PyGuitarPro Documentation

Release 0.6

Sviatoslav Abakumov

Jan 19, 2020

Contents

PyGuitarPro is a package to read, write and manipulate GP3, GP4 and GP5 files. Initially PyGuitarPro is a Python port of AlphaTab which is a Haxe port of TuxGuitar.

To anyone wanting to create their own the best guitar tablature editor in Python this package will be the good thing to start with.

CHAPTER 1

Installation

Install the package with:

```
pip install pyguitarpro
```

For the latest development version:

```
git clone https://github.com/Perlence/PyGuitarPro
cd pyguitarpro
pip install -e .
```

CHAPTER 2

Quickstart

After the package has been installed, it's ready for hacking.

Reading `.gp*` files is as easy as:

```
import guitarpro
curl = guitarpro.parse('Mastodon - Curl of the Burl.gp5')
```

Writing `.gp*` files isn't that hard as well:

```
guitarpro.write(curl, 'Mastodon - Curl of the Burl 2.gp5')
```

All objects representing GP entities are hashable and comparable. This gives the great opportunity to apply *diff* algorithm to tabs, or even *diff3* algorithm to merge tablatures.

The package is also designed to convert tablatures between formats. To do this, simply change extension of the output file according to your desired format:

```
guitarpro.write(curl, 'Mastodon - Curl of the Burl.gp4')
```

Functions *guitarpro.parse()* and *guitarpro.write()* support not only filenames, but also file-like object:

```
from urllib.request import urlopen
stream = urlopen('https://github.com/Perlence/PyGuitarPro/raw/develop/tests/Mastodon
↪%20-%20Curl%20of%20the%20Burl.gp5')
curl = guitarpro.parse(stream)
```

Note: PyGuitarPro supports only GP3, GP4 and GP5 files. Support for GPX (Guitar Pro 6) files is out of scope of the project.

CHAPTER 3

API Reference

3.1 General functions

guitarpro.**parse**(*stream*, *encoding='cp1252'*)
Open a GP file and read its contents.

Parameters

- **stream** – path to a GP file or file-like object.
- **encoding** – decode strings in tablature using this charset. Given encoding must be an 8-bit charset.

guitarpro.**write**(*song*, *stream*, *version=None*, *encoding='cp1252'*)
Write a song into GP file.

Parameters

- **song** (guitarpro.models.Song) – a song to write.
- **stream** – path to save GP file or file-like object.
- **version** (*tuple*) – explicitly set version of GP file to save, e.g. `(5, 1, 0)`.
- **encoding** – encode strings into given 8-bit charset.

3.2 Models

exception guitarpro.models.**GPException**

class guitarpro.models.**RepeatGroup**(*measureHeaders=NOTHING*, *closings=NOTHING*, *openings=NOTHING*, *isClosed=False*)
This class can store the information about a group of measures which are repeated.

class guitarpro.models.**KeySignature**

class guitarpro.models.**LyricLine**(*startingMeasure=1*, *lyrics=''*)
A lyrics line.

class guitarpro.models.**Lyrics**(*trackChoice=0*, *lines=None*)
A collection of lyrics lines for a track.

class guitarpro.models.**Point**(*x*, *y*)
A point construct using floating point coordinates.

class guitarpro.models.**Padding**(*right*, *top*, *left*, *bottom*)
A padding construct.

class guitarpro.models.**HeaderFooterElements**
An enumeration of the elements which can be shown in the header and footer of a rendered song sheet.

All values can be combined using bit-operators as they are flags.

class guitarpro.models.**PageSetup**(*pageSize=Point(x=210, y=297)*, *pageMargin=Padding(right=10, top=15, left=10, bottom=10)*, *scoreSizeProportion=1.0*, *headerAndFooter=<HeaderFooterElements.all: 511>*, *title='%title%'*, *subtitle='%subtitle%'*, *artist='%artist%'*, *album='%album%'*, *words='Words by %words%'*, *music='Music by %music%'*, *wordsAndMusic='Words & Music by %WORDSMUSIC%'*, *copyright='Copyright %copyright%nAll Rights Reserved - International Copyright Secured'*, *pageNumber='Page %N%/%P%'*)
The page setup describes how the document is rendered.

Page setup contains page size, margins, paddings, and how the title elements are rendered.

Following template vars are available for defining the page texts:

- `%title%`: will be replaced with Song.title
- `%subtitle%`: will be replaced with Song.subtitle
- `%artist%`: will be replaced with Song.artist
- `%album%`: will be replaced with Song.album
- `%words%`: will be replaced with Song.words
- `%music%`: will be replaced with Song.music
- `%WORDSANDMUSIC%`: will be replaced with the according word and music values
- `%copyright%`: will be replaced with Song.copyright
- `%N%`: will be replaced with the current page number (if supported by layout)
- `%P%`: will be replaced with the number of pages (if supported by layout)

class guitarpro.models.**RSEEqualizer**(*knobs=NOTHING*, *gain=0.0*)
Equalizer found in master effect and track effect.

Attribute `RSEEqualizer.knobs` is a list of values in range from -6.0 to 5.9. Master effect has 10 knobs, track effect has 3 knobs. Gain is a value in range from -6.0 to 5.9 which can be found in both master and track effects and is named as "PRE" in Guitar Pro 5.

class guitarpro.models.**RSEMasterEffect**(*volume=0*, *reverb=0*, *equalizer=NOTHING*)
Master effect as seen in "Score information".

class guitarpro.models.**Song**(*versionTuple=None*, *clipboard=None*, *title=''*, *subtitle=''*, *artist=''*, *album=''*, *words=''*, *music=''*, *copyright=''*, *tab=''*, *instructions=''*, *notice=NOTHING*, *lyrics=NOTHING*, *pageSetup=NOTHING*, *tempoName='Moderate'*, *tempo=120*, *hideTempo=False*, *key=<KeySignature.CMajor: (0, 0)>*, *measureHeaders=None*, *tracks=None*, *masterEffect=NOTHING*, *currentRepeatGroup=NOTHING*)

The top-level node of the song model.

It contains basic information about the stored song.

class guitarpro.models.**Tempo**(*value=120*)

A song tempo in BPM.

class guitarpro.models.**MidiChannel**(*channel=0*, *effectChannel=1*, *instrument=25*, *volume=104*, *balance=64*, *chorus=0*, *reverb=0*, *phaser=0*, *tremolo=0*, *bank=0*)

A MIDI channel describes playing data for a track.

class guitarpro.models.**DirectionSign**(*name=''*)

A navigation sign like *Coda* or *Segno*.

class guitarpro.models.**Tuplet**(*enters=1*, *times=1*)

A *n:m* tuplet.

class guitarpro.models.**Duration**(*value=4*, *isDotted=False*, *tuplet=NOTHING*)

A duration.

class guitarpro.models.**TimeSignature**(*numerator=4*, *denominator=NOTHING*, *beams=NOTHING*)

A time signature.

class guitarpro.models.**TripletFeel**

An enumeration of different triplet feels.

none = 0

No triplet feel.

eighth = 1

Eighth triplet feel.

sixteenth = 2

Sixteenth triplet feel.

class guitarpro.models.**MeasureHeader**(*number=1*, *start=960*, *hasDoubleBar=False*, *keySignature=<KeySignature.CMajor: (0, 0)>*, *timeSignature=NOTHING*, *tempo=NOTHING*, *marker=None*, *isRepeatOpen=False*, *repeatAlternative=0*, *repeatClose=-1*, *tripletFeel=<TripletFeel.none: 0>*, *direction=None*, *fromDirection=None*)

A measure header contains metadata for measures over multiple tracks.

class guitarpro.models.**Color**(*r*, *g*, *b*, *a=1*)

An RGB Color.

class guitarpro.models.**Marker**(*title='Section'*, *color=Color(r=255, g=0, b=0, a=1)*)

A marker annotation for beats.

class guitarpro.models.**TrackSettings**(*tablature=True*, *notation=True*, *diagramsAreBelow=False*, *showRhythm=False*, *forceHorizontal=False*, *forceChannels=False*, *diagramList=True*, *diagramsInScore=False*, *autoLetRing=False*, *autoBrush=False*, *extendRhythmic=False*)

Settings of the track.

class guitarpro.models.**Accentuation**

Values of auto-accentuation on the beat found in track RSE settings.

none = 0

No auto-accentuation.

verySoft = 1

Very soft accentuation.

soft = 2

Soft accentuation.

medium = 3

Medium accentuation.

strong = 4

Strong accentuation.

veryStrong = 5

Very strong accentuation.

class guitarpro.models.**Track**(*song*, *number=1*, *fretCount=24*, *offset=0*, *isPercussionTrack=False*, *is12StringedGuitarTrack=False*, *isBanjoTrack=False*, *isVisible=True*, *isSolo=False*, *isMute=False*, *indicateTuning=False*, *name='Track 1'*, *measures=None*, *strings=None*, *port=1*, *channel=NOTHING*, *color=Color(r=255, g=0, b=0, a=1)*, *settings=NOTHING*, *useRSE=False*, *rse=NOTHING*)

A track contains multiple measures.

class guitarpro.models.**GuitarString**(*number*, *value*)

A guitar string with a special tuning.

class guitarpro.models.**MeasureClef**

An enumeration of available clefs.

class guitarpro.models.**LineBreak**

A line break directive.

none = 0

No line break.

break_ = 1

Break line.

protect = 2

Protect the line from breaking.

class guitarpro.models.**Measure**(*track*, *header*, *clef=<MeasureClef.treble: 0>*, *voices=None*, *lineBreak=<LineBreak.none: 0>*)

A measure contains multiple voices of beats.

class guitarpro.models.**VoiceDirection**

Voice directions indicating the direction of beams.

class guitarpro.models.**Voice**(*measure*, *beats=NOTHING*, *direction=<VoiceDirection.none: 0>*)

A voice contains multiple beats.

class guitarpro.models.**BeatStrokeDirection**
All beat stroke directions.

class guitarpro.models.**BeatStroke**(*direction=<BeatStrokeDirection.none: 0>*, *value=0*)
A stroke effect for beats.

class guitarpro.models.**SlapEffect**
Characteristic of articulation.

none = 0
No slap effect.

tapping = 1
Tapping.

slapping = 2
Slapping.

popping = 3
Popping.

class guitarpro.models.**BeatEffect**(*stroke=NOTHING*, *hasRasgueado=False*, *pickStroke=<BeatStrokeDirection.none: 0>*, *chord=None*, *fadeIn=False*, *tremoloBar=None*, *mixTableChange=None*, *slapEffect=<SlapEffect.none: 0>*, *vibrato=None*)
This class contains all beat effects.

class guitarpro.models.**TupletBracket**

class guitarpro.models.**BeatDisplay**(*breakBeam=False*, *forceBeam=False*, *beamDirection=<VoiceDirection.none: 0>*, *tupletBracket=<TupletBracket.none: 0>*, *breakSecondary=0*, *breakSecondaryTuplet=False*, *forceBracket=False*)
Parameters of beat display.

class guitarpro.models.**Octave**
Octave signs.

class guitarpro.models.**BeatStatus**

class guitarpro.models.**Beat**(*voice*, *notes=NOTHING*, *duration=NOTHING*, *text=None*, *start=None*, *effect=NOTHING*, *index=None*, *octave=<Octave.none: 0>*, *display=NOTHING*, *status=<BeatStatus.empty: 0>*)
A beat contains multiple notes.

class guitarpro.models.**HarmonicEffect**
A harmonic note effect.

class guitarpro.models.**NaturalHarmonic**

class guitarpro.models.**ArtificialHarmonic**(*pitch=None*, *octave=None*)

class guitarpro.models.**TappedHarmonic**(*fret=None*)

class guitarpro.models.**PinchHarmonic**

class guitarpro.models.**SemiHarmonic**

class guitarpro.models.**GraceEffectTransition**
All transition types for grace notes.

none = 0
No transition.

slide = 1
Slide from the grace note to the real one.

bend = 2
Perform a bend from the grace note to the real one.

hammer = 3
Perform a hammer on.

class guitarpro.models.**Velocities**
A collection of velocities / dynamics.

class guitarpro.models.**GraceEffect**(*duration=1*, *fret=0*, *isDead=False*, *isOnBeat=False*, *transition=<GraceEffectTransition.none: 0>*, *velocity=95*)
A grace note effect.

durationTime
Get the duration of the effect.

class guitarpro.models.**TrillEffect**(*fret=0*, *duration=NOTHING*)
A trill effect.

class guitarpro.models.**TremoloPickingEffect**(*duration=NOTHING*)
A tremolo picking effect.

class guitarpro.models.**SlideType**
An enumeration of all supported slide types.

class guitarpro.models.**Fingering**
Left and right hand fingering used in tabs and chord diagram editor.

unknown = -2
Unknown (used only in chord editor).

open = -1
Open or muted.

thumb = 0
Thumb.

index = 1
Index finger.

middle = 2
Middle finger.

annular = 3
Annular finger.

little = 4
Little finger.

class guitarpro.models.**NoteEffect**(*accentuatedNote=False*, *bend=None*, *ghostNote=False*, *grace=None*, *hammer=False*, *harmonic=None*, *heavyAccentuatedNote=False*, *leftHandFinger=<Fingering.open: -1>*, *letRing=False*, *palmMute=False*, *rightHandFinger=<Fingering.open: -1>*, *slides=NOTHING*, *staccato=False*, *tremoloPicking=None*, *trill=None*, *vibrato=False*)
Contains all effects which can be applied to one note.

class guitarpro.models.**NoteType**

class guitarpro.models.**Note**(*beat*, *value=0*, *velocity=95*, *string=0*, *effect=NOTHING*, *durationPercent=1.0*, *swapAccidentals=False*, *type=<NoteType.rest: 0>*)

Describes a single note.

class guitarpro.models.**Chord**(*length*, *sharp=None*, *root=None*, *type=None*, *extension=None*, *bass=None*, *tonality=None*, *add=None*, *name=''*, *fifth=None*, *ninth=None*, *eleventh=None*, *firstFret=None*, *strings=NOTHING*, *barres=NOTHING*, *omissions=NOTHING*, *fingerings=NOTHING*, *show=None*, *newFormat=None*)

A chord annotation for beats.

class guitarpro.models.**ChordType**

Type of the chord.

major = 0

Major chord.

seventh = 1

Dominant seventh chord.

majorSeventh = 2

Major seventh chord.

sixth = 3

Add sixth chord.

minor = 4

Minor chord.

minorSeventh = 5

Minor seventh chord.

minorMajor = 6

Minor major seventh chord.

minorSixth = 7

Minor add sixth chord.

suspendedSecond = 8

Suspended second chord.

suspendedFourth = 9

Suspended fourth chord.

seventhSuspendedSecond = 10

Seventh suspended second chord.

seventhSuspendedFourth = 11

Seventh suspended fourth chord.

diminished = 12

Diminished chord.

augmented = 13

Augmented chord.

power = 14

Power chord.

class guitarpro.models.**Barre**(*fret*, *start=0*, *end=0*)

A single barre.

Parameters

- **start** – first string from the bottom of the barre.
- **end** – last string on the top of the barre.

class guitarpro.models.**ChordAlteration**
Tonality of the chord.

perfect = 0
Perfect.

diminished = 1
Diminished.

augmented = 2
Augmented.

class guitarpro.models.**ChordExtension**
Extension type of the chord.

none = 0
No extension.

ninth = 1
Ninth chord.

eleventh = 2
Eleventh chord.

thirteenth = 3
Thirteenth chord.

class guitarpro.models.**PitchClass**(*just*, *accidental=None*, *value=None*, *intonation=None*)
A pitch class.

Constructor provides several overloads. Each overload provides keyword argument *intonation* that may be either "sharp" or "flat".

First of overloads is (tone, accidental):

Parameters

- **tone** – integer of whole-tone.
- **accidental** – flat (-1), none (0) or sharp (1).

```
>>> p = PitchClass(4, -1)
>>> p
PitchClass(just=4, accidental=-1, value=3, intonation='flat')
>>> print(p)
Eb
>>> p = PitchClass(4, -1, intonation='sharp')
>>> p
PitchClass(just=4, accidental=-1, value=3, intonation='sharp')
>>> print(p)
D#
```

Second, semitone number can be directly passed to constructor:

Parameters **semitone** – integer of semitone.

```
>>> p = PitchClass(3)
>>> print(p)
Eb
```

(continues on next page)

(continued from previous page)

```
>>> p = PitchClass(3, intonation='sharp')
>>> print(p)
D#
```

And last, but not least, note name:

Parameters **name** – string representing note.

```
>>> p = PitchClass('D#')
>>> print(p)
D#
```

class guitarpro.models.**BeatText**(*value=''*)
A text annotation for beats.

class guitarpro.models.**MixTableItem**(*value=0*, *duration=0*, *allTracks=False*)
A mix table item describes a mix parameter, e.g. volume or reverb.

class guitarpro.models.**MixTableChange**(*instrument=None*, *volume=None*, *balance=None*, *chorus=None*, *reverb=None*, *phaser=None*, *tremolo=None*, *tempoName=''*, *tempo=None*, *hideTempo=True*, *wah=None*, *useRSE=False*, *rse=NOTHING*)
A MixTableChange describes a change in mix parameters.

class guitarpro.models.**BendType**
All Bend presets.

none = 0
No Preset.

bend = 1
A simple bend.

bendRelease = 2
A bend and release afterwards.

bendReleaseBend = 3
A bend, then release and rebend.

prebend = 4
Prebend.

prebendRelease = 5
Prebend and then release.

dip = 6
Dip the bar down and then back up.

dive = 7
Dive the bar.

releaseUp = 8
Release the bar up.

invertedDip = 9
Dip the bar up and then back down.

return_ = 10
Return the bar.

releaseDown = 11
Release the bar down.

class guitarpro.models.**BendPoint**(*position=0*, *value=None*, *vibrato=False*)
A single point within the BendEffect.

getTime(*duration*)
Gets the exact time when the point need to be played (MIDI).

Parameters **duration** – the full duration of the effect.

class guitarpro.models.**BendEffect**(*type=<BendType.none: 0>*, *value=0*, *points=NOTHING*)
This effect is used to describe string bends and tremolo bars.

semitoneLength = 1
The note offset per bend point offset.

maxPosition = 12
The max position of the bend points (x axis)

maxValue = 12
The max value of the bend points (y axis)

CHAPTER 4

Guitar Pro File Format

4.1 Basic Guitar Pro 3—5 types

4.1.1 Byte

Values of type *Byte* are stored in 1 byte.

4.1.2 Signed byte

Values of type *Signed byte* are stored in 1 byte.

4.1.3 Bool

Values of type *Bool* are stored in 1 byte.

4.1.4 Short

Values of type *Short* are stored in 2 little-endian bytes.

4.1.5 Int

Values of type *Int* are stored in 4 little-endian bytes.

4.1.6 Float

Values of type *Float* are stored in 4 little-endian bytes.

4.1.7 Double

Values of type *Double* are stored in 8 little-endian bytes.

4.1.8 ByteSizeString

Values of type *ByteSizeString* are represented by length of the string (1 *Byte*) followed by characters encoded in an 8-bit charset.

4.1.9 IntSizeString

Values of type *IntSizeString* are represented by length of the string (1 *Int*) followed by characters encoded in an 8-bit charset.

4.1.10 IntByteSizeString

Values of type *IntByteSizeString* are represented by an *Int* holding length of the string increased by 1, a *Byte* holding length of the string and finally followed by characters encoded in an 8-bit charset.

4.2 Guitar Pro 3 format

class guitarpro.gp3.**GP3File**(*data*, *encoding*, *version=None*, *versionTuple=None*)

A reader for GuitarPro 3 files.

readSong()

Read the song.

A song consists of score information, triplet feel, tempo, song key, MIDI channels, measure and track count, measure headers, tracks, measures.

- Version: *ByteSizeString* of size 30.
- Score information. See *readInfo()*.
- Triplet feel: *Bool*. If value is true, then triplet feel is set to eigth.
- Tempo: *Int*.
- Key: *Int*. Key signature of the song.
- MIDI channels. See *readMidiChannels()*.
- Number of measures: *Int*.
- Number of tracks: *Int*.
- Measure headers. See *readMeasureHeaders()*.
- Tracks. See *readTracks()*.
- Measures. See *readMeasures()*.

readInfo(*song*)

Read score information.

Score information consists of sequence of *IntByteSizeStrings*:

- title

- subtitle
- artist
- album
- words
- copyright
- tabbed by
- instructions

The sequence if followed by notice. Notice starts with the number of notice lines stored in *Int*. Each line is encoded in *IntByteSizeString*.

readMidiChannels()
Read MIDI channels.

Guitar Pro format provides 64 channels (4 MIDI ports by 16 channels), the channels are stored in this order:

- port1/channel1
- port1/channel2
- ...
- port1/channel16
- port2/channel1
- ...
- port4/channel16

Each channel has the following form:

- Instrument: *Int*.
- Volume: *Byte*.
- Balance: *Byte*.
- Chorus: *Byte*.
- Reverb: *Byte*.
- Phaser: *Byte*.
- Tremolo: *Byte*.
- blank1: *Byte*.
- blank2: *Byte*.

readMeasureHeaders(*song*, *measureCount*)
Read measure headers.

The *measures* are written one after another, their number have been specified previously.

Parameters **measureCount** – number of measures to expect.

readMeasureHeader(*number*, *song*, *previous=None*)
Read measure header.

The first byte is the measure's flags. It lists the data given in the current measure.

- *0x01*: numerator of the key signature

- *0x02*: denominator of the key signature
- *0x04*: beginning of repeat
- *0x08*: end of repeat
- *0x10*: number of alternate ending
- *0x20*: presence of a marker
- *0x40*: tonality of the measure
- *0x80*: presence of a double bar

Each of these elements is present only if the corresponding bit is a 1.

The different elements are written (if they are present) from lowest to highest bit.

Exceptions are made for the double bar and the beginning of repeat whose sole presence is enough, complementary data is not necessary.

- Numerator of the key signature: *Byte*.
- Denominator of the key signature: *Byte*.
- End of repeat: *Byte*. Number of repeats until the previous beginning of repeat.
- Number of alternate ending: *Byte*.
- Marker: see `GP3File.readMarker()`.
- Tonality of the measure: 2 *Bytes*. These values encode a key signature change on the current piece. First byte is key signature root, second is key signature type.

readMarker(*header*)
Read marker.

The markers are written in two steps. First is written an integer equal to the marker's name length + 1, then a string containing the marker's name. Finally the marker's color is written.

readColor()
Read color.

Colors are used by `guitarpro.models.Marker` and `guitarpro.models.Track`. They consist of 3 consecutive bytes and one blank byte.

readTracks(*song*, *trackCount*, *channels*)
Read tracks.

The tracks are written one after another, their number having been specified previously in `GP3File.readSong()`.

Parameters **trackCount** – number of tracks to expect.

readTrack(*track*, *channels*)
Read track.

The first byte is the track's flags. It presides the track's attributes:

- *0x01*: drums track
- *0x02*: 12 stringed guitar track
- *0x04*: banjo track
- *0x08*: *blank*
- *0x10*: *blank*

- *0x20*: *blank*
- *0x40*: *blank*
- *0x80*: *blank*

Flags are followed by:

- Name: *ByteSizeString*. A 40 characters long string containing the track's name.
- **Number of strings: *Int*. An integer equal to the number** of strings of the track.
- Tuning of the strings: List of 7 *Ints*. The tuning of the strings is stored as a 7-integers table, the "Number of strings" first integers being really used. The strings are stored from the highest to the lowest.
- Port: *Int*. The number of the MIDI port used.
- Channel. See `GP3File.readChannel()`.
- Number of frets: *Int*. The number of frets of the instrument.
- Height of the capo: *Int*. The number of the fret on which a capo is set. If no capo is used, the value is 0.
- Track's color. The track's displayed color in Guitar Pro.

readChannel(*channels*)
Read MIDI channel.

MIDI channel in Guitar Pro is represented by two integers. First is zero-based number of channel, second is zero-based number of channel used for effects.

readMeasures(*song*)
Read measures.

Measures are written in the following order:

- measure 1/track 1
- measure 1/track 2
- ...
- measure 1/track m
- measure 2/track 1
- measure 2/track 2
- ...
- measure 2/track m
- ...
- measure n/track 1
- measure n/track 2
- ...
- measure n/track m

readMeasure(*measure*)
Read measure.

The measure is written as number of beats followed by sequence of beats.

readBeat(*start*, *voice*)
Read beat.

The first byte is the beat flags. It lists the data present in the current beat:

- *0x01*: dotted notes
- *0x02*: presence of a chord diagram
- *0x04*: presence of a text
- *0x08*: presence of effects
- *0x10*: presence of a mix table change event
- *0x20*: the beat is a n-tuplet
- *0x40*: status: True if the beat is empty of if it is a rest
- *0x80*: *blank*

Flags are followed by:

- Status: *Byte*. If flag at *0x40* is true, read one byte. If value of the byte is *0x00* then beat is empty, if value is *0x02* then the beat is rest.
- Beat duration: *Byte*. See `readDuration()`.
- Chord diagram. See `readChord()`.
- Text. See `readText()`.
- Beat effects. See `readBeatEffects()`.
- Mix table change effect. See `readMixTableChange()`.

getBeat(*voice*, *start*)
Get beat from measure by start time.

readDuration(*flags*)
Read beat duration.

Duration is composed of byte signifying duration and an integer that maps to `guitarpro.models.Tuplet`.

The byte maps to following values:

- *-2*: whole note
- *-1*: half note
- *0*: quarter note
- *1*: eighth note
- *2*: sixteenth note
- *3*: thirty-second note

If flag at *0x20* is true, the tuplet is read.

readChord(*stringCount*)
Read chord diagram.

First byte is chord header. If it's set to 0, then following chord is written in default (GP3) format. If chord header is set to 1, then chord diagram in encoded in more advanced (GP4) format.

readOldChord(*chord*)
Read chord diagram encoded in GP3 format.

Chord diagram is read as follows:

- Name: *IntByteSizeString*. Name of the chord, e.g. *Em*.
- First fret: *Int*. The fret from which the chord is displayed in chord editor.
- List of frets: 6 *Ints*. Frets are listed in order: fret on the string 1, fret on the string 2, ..., fret on the string 6. If string is untouched then the values of fret is *-1*.

readNewChord(*chord*)
Read new-style (GP4) chord diagram.

New-style chord diagram is read as follows:

- Sharp: *Bool*. If true, display all semitones as sharps, otherwise display as flats.
- Blank space, 3 *Bytes*.
- Root: *Int*. Values are:
  - -1 for customized chords
  - 0: C
  - 1: C#
  - ...
- Type: *Int*. Determines the chord type as followed. See `guitarpro.models.ChordType` for mapping.
- Chord extension: *Int*. See `guitarpro.models.ChordExtension` for mapping.
- Bass note: *Int*. Lowest note of chord as in *C/Am*.
- Tonality: *Int*. See `guitarpro.models.ChordAlteration` for mapping.
- Add: *Bool*. Determines if an "add" (added note) is present in the chord.
- Name: *ByteSizeString*. Max length is 22.
- Fifth alteration: *Int*. Maps to `guitarpro.models.ChordAlteration`.
- Ninth alteration: *Int*. Maps to `guitarpro.models.ChordAlteration`.
- Eleventh alteration: *Int*. Maps to `guitarpro.models.ChordAlteration`.
- List of frets: 6 *Ints*. Fret values are saved as in default format.
- Count of barres: *Int*. Maximum count is 2.
- Barre frets: 2 *Ints*.
- Barre start strings: 2 *Ints*.
- Barre end string: 2 *Ints*.
- Omissions: 7 *Bools*. If the value is true then note is played in chord.
- Blank space, 1 *Byte*.

readText()
Read beat text.

Text is stored in *IntByteSizeString*.

readBeatEffects(*effect*)
Read beat effects.

The first byte is effects flags:

- *0x01*: vibrato
- *0x02*: wide vibrato
- *0x04*: natural harmonic
- *0x08*: artificial harmonic
- *0x10*: fade in
- *0x20*: tremolo bar or slap effect
- *0x40*: beat stroke direction
- *0x80*: *blank*
- Tremolo bar or slap effect: *Byte*. If it's 0 then tremolo bar should be read (see *readTremoloBar()*). Else it's tapping and values of the byte map to:
  - *1*: tap
  - *2*: slap
  - *3*: pop
- Beat stroke direction. See *readBeatStroke()*.

readTremoloBar()
Read tremolo bar beat effect.

The only type of tremolo bar effect Guitar Pro 3 supports is *dip*. The value of the effect is encoded in *Int* and shows how deep tremolo bar is pressed.

readBeatStroke()
Read beat stroke.

Beat stroke consists of two *Bytes* which correspond to stroke up and stroke down speed. See *guitarpro.models.BeatStrokeDirection* for value mapping.

toStrokeValue(*value*)
Unpack stroke value.

Stroke value maps to:

- *1*: hundred twenty-eighth
- *2*: sixty-fourth
- *3*: thirty-second
- *4*: sixteenth
- *5*: eighth
- *6*: quarter

readMixTableChange(*measure*)
Read mix table change.

List of values is read first. See *readMixTableChangeValues()*.

List of values is followed by the list of durations for parameters that have changed. See *readMixTableChangeDurations()*.

readMixTableChangeValues(*tableChange*, *measure*)
Read mix table change values.

Mix table change values consist of 7 *SignedBytes* and an *Int*, which correspond to:

- instrument
- volume
- balance
- chorus
- reverb
- phaser
- tremolo
- tempo

If signed byte is *-1* then corresponding parameter hasn't changed.

readMixTableChangeDurations(*tableChange*)
Read mix table change durations.

Durations are read for each non-null `MixTableItem`. Durations are encoded in *Signed byte*.

readNotes(*track*, *beat*, *duration*, *effect=None*)
Read notes.

First byte lists played strings:

- *0x01*: 7th string
- *0x02*: 6th string
- *0x04*: 5th string
- *0x08*: 4th string
- *0x10*: 3th string
- *0x20*: 2th string
- *0x40*: 1th string
- *0x80*: *blank*

readNote(*note*, *guitarString*, *track*)
Read note.

The first byte is note flags:

- *0x01*: time-independent duration
- *0x02*: heavy accentuated note
- *0x04*: ghost note
- *0x08*: presence of note effects
- *0x10*: dynamics
- *0x20*: fret
- *0x40*: accentuated note
- *0x80*: right hand or left hand fingering

Flags are followed by:

- Note type: *Byte*. Note is normal if values is 1, tied if value is 2, dead if value is 3.
- Time-independent duration: 2 *SignedBytes*. Correspond to duration and tuplet. See `readDuration()` for reference.
- Note dynamics: *Signed byte*. See `unpackVelocity()`.
- Fret number: *Signed byte*. If flag at *0x20* is set then read fret number.
- Fingering: 2 *SignedBytes*. See `guitarpro.models.Fingering`.
- Note effects. See `readNoteEffects()`.

unpackVelocity(*dyn*)
Convert Guitar Pro dynamic value to raw MIDI velocity.

getTiedNoteValue(*stringIndex*, *track*)
Get note value of tied note.

readNoteEffects(*note*)
Read note effects.

First byte is note effects flags:

- *0x01*: bend presence
- *0x02*: hammer-on/pull-off
- *0x04*: slide
- *0x08*: let-ring
- *0x10*: grace note presence

Flags are followed by:

- Bend. See `readBend()`.
- Grace note. See `readGrace()`.

readBend()
Read bend.

Encoded as:

- Bend type: *Signed byte*. See `guitarpro.models.BendType`.
- Bend value: *Int*.
- Number of bend points: *Int*.
- List of points. Each point consists of:
  - Position: *Int*. Shows where point is set along *x*-axis.
  - Value: *Int*. Shows where point is set along *y*-axis.
  - Vibrato: *Bool*.

readGrace()
Read grace note effect.

- Fret: *Signed byte*. Number of fret.
- Dynamic: *Byte*. Dynamic of a grace note, as in `guitarpro.models.Note.velocity`.
- Transition: *Byte*. See `guitarpro.models.GraceEffectTransition`.

- Duration: *Byte*. Values are:
  - *1*: Thirty-second note.
  - *2*: Twenty-fourth note.
  - *3*: Sixteenth note.

4.3 Guitar Pro 4 format

class guitarpro.gp4.**GP4File**(*data*, *encoding*, *version=None*, *versionTuple=None*)

A reader for GuitarPro 4 files.

readSong()

Read the song.

A song consists of score information, triplet feel, lyrics, tempo, song key, MIDI channels, measure and track count, measure headers, tracks, measures.

- Version: *ByteSizeString* of size 30.
- Score information. See `readInfo()`.
- Triplet feel: *Bool*. If value is true, then triplet feel is set to eigth.
- Lyrics. See *readLyrics()*.
- Tempo: *Int*.
- Key: *Int*. Key signature of the song.
- Octave: *Signed byte*. Reserved for future uses.
- MIDI channels. See `readMidiChannels()`.
- Number of measures: *Int*.
- Number of tracks: *Int*.
- Measure headers. See `readMeasureHeaders()`.
- Tracks. See `readTracks()`.
- Measures. See `readMeasures()`.

readLyrics()

Read lyrics.

First, read an *Int* that points to the track lyrics are bound to. Then it is followed by 5 lyric lines. Each one constists of number of starting measure encoded in *Int* and *IntSizeString* holding text of the lyric line.

readNewChord(*chord*)

Read new-style (GP4) chord diagram.

New-style chord diagram is read as follows:

- Sharp: *Bool*. If true, display all semitones as sharps, otherwise display as flats.
- Blank space, 3 *Bytes*.
- Root: *Byte*. Values are:
  - -1 for customized chords
  - 0: C

  - 1: C#
  - ...

- Type: *Byte*. Determines the chord type as followed. See `guitarpro.models.ChordType` for mapping.
- Chord extension: *Byte*. See `guitarpro.models.ChordExtension` for mapping.
- Bass note: *Int*. Lowest note of chord as in *C/Am*.
- Tonality: *Int*. See `guitarpro.models.ChordAlteration` for mapping.
- Add: *Bool*. Determines if an "add" (added note) is present in the chord.
- Name: *ByteSizeString*. Max length is 22.
- Fifth tonality: *Byte*. Maps to `guitarpro.models.ChordExtension`.
- Ninth tonality: *Byte*. Maps to `guitarpro.models.ChordExtension`.
- Eleventh tonality: *Byte*. Maps to `guitarpro.models.ChordExtension`.
- List of frets: 6 *Ints*. Fret values are saved as in default format.
- Count of barres: *Byte*. Maximum count is 5.
- Barre frets: 5 *Bytes*.
- Barre start strings: 5 *Bytes*.
- Barre end string: 5 *Bytes*.
- Omissions: 7 *Bools*. If the value is true then note is played in chord.
- Blank space, 1 *Byte*.
- Fingering: 7 *SignedBytes*. For value mapping, see `guitarpro.models.Fingering`.

readBeatEffects(*effect*)

Read beat effects.

Beat effects are read using two byte flags.

The first byte of flags is:

- *0x01*: *blank*
- *0x02*: wide vibrato
- *0x04*: *blank*
- *0x08*: *blank*
- *0x10*: fade in
- *0x20*: slap effect
- *0x40*: beat stroke
- *0x80*: *blank*

The second byte of flags is:

- *0x01*: rasgueado
- *0x02*: pick stroke
- *0x04*: tremolo bar
- *0x08*: *blank*

- *0x10*: *blank*
- *0x20*: *blank*
- *0x40*: *blank*
- *0x80*: *blank*

Flags are followed by:

- Slap effect: *Signed byte*. For value mapping see *guitarpro.models.SlapEffect*.
- Tremolo bar. See *readTremoloBar()*.
- Beat stroke. See `readBeatStroke()`.
- Pick stroke: *Signed byte*. For value mapping see *guitarpro.models.BeatStrokeDirection*.

readTremoloBar()
Read tremolo bar beat effect.

The only type of tremolo bar effect Guitar Pro 3 supports is *dip*. The value of the effect is encoded in *Int* and shows how deep tremolo bar is pressed.

readMixTableChange(*measure*)
Read mix table change.

Mix table change in Guitar Pro 4 format extends Guitar Pro 3 format. It constists of *values*, *durations*, and, new to GP3, *flags*.

readMixTableChangeFlags(*tableChange*)
Read mix table change flags.

The meaning of flags:

- *0x01*: change volume for all tracks
- *0x02*: change balance for all tracks
- *0x04*: change chorus for all tracks
- *0x08*: change reverb for all tracks
- *0x10*: change phaser for all tracks
- *0x20*: change tremolo for all tracks

readNoteEffects(*note*)
Read note effects.

The effects presence for the current note is set by the 2 bytes of flags.

First set of flags:

- *0x01*: bend
- *0x02*: hammer-on/pull-off
- *0x04*: *blank*
- *0x08*: let-ring
- *0x10*: grace note
- *0x20*: *blank*
- *0x40*: *blank*

- *0x80*: *blank*

Second set of flags:

- *0x01*: staccato
- *0x02*: palm mute
- *0x04*: tremolo picking
- *0x08*: slide
- *0x10*: harmonic
- *0x20*: trill
- *0x40*: vibrato
- *0x80*: *blank*

Flags are followed by:

- Bend. See `readBend()`.
- Grace note. See `readGrace()`.
- Tremolo picking. See *`readTremoloPicking()`*.
- Slide. See *`readSlides()`*.
- Harmonic. See *`readHarmonic()`*.
- Trill. See *`readTrill()`*.

readTremoloPicking()
Read tremolo picking.

Tremolo constists of picking speed encoded in *Signed byte*. For value mapping refer to *`fromTremoloValue()`*.

fromTremoloValue(*value*)
Convert tremolo picking speed to actual duration.

Values are:

- *1*: eighth
- *2*: sixteenth
- *3*: thirtySecond

readSlides()
Read slides.

Slide is encoded in *Signed byte*. See *`guitarpro.models.SlideType`* for value mapping.

readHarmonic(*note*)
Read harmonic.

Harmonic is encoded in *Signed byte*. Values correspond to:

- *1*: natural harmonic
- *3*: tapped harmonic
- *4*: pinch harmonic
- *5*: semi-harmonic
- *15*: artificial harmonic on (*n* + *5*)th fret

- *17*: artificial harmonic on (*n* + *7*)th fret
- *22*: artificial harmonic on (*n* + *12*)th fret

readTrill()
Read trill.

- Fret: *Signed byte*.
- Period: *Signed byte*. See `fromTrillPeriod()`.

fromTrillPeriod(*period*)
Convert trill period to actual duration.

Values are:

- *1*: sixteenth
- *2*: thirty-second
- *3*: sixty-fourth

4.4 Guitar Pro 5 format

class guitarpro.gp5.**GP5File**(*data*, *encoding*, *version=None*, *versionTuple=None*)
A reader for GuitarPro 5 files.

readSong()
Read the song.

A song consists of score information, triplet feel, lyrics, tempo, song key, MIDI channels, measure and track count, measure headers, tracks, measures.

- Version: *ByteSizeString* of size 30.
- Score information. See `readInfo()`.
- Lyrics. See `readLyrics()`.
- RSE master effect. See `readRSEInstrument()`.
- Tempo name: *IntByteSizeString*.
- Tempo: *Int*.
- Hide tempo: *Bool*. Don't display tempo on the sheet if set.
- Key: *Int*. Key signature of the song.
- Octave: *Int*. Octave of the song.
- MIDI channels. See `readMidiChannels()`.
- Directions. See `readDirections()`.
- Master reverb. See `readMasterReverb()`.
- Number of measures: *Int*.
- Number of tracks: *Int*.
- Measure headers. See `readMeasureHeaders()`.
- Tracks. See `readTracks()`.
- Measures. See `readMeasures()`.

readInfo(*song*)
Read score information.

Score information consists of sequence of *IntByteSizeStrings*:

- title
- subtitle
- artist
- album
- words
- music
- copyright
- tabbed by
- instructions

The sequence if followed by notice. Notice starts with the number of notice lines stored in *Int*. Each line is encoded in *IntByteSizeString*.

readRSEMasterEffect()
Read RSE master effect.

Persistence of RSE master effect was introduced in Guitar Pro 5.1. It is read as:

- Master volume: *Int*. Values are in range from 0 to 200.
- 10-band equalizer. See `readEqualizer()`.

readEqualizer(*knobsNumber*)
Read equalizer values.

Equalizers are used in RSE master effect and Track RSE. They consist of *n* *SignedBytes* for each *n* bands and one *Signed byte* for gain (PRE) fader.

Volume values are stored as opposite to actual value. See `unpackVolumeValue()`.

unpackVolumeValue(*value*)
Unpack equalizer volume value.

Equalizer volumes are float but stored as *SignedBytes*.

readPageSetup()
Read page setup.

Page setup is read as follows:

- Page size: 2 *Ints*. Width and height of the page.
- Page padding: 4 *Ints*. Left, right, top, bottom padding of the page.
- Score size proportion: *Int*.
- Header and footer elements: *Short*. See `guitarpro.models.HeaderFooterElements` for value mapping.
- List of placeholders:
  - title
  - subtitle
  - artist

- album
- words
- music
- wordsAndMusic
- copyright1, e.g. *"Copyright %copyright%"*
- copyright2, e.g. *"All Rights Reserved - International Copyright Secured"*
- pageNumber

readDirections()
Read directions.

Directions is a list of 19 *ShortInts* each pointing at the number of measure.

Directions are read in the following order.

- Coda
- Double Coda
- Segno
- Segno Segno
- Fine
- Da Capo
- Da Capo al Coda
- Da Capo al Double Coda
- Da Capo al Fine
- Da Segno
- Da Segno al Coda
- Da Segno al Double Coda
- Da Segno al Fine
- Da Segno Segno
- Da Segno Segno al Coda
- Da Segno Segno al Double Coda
- Da Segno Segno al Fine
- Da Coda
- Da Double Coda

readMeasureHeaders(*song*, *measureCount*, *directions*)
Read measure headers.

The *measures* are written one after another, their number have been specified previously.

Parameters **measureCount** – number of measures to expect.

readMeasureHeader(*number*, *song*, *previous=None*)
Read measure header.

Measure header format in Guitar Pro 5 differs from one if Guitar Pro 3.

First, there is a blank byte if measure is not first. Then measure header is read as in GP3's `guitarpro.gp3.readMeasureHeader()`. Then measure header is read as follows:

- Time signature beams: 4 *Bytes*. Appears If time signature was set, i.e. flags *0x01* and *0x02* are both set.
- Blank *Byte* if flag at *0x10* is set.
- Triplet feel: *Byte*. See *guitarpro.models.TripletFeel*.

readTracks(*song*, *trackCount*, *channels*)
Read tracks.

Tracks in Guitar Pro 5 have almost the same format as in Guitar Pro 3. If it's Guitar Pro 5.0 then 2 blank bytes are read after `guitarpro.gp3.readTracks()`. If format version is higher than 5.0, 1 blank byte is read.

readTrack(*track*, *channels*)
Read track.

If it's Guitar Pro 5.0 format and track is first then one blank byte is read.

Then go track's flags. It presides the track's attributes:

- *0x01*: drums track
- *0x02*: 12 stringed guitar track
- *0x04*: banjo track
- *0x08*: track visibility
- *0x10*: track is soloed
- *0x20*: track is muted
- *0x40*: RSE is enabled
- *0x80*: show tuning in the header of the sheet.

Flags are followed by:

- Name: *String*. A 40 characters long string containing the track's name.
- Number of strings: *Int*. An integer equal to the number of strings of the track.
- Tuning of the strings: *Table of integers*. The tuning of the strings is stored as a 7-integers table, the "Number of strings" first integers being really used. The strings are stored from the highest to the lowest.
- Port: *Int*. The number of the MIDI port used.
- Channel. See `GP3File.readChannel()`.
- Number of frets: *Int*. The number of frets of the instrument.
- Height of the capo: *Int*. The number of the fret on which a capo is set. If no capo is used, the value is 0.
- Track's color. The track's displayed color in Guitar Pro.

The properties are followed by second set of flags stored in a *Short*.

- *0x0001*: show tablature
- *0x0002*: show standard notation
- *0x0004*: chord diagrams are below standard notation

- *0x0008*: show rhythm with tab
- *0x0010*: force horizontal beams
- *0x0020*: force channels 11 to 16
- *0x0040*: diagram list on top of the score
- *0x0080*: diagrams in the score
- *0x0200*: auto let-ring
- *0x0400*: auto brush
- *0x0800*: extend rhythmic inside the tab

Then follow:

- Auto accentuation: *Byte*. See `guitarpro.models.Accentuation`.
- MIDI bank: *Byte*.
- Track RSE. See `readTrackRSE()`.

readTrackRSE(*trackRSE*)
Read track RSE.

In GuitarPro 5.1 track RSE is read as follows:

- Humanize: *Byte*.
- Unknown space: 6 *Ints*.
- RSE instrument. See `readRSEInstrument()`.
- 3-band track equalizer. See `readEqualizer()`.
- RSE instrument effect. See `readRSEInstrumentEffect()`.

readRSEInstrument()
Read RSE instrument.

- MIDI instrument number: *Int*.
- Unknown *Int*.
- Sound bank: *Int*.
- Effect number: *Int*. Vestige of Guitar Pro 5.0 format.

readRSEInstrumentEffect(*rseInstrument*)
Read RSE instrument effect name.

This feature was introduced in Guitar Pro 5.1.

- Effect name: *IntByteSizeString*.
- Effect category: *IntByteSizeString*.

readMeasure(*measure*)
Read measure.

Guitar Pro 5 stores twice more measures compared to Guitar Pro 3. One measure consists of two sub-measures for each of two voices.

Sub-measures are followed by a `LineBreak` stored in *Byte*.

readBeat(*start*, *voice*)
Read beat.

First, beat is read is in Guitar Pro 3 `guitarpro.gp3.readBeat()`. Then it is followed by set of flags stored in *Short*.

- *0x0001*: break beams
- *0x0002*: direct beams down
- *0x0004*: force beams
- *0x0008*: direct beams up
- *0x0010*: ottava (8va)
- *0x0020*: ottava bassa (8vb)
- *0x0040*: quindicesima (15ma)
- *0x0100*: quindicesima bassa (15mb)
- *0x0200*: start tuplet bracket here
- *0x0400*: end tuplet bracket here
- *0x0800*: break secondary beams
- *0x1000*: break secondary tuplet
- *0x2000*: force tuplet bracket
- Break secondary beams: *Byte*. Appears if flag at *0x0800* is set. Signifies how much beams should be broken.

readBeatStroke()
Read beat stroke.

Beat stroke consists of two *Bytes* which correspond to stroke down and stroke up speed. See *guitarpro.models.BeatStroke* for value mapping.

readMixTableChange(*measure*)
Read mix table change.

Mix table change was modified to support RSE instruments. It is read as in Guitar Pro 3 and is followed by:

- Wah effect. See *readWahEffect()*.
- RSE instrument effect. See *readRSEInstrumentEffect()*.

readMixTableChangeValues(*tableChange*, *measure*)
Read mix table change values.

Mix table change values consist of:

- Instrument: *Signed byte*.
- RSE instrument. See *readRSEInstrument*.
- Volume: *Signed byte*.
- Balance: *Signed byte*.
- Chorus: *Signed byte*.
- Reverb: *Signed byte*.
- Phaser: *Signed byte*.

- Tremolo: *Signed byte*.
- Tempo name: *IntByteSizeString*.
- Tempo: *Int*.

If the value is -1 then corresponding parameter hasn't changed.

readMixTableChangeDurations (*tableChange*)
Read mix table change durations.

Durations are read for each non-null `MixTableItem`. Durations are encoded in *Signed byte*.

If tempo did change, then one *Bool* is read. If it's true, then tempo change won't be displayed on the score.

readMixTableChangeFlags (*tableChange*)
Read mix table change flags.

Mix table change flags are read as in Guitar Pro 4 `guitarpro.gp4.readMixTableChangeFlags()`, with one additional flag:

- *0x40*: use RSE
- *0x80*: show wah-wah

readWahEffect (*flags*)
Read wah-wah.

- Wah value: *Signed byte*. See `guitarpro.models.WahEffect` for value mapping.

readNote (*note*, *guitarString*, *track*)
Read note.

The first byte is note flags:

- *0x01*: duration percent
- *0x02*: heavy accentuated note
- *0x04*: ghost note
- *0x08*: presence of note effects
- *0x10*: dynamics
- *0x20*: fret
- *0x40*: accentuated note
- *0x80*: right hand or left hand fingering

Flags are followed by:

- Note type: *Byte*. Note is normal if values is 1, tied if value is 2, dead if value is 3.
- Note dynamics: *Signed byte*. See `unpackVelocity()`.
- Fret number: *Signed byte*. If flag at *0x20* is set then read fret number.
- Fingering: 2 *SignedBytes*. See `guitarpro.models.Fingering`.
- Duration percent: *Double*.
- Second set of flags: *Byte*.
  - *0x02*: swap accidentals.
- Note effects. See `guitarpro.gp4.readNoteEffects()`.

readGrace()
Read grace note effect.

- Fret: *Signed byte*. Number of fret.
- Dynamic: *Byte*. Dynamic of a grace note, as in `guitarpro.models.Note.velocity`.
- Transition: *Byte*. See *guitarpro.models.GraceEffectTransition*.
- Duration: *Byte*. Values are:
  - *1*: Thirty-second note.
  - *2*: Twenty-fourth note.
  - *3*: Sixteenth note.
- Flags: *Byte*.
  - *0x01*: grace note is muted (dead)
  - *0x02*: grace note is on beat

readSlides()
Read slides.

First *Byte* stores slide types:

- *0x01*: shift slide
- *0x02*: legato slide
- *0x04*: slide out downwards
- *0x08*: slide out upwards
- *0x10*: slide into from below
- *0x20*: slide into from above

readHarmonic(*note*)
Read harmonic.

First *Byte* is harmonic type:

- *1*: natural harmonic
- *2*: artificial harmonic
- *3*: tapped harmonic
- *4*: pinch harmonic
- *5*: semi-harmonic

In case harmonic types is artificial, following data is read:

- Note: *Byte*.
- Accidental: *Signed byte*.
- Octave: *Byte*.

If harmonic type is tapped:

- Fret: *Byte*.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

H

I

K

L

M

N

O

P

R

S

T

U

V

W