The Python function 'pack'

DSP Lab (EE 4163 / EL 6183)

`PyAudio` reads and writes to audio devices via binary strings (in Python 3, they are 'byte' data type). Therefore, to write numbers to an audio device, we use the Python function `pack()`, which converts numbers to binary strings. This conversion is needed because `PyAudio` can only read and write the audio signal values as *binary strings*. (In Python 3, they are data type 'bytes' .) The function `unpack()` converts strings back into numbers.

The `pack` and `unpack` functions are in the `struct` module. Documentation for `unpack()` and `pack()` is in the documentation for the `struct` module:

https://docs.python.org/3/library/struct.html

The conversion between numbers and binary strings is controlled by the formatting value (e.g., 'B', 'h', 'i'). The format 'h' converts between signed 16-bit integers and binary strings (`short` in the C language).

A binary string can be represented compactly using hexadecimal symbols (e.g., `'\x00'`). For example, if the format is a signed 16-bit integer, then the value '0' as a binary `string` can be displayed as `'\x00\x00'`, and value '1' as a `string '\x01\x00'`, etc., where the lower byte (8 bits) is on the left and high byte (8 bits) is on the right. More specifically, `'\x01\x00'` is the binary number

$$\texttt{1 as an integer} \longleftrightarrow \underbrace{\underbrace{\texttt{'\x01}}_{\text{low}}\ \underbrace{\texttt{\x00'}}_{\text{high}}}_{\text{2 bytes in hex}} \longleftrightarrow \underbrace{\underbrace{\texttt{0000 0000}}_{\text{high}}\underbrace{\texttt{0000 0001}}_{\text{low}}}_{\text{two bytes in binary}} \tag{1}$$