cedargrove_midi_tools

MIDI Tools

A CircuitPython method collection for processing MIDI notes and Control Change codes. It currently consists of six helpers for converting MIDI note values to and from frequency values and note name representations, and to provide descriptions of MIDI Control Change (CC) controller codes.

- Author(s): JG for Cedar Grove Studios

Implementation Notes

Hardware:

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases

note_or_name(*value*)

Bidirectionally translates a MIDI sequential note value to a note name or a note name to a MIDI sequential note value. Note values are of integer type in the range of 0 to 127 (inclusive). Note names are character strings expressed in the format NoteOctave, such as 'C4' or 'G#7'. Note names can range from 'C-1' (note value 0) to 'F#9' (note value 127). If the input value is outside of the note value or name range, the value of None is returned.

Parameters:	• **value** – The note name or note value input. Note value is an integer; note name is a string. No default value.

Example:

```
>>> from cedargrove_midi_tools import note_or_name
>>> note_or_name('G5')
79
>>> note_or_name(79)
'G5'
```

note_to_name(note)

Translates a MIDI sequential note value to a note name. Note values are of integer type in the range of 0 to 127 (inclusive). Note names are character strings expressed in the NoteOctave format, such as 'C4' or 'G#7'. Note names can range from 'C-1' (note value 0) to 'F#9' (note value 127). If the input value is outside of that range, the value of None is returned.

Parameters:	• **note** – The note value input in the range of 0 to 127 (inclusive). No default value.

name_to_note(name)

Translates a note name to a MIDI sequential note value. Note names are character strings expressed in the NoteOctave format, such as 'C4' or 'G#7'. Note names can range from 'C-1' (note value 0) to 'F#9' (note value 127). Note values are of integer type in the range of 0 to 127 (inclusive). If the input value is outside of that range, the value of None is returned.

Parameters:	• **name** – The note name input in NoteOctave format. No default value.

Example:

```
>>> from cedargrove_midi_tools import note_to_name, name_to_note
>>> note_to_name(70)
'A#4'
>>> name_to_note('A#4')
70
```

note_to_frequency(note)

Translates a MIDI sequential note value to its corresponding frequency in Hertz (Hz). Note values are of integer type in the range of 0 to 127 (inclusive). Frequency values are floating point. If the input note value is less than 0 or greater than 127, the input is invalid and the value of None is returned. Ref: MIDI Tuning Standard formula: https://en.wikipedia.org/wiki/MIDI_tuning_standard

Parameters:	• **note** – The MIDI note value input in the range of 0 to 127 (inclusive). No default.

frequency_to_note(frequency)

Translates a frequency in Hertz (Hz) to the closest MIDI sequential note value. Frequency values are floating point. Note values are of integer type in the range of 0 to 127 (inclusive). If the input frequency is less than the corresponding frequency for note 0 or greater than note 127, the input is invalid and the value of None is returned. Ref: MIDI Tuning Standard formula: https://en.wikipedia.org/wiki/MIDI_tuning_standard

Parameters:	• **frequency** – The frequency value input in Hz. No default.

Example:

```
>>> from cedargrove_midi_tools import note_to_frequency, frequency_to_note
>>> note_to_frequency(60)
261.625
>>> frequency_to_note(261.63)
60
```

cc_code_to_description(cc_code)

Provides a controller description decoded from a Control Change controller code value. Ref: https://www.midi.org/specifications-old/item/table-3-control-change-messages-data-bytes-2

Parameters:	• **cc_code** – The Control Change code value in the range of 0 to 127. No default value.

Example:

```
>>> from cedargrove_midi_tools import cc_code_to_description
>>> cc_code_to_description(24)
'Ctrl_24'
>>> cc_code_to_description(1)
'Modulation'
```