)
```

Of course, replace *<LOGIN>* with your actual login.

CHAPTER

FOUR

SOME RESOURCES. . .

- https://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling
- https://www.sigidwiki.com/wiki/Dual_Tone_Multi_Frequency_(DTMF)
- http://onlinetonegenerator.com/dtmf.html