Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.midi.realtime

Previous topic

music21.midi.percussion

Next topic

music21.midi.translate

Table of Contents

- music21.midi.realtime
  - StreamPlayer
    - `StreamPlayer`
      - `StreamPlayer.getStringOrBytesIOFile()`
      - `StreamPlayer.play()`
      - `StreamPlayer.playStringIOFile()`
      - `StreamPlayer.stop()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Quick search

This Page

- Show Source

music21.midi.realtime¶

Objects for realtime playback of Music21 Streams as MIDI.

From an idea of Joe “Codeswell”:

https://joecodeswell.wordpress.com/2012/06/13/how-to-produce-python-controlled-audio-output-from-music-made-with-music21

https://stackoverflow.com/questions/10983462/how-can-i-produce-real-time-audio-output-from-music-made-with-music21

Requires pygame: http://www.pygame.org/download.shtml

StreamPlayer¶

class music21.midi.realtime.StreamPlayer(*streamIn: Stream*, *reinitMixer: bool = False*, *mixerFreq: int = 44100*, *mixerBitSize: int = -16*, *mixerChannels: int = 2*, *mixerBuffer: int = 1024*)¶

> Create a player for a stream that plays its midi version in realtime using pygame.
>
> Set up a detuned piano (where each key has a random but consistent detuning from 30 cents flat to sharp) and play a Bach Chorale on it in real time.

```
>>> import random
>>> keyDetune = []
>>> for i in range(127):
...    keyDetune.append(random.randint(-30, 30))
```

```
>>> b = corpus.parse('bwv66.6')
>>> for n in b.flatten().notes:
...    n.pitch.microtone = keyDetune[n.pitch.midi]
>>> sp = midi.realtime.StreamPlayer(b)
>>> sp.play()
```

> The stream is stored (unaltered) in *StreamPlayer.streamIn*, and can be changed any time the midi file is not playing.
>
> A number of mixer controls can be passed in with keywords:

- mixerFreq (default 44100 – CD quality)
- mixerBitSize (default -16 (=unsigned 16bit) –

  really, are you going to do 24bit audio with Python?? :-))
- mixerChannels (default 2 = stereo)

- mixerBuffer (default 1024 = number of samples)

`StreamPlayer` methods

StreamPlayer.getStringOrBytesIOFile()¶

StreamPlayer.play(*busyFunction=None*, *busyArgs=None*, *endFunction=None*, *endArgs=None*, *busyWaitMilliseconds=50*, ***, *playForMilliseconds=inf*, *blocked=True*)¶

> busyFunction is a function that is called with busyArgs when the music is busy every busyWaitMilliseconds.
>
> endFunction is a function that is called with endArgs when the music finishes playing.
>
> playForMilliseconds is the amount of time in milliseconds after which the playback will be automatically stopped.
>
> If blocked is False, the method will finish before ending the stream, allowing you to completely control whether to stop it. Ignore every other arguments

StreamPlayer.playStringIOFile(*stringIOFile*, *busyFunction=None*, *busyArgs=None*, *endFunction=None*, *endArgs=None*, *busyWaitMilliseconds=50*, ***, *playForMilliseconds=inf*, *blocked=True*)¶

> busyFunction is a function that is called with busyArgs when the music is busy every busyWaitMilliseconds.
>
> endFunction is a function that is called with endArgs when the music finishes playing.
>
> playForMilliseconds is the amount of time in milliseconds after which the playback will be automatically stopped.
>
> If blocked is False, the method will finish before ending the stream, allowing you to completely control whether to stop it. Ignore every other arguments but for stringIOFile

StreamPlayer.stop()¶

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.midi.realtime

© Copyright 2006-2023 Michael Scott Asato Cuthbert. Last updated on Jun 12, 2023.

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.musedata.translate

Previous topic

music21.musedata.base40

Next topic

music21.musicxml.archiveTools

Table of Contents

- music21.musedata.translate
  - Functions
    - `museDataWorkToStreamScore()`
    - `musedataPartToStreamPart()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.metadata.primitives

Next topic

music21.meter.base

Table of Contents

- music21.metadata.properties
  - PropertyDescription
    - `PropertyDescription`
      - `PropertyDescription.__eq__()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.volpiano

Next topic

Installation

Table of Contents

- music21.volume
  - Volume
    - `Volume`
      - `Volume.cachedRealized`
      - `Volume.cachedRealizedStr`
      - `Volume.realized`
      - `Volume.velocity`
      - `Volume.velocityScalar`
      - `Volume.getDynamicContext()`
      - `Volume.getRealized()`
      - `Volume.getRealizedStr()`
      - `Volume.mergeAttributes()`
  - Functions
    - `realizeVolume()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.musicxml.xmlSoundParser

Next topic

music21.note

Table of Contents

- music21.musicxml.xmlToM21
    - MeasureParser
        - `MeasureParser`
            - `MeasureParser.addToStaffReference()`
            - `MeasureParser.findM21VoiceFromXmlVoice()`
            - `MeasureParser.getStaffNumber()`
            - `MeasureParser.handleClef()`
            - `MeasureParser.handleFingering()`
            - `MeasureParser.handleKeySignature()`
            - `MeasureParser.handleMeasureStyle()`
            - `MeasureParser.handleStaffDetails()`
            - `MeasureParser.handleTimeSignature()`
            - `MeasureParser.insertCoreAndRef()`
            - `MeasureParser.insertInMeasureOrVoice()`
            - `MeasureParser.mxKeyOctaves()`
            - `MeasureParser.nonTraditionalKeySignature()`
            - `MeasureParser.parse()`
            - `MeasureParser.parseAttributesTag()`
            - `MeasureParser.parseMeasureAttributes()`
            - `MeasureParser.parseMeasureNumbers()`
            - `MeasureParser.setDirectionInDirectionType()`
            - `MeasureParser.setDynamicsDirection()`
            - `MeasureParser.setHarmonic()`
            - `MeasureParser.setSound()`
            - `MeasureParser.updateLyricsFromList()`
            - `MeasureParser.updateVoiceInformation()`
            - `MeasureParser.xmlBackup()`
            - `MeasureParser.xmlBarline()`
            - `MeasureParser.xmlDirection()`
            - `MeasureParser.xmlDirectionTypeToSpanners()`
            - `MeasureParser.xmlForward()`
            - `MeasureParser.xmlGraceToGrace()`
            - `MeasureParser.xmlHarmony()`
            - `MeasureParser.xmlNotations()`
            - `MeasureParser.xmlNotationsToSpanners()`
            - `MeasureParser.xmlNoteToGeneralNoteHelper()`
            - `MeasureParser.xmlNotehead()`

        - MeasureParser.xmlOneSpanner()
        - MeasureParser.xmlOrnamentToExpression()
        - MeasureParser.xmlPrint()
        - MeasureParser.xmlSound()
        - MeasureParser.xmlStaffLayoutFromStaffDetails()
        - MeasureParser.xmlTechnicalToArticulation()
        - MeasureParser.xmlToAccidental()
        - MeasureParser.xmlToArticulation()
        - MeasureParser.xmlToBarline()
        - MeasureParser.xmlToBeam()
        - MeasureParser.xmlToBeams()
        - MeasureParser.xmlToChord()
        - MeasureParser.xmlToChordSymbol()
        - MeasureParser.xmlToClef()
        - MeasureParser.xmlToDuration()
        - MeasureParser.xmlToKeySignature()
        - MeasureParser.xmlToLyric()
        - MeasureParser.xmlToNote()
        - MeasureParser.xmlToOffset()
        - MeasureParser.xmlToPitch()
        - MeasureParser.xmlToRehearsalMark()
        - MeasureParser.xmlToRepeat()
        - MeasureParser.xmlToRest()
        - MeasureParser.xmlToSimpleNote()
        - MeasureParser.xmlToTempoIndication()
        - MeasureParser.xmlToTextExpression()
        - MeasureParser.xmlToTie()
        - MeasureParser.xmlToTimeSignature()
        - MeasureParser.xmlToTremolo()
        - MeasureParser.xmlToTuplets()
        - MeasureParser.xmlToUnpitched()
        - MeasureParser.xmlTransposeToInterval()
- MusicXMLImporter
    - MusicXMLImporter
        - MusicXMLImporter.creatorToContributor()
        - MusicXMLImporter.identificationToMetadata()
        - MusicXMLImporter.isRecognizableMetadataKey()
        - MusicXMLImporter.parsePartList()
        - MusicXMLImporter.parseXMLText()
        - MusicXMLImporter.partGroups()
        - MusicXMLImporter.processEncoding()
        - MusicXMLImporter.readFile()
        - MusicXMLImporter.rightsToCopyright()
        - MusicXMLImporter.scoreFromFile()
        - MusicXMLImporter.styleFromXmlDefaults()
        - MusicXMLImporter.xmlAppearanceToStyle()
        - MusicXMLImporter.xmlCreditToTextBox()
        - MusicXMLImporter.xmlDefaultsToScoreLayout()
        - MusicXMLImporter.xmlMetadata()
        - MusicXMLImporter.xmlPartToPart()
        - MusicXMLImporter.xmlRootToScore()
- PartGroup
    - PartGroup
        - PartGroup.add()
- PartParser
    - PartParser
        - PartParser.adjustTimeAttributesFromMeasure()
        - PartParser.applyMultiMeasureRest()

        - `PartParser.getDefaultInstrument()`
        - `PartParser.parse()`
        - `PartParser.parseMeasures()`
        - `PartParser.parseXmlScorePart()`
        - `PartParser.reclassifyInstrumentFromName()`
        - `PartParser.removeEndForwardRest()`
        - `PartParser.separateOutPartStaves()`
        - `PartParser.setLastMeasureInfo()`
        - `PartParser.updateTransposition()`
        - `PartParser.xmlMeasureToMeasure()`
- XMLParserBase
    - `XMLParserBase`
        - `XMLParserBase.setColor()`
        - `XMLParserBase.setEditorial()`
        - `XMLParserBase.setFont()`
        - `XMLParserBase.setLineStyle()`
        - `XMLParserBase.setPlacement()`
        - `XMLParserBase.setPosition()`
        - `XMLParserBase.setPrintObject()`
        - `XMLParserBase.setPrintStyle()`
        - `XMLParserBase.setPrintStyleAlign()`
        - `XMLParserBase.setStyleAttributes()`
        - `XMLParserBase.setTextFormatting()`
        - `XMLParserBase.xmlPageLayoutToPageLayout()`
        - `XMLParserBase.xmlPrintToPageLayout()`
        - `XMLParserBase.xmlPrintToSystemLayout()`
        - `XMLParserBase.xmlStaffLayoutToStaffLayout()`
        - `XMLParserBase.xmlSystemLayoutToSystemLayout()`
- Functions
    - `musicXMLTypeToType()`
    - `strippedText()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.serial

Next topic

music21.sites

Table of Contents

- music21.sieve
  - CompressionSegment
    - `CompressionSegment`
  - PitchSieve
    - `PitchSieve`
      - `PitchSieve.getIntervalSequence()`
  - PrimeSegment
    - `PrimeSegment`
  - Residual
    - `Residual`
      - `Residual.__eq__()`
      - `Residual.copy()`
      - `Residual.period()`
      - `Residual.represent()`
      - `Residual.segment()`
      - `Residual.setSegmentFormat()`
      - `Residual.setZ()`
      - `Residual.setZRange()`
  - Sieve
    - `Sieve`
      - `Sieve.collect()`
      - `Sieve.compress()`
      - `Sieve.expand()`
      - `Sieve.period()`
      - `Sieve.represent()`
      - `Sieve.segment()`
      - `Sieve.setSegmentFormat()`
      - `Sieve.setZ()`
      - `Sieve.setZRange()`
  - Functions
    - `discreteBinaryPad()`
    - `eratosthenes()`
    - `rabinMiller()`
    - `unitNormEqual()`
    - `unitNormRange()`
    - `unitNormStep()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.style

Next topic

music21.tempo

Table of Contents

- music21.tablature
    - BassGuitarFretBoard
        - `BassGuitarFretBoard`
    - ChordWithFretBoard
        - `ChordWithFretBoard`
            - `ChordWithFretBoard.getFretNotesFromFigure()`
    - FirstFret
        - `FirstFret`
    - FretBoard
        - `FretBoard`
            - `FretBoard.fretNotesLowestFirst()`
            - `FretBoard.getFretNoteByString()`
            - `FretBoard.getPitches()`
    - FretNote
        - `FretNote`
    - GuitarFretBoard
        - `GuitarFretBoard`
    - MandolinFretBoard
        - `MandolinFretBoard`
    - UkeleleFretBoard
        - `UkeleleFretBoard`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.musicxml.partStaffExporter

Next topic

music21.musicxml.xmlObjects

Table of Contents

- music21.musicxml.test_m21ToXml
  - Functions
    - `stripInnerSpaces()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.tree.toStream

Next topic

music21.tree.verticality

Table of Contents

- music21.tree.trees
  - ElementTree
    - `ElementTree`
      - `ElementTree.endTime`
      - `ElementTree.source`
      - `ElementTree.__eq__()`
      - `ElementTree.__getitem__()`
      - `ElementTree.getNodeByIndex()`
      - `ElementTree.getPositionFromElementUnsafe()`
      - `ElementTree.highestPosition()`
      - `ElementTree.index()`
      - `ElementTree.insert()`
      - `ElementTree.iterNodes()`
      - `ElementTree.lowestPosition()`
      - `ElementTree.populateFromSortedList()`
  - OffsetTree
    - `OffsetTree`
      - `OffsetTree.__getitem__()`
      - `OffsetTree.allOffsets()`
      - `OffsetTree.allTimePoints()`
      - `OffsetTree.append()`
      - `OffsetTree.copy()`
      - `OffsetTree.elementEndTime()`
      - `OffsetTree.elementsOverlappingOffset()`
      - `OffsetTree.elementsStartingAt()`
      - `OffsetTree.elementsStoppingAt()`
      - `OffsetTree.getPositionFromElementUnsafe()`
      - `OffsetTree.getVerticalityAt()`
      - `OffsetTree.overlapTimePoints()`
      - `OffsetTree.removeElements()`
      - `OffsetTree.simultaneityDict()`

Table of Contents

- About *music21*
- User’s Guide

- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.romanText.translate

Next topic

music21.romanText.writeRoman

Table of Contents

- music21.romanText.tsvConverter
  - M21toTSV
    - `M21toTSV`
      - `M21toTSV.m21ToTsv()`
      - `M21toTSV.write()`
  - TabChord
    - `TabChord`
  - TabChordBase
    - `TabChordBase`
      - `TabChordBase.combinedChord`
      - `TabChordBase.populateFromRow()`
      - `TabChordBase.tabToM21()`
  - TabChordV2
    - `TabChordV2`
      - `TabChordV2.beat`
      - `TabChordV2.global_key`
      - `TabChordV2.local_key`
      - `TabChordV2.measure`
  - TsvHandler
    - `TsvHandler`
      - `TsvHandler.prepStream()`
      - `TsvHandler.toM21Stream()`
      - `TsvHandler.tsvToChords()`
  - Functions
    - `characterSwaps()`
    - `getForm()`
    - `getLocalKey()`
    - `getSecondaryKey()`
    - `handleAddedTones()`
    - `isMinor()`
    - `localKeyAsRn()`

Table of Contents

- About *music21*

- User’s Guide

- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.stream.tools

Next topic

music21.tablature

Table of Contents

- music21.style
  - BeamStyle
    - `BeamStyle`
  - BezierStyle
    - `BezierStyle`
  - Enclosure
    - `Enclosure`
  - LineStyle
    - `LineStyle`
  - NoteStyle
    - `NoteStyle`
      - `NoteStyle.accidentalStyle`
      - `NoteStyle.noteSize`
      - `NoteStyle.stemStyle`
  - StreamStyle
    - `StreamStyle`
  - Style
    - `Style`
      - `Style.absoluteY`
      - `Style.enclosure`
      - `Style.hideObjectOnPrint`
      - `Style.units`
  - StyleMixin
    - `StyleMixin`
      - `StyleMixin.hasEditorialInformation`
      - `StyleMixin.hasStyleInformation`
      - `StyleMixin.editorial`
      - `StyleMixin.style`
  - TextStyle
    - `TextStyle`
      - `TextStyle.alignHorizontal`
      - `TextStyle.alignVertical`
      - `TextStyle.fontFamily`
      - `TextStyle.fontSize`
      - `TextStyle.fontStyle`
      - `TextStyle.fontWeight`
      - `TextStyle.justify`

        - `TextStyle.letterSpacing`
    - TextStylePlacement
        - `TextStylePlacement`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.stream.base

Previous topic

music21.spanner

Next topic

music21.stream.core

Table of Contents

- music21.stream.base
  - Stream
    - `Stream`
      - `Stream.beat`
      - `Stream.beatDuration`
      - `Stream.beatStr`
      - `Stream.beatStrength`
      - `Stream.flat`
      - `Stream.highestOffset`
      - `Stream.highestTime`
      - `Stream.isGapless`
      - `Stream.lowestOffset`
      - `Stream.notes`
      - `Stream.notesAndRests`
      - `Stream.pitches`
      - `Stream.secondsMap`
      - `Stream.spanners`
      - `Stream.voices`
      - `Stream.atSoundingPitch`
      - `Stream.clef`
      - `Stream.duration`
      - `Stream.elements`
      - `Stream.finalBarline`
      - `Stream.keySignature`
      - `Stream.metadata`
      - `Stream.seconds`
      - `Stream.staffLines`
      - `Stream.timeSignature`
      - `Stream.__eq__()`
      - `Stream.__getitem__()`
      - `Stream.activateVariants()`
      - `Stream.addGroupForElements()`
      - `Stream.allPlayingWhileSounding()`
      - `Stream.analyze()`
      - `Stream.append()`
      - `Stream.attachIntervalsBetweenStreams()`
      - `Stream.attachMelodicIntervals()`

- Stream.augmentOrDiminish()
- Stream.beatAndMeasureFromOffset()
- Stream.chordify()
- Stream.clear()
- Stream.cloneEmpty()
- Stream.containerInHierarchy()
- Stream.elementOffset()
- Stream.expandRepeats()
- Stream.explode()
- Stream.extendDuration()
- Stream.extendTies()
- Stream.extractContext()
- Stream.findConsecutiveNotes()
- Stream.findGaps()
- Stream.first()
- Stream.flatten()
- Stream.flattenUnnecessaryVoices()
- Stream.getElementAfterElement()
- Stream.getElementAtOrBefore()
- Stream.getElementBeforeOffset()
- Stream.getElementById()
- Stream.getElementsByClass()
- Stream.getElementsByGroup()
- Stream.getElementsByOffset()
- Stream.getElementsNotOfClass()
- Stream.getInstrument()
- Stream.getInstruments()
- Stream.getOverlaps()
- Stream.getTimeSignatures()
- Stream.hasElement()
- Stream.hasElementOfClass()
- Stream.hasMeasures()
- Stream.hasPartLikeStreams()
- Stream.hasVoices()
- Stream.haveAccidentalsBeenMade()
- Stream.index()
- Stream.insert()
- Stream.insertAndShift()
- Stream.insertIntoNoteOrChord()
- Stream.invertDiatonic()
- Stream.isSequence()
- Stream.isTwelveTone()
- Stream.isWellFormedNotation()
- Stream.iter()
- Stream.last()
- Stream.lyrics()
- Stream.makeAccidentals()
- Stream.makeBeams()
- Stream.makeImmutable()
- Stream.makeMeasures()
- Stream.makeMutable()
- Stream.makeNotation()
- Stream.makeRests()
- Stream.makeTies()
- Stream.makeVoices()
- Stream.measure()
- Stream.measureOffsetMap()
- Stream.measures()

        - Stream.melodicIntervals()
        - Stream.mergeAttributes()
        - Stream.mergeElements()
        - Stream.metronomeMarkBoundaries()
        - Stream.offsetMap()
        - Stream.playingWhenAttacked()
        - Stream.plot()
        - Stream.pop()
        - Stream.quantize()
        - Stream.recurse()
        - Stream.recurseRepr()
        - Stream.remove()
        - Stream.removeByClass()
        - Stream.removeByNotOfClass()
        - Stream.repeatAppend()
        - Stream.repeatInsert()
        - Stream.replace()
        - Stream.scaleDurations()
        - Stream.scaleOffsets()
        - Stream.setDerivationMethod()
        - Stream.setElementOffset()
        - Stream.shiftElements()
        - Stream.show()
        - Stream.showVariantAsOssialikePart()
        - Stream.simultaneousAttacks()
        - Stream.sliceAtOffsets()
        - Stream.sliceByBeat()
        - Stream.sliceByGreatestDivisor()
        - Stream.sliceByQuarterLengths()
        - Stream.sort()
        - Stream.sorted()
        - Stream.splitAtDurations()
        - Stream.splitAtQuarterLength()
        - Stream.splitByClass()
        - Stream.storeAtEnd()
        - Stream.stripTies()
        - Stream.template()
        - Stream.toSoundingPitch()
        - Stream.toWrittenPitch()
        - Stream.transferOffsetToElements()
        - Stream.transpose()
        - Stream.voicesToParts()
        - Stream.write()
        - Stream.autoSort
        - Stream.definesExplicitPageBreaks
        - Stream.definesExplicitSystemBreaks
        - Stream.isFlat
        - Stream.isSorted
        - Stream.recursionType
- Measure
    - Measure
        - Measure.barDuration
        - Measure.leftBarline
        - Measure.rightBarline
        - Measure.barDurationProportion()
        - Measure.bestTimeSignature()
        - Measure.makeNotation()
        - Measure.measureNumberWithSuffix()

    - Measure.mergeAttributes()
    - Measure.padAsAnacrusis()
    - Measure.clefIsNew
    - Measure.keyIsNew
    - Measure.layoutWidth
    - Measure.number
    - Measure.numberSuffix
    - Measure.paddingLeft
    - Measure.paddingRight
    - Measure.showNumber
    - Measure.timeSignatureIsNew
- Part
  - Part
    - Part.partAbbreviation
    - Part.partName
    - Part.makeAccidentals()
    - Part.mergeAttributes()
- Score
  - Score
    - Score.parts
    - Score.expandRepeats()
    - Score.implode()
    - Score.makeNotation()
    - Score.measure()
    - Score.measureOffsetMap()
    - Score.measures()
    - Score.partsToVoices()
    - Score.sliceByGreatestDivisor()
- Opus
  - Opus
    - Opus.scores
    - Opus.getNumbers()
    - Opus.getScoreByNumber()
    - Opus.getScoreByTitle()
    - Opus.mergeScores()
    - Opus.show()
    - Opus.write()
- Voice
  - Voice
- SpannerStorage
  - SpannerStorage
    - SpannerStorage.coreSelfActiveSite()
    - SpannerStorage.coreStoreAtEnd()
    - SpannerStorage.replace()
- VariantStorage
  - VariantStorage
- OffsetMap
  - OffsetMap
- BestQuantizationMatch
  - BestQuantizationMatch
- PartStaff
  - PartStaff
- System
  - System

Table of Contents

- About *music21*

- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.noteworthy.binaryTranslate

Next topic

music21.omr.correctors

Table of Contents

- music21.noteworthy.translate
  - NoteworthyTranslator
    - `NoteworthyTranslator`
      - `NoteworthyTranslator.createBarlines()`
      - `NoteworthyTranslator.createClef()`
      - `NoteworthyTranslator.createDynamicVariance()`
      - `NoteworthyTranslator.createDynamics()`
      - `NoteworthyTranslator.createKey()`
      - `NoteworthyTranslator.createLyrics()`
      - `NoteworthyTranslator.createOtherRepetitions()`
      - `NoteworthyTranslator.createPart()`
      - `NoteworthyTranslator.createTimeSignature()`
      - `NoteworthyTranslator.getMultiplePitchesFromPositionInfo()`
      - `NoteworthyTranslator.getOnePitchFromPosition()`
      - `NoteworthyTranslator.getPitchFromPositionInfo()`
      - `NoteworthyTranslator.getStepAndOctaveFromPosition()`
      - `NoteworthyTranslator.parseFile()`
      - `NoteworthyTranslator.parseList()`
      - `NoteworthyTranslator.parseString()`
      - `NoteworthyTranslator.setDurationForObject()`
      - `NoteworthyTranslator.setTieFromPitchInfo()`
      - `NoteworthyTranslator.translateChord()`
      - `NoteworthyTranslator.translateNote()`
      - `NoteworthyTranslator.translateRest()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference

- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.sorting

Previous topic

music21.sites

Next topic

music21.spanner

Table of Contents

- music21.sorting
  - SortTuple
    - `SortTuple`
      - `SortTuple.__eq__()`
      - `SortTuple.add()`
      - `SortTuple.modify()`
      - `SortTuple.shortRepr()`
      - `SortTuple.sub()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.metadata

Previous topic

music21.mei.base

Next topic

music21.metadata.bundles

Table of Contents

- music21.metadata
  - AmbitusShort
    - `AmbitusShort`
  - Metadata
    - `Metadata`
      - `Metadata.bestTitle`
      - `Metadata.contributors`
      - `Metadata.software`
      - `Metadata.alternativeTitle`
      - `Metadata.composer`
      - `Metadata.composers`
      - `Metadata.copyright`
      - `Metadata.corpusFilePath`
      - `Metadata.date`
      - `Metadata.dateCreated`
      - `Metadata.fileFormat`
      - `Metadata.fileNumber`
      - `Metadata.filePath`
      - `Metadata.librettist`
      - `Metadata.librettists`
      - `Metadata.localeOfComposition`
      - `Metadata.lyricist`
      - `Metadata.lyricists`
      - `Metadata.movementName`
      - `Metadata.movementNumber`
      - `Metadata.number`
      - `Metadata.opusNumber`
      - `Metadata.title`
      - `Metadata.__getitem__()`
      - `Metadata.add()`
      - `Metadata.addContributor()`
      - `Metadata.addCustom()`
      - `Metadata.all()`
      - `Metadata.getContributorsByRole()`
      - `Metadata.getCustom()`
      - `Metadata.isContributorUniqueName()`
      - `Metadata.isStandardName()`

        - `Metadata.namespaceNameToUniqueName()`
        - `Metadata.search()`
        - `Metadata.setCustom()`
        - `Metadata.uniqueNameToNamespaceName()`
  - RichMetadata
    - `RichMetadata`
      - `RichMetadata.all()`
      - `RichMetadata.getSourcePath()`
      - `RichMetadata.merge()`
      - `RichMetadata.update()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Table of Contents

- music21.tree.fromStream
  - Functions
    - `asTimespans()`
    - `asTree()`
    - `listOfTreesByClass()`
    - `makeFastShallowTreeFromSortedStream()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Quick search

This Page

- Show Source

music21.tree.fromStream¶

Tools for creating timespans (fast, manipulable objects) from Streams

Functions¶

music21.tree.fromStream.asTimespans(*inputStream*, *, *flatten: str | bool = False*, *classList: Sequence[type[music21.base.Music21Object]] | None = None*) → TimespanTree¶

> Recurses through a score and constructs a `TimespanTree`. Use Stream.asTimespans() generally since that caches the TimespanTree.
>
> ```
> >>> score = corpus.parse('bwv66.6')
> >>> scoreTree = tree.fromStream.asTimespans(score, flatten=True,
> ...                                         classList=(note.Note, chord.Chord))
> >>> scoreTree
> <TimespanTree {165} (0.0 to 36.0) <music21.stream.Score ...>>
> >>> for x in scoreTree[:5]:
> ...     x
> ...
> <PitchedTimespan (0.0 to 0.5) <music21.note.Note C#>>
> <PitchedTimespan (0.0 to 0.5) <music21.note.Note A>>
> <PitchedTimespan (0.0 to 0.5) <music21.note.Note A>>
> <PitchedTimespan (0.0 to 1.0) <music21.note.Note E>>
> <PitchedTimespan (0.5 to 1.0) <music21.note.Note B>>
> ```
>
> ```
> >>> scoreTree = tree.fromStream.asTimespans(score, flatten=False, classList=())
> ```
>
> Each of these has 11 elements – mainly the Measures
>
> ```
> >>> for x in scoreTree:
> ...     x
> ...
> <ElementTimespan (0.0 to 0.0) <music21.metadata.Metadata object at 0x...>>
> <ElementTimespan (0.0 to 0.0) <music21.layout.StaffGroup ...>>
> <TimespanTree {11} (0.0 to 36.0) <music21.stream.Part Soprano>>
> <TimespanTree {11} (0.0 to 36.0) <music21.stream.Part Alto>>
> <TimespanTree {11} (0.0 to 36.0) <music21.stream.Part Tenor>>
> <TimespanTree {11} (0.0 to 36.0) <music21.stream.Part Bass>>
> ```
>
> ```
> >>> tenorElements = scoreTree[4]
> >>> tenorElements
> <TimespanTree {11} (0.0 to 36.0) <music21.stream.Part Tenor>>
> ```
>
> ```
> >>> tenorElements.source
> <music21.stream.Part Tenor>
> ```
>
> ```
> >>> tenorElements.source is score[3]
> True
> ```

music21.tree.fromStream.asTree(*inputStream: StreamType*, *, *flatten: Literal['semiFlat'] | bool = False*, *classList: Sequence[type] | None = None*, *useTimespans: bool = False*, *groupOffsets: bool = False*) → OffsetTree | ElementTree | TimespanTree¶

Converts a Stream and constructs an `ElementTree` based on this.

Use Stream.asTree() generally since that caches the ElementTree.

```
>>> score = tree.makeExampleScore()
>>> elementTree = tree.fromStream.asTree(score)
>>> elementTree
<ElementTree {2} (0.0 <0.-20...> to 8.0) <music21.stream.Score exampleScore>>
>>> for x in elementTree.iterNodes():
...     x
<ElementNode: Start:0.0 <0.-20...> Indices:(l:0 *0* r:1) Payload:<music21.stream.Part
<ElementNode: Start:0.0 <0.-20...> Indices:(l:0 *1* r:2) Payload:<music21.stream.Part
```

```
>>> etFlat = tree.fromStream.asTree(score, flatten=True)
>>> etFlat
<ElementTree {20} (0.0 <0.-25...> to 8.0) <music21.stream.Score exampleScore>>
```

The elementTree’s classSortOrder has changed to -25 to match the lowest positioned element in the score, which is an Instrument object (classSortOrder=-25)

```
>>> for x in etFlat.iterNodes():
...     x
<ElementNode: Start:0.0 <0.-25...> Indices:(l:0 *0* r:2)
    Payload:<music21.instrument.Instrument 'PartA: : '>>
<ElementNode: Start:0.0 <0.-25...> Indices:(l:1 *1* r:2)
    Payload:<music21.instrument.Instrument 'PartB: : '>>
<ElementNode: Start:0.0 <0.0...> Indices:(l:0 *2* r:4) Payload:<music21.clef.BassClef>
<ElementNode: Start:0.0 <0.0...> Indices:(l:3 *3* r:4) Payload:<music21.clef.BassClef>
...
<ElementNode: Start:0.0 <0.20...> Indices:(l:5 *6* r:8) Payload:<music21.note.Note C>>
<ElementNode: Start:0.0 <0.20...> Indices:(l:7 *7* r:8) Payload:<music21.note.Note C#>
<ElementNode: Start:1.0 <0.20...> Indices:(l:0 *8* r:20) Payload:<music21.note.Note D>
...
<ElementNode: Start:7.0 <0.20...> Indices:(l:15 *17* r:20) Payload:<music21.note.Note
<ElementNode: Start:End <0.-5...> Indices:(l:18 *18* r:20)
    Payload:<music21.bar.Barline type=final>>
<ElementNode: Start:End <0.-5...> Indices:(l:19 *19* r:20)
    Payload:<music21.bar.Barline type=final>>
```

```
>>> etFlat.getPositionAfter(0.5)
SortTuple(atEnd=0, offset=1.0, priority=0, classSortOrder=20, isNotGrace=1, insertInde
```

```
>>> etFlatNotes = tree.fromStream.asTree(score, flatten=True, classList=(note.Note,))
>>> etFlatNotes
<ElementTree {12} (0.0 <0.20...> to 8.0) <music21.stream.Score exampleScore>>
```

music21.tree.fromStream.listOfTreesByClass(*inputStream: StreamType*, *, *classLists: Sequence[Sequence[type[M21ObjType]]] = ()*, *currentParentage: tuple[stream.Stream, ...] | None = None*, *initialOffset: float = 0.0*, *flatten: bool | str = False*, *useTimespans: bool = False*) → list[trees.OffsetTree | timespanTree.TimespanTree]¶

To be DEPRECATED in v8: this is no faster than calling streamToTimespanTree multiple times with different classLists.

Recurses through *inputStream*, and constructs TimespanTrees for each encountered substream and PitchedTimespan for each encountered non-stream element.

classLists should be a sequence of elements contained in *classSet*. One TimespanTree will be constructed for each element in *classLists*, in a single optimized pass through the *inputStream*.

This is used internally by *streamToTimespanTree*.

```
>>> score = tree.makeExampleScore()
```

Get everything in the score

```
>>> treeList = tree.fromStream.listOfTreesByClass(score, useTimespans=True)
>>> treeList
[<TimespanTree {2} (-inf to inf) <music21.stream.Score ...>>]
>>> tl0 = treeList[0]
>>> for t in tl0:
...     print(t)
<TimespanTree {4} (-inf to inf) <music21.stream.Part ...>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 1 offset=0.0>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 2 offset=2.0>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 3 offset=4.0>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 4 offset=6.0>>
<TimespanTree {4} (-inf to inf) <music21.stream.Part ...>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 1 offset=0.0>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 2 offset=2.0>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 3 offset=4.0>>
    <TimespanTree {0} (-inf to inf) <music21.stream.Measure 4 offset=6.0>>
```

Now filter the Notes and the Clefs & TimeSignatures of the score (flattened) into a list of two TimespanTrees

```
>>> classLists = ((note.Note,), (clef.Clef, meter.TimeSignature))
>>> treeList = tree.fromStream.listOfTreesByClass(score, useTimespans=True,
...                                               classLists=classLists, flatten=True)
>>> treeList
[<TimespanTree {12} (0.0 to 8.0) <music21.stream.Score ...>>,
 <TimespanTree {4} (0.0 to 0.0) <music21.stream.Score ...>>]
```

- Changed in v8: it is now a stickler that classLists must be sequences of sequences,

  such as tuples of tuples.

music21.tree.fromStream.makeFastShallowTreeFromSortedStream(*inputStream: stream.Stream*, *, *outputTree: trees.OffsetTree* | *trees.ElementTree*, *classList: Sequence[type] | None = None*) → trees.OffsetTree | trees.ElementTree¶

Use populateFromSortedList to quickly make a tree from a stream.

This only works if the stream is flat (or we are not flattening) and sorts have already been run, and we are not making an OffsetTree.

Returns the same outputTree that was put in, only with elements in it.

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.tree.fromStream

© Copyright 2006-2023 Michael Scott Asato Cuthbert. Last updated on Jun 12, 2023.

Navigation

Previous topic

music21.musicxml.test_m21ToXml

Next topic

music21.musicxml.xmlSoundParser

Table of Contents

- music21.musicxml.xmlObjects
  - Functions
    - `booleanToYesNo()`
    - `fractionToPercent()`
    - `isValidXSDID()`
    - `yesNoToBoolean()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference**

Previous topic

User’s Guide, Chapter 61: TimespanTrees and Verticalities

Next topic

music21.abcFormat

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.meter.core

Next topic

music21.midi

Table of Contents

- music21.meter.tools
  - MeterTerminalTuple
    - `MeterTerminalTuple`
  - Functions
    - `divisionOptionsAdditiveMultiples()`
    - `divisionOptionsAdditiveMultiplesDownward()`
    - `divisionOptionsAdditiveMultiplesEvenDivision()`
    - `divisionOptionsAdditiveMultiplesUpward()`
    - `divisionOptionsFractionsDownward()`
    - `divisionOptionsFractionsUpward()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.tree

Next topic

music21.tree.core

Table of Contents

- music21.tree.analysis
  - Horizontality
    - `Horizontality`
      - `Horizontality.hasNeighborTone`
      - `Horizontality.hasNoMotion`
      - `Horizontality.hasPassingTone`
      - `Horizontality.__getitem__()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Quick search

This Page

- Show Source

music21.tree.analysis¶

Tools for performing voice-leading analysis with trees.

Horizontality¶

class music21.tree.analysis.Horizontality(*timespans=None*)¶

> A horizontality of consecutive PitchedTimespan objects.
>
> It must be initiated with a list or tuple of Timespan objects.

`Horizontality` read-only properties

Horizontality.hasNeighborTone¶

> Is true if the Horizontality contains a neighbor tone.

Horizontality.hasNoMotion¶

> Is true if the Horizontality contains no motion (including enharmonic restatements)

Horizontality.hasPassingTone¶

> Is true if the Horizontality contains a passing tone; currently defined as three tones in one direction.
>
> (TODO: better check)

`Horizontality` methods

Horizontality.__getitem__(*item*)¶

© Copyright 2006-2023 Michael Scott Asato Cuthbert. Last updated on Jun 12, 2023.

Navigation

Previous topic

music21.meter.base

Next topic

music21.meter.tools

Table of Contents

- music21.meter.core
  - MeterSequence
    - `MeterSequence`
      - `MeterSequence.denominator`
      - `MeterSequence.depth`
      - `MeterSequence.flat`
      - `MeterSequence.flatWeight`
      - `MeterSequence.numerator`
      - `MeterSequence.partitionDisplay`
      - `MeterSequence.partitionStr`
      - `MeterSequence.weight`
      - `MeterSequence.__getitem__()`
      - `MeterSequence.flatten()`
      - `MeterSequence.getLevel()`
      - `MeterSequence.getLevelList()`
      - `MeterSequence.getLevelSpan()`
      - `MeterSequence.getLevelWeight()`
      - `MeterSequence.getPartitionOptions()`
      - `MeterSequence.isUniformPartition()`
      - `MeterSequence.load()`
      - `MeterSequence.offsetToAddress()`
      - `MeterSequence.offsetToDepth()`
      - `MeterSequence.offsetToIndex()`
      - `MeterSequence.offsetToSpan()`
      - `MeterSequence.offsetToWeight()`
      - `MeterSequence.partition()`
      - `MeterSequence.partitionByCount()`
      - `MeterSequence.partitionByList()`
      - `MeterSequence.partitionByOtherMeterSequence()`
      - `MeterSequence.setLevelWeight()`
      - `MeterSequence.subdivideNestedHierarchy()`
      - `MeterSequence.subdividePartitionsEqual()`
  - MeterTerminal
    - `MeterTerminal`
      - `MeterTerminal.depth`
      - `MeterTerminal.denominator`
      - `MeterTerminal.duration`

- `MeterTerminal.numerator`
- `MeterTerminal.weight`
- `MeterTerminal.ratioEqual()`
- `MeterTerminal.subdivide()`
- `MeterTerminal.subdivideByCount()`
- `MeterTerminal.subdivideByList()`
- `MeterTerminal.subdivideByOther()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.tree.timespanTree

Next topic

music21.tree.trees

Table of Contents

- music21.tree.toStream
  - Functions
    - `chordified()`
    - `partwise()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.scale.intervalNetwork

Next topic

music21.search.base

Table of Contents

- music21.scale.scala
  - ScalaData
    - `ScalaData`
      - `ScalaData.getAdjacentCents()`
      - `ScalaData.getCentsAboveTonic()`
      - `ScalaData.getFileString()`
      - `ScalaData.getIntervalSequence()`
      - `ScalaData.parse()`
      - `ScalaData.setAdjacentCents()`
      - `ScalaData.setIntervalSequence()`
  - ScalaFile
    - `ScalaFile`
      - `ScalaFile.close()`
      - `ScalaFile.open()`
      - `ScalaFile.openFileLike()`
      - `ScalaFile.read()`
      - `ScalaFile.readstr()`
      - `ScalaFile.write()`
      - `ScalaFile.writestr()`
  - ScalaPitch
    - `ScalaPitch`
      - `ScalaPitch.parse()`
  - Functions
    - `getPaths()`
    - `parse()`
    - `search()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation

- Developer Reference
- Documentation and tests in progress

Quick search

This Page

- Show Source

music21.scale.scala¶

This module defines classes for representing Scala scale data, including Scala pitch representations, storage, and files.

The Scala format is defined at the following URL: https://www.huygens-fokker.org/scala/scl_format.html

We thank Manuel Op de Coul for allowing us to include the repository (as of May 11, 2011) with music21

Scala files are encoded as latin-1 (ISO-8859) text

Utility functions are also provided to search and find scales in the Scala scale archive. File names can be found with the `search()` function.

To create a `ScalaScale` instance, simply provide a root pitch and the name of the scale. Scale names are given as the scala .scl filename.

```
>>> mbiraScales = scale.scala.search('mbira')
>>> mbiraScales
['mbira_banda.scl', 'mbira_banda2.scl', 'mbira_gondo.scl', 'mbira_kunaka.scl',
 'mbira_kunaka2.scl', 'mbira_mude.scl', 'mbira_mujuru.scl', 'mbira_zimb.scl']
```

For most people you’ll want to do something like this:

```
>>> sc = scale.ScalaScale('a4', 'mbira_banda.scl')
>>> [str(p) for p in sc.pitches]
['A4', 'B4(-15c)', 'C#5(-11c)', 'E-5(-7c)', 'E~5(+6c)', 'F#5(+14c)', 'G~5(+1c)', 'B-5(+2c)'
```

ScalaData¶

class music21.scale.scala.ScalaData(*sourceString=None*, *fileName=None*)¶

Object representation of data stored in a Scala scale file. This object is used to access Scala information stored in a file. To create a music21 scale with a Scala file, use `ScalaScale`.

This is not called ScalaScale, as this name clashes with the `ScalaScale` that uses this object.

```
>>> import os
>>> sf = scale.scala.ScalaFile()
>>> fp = common.getSourceFilePath() / 'scale' / 'scala' / 'scl' / 'tanaka.scl'
>>> sf.open(fp)
>>> sd = sf.read()
```

```
>>> print(sd.description)  # converted to unicode...
26-note choice system of Shohé Tanaka, Studien i.G.d. reinen Stimmung (1890)
>>> sd.pitchCount
26
```

Distances from the tonic:

```
>>> cat = sd.getCentsAboveTonic()
>>> len(cat)
26
>>> list(int(round(x)) for x in cat[0:4])
[71, 92, 112, 182]
>>> sd.pitchValues[0]
<music21.scale.scala.ScalaPitch object at 0x10b16fac8>
>>> sd.pitchValues[0].cents
70.6724...
```

This will not add up with centsAboveTonic above, due to rounding

```
>>> adj = sd.getAdjacentCents()
>>> list(int(round(x)) for x in adj[0:4])
[71, 22, 20, 71]
```

Interval Sequences

```
>>> intSeq = sd.getIntervalSequence()
>>> intSeq[0:4]
[<music21.interval.Interval m2 (-29c)>,
 <music21.interval.Interval P1 (+22c)>,
 <music21.interval.Interval P1 (+20c)>,
 <music21.interval.Interval m2 (-29c)>]
```

Tweak the file and be ready to write it back out:

```
>>> sd.pitchValues[0].cents = 73.25
>>> sd.fileName = 'tanaka2.scl'
>>> sd.description = 'Tweaked version of tanaka.scl'
>>> fs = sd.getFileString()
>>> print(fs)
! tanaka2.scl
!
Tweaked version of tanaka.scl
26
!
73.25
92.17...
111.73...
182.40...
```

Be sure to reencode *fs* as *latin-1* before writing to disk.

```
>>> sf.close()
```

ScalaData methods

ScalaData.getAdjacentCents()¶

Get cents values between adjacent intervals.

ScalaData.getCentsAboveTonic()¶

Return a list of cent values above the implied tonic.

ScalaData.getFileString()¶

Return a unicode-string suitable for writing a Scala file

The unicode string should be encoded in Latin-1 for maximum Scala compatibility.

ScalaData.getIntervalSequence()¶

Get the scale as a list of Interval objects.

ScalaData.parse()¶

Parse a scala file delivered as a long string with line breaks

ScalaData.setAdjacentCents(*centList*)¶

Given a list of adjacent cent values, create the necessary ScalaPitch objects and update them

ScalaData.setIntervalSequence(*iList*)¶

Set the scale from a list of Interval objects.

ScalaFile¶

class music21.scale.scala.ScalaFile(*data=None*)¶

Interface for reading and writing scala files. On reading, returns a `ScalaData` object.

```
>>> import os
>>> sf = scale.scala.ScalaFile()
>>> fp = common.getSourceFilePath() / 'scale' / 'scala' / 'scl' / 'tanaka.scl'
>>> sf.open(fp)
>>> sd = sf.read()
>>> sd
<music21.scale.scala.ScalaData object at 0x10b170e10>
>>> sd is sf.data
True
>>> sf.fileName.endswith('tanaka.scl')
True
>>> sd.pitchCount
26
>>> sf.close()
```

ScalaFile methods

ScalaFile.close()¶

ScalaFile.open(*fp*, *mode='r'*)¶

Open a file for reading

ScalaFile.openFileLike(*fileLike*)¶

Assign a file-like object, such as those provided by StringIO, as an open file object.

ScalaFile.read()¶

Read a file. Note that this calls readstr, which processes all tokens.

If *number* is given, a work number will be extracted if possible.

ScalaFile.readstr(*strSrc*)¶

Read a string and process all Tokens. Returns a ABCHandler instance.

ScalaFile.write()¶

ScalaFile.writestr()¶

ScalaPitch¶

class music21.scale.scala.ScalaPitch(*sourceString=None*)¶

Representation of a scala pitch notation

```
>>> sp = scale.scala.ScalaPitch(' 1066.667 cents')
>>> print(sp.parse())
1066.667
```

```
>>> sp = scale.scala.ScalaPitch(' 2/1')
>>> sp.parse()
1200.0
>>> sp.parse('100.0 C#')
100.0
>>> [sp.parse(x) for x in ['89/84', '55/49', '44/37', '63/50', '4/3', '99/70', '442/2
...      '27/17', '37/22', '98/55', '15/8', '2/1']]
[100.0992..., 199.9798..., 299.9739..., 400.10848..., 498.04499...,
 600.0883..., 699.9976..., 800.9095..., 900.0260...,
 1000.0201..., 1088.2687..., 1200.0]
```

`ScalaPitch` methods

ScalaPitch.parse(*sourceString=None*)¶

Parse the source string and set self.cents.

Functions¶

music21.scale.scala.getPaths()¶

Get all scala scale paths. This is called once or the module and cached as SCALA_PATHS, which should be used instead of calls to this function.

```
>>> a = scale.scala.getPaths()
>>> len(a) >= 3800
True
```

music21.scale.scala.parse(*target*)¶

Get a `ScalaData` object from the bundled SCL archive or a file path.

```
>>> ss = scale.scala.parse('balafon6')
>>> ss.description
```

```
'Observed balafon tuning from Burma, Helmholtz/Ellis p. 518, nr.84'
>>> [str(i) for i in ss.getIntervalSequence()]
['<music21.interval.Interval m2 (+14c)>', '<music21.interval.Interval M2 (+36c)>',
'<music21.interval.Interval M2>', '<music21.interval.Interval m2 (+37c)>',
'<music21.interval.Interval M2 (-49c)>', '<music21.interval.Interval M2 (-6c)>',
'<music21.interval.Interval M2 (-36c)>']
```

```
>>> scale.scala.parse('incorrectFileName.scl') is None
True
```

```
>>> ss = scale.scala.parse('barbourChrom1')
>>> print(ss.description)
Barbour's #1 Chromatic
>>> ss.fileName
'barbour_chrom1.scl'
```

```
>>> ss = scale.scala.parse('blackj_gws.scl')
>>> ss.description
'Detempered Blackjack in 1/4 kleismic marvel tuning'
```

music21.scale.scala.search(*target*)¶

Search the scala archive for matches based on a string

```
>>> mbiraScales = scale.scala.search('mbira')
>>> mbiraScales
['mbira_banda.scl', 'mbira_banda2.scl', 'mbira_gondo.scl', 'mbira_kunaka.scl',
 'mbira_kunaka2.scl', 'mbira_mude.scl', 'mbira_mujuru.scl', 'mbira_zimb.scl']
```

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.scale.scala

© Copyright 2006-2023 Michael Scott Asato Cuthbert. Last updated on Jun 12, 2023.

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.midi.translate

Previous topic

music21.midi.realtime

Next topic

music21.musedata

Table of Contents

- music21.midi.translate
  - Functions
    - `streamToMidiFile()`
    - `midiFileToStream()`
    - `assignPacketsToChannels()`
    - `channelInstrumentData()`
    - `chordToMidiEvents()`
    - `conductorStream()`
    - `durationToMidiTicks()`
    - `elementToMidiEventList()`
    - `filterPacketsByTrackId()`
    - `getEndEvents()`
    - `getMetaEvents()`
    - `getNotesFromEvents()`
    - `getPacketFromMidiEvent()`
    - `getStartEvents()`
    - `getTimeForEvents()`
    - `insertConductorEvents()`
    - `instrumentToMidiEvents()`
    - `keySignatureToMidiEvents()`
    - `midiAsciiStringToBinaryString()`
    - `midiEventsToChord()`
    - `midiEventsToInstrument()`
    - `midiEventsToKey()`
    - `midiEventsToNote()`
    - `midiEventsToTempo()`
    - `midiEventsToTimeSignature()`
    - `midiFilePathToStream()`
    - `midiStringToStream()`
    - `midiTrackToStream()`
    - `midiTracksToStreams()`
    - `music21ObjectToMidiFile()`
    - `noteToMidiEvents()`
    - `offsetToMidiTicks()`
    - `packetStorageFromSubstreamList()`
    - `packetsToDeltaSeparatedEvents()`
    - `packetsToMidiTrack()`

    - `prepareStreamForMidi()`
    - `streamHierarchyToMidiTracks()`
    - `streamToPackets()`
    - `tempoToMidiEvents()`
    - `ticksToDuration()`
    - `timeSignatureToMidiEvents()`
    - `updatePacketStorageWithChannelInfo()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.search.segment

Next topic

music21.serial

Table of Contents

- music21.search.serial
  - ContiguousSegmentOfNotes
    - `ContiguousSegmentOfNotes`
      - `ContiguousSegmentOfNotes.originalCenteredTransformationsFromMatched`
      - `ContiguousSegmentOfNotes.startMeasureNumber`
      - `ContiguousSegmentOfNotes.startOffset`
      - `ContiguousSegmentOfNotes.zeroCenteredTransformationsFromMatched`
      - `ContiguousSegmentOfNotes.getActiveMatchedRows()`
      - `ContiguousSegmentOfNotes.getDistinctPitchClasses()`
      - `ContiguousSegmentOfNotes.readPitchClassesFromBottom()`
      - `ContiguousSegmentOfNotes.activeSegment`
      - `ContiguousSegmentOfNotes.containerStream`
      - `ContiguousSegmentOfNotes.matchedSegment`
      - `ContiguousSegmentOfNotes.partNumber`
      - `ContiguousSegmentOfNotes.segment`
  - ContiguousSegmentSearcher
    - `ContiguousSegmentSearcher`
      - `ContiguousSegmentSearcher.addActiveChords()`
      - `ContiguousSegmentSearcher.byLength()`
      - `ContiguousSegmentSearcher.getSearchBoundMethod()`
      - `ContiguousSegmentSearcher.searchIgnoreAllExclude()`
      - `ContiguousSegmentSearcher.searchIgnoreAllInclude()`
      - `ContiguousSegmentSearcher.searchIncludeAllExclude()`
      - `ContiguousSegmentSearcher.searchIncludeAllInclude()`
      - `ContiguousSegmentSearcher.searchRowsOnlyExclude()`
      - `ContiguousSegmentSearcher.searchRowsOnlyInclude()`
      - `ContiguousSegmentSearcher.searchSkipConsecutiveExclude()`
      - `ContiguousSegmentSearcher.searchSkipConsecutiveInclude()`
  - MultisetSegmentMatcher
    - `MultisetSegmentMatcher`
      - `MultisetSegmentMatcher.equalSubset()`
  - SegmentMatcher
    - `SegmentMatcher`
      - `SegmentMatcher.includeChords`
      - `SegmentMatcher.reps`
      - `SegmentMatcher.checkSearchedAlready()`
      - `SegmentMatcher.equalSubset()`

        - SegmentMatcher.find()
        - SegmentMatcher.findOneIgnoreAll()
        - SegmentMatcher.findOneOtherReps()
        - SegmentMatcher.getContiguousSegmentsByLength()
        - SegmentMatcher.normalize()
  - TransformedSegmentMatcher
    - TransformedSegmentMatcher
      - TransformedSegmentMatcher.checkSearchedAlready()
      - TransformedSegmentMatcher.equalSubset()
      - TransformedSegmentMatcher.getTransformations()
      - TransformedSegmentMatcher.normalize()
  - TransposedInvertedMultisetMatcher
    - TransposedInvertedMultisetMatcher
      - TransposedInvertedMultisetMatcher.checkSearchedAlready()
      - TransposedInvertedMultisetMatcher.equalSubset()
  - TransposedMultisetMatcher
    - TransposedMultisetMatcher
      - TransposedMultisetMatcher.checkSearchedAlready()
      - TransposedMultisetMatcher.equalSubset()
  - TransposedSegmentMatcher
    - TransposedSegmentMatcher
      - TransposedSegmentMatcher.normalize()
  - Functions
    - labelMultisets()
    - labelSegments()
    - labelTransformedSegments()
    - labelTransposedAndInvertedMultisets()
    - labelTransposedMultisets()
    - labelTransposedSegments()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.romanText.tsvConverter

Next topic

music21.scale

Table of Contents

- music21.romanText.writeRoman
  - RnWriter
    - `RnWriter`
      - `RnWriter.getChordString()`
      - `RnWriter.prepSequentialListOfLines()`
      - `RnWriter.prepTitle()`
  - Functions
    - `intBeat()`
    - `rnString()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.scale.scala

Next topic

music21.search.lyrics

Table of Contents

- music21.search.base
  - SearchMatch
    - `SearchMatch`
      - `SearchMatch.elStart`
      - `SearchMatch.els`
      - `SearchMatch.index`
      - `SearchMatch.iterator`
  - StreamSearcher
    - `StreamSearcher`
      - `StreamSearcher.noteNameAlgorithm()`
      - `StreamSearcher.rhythmAlgorithm()`
      - `StreamSearcher.run()`
      - `StreamSearcher.wildcardAlgorithm()`
  - Wildcard
    - `Wildcard`
  - WildcardDuration
    - `WildcardDuration`
  - Functions
    - `approximateNoteSearch()`
    - `approximateNoteSearchNoRhythm()`
    - `approximateNoteSearchOnlyRhythm()`
    - `approximateNoteSearchWeighted()`
    - `mostCommonMeasureRhythms()`
    - `noteNameRhythmicSearch()`
    - `noteNameSearch()`
    - `rhythmicSearch()`
    - `streamSearchBase()`
    - `translateDiatonicStreamToString()`
    - `translateDurationToBytes()`
    - `translateIntervalsAndSpeed()`
    - `translateNoteTieToByte()`
    - `translateNoteToByte()`
    - `translateNoteWithDurationToBytes()`
    - `translateStreamToString()`
    - `translateStreamToStringNoRhythm()`
    - `translateStreamToStringOnlyRhythm()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.tree

Previous topic

music21.tinyNotation

Next topic

music21.tree.analysis

Table of Contents

- music21.tree
  - Functions
    - `makeExampleScore()`

Table of Contents

- About *music21*

- User’s Guide

- **Module Reference**

- Installation

- Developer Reference

- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.musicxml.m21ToXml

Previous topic

music21.musicxml.lilypondTestSuite

Next topic

music21.musicxml.partStaffExporter

Table of Contents

- music21.musicxml.m21ToXml
  - GeneralObjectExporter
    - `GeneralObjectExporter`
      - `GeneralObjectExporter.fromDiatonicScale()`
      - `GeneralObjectExporter.fromDuration()`
      - `GeneralObjectExporter.fromDynamic()`
      - `GeneralObjectExporter.fromGeneralNote()`
      - `GeneralObjectExporter.fromGeneralObject()`
      - `GeneralObjectExporter.fromMeasure()`
      - `GeneralObjectExporter.fromMusic21Object()`
      - `GeneralObjectExporter.fromPart()`
      - `GeneralObjectExporter.fromPitch()`
      - `GeneralObjectExporter.fromScale()`
      - `GeneralObjectExporter.fromScore()`
      - `GeneralObjectExporter.fromStream()`
      - `GeneralObjectExporter.fromVoice()`
      - `GeneralObjectExporter.parse()`
      - `GeneralObjectExporter.parseWellformedObject()`
  - MeasureExporter
    - `MeasureExporter`
      - `MeasureExporter.appendArpeggioMarkSpannersToNotations()`
      - `MeasureExporter.arpeggioMarkToMxExpression()`
      - `MeasureExporter.articulationToXmlArticulation()`
      - `MeasureExporter.articulationToXmlTechnical()`
      - `MeasureExporter.barlineToXml()`
      - `MeasureExporter.beamToXml()`
      - `MeasureExporter.beamsToXml()`
      - `MeasureExporter.chordSymbolToXml()`
      - `MeasureExporter.chordToXml()`
      - `MeasureExporter.chordWithFretBoardToXml()`
      - `MeasureExporter.clefToXml()`
      - `MeasureExporter.codaToXml()`
      - `MeasureExporter.dealWithNotehead()`
      - `MeasureExporter.durationXml()`
      - `MeasureExporter.dynamicToXml()`
      - `MeasureExporter.expressionToXml()`
      - `MeasureExporter.fretBoardToXml()`

      - MeasureExporter.fretNoteToXml()
      - MeasureExporter.getArpeggioNumber()
      - MeasureExporter.intervalToXmlTranspose()
      - MeasureExporter.keySignatureToXml()
      - MeasureExporter.lyricToXml()
      - MeasureExporter.mainElementsParse()
      - MeasureExporter.measureStyle()
      - MeasureExporter.moveBackward()
      - MeasureExporter.moveForward()
      - MeasureExporter.noChordToXml()
      - MeasureExporter.noteToNotations()
      - MeasureExporter.noteToXml()
      - MeasureExporter.noteheadToXml()
      - MeasureExporter.objectAttachedSpannersToNotations()
      - MeasureExporter.objectAttachedSpannersToTechnicals()
      - MeasureExporter.ornamentToMxAccidentalMarks()
      - MeasureExporter.parse()
      - MeasureExporter.parseFlatElements()
      - MeasureExporter.parseOneElement()
      - MeasureExporter.pitchToXml()
      - MeasureExporter.placeInDirection()
      - MeasureExporter.rehearsalMarkToXml()
      - MeasureExporter.relatedSpanners()
      - MeasureExporter.repeatToXml()
      - MeasureExporter.restToXml()
      - MeasureExporter.romanNumeralToXml()
      - MeasureExporter.segnoToXml()
      - MeasureExporter.setBarline()
      - MeasureExporter.setHarmonic()
      - MeasureExporter.setLeftBarline()
      - MeasureExporter.setLineStyle()
      - MeasureExporter.setMxAttributes()
      - MeasureExporter.setMxAttributesObjectForStartOfMeasure()
      - MeasureExporter.setMxPrint()
      - MeasureExporter.setNoteInstrument()
      - MeasureExporter.setOffsetOptional()
      - MeasureExporter.setRbSpanners()
      - MeasureExporter.setRightBarline()
      - MeasureExporter.setTranspose()
      - MeasureExporter.staffLayoutToXmlPrint()
      - MeasureExporter.staffLayoutToXmlStaffDetails()
      - MeasureExporter.tempoIndicationToXml()
      - MeasureExporter.textExpressionToXml()
      - MeasureExporter.tieToXmlTie()
      - MeasureExporter.tieToXmlTied()
      - MeasureExporter.timeSignatureToXml()
      - MeasureExporter.tupletToTimeModification()
      - MeasureExporter.tupletToXmlTuplet()
      - MeasureExporter.unpitchedToXml()
      - MeasureExporter.wrapObjectInAttributes()
  - PartExporter
    - PartExporter
      - PartExporter.fixupNotationFlat()
      - PartExporter.fixupNotationMeasured()
      - PartExporter.getXmlScorePart()
      - PartExporter.instrumentSetup()
      - PartExporter.instrumentToXmlMidiInstrument()
      - PartExporter.instrumentToXmlScoreInstrument()

        - PartExporter.mergeInstrumentStreamPartStaffAware()
        - PartExporter.parse()
        - PartExporter.previousPartStaffInGroup
- ScoreExporter
    - ScoreExporter
        - ScoreExporter.addStyleToXmlDefaults()
        - ScoreExporter.contributorToXmlCreator()
        - ScoreExporter.emptyObject()
        - ScoreExporter.getSupports()
        - ScoreExporter.metadataToMiscellaneous()
        - ScoreExporter.parse()
        - ScoreExporter.parseFlatScore()
        - ScoreExporter.parsePartlikeScore()
        - ScoreExporter.postPartProcess()
        - ScoreExporter.renumberVoicesWithinStaffGroups()
        - ScoreExporter.scorePreliminaries()
        - ScoreExporter.setDefaults()
        - ScoreExporter.setEncoding()
        - ScoreExporter.setIdentification()
        - ScoreExporter.setMeterStream()
        - ScoreExporter.setPartExporterStaffGroups()
        - ScoreExporter.setPartList()
        - ScoreExporter.setPartsAndRefStream()
        - ScoreExporter.setScoreHeader()
        - ScoreExporter.setScoreLayouts()
        - ScoreExporter.setTitles()
        - ScoreExporter.staffGroupToXmlPartGroup()
        - ScoreExporter.styleToXmlAppearance()
        - ScoreExporter.textBoxToXmlCredit()
- XMLExporterBase
    - XMLExporterBase
        - XMLExporterBase.accidentalToMx()
        - XMLExporterBase.accidentalToMxAccidentalMark()
        - XMLExporterBase.addDividerComment()
        - XMLExporterBase.asBytes()
        - XMLExporterBase.dump()
        - XMLExporterBase.pageLayoutToXmlPageLayout()
        - XMLExporterBase.pageLayoutToXmlPrint()
        - XMLExporterBase.setColor()
        - XMLExporterBase.setEditorial()
        - XMLExporterBase.setFont()
        - XMLExporterBase.setPosition()
        - XMLExporterBase.setPrintObject()
        - XMLExporterBase.setPrintStyle()
        - XMLExporterBase.setPrintStyleAlign()
        - XMLExporterBase.setStyleAttributes()
        - XMLExporterBase.setTextFormatting()
        - XMLExporterBase.staffLayoutToXmlStaffLayout()
        - XMLExporterBase.systemLayoutToXmlPrint()
        - XMLExporterBase.systemLayoutToXmlSystemLayout()
        - XMLExporterBase.xmlHeader()
- Functions
    - getMetadataFromContext()
    - normalizeColor()
    - typeToMusicXMLType()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.stream.base

Next topic

music21.stream.enums

Table of Contents

- music21.stream.core
  - StreamCore
    - `StreamCore`
      - `StreamCore.spannerBundle`
      - `StreamCore.asTimespans()`
      - `StreamCore.asTree()`
      - `StreamCore.coreAppend()`
      - `StreamCore.coreCopyAsDerivation()`
      - `StreamCore.coreElementsChanged()`
      - `StreamCore.coreGatherMissingSpanners()`
      - `StreamCore.coreGetElementByMemoryLocation()`
      - `StreamCore.coreGuardBeforeAddElement()`
      - `StreamCore.coreHasElementByMemoryLocation()`
      - `StreamCore.coreInsert()`
      - `StreamCore.coreSelfActiveSite()`
      - `StreamCore.coreSetElementOffset()`
      - `StreamCore.coreStoreAtEnd()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.meter.tools

Next topic

music21.midi.percussion

Table of Contents

- music21.midi
  - ChannelModeMessages
    - `ChannelModeMessages`
  - ChannelVoiceMessages
    - `ChannelVoiceMessages`
  - DeltaTime
    - `DeltaTime`
      - `DeltaTime.getBytes()`
      - `DeltaTime.readUntilLowByte()`
  - MetaEvents
    - `MetaEvents`
  - MidiEvent
    - `MidiEvent`
      - `MidiEvent.sortOrder`
      - `MidiEvent.data`
      - `MidiEvent.pitch`
      - `MidiEvent.velocity`
      - `MidiEvent.getBytes()`
      - `MidiEvent.isDeltaTime()`
      - `MidiEvent.isNoteOff()`
      - `MidiEvent.isNoteOn()`
      - `MidiEvent.matchedNoteOff()`
      - `MidiEvent.parseChannelVoiceMessage()`
      - `MidiEvent.read()`
      - `MidiEvent.setPitchBend()`
  - MidiFile
    - `MidiFile`
      - `MidiFile.close()`
      - `MidiFile.open()`
      - `MidiFile.openFileLike()`
      - `MidiFile.read()`
      - `MidiFile.readstr()`
      - `MidiFile.write()`
      - `MidiFile.writeMThdStr()`
      - `MidiFile.writestr()`
  - MidiTrack
    - `MidiTrack`

    - `MidiTrack.length`
    - `MidiTrack.getBytes()`
    - `MidiTrack.getChannels()`
    - `MidiTrack.getProgramChanges()`
    - `MidiTrack.hasNotes()`
    - `MidiTrack.processDataToEvents()`
    - `MidiTrack.read()`
    - `MidiTrack.setChannel()`
    - `MidiTrack.updateEvents()`
- SysExEvents
  - `SysExEvents`
- Functions
  - `charToBinary()`
  - `getNumber()`
  - `getNumbersAsList()`
  - `getVariableLengthNumber()`
  - `intsToHexBytes()`
  - `putNumber()`
  - `putNumbersAsList()`
  - `putVariableLengthNumber()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.tinyNotation

Previous topic

music21.tie

Next topic

music21.tree

Table of Contents

- music21.tinyNotation
  - Converter
    - `Converter`
      - `Converter.load()`
      - `Converter.parse()`
      - `Converter.parseEndStates()`
      - `Converter.parseModifiers()`
      - `Converter.parseOne()`
      - `Converter.parseStartStates()`
      - `Converter.postParse()`
      - `Converter.setupRegularExpressions()`
      - `Converter.splitPreTokens()`
  - Token
    - `Token`
      - `Token.parse()`
  - State
    - `State`
      - `State.affectTokenAfterParse()`
      - `State.affectTokenAfterParseBeforeModifiers()`
      - `State.affectTokenBeforeParse()`
      - `State.end()`
      - `State.start()`
  - Modifier
    - `Modifier`
      - `Modifier.postParse()`
      - `Modifier.preParse()`
  - IdModifier
    - `IdModifier`
      - `IdModifier.postParse()`
  - LyricModifier
    - `LyricModifier`
      - `LyricModifier.postParse()`
  - NoteOrRestToken
    - `NoteOrRestToken`
      - `NoteOrRestToken.applyDuration()`
      - `NoteOrRestToken.dots()`
      - `NoteOrRestToken.durationType()`

- NoteToken
  - `NoteToken`
    - `NoteToken.editorialAccidental()`
    - `NoteToken.flats()`
    - `NoteToken.highOctave()`
    - `NoteToken.lowOctave()`
    - `NoteToken.natural()`
    - `NoteToken.parse()`
    - `NoteToken.processPitchMap()`
    - `NoteToken.sharps()`
- QuadrupletState
  - `QuadrupletState`
- RestToken
  - `RestToken`
    - `RestToken.parse()`
- TieState
  - `TieState`
    - `TieState.end()`
- TimeSignatureToken
  - `TimeSignatureToken`
    - `TimeSignatureToken.parse()`
- TripletState
  - `TripletState`
- TupletState
  - `TupletState`
    - `TupletState.affectTokenAfterParse()`
    - `TupletState.end()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.repeat

Next topic

music21.romanText.clercqTemperley

Table of Contents

- music21.roman
  - RomanNumeral
    - `RomanNumeral`
      - `RomanNumeral.figureAndKey`
      - `RomanNumeral.scaleDegreeWithAlteration`
      - `RomanNumeral.figure`
      - `RomanNumeral.functionalityScore`
      - `RomanNumeral.key`
      - `RomanNumeral.romanNumeral`
      - `RomanNumeral.__eq__()`
      - `RomanNumeral.adjustMinorVIandVIIByQuality()`
      - `RomanNumeral.bassScaleDegreeFromNotation()`
      - `RomanNumeral.isMixture()`
      - `RomanNumeral.isNeapolitan()`
      - `RomanNumeral.transpose()`
      - `RomanNumeral.addedSteps`
      - `RomanNumeral.bracketedAlterations`
      - `RomanNumeral.caseMatters`
      - `RomanNumeral.figuresNotationObj`
      - `RomanNumeral.figuresWritten`
      - `RomanNumeral.frontAlterationAccidental`
      - `RomanNumeral.frontAlterationString`
      - `RomanNumeral.frontAlterationTransposeInterval`
      - `RomanNumeral.impliedQuality`
      - `RomanNumeral.impliedScale`
      - `RomanNumeral.omittedSteps`
      - `RomanNumeral.pivotChord`
      - `RomanNumeral.primaryFigure`
      - `RomanNumeral.romanNumeralAlone`
      - `RomanNumeral.scaleCardinality`
      - `RomanNumeral.scaleDegree`
      - `RomanNumeral.secondaryRomanNumeral`
      - `RomanNumeral.secondaryRomanNumeralKey`
      - `RomanNumeral.seventhMinor`
      - `RomanNumeral.sixthMinor`
      - `RomanNumeral.useImpliedScale`
  - ChordFigureTuple

    - ChordFigureTuple
  - FigureTuple
    - FigureTuple
  - Minor67Default
    - Minor67Default
  - Functions
    - correctRNAlterationForMinor()
    - correctSuffixForChordQuality()
    - expandShortHand()
    - figureTupleSolo()
    - figureTuples()
    - identifyAsTonicOrDominant()
    - postFigureFromChordAndKey()
    - romanInversionName()
    - romanNumeralFromChord()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.musicxml.xmlObjects

Next topic

music21.musicxml.xmlToM21

Table of Contents

- music21.musicxml.xmlSoundParser
  - SoundTagMixin
    - `SoundTagMixin`
      - `SoundTagMixin.soundTagParser()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.musicxml.archiveTools

Previous topic

music21.musedata.translate

Next topic

music21.musicxml.helpers

Table of Contents

- music21.musicxml.archiveTools
  - Functions
    - `compressAllXMLFiles()`
    - `compressXML()`
    - `uncompressMXL()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.stream.enums

Next topic

music21.stream.iterator

Table of Contents

- music21.stream.filters
  - ClassFilter
    - `ClassFilter`
      - `ClassFilter.__eq__()`
  - ClassNotFilter
    - `ClassNotFilter`
  - GroupFilter
    - `GroupFilter`
  - IdFilter
    - `IdFilter`
  - IsFilter
    - `IsFilter`
      - `IsFilter.reset()`
  - IsNotFilter
    - `IsNotFilter`
      - `IsNotFilter.reset()`
  - OffsetFilter
    - `OffsetFilter`
      - `OffsetFilter.isElementOffsetInRange()`
  - OffsetHierarchyFilter
    - `OffsetHierarchyFilter`
  - StreamFilter
    - `StreamFilter`
      - `StreamFilter.reset()`

Table of Contents

- About *music21*

- User’s Guide

- **Module Reference**

- Installation

- Developer Reference

- Documentation and tests in progress

Navigation

Previous topic

music21.search.lyrics

Next topic

music21.search.serial

Table of Contents

- music21.search.segment
  - Functions
    - `getDifflibOrPyLev()`
    - `indexOnePath()`
    - `indexScoreFilePaths()`
    - `indexScoreParts()`
    - `loadScoreDict()`
    - `saveScoreDict()`
    - `scoreSimilarity()`
    - `translateMonophonicPartToSegments()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.stream.tools

Previous topic

music21.stream.streamStatus

Next topic

music21.style

Table of Contents

- music21.stream.tools
  - Functions
    - `removeDuplicates()`

Table of Contents

- About *music21*

- User’s Guide

- **Module Reference**

- Installation

- Developer Reference

- Documentation and tests in progress

Navigation

Previous topic

music21.tempo

Next topic

music21.tie

Table of Contents

- music21.text
  - TextBox
    - `TextBox`
      - `TextBox.content`
      - `TextBox.page`
  - LanguageDetector
    - `LanguageDetector`
      - `LanguageDetector.mostLikelyLanguage()`
      - `LanguageDetector.mostLikelyLanguageNumeric()`
      - `LanguageDetector.readExcerpts()`
  - Trigram
    - `Trigram`
      - `Trigram.length`
      - `Trigram.likely()`
      - `Trigram.makeWords()`
      - `Trigram.measure()`
      - `Trigram.parseExcerpt()`
      - `Trigram.similarity()`
  - Functions
    - `assembleAllLyrics()`
    - `assembleLyrics()`
    - `postpendArticle()`
    - `prependArticle()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.stream.iterator

Next topic

music21.stream.streamStatus

Table of Contents

- music21.stream.makeNotation
  - Functions
    - `consolidateCompletedTuplets()`
    - `getTiePitchSet()`
    - `iterateBeamGroups()`
    - `makeAccidentalsInMeasureStream()`
    - `makeBeams()`
    - `makeMeasures()`
    - `makeOrnamentalAccidentals()`
    - `makeRests()`
    - `makeTies()`
    - `makeTupletBrackets()`
    - `moveNotesToVoices()`
    - `ornamentalPitches()`
    - `realizeOrnaments()`
    - `saveAccidentalDisplayStatus()`
    - `setStemDirectionForBeamGroups()`
    - `setStemDirectionOneGroup()`
    - `splitElementsToCompleteTuplets()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Table of Contents

- music21.stream.iterator
  - StreamIterator
    - `StreamIterator`
      - `StreamIterator.activeElementList`
      - `StreamIterator.notes`
      - `StreamIterator.notesAndRests`
      - `StreamIterator.parts`
      - `StreamIterator.spanners`
      - `StreamIterator.voices`
      - `StreamIterator.__getitem__()`
      - `StreamIterator.addFilter()`
      - `StreamIterator.cleanup()`
      - `StreamIterator.clone()`
      - `StreamIterator.first()`
      - `StreamIterator.getElementById()`
      - `StreamIterator.getElementsByClass()`
      - `StreamIterator.getElementsByGroup()`
      - `StreamIterator.getElementsByOffset()`
      - `StreamIterator.getElementsByQuerySelector()`
      - `StreamIterator.getElementsNotOfClass()`
      - `StreamIterator.last()`
      - `StreamIterator.matchesFilters()`
      - `StreamIterator.matchingElements()`
      - `StreamIterator.removeFilter()`
      - `StreamIterator.reset()`
      - `StreamIterator.resetCaches()`
      - `StreamIterator.stream()`
      - `StreamIterator.updateActiveInformation()`
  - RecursiveIterator
    - `RecursiveIterator`
      - `RecursiveIterator.currentHierarchyOffset()`
      - `RecursiveIterator.getElementsByClass()`
      - `RecursiveIterator.getElementsByOffsetInHierarchy()`
      - `RecursiveIterator.iteratorStack()`
      - `RecursiveIterator.matchingElements()`
      - `RecursiveIterator.reset()`
      - `RecursiveIterator.streamStack()`

- OffsetIterator
    - `OffsetIterator`
        - `OffsetIterator.getElementsByClass()`
        - `OffsetIterator.reset()`
- ActiveInformation
    - `ActiveInformation`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.omr.correctors

Next topic

music21.percussion

Table of Contents

- music21.omr.evaluators
  - OmrGroundTruthPair
    - `OmrGroundTruthPair`
      - `OmrGroundTruthPair.debug`
      - `OmrGroundTruthPair.deleteCost()`
      - `OmrGroundTruthPair.getDifferences()`
      - `OmrGroundTruthPair.getGroundScore()`
      - `OmrGroundTruthPair.getOmrScore()`
      - `OmrGroundTruthPair.hashAll()`
      - `OmrGroundTruthPair.insertCost()`
      - `OmrGroundTruthPair.minEditDist()`
      - `OmrGroundTruthPair.parseAll()`
      - `OmrGroundTruthPair.substCost()`
  - Functions
    - `autoCorrelationBestMeasure()`
    - `evaluateCorrectingModel()`

Table of Contents

- About *music21*

- User’s Guide

- **Module Reference**

- Installation

- Developer Reference

- Documentation and tests in progress

Navigation

Previous topic

music21.lily.translate

Next topic

music21.metadata

Table of Contents

- music21.mei.base
  - MeiToM21Converter
    - `MeiToM21Converter`
      - `MeiToM21Converter.run()`
  - Functions
    - `accidFromElement()`
    - `articFromElement()`
    - `beamFromElement()`
    - `chordFromElement()`
    - `clefFromElement()`
    - `dotFromElement()`
    - `instrDefFromElement()`
    - `layerFromElement()`
    - `measureFromElement()`
    - `noteFromElement()`
    - `spaceFromElement()`
    - `mSpaceFromElement()`
    - `restFromElement()`
    - `mRestFromElement()`
    - `scoreFromElement()`
    - `sectionFromElement()`
    - `scoreDefFromElement()`
    - `staffFromElement()`
    - `staffDefFromElement()`
    - `staffGrpFromElement()`
    - `tupletFromElement()`
    - `addSlurs()`
    - `allPartsPresent()`
    - `barLineFromElement()`
    - `beamTogether()`
    - `getVoiceId()`
    - `makeDuration()`
    - `makeMetadata()`
    - `metaSetComposer()`
    - `metaSetDate()`
    - `metaSetTitle()`
    - `removeOctothorpe()`

        - safePitch()
        - scaleToTuplet()
        - sectionScoreCore()
        - sylFromElement()
        - verseFromElement()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.midi.translate

Next topic

music21.musedata.base40

Table of Contents

- music21.musedata
    - MuseDataWork
        - `MuseDataWork`
            - `MuseDataWork.addFile()`
            - `MuseDataWork.addString()`
            - `MuseDataWork.getParts()`
    - MuseDataDirectory
        - `MuseDataDirectory`
            - `MuseDataDirectory.getPaths()`
            - `MuseDataDirectory.isMusedataFile()`
    - MuseDataFile
        - `MuseDataFile`
            - `MuseDataFile.close()`
            - `MuseDataFile.open()`
            - `MuseDataFile.read()`
            - `MuseDataFile.readstr()`
    - MuseDataMeasure
        - `MuseDataMeasure`
            - `MuseDataMeasure.getBarObject()`
            - `MuseDataMeasure.getMeasureObject()`
            - `MuseDataMeasure.getRecords()`
            - `MuseDataMeasure.hasNotes()`
            - `MuseDataMeasure.hasVoices()`
    - MuseDataMeasureIterator
        - `MuseDataMeasureIterator`
    - MuseDataPart
        - `MuseDataPart`
            - `MuseDataPart.getClefObject()`
            - `MuseDataPart.getDirective()`
            - `MuseDataPart.getDivisionsPerQuarterNote()`
            - `MuseDataPart.getGroupMembershipNumber()`
            - `MuseDataPart.getGroupMemberships()`
            - `MuseDataPart.getGroupMembershipsTotal()`
            - `MuseDataPart.getKeyParameters()`
            - `MuseDataPart.getKeySignature()`
            - `MuseDataPart.getMeasures()`
            - `MuseDataPart.getMovementNumber()`

        - `MuseDataPart.getMovementTitle()`
        - `MuseDataPart.getPartName()`
        - `MuseDataPart.getSource()`
        - `MuseDataPart.getTimeSignatureObject()`
        - `MuseDataPart.getTimeSignatureParameters()`
        - `MuseDataPart.getTranspositionIntervalObject()`
        - `MuseDataPart.getWorkNumber()`
        - `MuseDataPart.getWorkTitle()`
        - `MuseDataPart.update()`
    - MuseDataRecord
        - `MuseDataRecord`
            - `MuseDataRecord.getArticulationObjects()`
            - `MuseDataRecord.getBeams()`
            - `MuseDataRecord.getDots()`
            - `MuseDataRecord.getDynamicObjects()`
            - `MuseDataRecord.getExpressionObjects()`
            - `MuseDataRecord.getLyrics()`
            - `MuseDataRecord.getPitchObject()`
            - `MuseDataRecord.getQuarterLength()`
            - `MuseDataRecord.hasCautionaryAccidental()`
            - `MuseDataRecord.isBack()`
            - `MuseDataRecord.isChord()`
            - `MuseDataRecord.isCueOrGrace()`
            - `MuseDataRecord.isNote()`
            - `MuseDataRecord.isRest()`
            - `MuseDataRecord.isTied()`
    - MuseDataRecordIterator
        - `MuseDataRecordIterator`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.ipython21

Next topic

music21.key

Table of Contents

- music21.ipython21.converters
  - Functions
    - `displayMusic21jMIDI()`
    - `showImageThroughMuseScore()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.metadata.caching

Next topic

music21.metadata.properties

Table of Contents

- music21.metadata.primitives
  - Text
    - `Text`
      - `Text.language`
      - `Text.__eq__()`
      - `Text.getNormalizedArticle()`
  - Date
    - `Date`
      - `Date.datetime`
      - `Date.hasError`
      - `Date.hasTime`
      - `Date.__eq__()`
      - `Date.errorToSymbol()`
      - `Date.load()`
      - `Date.loadDatetime()`
      - `Date.loadOther()`
      - `Date.loadStr()`
  - DateSingle
    - `DateSingle`
  - DateRelative
    - `DateRelative`
      - `DateRelative.relevance`
  - DateBetween
    - `DateBetween`
      - `DateBetween.relevance`
  - DateSelection
    - `DateSelection`
      - `DateSelection.relevance`
  - Contributor
    - `Contributor`
      - `Contributor.name`
      - `Contributor.names`
      - `Contributor.role`
      - `Contributor.__eq__()`
      - `Contributor.abbreviationToRole()`
      - `Contributor.age()`
      - `Contributor.roleToAbbreviation()`

  - Copyright
    - `Copyright`
      - `Copyright.__eq__()`
  - Creator
    - `Creator`
  - DatePrimitive
    - `DatePrimitive`
      - `DatePrimitive.datetime`
      - `DatePrimitive.relevance`
      - `DatePrimitive.__eq__()`
  - Imprint
    - `Imprint`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.romanText.writeRoman

Next topic

music21.scale.intervalNetwork

Table of Contents

- music21.scale
  - ConcreteScale
    - `ConcreteScale`
      - `ConcreteScale.chord`
      - `ConcreteScale.isConcrete`
      - `ConcreteScale.name`
      - `ConcreteScale.pitches`
      - `ConcreteScale.abstract`
      - `ConcreteScale.__eq__()`
      - `ConcreteScale.derive()`
      - `ConcreteScale.deriveAll()`
      - `ConcreteScale.deriveByDegree()`
      - `ConcreteScale.deriveRanked()`
      - `ConcreteScale.findMissing()`
      - `ConcreteScale.getChord()`
      - `ConcreteScale.getDegreeMaxUnique()`
      - `ConcreteScale.getPitches()`
      - `ConcreteScale.getScalaData()`
      - `ConcreteScale.getScaleDegreeAndAccidentalFromPitch()`
      - `ConcreteScale.getScaleDegreeFromPitch()`
      - `ConcreteScale.getTonic()`
      - `ConcreteScale.intervalBetweenDegrees()`
      - `ConcreteScale.isNext()`
      - `ConcreteScale.match()`
      - `ConcreteScale.next()`
      - `ConcreteScale.nextPitch()`
      - `ConcreteScale.pitchFromDegree()`
      - `ConcreteScale.pitchesFromScaleDegrees()`
      - `ConcreteScale.romanNumeral()`
      - `ConcreteScale.show()`
      - `ConcreteScale.solfeg()`
      - `ConcreteScale.transpose()`
      - `ConcreteScale.tune()`
      - `ConcreteScale.write()`
  - AbstractScale
    - `AbstractScale`
      - `AbstractScale.buildNetwork()`

- AbstractScale.buildNetworkFromPitches()
  - AbstractScale.fixDefaultOctaveForPitchList()
  - AbstractScale.getDegreeMaxUnique()
  - AbstractScale.getIntervals()
  - AbstractScale.getNewTonicPitch()
  - AbstractScale.getPitchFromNodeDegree()
  - AbstractScale.getRealization()
  - AbstractScale.getRelativeNodeDegree()
  - AbstractScale.getScalaData()
  - AbstractScale.nextPitch()
  - AbstractScale.realizePitchByDegree()
  - AbstractScale.show()
  - AbstractScale.write()
- AbstractCyclicalScale
  - AbstractCyclicalScale
    - AbstractCyclicalScale.buildNetwork()
- AbstractDiatonicScale
  - AbstractDiatonicScale
    - AbstractDiatonicScale.buildNetwork()
- AbstractHarmonicMinorScale
  - AbstractHarmonicMinorScale
    - AbstractHarmonicMinorScale.buildNetwork()
- AbstractMelodicMinorScale
  - AbstractMelodicMinorScale
    - AbstractMelodicMinorScale.buildNetwork()
- AbstractOctatonicScale
  - AbstractOctatonicScale
    - AbstractOctatonicScale.buildNetwork()
- AbstractOctaveRepeatingScale
  - AbstractOctaveRepeatingScale
    - AbstractOctaveRepeatingScale.buildNetwork()
- AbstractRagAsawari
  - AbstractRagAsawari
    - AbstractRagAsawari.buildNetwork()
- AbstractRagMarwa
  - AbstractRagMarwa
    - AbstractRagMarwa.buildNetwork()
- AbstractWeightedHexatonicBlues
  - AbstractWeightedHexatonicBlues
    - AbstractWeightedHexatonicBlues.buildNetwork()
- ChromaticScale
  - ChromaticScale
- CyclicalScale
  - CyclicalScale
- DiatonicScale
  - DiatonicScale
    - DiatonicScale.getDominant()
    - DiatonicScale.getLeadingTone()
    - DiatonicScale.getParallelMajor()
    - DiatonicScale.getParallelMinor()
    - DiatonicScale.getRelativeMajor()
    - DiatonicScale.getRelativeMinor()
    - DiatonicScale.getTonic()
- DorianScale
  - DorianScale
- HarmonicMinorScale
  - HarmonicMinorScale
- HypoaeolianScale

    - `HypoaeolianScale`
  - HypodorianScale
    - `HypodorianScale`
  - HypolocrianScale
    - `HypolocrianScale`
  - HypolydianScale
    - `HypolydianScale`
  - HypomixolydianScale
    - `HypomixolydianScale`
  - HypophrygianScale
    - `HypophrygianScale`
  - LocrianScale
    - `LocrianScale`
  - LydianScale
    - `LydianScale`
  - MajorScale
    - `MajorScale`
  - MelodicMinorScale
    - `MelodicMinorScale`
  - MinorScale
    - `MinorScale`
  - MixolydianScale
    - `MixolydianScale`
  - OctatonicScale
    - `OctatonicScale`
  - OctaveRepeatingScale
    - `OctaveRepeatingScale`
  - PhrygianScale
    - `PhrygianScale`
  - RagAsawari
    - `RagAsawari`
  - RagMarwa
    - `RagMarwa`
  - ScalaScale
    - `ScalaScale`
  - Scale
    - `Scale`
      - `Scale.isConcrete`
      - `Scale.name`
      - `Scale.extractPitchList()`
  - SieveScale
    - `SieveScale`
  - WeightedHexatonicBlues
    - `WeightedHexatonicBlues`
  - WholeToneScale
    - `WholeToneScale`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference

- Documentation and tests in progress

Previous topic

music21.metadata.properties

Next topic

music21.meter.core

Table of Contents

- music21.meter.base
  - TimeSignature
    - `TimeSignature`
      - `TimeSignature.beatCountName`
      - `TimeSignature.beatDivisionCount`
      - `TimeSignature.beatDivisionCountName`
      - `TimeSignature.beatDivisionDurations`
      - `TimeSignature.beatDuration`
      - `TimeSignature.beatLengthToQuarterLengthRatio`
      - `TimeSignature.beatSubDivisionDurations`
      - `TimeSignature.classification`
      - `TimeSignature.quarterLengthToBeatLengthRatio`
      - `TimeSignature.barDuration`
      - `TimeSignature.beatCount`
      - `TimeSignature.denominator`
      - `TimeSignature.numerator`
      - `TimeSignature.ratioString`
      - `TimeSignature.summedNumerator`
      - `TimeSignature.averageBeatStrength()`
      - `TimeSignature.getAccent()`
      - `TimeSignature.getAccentWeight()`
      - `TimeSignature.getBeams()`
      - `TimeSignature.getBeat()`
      - `TimeSignature.getBeatDepth()`
      - `TimeSignature.getBeatDuration()`
      - `TimeSignature.getBeatOffsets()`
      - `TimeSignature.getBeatProgress()`
      - `TimeSignature.getBeatProportion()`
      - `TimeSignature.getBeatProportionStr()`
      - `TimeSignature.getMeasureOffsetOrMeterModulusOffset()`
      - `TimeSignature.getOffsetFromBeat()`
      - `TimeSignature.load()`
      - `TimeSignature.ratioEqual()`
      - `TimeSignature.resetValues()`
      - `TimeSignature.setAccentWeight()`
      - `TimeSignature.setDisplay()`
      - `TimeSignature.accentSequence`

            - `TimeSignature.beamSequence`
            - `TimeSignature.beatSequence`
            - `TimeSignature.displaySequence`
            - `TimeSignature.symbol`
            - `TimeSignature.symbolizeDenominator`
        - SenzaMisuraTimeSignature
            - `SenzaMisuraTimeSignature`
        - TimeSignatureBase
            - `TimeSignatureBase`
        - Functions
            - `bestTimeSignature()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.prebase

Previous topic

music21.pitch

Next topic

music21.repeat

Table of Contents

- music21.prebase
  - ProtoM21Object
    - `ProtoM21Object`
      - `ProtoM21Object.classSet`
      - `ProtoM21Object.classes`
      - `ProtoM21Object.isClassOrSubclass()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.voiceLeading

Next topic

music21.volume

Table of Contents

- music21.volpiano
  - ColumnBreak
    - `ColumnBreak`
  - ErrorLevel
    - `ErrorLevel`
  - LineBreak
    - `LineBreak`
  - Neume
    - `Neume`
  - PageBreak
    - `PageBreak`
  - Functions
    - `fromStream()`
    - `toPart()`

Table of Contents

- About *music21*
- User's Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.text

Next topic

music21.tinyNotation

Table of Contents

- music21.tie
  - Tie
    - `Tie`
      - `Tie.__eq__()`
      - `Tie.placement`
      - `Tie.style`
      - `Tie.type`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.scale.intervalNetwork

Previous topic

music21.scale

Next topic

music21.scale.scala

Table of Contents

- music21.scale.intervalNetwork
  - IntervalNetwork
    - IntervalNetwork
      - IntervalNetwork.degreeMax
      - IntervalNetwork.degreeMaxUnique
      - IntervalNetwork.degreeMin
      - IntervalNetwork.terminusHighNodes
      - IntervalNetwork.terminusLowNodes
      - IntervalNetwork.__eq__()
      - IntervalNetwork.clear()
      - IntervalNetwork.degreeModulus()
      - IntervalNetwork.fillArbitrary()
      - IntervalNetwork.fillBiDirectedEdges()
      - IntervalNetwork.fillDirectedEdges()
      - IntervalNetwork.fillMelodicMinor()
      - IntervalNetwork.filterPitchList()
      - IntervalNetwork.find()
      - IntervalNetwork.findMissing()
      - IntervalNetwork.getNeighborNodeIds()
      - IntervalNetwork.getNetworkxGraph()
      - IntervalNetwork.getNext()
      - IntervalNetwork.getNodeDegreeDictionary()
      - IntervalNetwork.getPitchFromNodeDegree()
      - IntervalNetwork.getRelativeNodeDegree()
      - IntervalNetwork.getRelativeNodeId()
      - IntervalNetwork.getUnalteredPitch()
      - IntervalNetwork.match()
      - IntervalNetwork.nextPitch()
      - IntervalNetwork.nodeIdToDegree()
      - IntervalNetwork.nodeIdToEdgeDirections()
      - IntervalNetwork.nodeNameToNodes()
      - IntervalNetwork.plot()
      - IntervalNetwork.processAlteredNodes()
      - IntervalNetwork.realize()
      - IntervalNetwork.realizeAscending()
      - IntervalNetwork.realizeDescending()
      - IntervalNetwork.realizeIntervals()

        - `IntervalNetwork.realizeMinMax()`
        - `IntervalNetwork.realizePitch()`
        - `IntervalNetwork.realizePitchByDegree()`
        - `IntervalNetwork.realizeTermini()`
        - `IntervalNetwork.transposePitchAndApplySimplification()`
        - `IntervalNetwork.weightedSelection()`
  - Node
    - `Node`
      - `Node.__eq__()`
  - Edge
    - `Edge`
      - `Edge.connections`
      - `Edge.__eq__()`
      - `Edge.addBiDirectedConnections()`
      - `Edge.addDirectedConnection()`
      - `Edge.getConnections()`
  - BoundIntervalNetwork
    - `BoundIntervalNetwork`
  - Direction
    - `Direction`
  - Terminus
    - `Terminus`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.note

Next topic

music21.noteworthy.translate

Table of Contents

- music21.noteworthy.binaryTranslate
  - NWCConverter
    - `NWCConverter`
      - `NWCConverter.advanceToNotNUL()`
      - `NWCConverter.byteToInt()`
      - `NWCConverter.byteToSignedInt()`
      - `NWCConverter.dumpToNWCText()`
      - `NWCConverter.fileVersion()`
      - `NWCConverter.isValidNWCFile()`
      - `NWCConverter.parse()`
      - `NWCConverter.parseFile()`
      - `NWCConverter.parseHeader()`
      - `NWCConverter.parseString()`
      - `NWCConverter.readBytes()`
      - `NWCConverter.readLEShort()`
      - `NWCConverter.readToNUL()`
      - `NWCConverter.skipBytes()`
      - `NWCConverter.toStream()`
  - NWCObject
    - `NWCObject`
      - `NWCObject.barline()`
      - `NWCObject.clef()`
      - `NWCObject.dynamic()`
      - `NWCObject.dynamicVariation()`
      - `NWCObject.ending()`
      - `NWCObject.flowDir()`
      - `NWCObject.instrument()`
      - `NWCObject.keySig()`
      - `NWCObject.mpc()`
      - `NWCObject.note()`
      - `NWCObject.noteChordMember()`
      - `NWCObject.parse()`
      - `NWCObject.pedal()`
      - `NWCObject.performance()`
      - `NWCObject.rest()`
      - `NWCObject.restChordMember()`
      - `NWCObject.setDurationForObject()`

- NWCObject.tempo()
- NWCObject.tempoVariation()
- NWCObject.textObj()
- NWCObject.timeSig()
- NWCStaff
  - NWCStaff
    - NWCStaff.dump()
    - NWCStaff.parse()
    - NWCStaff.parseHeader()
    - NWCStaff.parseLyrics()
    - NWCStaff.parseObjects()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.tree.core

Previous topic

music21.tree.analysis

Next topic

music21.tree.fromStream

Table of Contents

- music21.tree.core
  - AVLNode
    - `AVLNode`
      - `AVLNode.debug()`
      - `AVLNode.moveAttributes()`
      - `AVLNode.rebalance()`
      - `AVLNode.rotateLeftLeft()`
      - `AVLNode.rotateLeftRight()`
      - `AVLNode.rotateRightLeft()`
      - `AVLNode.rotateRightRight()`
      - `AVLNode.update()`
      - `AVLNode.balance`
      - `AVLNode.height`
      - `AVLNode.leftChild`
      - `AVLNode.payload`
      - `AVLNode.position`
      - `AVLNode.rightChild`
  - AVLTree
    - `AVLTree`
      - `AVLTree.createNodeAtPosition()`
      - `AVLTree.debug()`
      - `AVLTree.getNodeAfter()`
      - `AVLTree.getNodeBefore()`
      - `AVLTree.getNodeByPosition()`
      - `AVLTree.getPositionAfter()`
      - `AVLTree.getPositionBefore()`
      - `AVLTree.populateFromSortedList()`
      - `AVLTree.removeNode()`

Table of Contents

- About *music21*

- User’s Guide

- **Module Reference**

- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.omr.evaluators

Next topic

music21.pitch

Table of Contents

- music21.percussion
  - PercussionChord
    - `PercussionChord`
      - `PercussionChord.notes`
      - `PercussionChord.pitches`
      - `PercussionChord.__eq__()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.musicxml.m21ToXml

Next topic

music21.musicxml.test_m21ToXml

Table of Contents

- music21.musicxml.partStaffExporter
  - PartStaffExporterMixin
    - `PartStaffExporterMixin`
      - `PartStaffExporterMixin.addStaffTagsMultiStaffParts()`
      - `PartStaffExporterMixin.cleanUpSubsequentPartStaffs()`
      - `PartStaffExporterMixin.getRootForPartStaff()`
      - `PartStaffExporterMixin.joinPartStaffs()`
      - `PartStaffExporterMixin.joinableGroups()`
      - `PartStaffExporterMixin.moveMeasureContents()`
      - `PartStaffExporterMixin.movePartStaffMeasureContents()`
      - `PartStaffExporterMixin.processSubsequentPartStaff()`
      - `PartStaffExporterMixin.setEarliestAttributesAndClefsPartStaff()`
  - Functions
    - `addStaffTags()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.tablature

Next topic

music21.text

Table of Contents

- music21.tempo
  - MetronomeMark
    - `MetronomeMark`
      - `MetronomeMark.number`
      - `MetronomeMark.numberSounding`
      - `MetronomeMark.referent`
      - `MetronomeMark.text`
      - `MetronomeMark.durationToSeconds()`
      - `MetronomeMark.getEquivalentByReferent()`
      - `MetronomeMark.getMaintainedNumberWithReferent()`
      - `MetronomeMark.getQuarterBPM()`
      - `MetronomeMark.getTextExpression()`
      - `MetronomeMark.secondsPerQuarter()`
      - `MetronomeMark.secondsToDuration()`
      - `MetronomeMark.setQuarterBPM()`
      - `MetronomeMark.placement`
  - TempoText
    - `TempoText`
      - `TempoText.text`
      - `TempoText.applyTextFormatting()`
      - `TempoText.getMetronomeMark()`
      - `TempoText.getTextExpression()`
      - `TempoText.isCommonTempoText()`
      - `TempoText.setTextExpression()`
  - MetricModulation
    - `MetricModulation`
      - `MetricModulation.number`
      - `MetricModulation.newMetronome`
      - `MetricModulation.newReferent`
      - `MetricModulation.oldMetronome`
      - `MetricModulation.oldReferent`
      - `MetricModulation.setEqualityByReferent()`
      - `MetricModulation.setOtherByReferent()`
      - `MetricModulation.updateByContext()`
  - TempoIndication
    - `TempoIndication`
      - `TempoIndication.getPreviousMetronomeMark()`

        - `TempoIndication.getSoundingMetronomeMark()`
  - AccelerandoSpanner
    - `AccelerandoSpanner`
  - RitardandoSpanner
    - `RitardandoSpanner`
  - TempoChangeSpanner
    - `TempoChangeSpanner`
  - Functions
    - `interpolateElements()`
    - `convertTempoByReferent()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Quick search

- Show Source

This Page

- Show Source

music21.tempo¶

This module defines objects for describing tempo and changes in tempo.

MetronomeMark¶

class music21.tempo.MetronomeMark(*text: str | int | None = None*, *number: OffsetQLIn | None = None*, *referent: OffsetQLIn | str | duration.Duration | base.Music21Object | None = None*, ***, *parentheses: bool = False*, *playbackOnly: bool = False*, *numberSounding: OffsetQLIn | None = None*, *numberImplicit: bool | None = None*, ***keywords*)¶

A way of specifying a particular tempo with a text string, a referent (a duration) and a number.

The *referent* attribute is a Duration object, or a string duration type or a floating-point quarter-length value used to create a Duration.

MetronomeMarks, as Music21Object subclasses, also have .duration object property independent of the *referent*.

```
>>> a = tempo.MetronomeMark('slow', 40, note.Note(type='half'))
>>> a.number
40
>>> a.referent
<music21.duration.Duration 2.0>
>>> a.referent.type
'half'
>>> print(a.text)
slow
```

Some text marks will automatically suggest a number.

```
>>> mm = tempo.MetronomeMark('adagio')
>>> mm.number
56
>>> mm.numberImplicit
True
```

For certain numbers, a text value can be set implicitly

```
>>> tm2 = tempo.MetronomeMark(number=208)
>>> print(tm2.text)
prestissimo
>>> tm2.referent
<music21.duration.Duration 1.0>
```

Unicode values work fine thanks to Python 3:

```
>>> marking = 'très vite'
>>> marking
'très vite'
>>> print(tempo.defaultTempoValues[marking])
144
>>> tm2 = tempo.MetronomeMark(marking)
>>> tm2.text.endswith('vite')
True
>>> tm2.number
144
```

For playback only (no score output) set numberSounding but no number:

```
>>> fast = tempo.MetronomeMark(numberSounding=168)
>>> fast
<music21.tempo.MetronomeMark Quarter=168 (playback only)>
```

`MetronomeMark` bases

- `TempoIndication`

- Music21Object
- ProtoM21Object

MetronomeMark read-only properties

Read-only properties inherited from Music21Object:

- beat
- beatDuration
- beatStr
- beatStrength
- hasEditorialInformation
- hasStyleInformation
- measureNumber

Read-only properties inherited from ProtoM21Object:

- classSet
- classes

MetronomeMark read/write properties

MetronomeMark.number¶

Get and set the number, or the numerical value of the Metronome.

```
>>> mm = tempo.MetronomeMark('slow')
>>> mm.number
56
>>> mm.numberImplicit
True
>>> mm.number = 52.5
>>> mm.number
52.5
>>> mm.numberImplicit
False
```

MetronomeMark.numberSounding¶

Get and set the numberSounding, or the numerical value of the Metronome that is used for playback independent of display. If numberSounding is None, number is assumed to be numberSounding.

```
>>> mm = tempo.MetronomeMark('slow')
>>> mm.number
56
>>> mm.numberImplicit
True
>>> mm.numberSounding is None
True
>>> mm.numberSounding = 120
>>> mm.numberSounding
120
```

MetronomeMark.referent¶

Get or set the referent, or the Duration object that is the reference for the tempo value in BPM.

MetronomeMark.text¶

Get or set a text string for this MetronomeMark. Internally implemented as a TempoText object, which stores the text in a TextExpression object.

```
>>> mm = tempo.MetronomeMark(number=123)
>>> mm.text == None
True
>>> mm.text = 'medium fast'
>>> print(mm.text)
medium fast
```

Read/write properties inherited from `Music21Object`:

- `activeSite`
- `derivation`
- `duration`
- `editorial`
- `id`
- `offset`
- `priority`
- `quarterLength`
- `seconds`
- `style`

`MetronomeMark` methods

MetronomeMark.durationToSeconds(*durationOrQuarterLength*)¶

Given a duration specified as a `Duration` object or a quarter length, return the resultant time in seconds at the tempo specified by this MetronomeMark.

```
>>> mm1 = tempo.MetronomeMark(referent=1.0, number=60.0)
>>> mm1.durationToSeconds(60)
60.0
>>> mm1.durationToSeconds(duration.Duration('16th'))
0.25
```

MetronomeMark.getEquivalentByReferent(*referent*)¶

Return a new MetronomeMark object that has an equivalent speed but different number and referent values based on a supplied referent (given as a Duration type, quarterLength, or Duration object).

```
>>> mm1 = tempo.MetronomeMark(number=60, referent=1.0)
>>> mm1.getEquivalentByReferent(0.5)
<music21.tempo.MetronomeMark larghetto Eighth=120>
>>> mm1.getEquivalentByReferent(duration.Duration('half'))
<music21.tempo.MetronomeMark larghetto Half=30>

>>> mm1.getEquivalentByReferent('longa')
<music21.tempo.MetronomeMark larghetto Imperfect Longa=3.75>
```

MetronomeMark.getMaintainedNumberWithReferent(*referent*)¶

Return a new MetronomeMark object that has an equivalent number but a new referent.

MetronomeMark.getQuarterBPM(*useNumberSounding=True*) → float | None¶

Get a BPM value where the beat is a quarter; must convert from the defined beat to a quarter beat. Will return None if no beat number is defined.

This mostly used for generating MusicXML <sound> tags when necessary.

```
>>> mm = tempo.MetronomeMark(number=60, referent='half')
>>> mm.getQuarterBPM()
120.0
>>> mm.referent = 'quarter'
>>> mm.getQuarterBPM()
60.0
```

MetronomeMark.getTextExpression(*returnImplicit=False*)¶

If there is a TextExpression available that is not implicit, return it; otherwise, return None.

```
>>> mm = tempo.MetronomeMark('presto')
>>> mm.number
184
>>> mm.numberImplicit
True
>>> mm.getTextExpression()
<music21.expressions.TextExpression 'presto'>
>>> mm.textImplicit
False
```

```
>>> mm = tempo.MetronomeMark(number=90)
>>> mm.numberImplicit
False
>>> mm.textImplicit
True
>>> mm.getTextExpression() is None
True
>>> mm.getTextExpression(returnImplicit=True)
<music21.expressions.TextExpression 'maestoso'>
```

MetronomeMark.secondsPerQuarter()¶

Return the duration in seconds for each quarter length (not necessarily the referent) of this MetronomeMark.

```
>>> mm1 = tempo.MetronomeMark(referent=1.0, number=60.0)
>>> mm1.secondsPerQuarter()
1.0
>>> mm1 = tempo.MetronomeMark(referent=2.0, number=60.0)
>>> mm1.secondsPerQuarter()
0.5
>>> mm1 = tempo.MetronomeMark(referent=2.0, number=30.0)
>>> mm1.secondsPerQuarter()
1.0
```

MetronomeMark.secondsToDuration(*seconds*)¶

Given a duration in seconds, return a `Duration` object equal to that time.

```
>>> mm1 = tempo.MetronomeMark(referent=1.0, number=60.0)
>>> mm1.secondsToDuration(0.25)
<music21.duration.Duration 0.25>
>>> mm1.secondsToDuration(0.5).type
'eighth'
>>> mm1.secondsToDuration(1)
<music21.duration.Duration 1.0>
```

MetronomeMark.setQuarterBPM(*value*, *setNumber=True*)¶

Given a value in BPM, use it to set the value of this MetronomeMark. BPM values are assumed to refer only to quarter notes; different beat values, if defined here, will be scaled

```
>>> mm = tempo.MetronomeMark(number=60, referent='half')
>>> mm.setQuarterBPM(240)  # set to 240 for a quarter
>>> mm.number  # a half is half as fast
120
```

Methods inherited from `TempoIndication`:

- `getPreviousMetronomeMark()`
- `getSoundingMetronomeMark()`

Methods inherited from Music21Object:

- __eq__()
- clearCache()
- containerHierarchy()
- contextSites()
- getAllContextsByClass()
- getContextByClass()
- getOffsetBySite()
- getOffsetInHierarchy()
- getSpannerSites()
- informSites()
- mergeAttributes()
- next()
- previous()
- purgeLocations()
- purgeOrphans()
- setOffsetBySite()
- show()
- sortTuple()
- splitAtDurations()
- splitAtQuarterLength()
- splitByQuarterLengths()
- write()

Methods inherited from ProtoM21Object:

- isClassOrSubclass()

MetronomeMark instance variables

MetronomeMark.placement¶

Staff placement: ‘above’, ‘below’, or None.

A setting of None implies that the placement will be determined by notation software and no particular placement is demanded.

This is not placed in the *.style* property, since for some expressions, the placement above or below an object has semantic meaning and is not purely presentational.

Instance variables inherited from Music21Object:

- classSortOrder
- groups
- isStream
- sites

TempoText¶

class music21.tempo.TempoText(*text=None*, ***keywords*)¶

```
>>> import music21
>>> tm = music21.tempo.TempoText('adagio')
>>> tm
<music21.tempo.TempoText 'adagio'>
>>> print(tm.text)
adagio
```

TempoText bases

- TempoIndication

- Music21Object
- ProtoM21Object

TempoText read-only properties

Read-only properties inherited from Music21Object:

- beat
- beatDuration
- beatStr
- beatStrength
- hasEditorialInformation
- hasStyleInformation
- measureNumber

Read-only properties inherited from ProtoM21Object:

- classSet
- classes

TempoText read/write properties

TempoText.text¶

Get or set the text as a string.

```
>>> import music21
>>> tm = music21.tempo.TempoText('adagio')
>>> tm.text
'adagio'
>>> tm.getTextExpression()
<music21.expressions.TextExpression 'adagio'>
```

Read/write properties inherited from Music21Object:

- activeSite
- derivation
- duration
- editorial
- id
- offset
- priority
- quarterLength
- seconds
- style

TempoText methods

TempoText.applyTextFormatting(*te=None*, *numberImplicit=False*)¶

Apply the default text formatting to the text expression version of this tempo mark

TempoText.getMetronomeMark()¶

Return a MetronomeMark object that is configured from this objects Text.

```
>>> tt = tempo.TempoText('slow')
>>> mm = tt.getMetronomeMark()
>>> mm.number
56
```

TempoText.getTextExpression(*numberImplicit=False*)¶

Return a TextExpression object for this text.

What is this a deepcopy and not the actual one?

TempoText.isCommonTempoText(*value=None*)¶

Return True or False if the supplied text seems like a plausible Tempo indications be used for this TempoText.

```
>>> tt = tempo.TempoText('adagio')
>>> tt.isCommonTempoText()
True
```

```
>>> tt = tempo.TempoText('Largo e piano')
>>> tt.isCommonTempoText()
True
```

```
>>> tt = tempo.TempoText('undulating')
>>> tt.isCommonTempoText()
False
```

TempoText.setTextExpression(*value*)¶

Given a TextExpression, set it in this object.

Methods inherited from `TempoIndication`:

- `getPreviousMetronomeMark()`
- `getSoundingMetronomeMark()`

Methods inherited from `Music21Object`:

- `__eq__()`
- `clearCache()`
- `containerHierarchy()`
- `contextSites()`
- `getAllContextsByClass()`
- `getContextByClass()`
- `getOffsetBySite()`
- `getOffsetInHierarchy()`
- `getSpannerSites()`
- `informSites()`
- `mergeAttributes()`
- `next()`
- `previous()`
- `purgeLocations()`
- `purgeOrphans()`
- `setOffsetBySite()`
- `show()`
- `sortTuple()`
- `splitAtDurations()`
- `splitAtQuarterLength()`
- `splitByQuarterLengths()`
- `write()`

Methods inherited from `ProtoM21Object`:

- `isClassOrSubclass()`

`TempoText` instance variables

Instance variables inherited from `Music21Object`:

- classSortOrder
- groups
- isStream
- sites

MetricModulation¶

class music21.tempo.MetricModulation(***keywords*)¶

A class for representing the relationship between two MetronomeMarks. Generally this relationship is one of equality, where the number is maintained but the referent that number is applied to each change.

The basic definition of a MetricModulation is given by supplying two MetronomeMarks, one for the oldMetronome, the other for the newMetronome. High level properties, oldReferent and newReferent, and convenience methods permit only setting the referent.

The *classicalStyle* attribute determines of the first MetronomeMark describes the new tempo, not the old (the reverse of expected usage).

The *maintainBeat* attribute determines if, after an equality statement, the beat is maintained. This is relevant for moving from 3/4 to 6/8, for example.

```
>>> s = stream.Stream()
>>> mm1 = tempo.MetronomeMark(number=60)
>>> s.append(mm1)
>>> s.repeatAppend(note.Note(quarterLength=1), 2)
>>> s.repeatAppend(note.Note(quarterLength=0.5), 4)
```

```
>>> mmod1 = tempo.MetricModulation()
>>> mmod1.oldReferent = 0.5  # can use Duration objects
>>> mmod1.newReferent = 'quarter'  # can use Duration objects
>>> s.append(mmod1)
>>> mmod1.updateByContext()  # get number from last MetronomeMark on Stream
>>> mmod1.newMetronome
<music21.tempo.MetronomeMark animato Quarter=120>
```

```
>>> s.append(note.Note())
>>> s.repeatAppend(note.Note(quarterLength=1.5), 2)
```

```
>>> mmod2 = tempo.MetricModulation()
>>> s.append(mmod2)  # if the obj is added to Stream, can set referents
>>> mmod2.oldReferent = 1.5  # will get number from previous MetronomeMark
>>> mmod2.newReferent = 'quarter'
>>> mmod2.newMetronome
<music21.tempo.MetronomeMark animato Quarter=80>
```

Note that an initial metric modulation can set old and new referents and get None as tempo numbers:

```
>>> mmod3 = tempo.MetricModulation()
>>> mmod3.oldReferent = 'half'
>>> mmod3.newReferent = '16th'
>>> mmod3
<music21.tempo.MetricModulation
    <music21.tempo.MetronomeMark
        Half=None>=<music21.tempo.MetronomeMark 16th=None>>
```

test w/ more sane referents that either the old or the new can change without a tempo number

```
>>> mmod3.oldReferent = 'quarter'
>>> mmod3.newReferent = 'eighth'
>>> mmod3
<music21.tempo.MetricModulation
    <music21.tempo.MetronomeMark
```

```
        Quarter=None>=<music21.tempo.MetronomeMark Eighth=None>>
>>> mmod3.oldMetronome
<music21.tempo.MetronomeMark Quarter=None>
>>> mmod3.oldMetronome.number = 60
```

New number automatically updates:

```
>>> mmod3
<music21.tempo.MetricModulation
    <music21.tempo.MetronomeMark larghetto
        Quarter=60>=<music21.tempo.MetronomeMark larghetto Eighth=60>>
```

MetricModulation bases

- TempoIndication
- Music21Object
- ProtoM21Object

MetricModulation read-only properties

MetricModulation.number¶

Get and the number of the MetricModulation, or the number assigned to the new MetronomeMark.

```
>>> s = stream.Stream()
>>> mm1 = tempo.MetronomeMark(number=60)
>>> s.append(mm1)
>>> s.repeatAppend(note.Note(quarterLength=1), 2)
>>> s.repeatAppend(note.Note(quarterLength=0.5), 4)

>>> mmod1 = tempo.MetricModulation()
>>> mmod1.oldReferent = 0.5  # can use Duration objects
>>> mmod1.newReferent = 'quarter'
>>> s.append(mmod1)
>>> mmod1.updateByContext()
>>> mmod1.newMetronome
<music21.tempo.MetronomeMark animato Quarter=120>
>>> mmod1.number
120
```

Read-only properties inherited from Music21Object:

- beat
- beatDuration
- beatStr
- beatStrength
- hasEditorialInformation
- hasStyleInformation
- measureNumber

Read-only properties inherited from ProtoM21Object:

- classSet
- classes

MetricModulation read/write properties

MetricModulation.newMetronome¶

Get or set the right MetronomeMark object for the new, or following value.

```
>>> mm1 = tempo.MetronomeMark(number=60, referent=1)
>>> mm1
<music21.tempo.MetronomeMark larghetto Quarter=60>
>>> mmod1 = tempo.MetricModulation()
>>> mmod1.newMetronome = mm1
>>> mmod1.newMetronome = 'junk'
Traceback (most recent call last):
music21.tempo.MetricModulationException: newMetronome property must be
    set with a MetronomeMark instance
```

MetricModulation.newReferent¶

Get or set the referent of the new MetronomeMark.

```
>>> mm1 = tempo.MetronomeMark(number=60, referent=1)
>>> mmod1 = tempo.MetricModulation()
>>> mmod1.newMetronome = mm1
>>> mmod1.newMetronome
<music21.tempo.MetronomeMark larghetto Quarter=60>
>>> mmod1.newReferent = 0.25
>>> mmod1.newMetronome
<music21.tempo.MetronomeMark larghetto 16th=240>
```

MetricModulation.oldMetronome¶

Get or set the left `MetronomeMark` object for the old, or previous value.

```
>>> mm1 = tempo.MetronomeMark(number=60, referent=1)
>>> mm1
<music21.tempo.MetronomeMark larghetto Quarter=60>
>>> mmod1 = tempo.MetricModulation()
>>> mmod1.oldMetronome = mm1
```

Note that we do need to have a proper MetronomeMark instance to figure this out:

```
>>> mmod1.oldMetronome = 'junk'
Traceback (most recent call last):
music21.tempo.MetricModulationException: oldMetronome property
    must be set with a MetronomeMark instance
```

MetricModulation.oldReferent¶

Get or set the referent of the old MetronomeMark.

```
>>> mm1 = tempo.MetronomeMark(number=60, referent=1)
>>> mmod1 = tempo.MetricModulation()
>>> mmod1.oldMetronome = mm1
>>> mmod1.oldMetronome
<music21.tempo.MetronomeMark larghetto Quarter=60>
>>> mmod1.oldReferent = 0.25
>>> mmod1.oldMetronome
<music21.tempo.MetronomeMark larghetto 16th=240>
```

Read/write properties inherited from `Music21Object`:

- `activeSite`
- `derivation`
- `duration`
- `editorial`
- `id`
- `offset`
- `priority`
- `quarterLength`
- `seconds`
- `style`

MetricModulation methods

MetricModulation.setEqualityByReferent(*side=None*, *referent=1.0*)¶

Set the other side of the metric modulation to an equality; side can be specified, or if one side is None, that side will be set.

```
>>> mm1 = tempo.MetronomeMark(number=60, referent=1)
>>> mmod1 = tempo.MetricModulation()
>>> mmod1.newMetronome = mm1
>>> mmod1.setEqualityByReferent(None, 2)
>>> mmod1
<music21.tempo.MetricModulation
     <music21.tempo.MetronomeMark larghetto
           Half=30>=<music21.tempo.MetronomeMark larghetto Quarter=60>>
```

MetricModulation.setOtherByReferent(*side: str | None = None*, *referent: str | int | float = 1.0*)¶

Set the other side of the metric modulation not based on equality, but on a direct translation of the tempo value.

referent can be a string type or an int/float quarter length

MetricModulation.updateByContext()¶

Update this metric modulation based on the context, or the surrounding MetronomeMarks or MetricModulations. The object needs to reside in a Stream for this to be effective.

Methods inherited from TempoIndication:

- getPreviousMetronomeMark()
- getSoundingMetronomeMark()

Methods inherited from Music21Object:

- __eq__()
- clearCache()
- containerHierarchy()
- contextSites()
- getAllContextsByClass()
- getContextByClass()
- getOffsetBySite()
- getOffsetInHierarchy()
- getSpannerSites()
- informSites()
- mergeAttributes()
- next()
- previous()
- purgeLocations()
- purgeOrphans()
- setOffsetBySite()
- show()
- sortTuple()
- splitAtDurations()
- splitAtQuarterLength()
- splitByQuarterLengths()
- write()

Methods inherited from ProtoM21Object:

- isClassOrSubclass()

MetricModulation instance variables

Instance variables inherited from `Music21Object`:

- `classSortOrder`
- `groups`
- `isStream`
- `sites`

TempoIndication¶

class music21.tempo.TempoIndication(*id: str* | *int* | *None = None*, *groups: Groups* | *None = None*, *sites: Sites* | *None = None*, *duration: Duration* | *None = None*, *activeSite: stream.Stream* | *None = None*, *style: Style* | *None = None*, *editorial: Editorial* | *None = None*, *offset: OffsetQL = 0.0*, *quarterLength: OffsetQLIn* | *None = None*, ***keywords*)¶

> A generic base class for all tempo indications to inherit. Can be used to filter out all types of tempo indications.

`TempoIndication` bases

- `Music21Object`
- `ProtoM21Object`

`TempoIndication` read-only properties

Read-only properties inherited from `Music21Object`:

- `beat`
- `beatDuration`
- `beatStr`
- `beatStrength`
- `hasEditorialInformation`
- `hasStyleInformation`
- `measureNumber`

Read-only properties inherited from `ProtoM21Object`:

- `classSet`
- `classes`

`TempoIndication` read/write properties

Read/write properties inherited from `Music21Object`:

- `activeSite`
- `derivation`
- `duration`
- `editorial`
- `id`
- `offset`
- `priority`
- `quarterLength`
- `seconds`
- `style`

`TempoIndication` methods

TempoIndication.getPreviousMetronomeMark()¶

Do activeSite and context searches to try to find the last relevant MetronomeMark or MetricModulation object. If a MetricModulation mark is found, return the new MetronomeMark, or the last relevant.

```
>>> s = stream.Stream()
>>> s.insert(0, tempo.MetronomeMark(number=120))
>>> mm1 = tempo.MetronomeMark(number=90)
>>> s.insert(20, mm1)
>>> mm1.getPreviousMetronomeMark()
<music21.tempo.MetronomeMark animato Quarter=120>
```

TempoIndication.getSoundingMetronomeMark(*found=None*)¶

Get the appropriate MetronomeMark from any sort of TempoIndication, regardless of class.

Methods inherited from `Music21Object`:

- `__eq__()`
- `clearCache()`
- `containerHierarchy()`
- `contextSites()`
- `getAllContextsByClass()`
- `getContextByClass()`
- `getOffsetBySite()`
- `getOffsetInHierarchy()`
- `getSpannerSites()`
- `informSites()`
- `mergeAttributes()`
- `next()`
- `previous()`
- `purgeLocations()`
- `purgeOrphans()`
- `setOffsetBySite()`
- `show()`
- `sortTuple()`
- `splitAtDurations()`
- `splitAtQuarterLength()`
- `splitByQuarterLengths()`
- `write()`

Methods inherited from `ProtoM21Object`:

- `isClassOrSubclass()`

`TempoIndication` instance variables

Instance variables inherited from `Music21Object`:

- `classSortOrder`
- `groups`
- `isStream`
- `sites`

AccelerandoSpanner¶

class music21.tempo.AccelerandoSpanner(**spannedElements: Music21Object | Sequence[Music21Object], **keywords*)¶

Spanner representing a speeding up.

`AccelerandoSpanner` bases

- TempoChangeSpanner
- Spanner
- Music21Object
- ProtoM21Object

AccelerandoSpanner read-only properties

Read-only properties inherited from Music21Object:

- beat
- beatDuration
- beatStr
- beatStrength
- hasEditorialInformation
- hasStyleInformation
- measureNumber

Read-only properties inherited from ProtoM21Object:

- classSet
- classes

AccelerandoSpanner read/write properties

Read/write properties inherited from Music21Object:

- activeSite
- derivation
- duration
- editorial
- id
- offset
- priority
- quarterLength
- seconds
- style

AccelerandoSpanner methods

Methods inherited from Spanner:

- __getitem__()
- addSpannedElements()
- fill()
- getFirst()
- getLast()
- getSpannedElementIds()
- getSpannedElements()
- getSpannedElementsByClass()
- hasSpannedElement()
- isFirst()
- isLast()
- purgeLocations()
- purgeOrphans()
- replaceSpannedElement()

Methods inherited from Music21Object:

- __eq__()
- clearCache()
- containerHierarchy()
- contextSites()
- getAllContextsByClass()
- getContextByClass()
- getOffsetBySite()
- getOffsetInHierarchy()
- getSpannerSites()
- informSites()
- mergeAttributes()
- next()
- previous()
- setOffsetBySite()
- show()
- sortTuple()
- splitAtDurations()
- splitAtQuarterLength()
- splitByQuarterLengths()
- write()

Methods inherited from ProtoM21Object:

- isClassOrSubclass()

AccelerandoSpanner instance variables

Instance variables inherited from Music21Object:

- classSortOrder
- groups
- isStream
- sites

RitardandoSpanner¶

class music21.tempo.RitardandoSpanner(**spannedElements: Music21Object* | *Sequence[Music21Object]*, ***keywords*)¶

Spanner representing a slowing down.

RitardandoSpanner bases

- TempoChangeSpanner
- Spanner
- Music21Object
- ProtoM21Object

RitardandoSpanner read-only properties

Read-only properties inherited from Music21Object:

- beat
- beatDuration
- beatStr
- beatStrength
- hasEditorialInformation
- hasStyleInformation
- measureNumber

Read-only properties inherited from ProtoM21Object:

- classSet
- classes

RitardandoSpanner read/write properties

Read/write properties inherited from Music21Object:

- activeSite
- derivation
- duration
- editorial
- id
- offset
- priority
- quarterLength
- seconds
- style

RitardandoSpanner methods

Methods inherited from Spanner:

- __getitem__()
- addSpannedElements()
- fill()
- getFirst()
- getLast()
- getSpannedElementIds()
- getSpannedElements()
- getSpannedElementsByClass()
- hasSpannedElement()
- isFirst()
- isLast()
- purgeLocations()
- purgeOrphans()
- replaceSpannedElement()

Methods inherited from Music21Object:

- __eq__()
- clearCache()
- containerHierarchy()
- contextSites()
- getAllContextsByClass()
- getContextByClass()
- getOffsetBySite()
- getOffsetInHierarchy()
- getSpannerSites()
- informSites()
- mergeAttributes()
- next()
- previous()
- setOffsetBySite()
- show()
- sortTuple()
- splitAtDurations()
- splitAtQuarterLength()
- splitByQuarterLengths()
- write()

Methods inherited from ProtoM21Object:

- isClassOrSubclass()

RitardandoSpanner instance variables

Instance variables inherited from Music21Object:

- classSortOrder
- groups
- isStream
- sites

TempoChangeSpanner¶

class music21.tempo.TempoChangeSpanner(**spannedElements: Music21Object* | *Sequence[Music21Object]*, ***keywords*)¶

> Spanners showing tempo-change. They do nothing right now.

TempoChangeSpanner bases

- Spanner
- Music21Object
- ProtoM21Object

TempoChangeSpanner read-only properties

Read-only properties inherited from Music21Object:

- beat
- beatDuration
- beatStr
- beatStrength
- hasEditorialInformation
- hasStyleInformation
- measureNumber

Read-only properties inherited from ProtoM21Object:

- classSet
- classes

TempoChangeSpanner read/write properties

Read/write properties inherited from Music21Object:

- activeSite
- derivation
- duration
- editorial
- id
- offset
- priority
- quarterLength
- seconds
- style

TempoChangeSpanner methods

Methods inherited from Spanner:

- __getitem__()
- addSpannedElements()
- fill()
- getFirst()
- getLast()
- getSpannedElementIds()
- getSpannedElements()
- getSpannedElementsByClass()
- hasSpannedElement()
- isFirst()
- isLast()
- purgeLocations()
- purgeOrphans()
- replaceSpannedElement()

Methods inherited from Music21Object:

- __eq__()
- clearCache()
- containerHierarchy()
- contextSites()
- getAllContextsByClass()
- getContextByClass()
- getOffsetBySite()
- getOffsetInHierarchy()
- getSpannerSites()
- informSites()
- mergeAttributes()
- next()
- previous()
- setOffsetBySite()
- show()
- sortTuple()
- splitAtDurations()
- splitAtQuarterLength()
- splitByQuarterLengths()
- write()

Methods inherited from ProtoM21Object:

- isClassOrSubclass()

TempoChangeSpanner instance variables

Instance variables inherited from Music21Object:

- classSortOrder
- groups
- isStream
- sites

Functions¶

music21.tempo.interpolateElements(*element1*, *element2*, *sourceStream*, *destinationStream*, *autoAdd=True*)¶

> Assume that element1 and element2 are two elements in sourceStream and destinationStream with other elements (say eA, eB, eC) between them. For instance, element1 could be the downbeat at offset 10 in sourceStream (a Stream representing a score) and offset 20.5 in destinationStream (which might be a Stream representing the timing of notes in particular recording at approximately but not exactly qtr = 30). Element2 could be the following downbeat in 4/4, at offset 14 in source but offset 25.0 in the recording:

```
>>> sourceStream = stream.Stream()
>>> destinationStream = stream.Stream()
>>> element1 = note.Note('C4', type='quarter')
>>> element2 = note.Note('G4', type='quarter')
>>> sourceStream.insert(10, element1)
>>> destinationStream.insert(20.5, element1)
>>> sourceStream.insert(14, element2)
>>> destinationStream.insert(25.0, element2)
```

Suppose eA, eB, and eC are three quarter notes that lie between element1 and element2 in sourceStream and destinationStream, as in:

```
>>> eA = note.Note('D4', type='quarter')
>>> eB = note.Note('E4', type='quarter')
>>> eC = note.Note('F4', type='quarter')
>>> sourceStream.insert(11, eA)
>>> sourceStream.insert(12, eB)
>>> sourceStream.insert(13, eC)
>>> destinationStream.append([eA, eB, eC])  # not needed if autoAdd were true
```

then running this function will cause eA, eB, and eC to have offsets 21.625, 22.75, and 23.875 respectively in destinationStream:

```
>>> tempo.interpolateElements(element1, element2,
...         sourceStream, destinationStream, autoAdd=False)
>>> for el in [eA, eB, eC]:
...    print(el.getOffsetBySite(destinationStream))
21.625
22.75
23.875
```

if the elements between element1 and element2 do not yet appear in destinationStream, they are automatically added unless autoAdd is False.

(with the default autoAdd, elements are automatically added to new streams):

```
>>> destStream2 = stream.Stream()
>>> destStream2.insert(10.1, element1)
>>> destStream2.insert(50.5, element2)
>>> tempo.interpolateElements(element1, element2, sourceStream, destStream2)
>>> for el in [eA, eB, eC]:
...    print('%.1f' % (el.getOffsetBySite(destStream2),))
20.2
30.3
40.4
```

(unless autoAdd is set to False, in which case a Tempo Exception arises:)

```
>>> destStream3 = stream.Stream()
>>> destStream3.insert(100, element1)
>>> destStream3.insert(500, element2)
>>> eA.id = 'blah'
>>> tempo.interpolateElements(element1, element2, sourceStream, destStream3, autoAdd=F
Traceback (most recent call last):
music21.tempo.TempoException: Could not find element <music21.note.Note D> with id ...
```

music21.tempo.convertTempoByReferent(*numberSrc: int | float*, *quarterLengthBeatSrc: int | float*, *quarterLengthBeatDst=1.0*) → float¶

Convert between equivalent tempi, where the speed stays the same but the beat referent and number change.

60 bpm at quarter, going to half

```
>>> tempo.convertTempoByReferent(60, 1, 2)
30.0
```

60 bpm at quarter, going to 16th

```
>>> tempo.convertTempoByReferent(60, 1, 0.25)
240.0
```

60 at dotted quarter, get quarter

```
>>> tempo.convertTempoByReferent(60, 1.5, 1)
90.0
```

60 at dotted quarter, get half

```
>>> tempo.convertTempoByReferent(60, 1.5, 2)
45.0
```

60 at dotted quarter, get trip

```
>>> tempo.convertTempoByReferent(60, 1.5, 1/3)
270.0
```

A Fraction instance can also be used:

```
>>> tempo.convertTempoByReferent(60, 1.5, common.opFrac(1/3))
270.0
```

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.tempo

© Copyright 2006-2023 Michael Scott Asato Cuthbert. Last updated on Jun 12, 2023.

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.pitch

Previous topic

music21.percussion

Next topic

music21.prebase

Table of Contents

- music21.pitch
  - Pitch
    - `Pitch`
      - `Pitch.alter`
      - `Pitch.french`
      - `Pitch.fullName`
      - `Pitch.german`
      - `Pitch.implicitOctave`
      - `Pitch.italian`
      - `Pitch.spanish`
      - `Pitch.unicodeName`
      - `Pitch.unicodeNameWithOctave`
      - `Pitch.accidental`
      - `Pitch.diatonicNoteNum`
      - `Pitch.freq440`
      - `Pitch.frequency`
      - `Pitch.groups`
      - `Pitch.microtone`
      - `Pitch.midi`
      - `Pitch.name`
      - `Pitch.nameWithOctave`
      - `Pitch.octave`
      - `Pitch.pitchClass`
      - `Pitch.pitchClassString`
      - `Pitch.ps`
      - `Pitch.step`
      - `Pitch.__eq__()`
      - `Pitch.convertMicrotonesToQuarterTones()`
      - `Pitch.convertQuarterTonesToMicrotones()`
      - `Pitch.getAllCommonEnharmonics()`
      - `Pitch.getCentShiftFromMidi()`
      - `Pitch.getEnharmonic()`
      - `Pitch.getHarmonic()`
      - `Pitch.getHigherEnharmonic()`
      - `Pitch.getLowerEnharmonic()`
      - `Pitch.getStringHarmonic()`
      - `Pitch.harmonicAndFundamentalFromPitch()`

        - Pitch.harmonicAndFundamentalStringFromPitch()
        - Pitch.harmonicFromFundamental()
        - Pitch.harmonicString()
        - Pitch.informClient()
        - Pitch.isEnharmonic()
        - Pitch.isTwelveTone()
        - Pitch.simplifyEnharmonic()
        - Pitch.transpose()
        - Pitch.transposeAboveTarget()
        - Pitch.transposeBelowTarget()
        - Pitch.updateAccidentalDisplay()
        - Pitch.spellingIsInferred
- Accidental
    - Accidental
        - Accidental.fullName
        - Accidental.unicode
        - Accidental.alter
        - Accidental.displayStatus
        - Accidental.displayType
        - Accidental.modifier
        - Accidental.name
        - Accidental.__eq__()
        - Accidental.inheritDisplay()
        - Accidental.isTwelveTone()
        - Accidental.listNames()
        - Accidental.set()
        - Accidental.setAttributeIndependently()
        - Accidental.displayLocation
        - Accidental.displaySize
        - Accidental.displayStyle
- Microtone
    - Microtone
        - Microtone.alter
        - Microtone.cents
        - Microtone.harmonicShift
        - Microtone.__eq__()
- Functions
    - convertPitchClassToStr()
    - isValidAccidentalName()
    - simplifyMultipleEnharmonics()
    - standardizeAccidentalName()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.lily.lilyObjects

Next topic

music21.mei.base

Table of Contents

- music21.lily.translate
  - LilypondConverter
    - `LilypondConverter`
      - `LilypondConverter.appendBeamCode()`
      - `LilypondConverter.appendContextFromChord()`
      - `LilypondConverter.appendContextFromNoteOrRest()`
      - `LilypondConverter.appendContextFromVariant()`
      - `LilypondConverter.appendM21ObjectToContext()`
      - `LilypondConverter.appendObjectsToContextFromStream()`
      - `LilypondConverter.appendStemCode()`
      - `LilypondConverter.baseNameFromPitch()`
      - `LilypondConverter.closeMeasure()`
      - `LilypondConverter.createPDF()`
      - `LilypondConverter.createPNG()`
      - `LilypondConverter.createSVG()`
      - `LilypondConverter.findLilyExec()`
      - `LilypondConverter.getLySpacersFromStream()`
      - `LilypondConverter.getSchemeForPadding()`
      - `LilypondConverter.loadFromMusic21Object()`
      - `LilypondConverter.loadObjectFromOpus()`
      - `LilypondConverter.loadObjectFromScore()`
      - `LilypondConverter.lyEmbeddedScmFromClef()`
      - `LilypondConverter.lyEmbeddedScmFromKeySignature()`
      - `LilypondConverter.lyEmbeddedScmFromTimeSignature()`
      - `LilypondConverter.lyGroupedMusicListFromScoreWithParts()`
      - `LilypondConverter.lyLyricElementFromM21Lyric()`
      - `LilypondConverter.lyMultipliedDurationFromDuration()`
      - `LilypondConverter.lyNewLyricsFromStream()`
      - `LilypondConverter.lyOssiaMusicFromVariant()`
      - `LilypondConverter.lyPartsAndOssiaInitFromScore()`
      - `LilypondConverter.lyPitchFromPitch()`
      - `LilypondConverter.lyPrefixCompositeMusicFromRelatedVariants()`
      - `LilypondConverter.lyPrefixCompositeMusicFromStream()`
      - `LilypondConverter.lyPrefixCompositeMusicFromVariant()`
      - `LilypondConverter.lyScoreBlockFromScore()`
      - `LilypondConverter.lySequentialMusicFromStream()`
      - `LilypondConverter.lySimpleMusicFromChord()`

    - `LilypondConverter.lySimpleMusicFromNoteOrRest()`
    - `LilypondConverter.newContext()`
    - `LilypondConverter.octaveCharactersFromPitch()`
    - `LilypondConverter.postEventsFromObject()`
    - `LilypondConverter.restoreContext()`
    - `LilypondConverter.runThroughLily()`
    - `LilypondConverter.setContextForTimeFraction()`
    - `LilypondConverter.setContextForTupletStart()`
    - `LilypondConverter.setContextForTupletStop()`
    - `LilypondConverter.setHeaderFromMetadata()`
    - `LilypondConverter.setupTools()`
    - `LilypondConverter.showPDF()`
    - `LilypondConverter.showPNG()`
    - `LilypondConverter.showSVG()`
    - `LilypondConverter.textFromMusic21Object()`
    - `LilypondConverter.writeLyFile()`
  - Functions
    - `makeLettersOnlyId()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.metadata

Next topic

music21.metadata.caching

Table of Contents

- music21.metadata.bundles
  - MetadataBundle
    - `MetadataBundle`
      - `MetadataBundle.filePath`
      - `MetadataBundle.name`
      - `MetadataBundle.corpus`
      - `MetadataBundle.__eq__()`
      - `MetadataBundle.__getitem__()`
      - `MetadataBundle.addFromPaths()`
      - `MetadataBundle.clear()`
      - `MetadataBundle.corpusPathToKey()`
      - `MetadataBundle.delete()`
      - `MetadataBundle.difference()`
      - `MetadataBundle.intersection()`
      - `MetadataBundle.isdisjoint()`
      - `MetadataBundle.issubset()`
      - `MetadataBundle.issuperset()`
      - `MetadataBundle.listSearchFields()`
      - `MetadataBundle.read()`
      - `MetadataBundle.search()`
      - `MetadataBundle.symmetric_difference()`
      - `MetadataBundle.union()`
      - `MetadataBundle.validate()`
      - `MetadataBundle.write()`
  - MetadataEntry
    - `MetadataEntry`
      - `MetadataEntry.corpusName`
      - `MetadataEntry.corpusPath`
      - `MetadataEntry.metadata`
      - `MetadataEntry.number`
      - `MetadataEntry.sourcePath`
      - `MetadataEntry.parse()`
      - `MetadataEntry.search()`
      - `MetadataEntry.show()`
  - Functions
    - `demo_bundle()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.search.lyrics

Previous topic

music21.search.base

Next topic

music21.search.segment

Table of Contents

- music21.search.lyrics
  - LyricSearcher
    - `LyricSearcher`
      - `LyricSearcher.indexText`
      - `LyricSearcher.indexTuples`
      - `LyricSearcher.index()`
      - `LyricSearcher.search()`
  - IndexedLyric
    - `IndexedLyric`
      - `IndexedLyric.modify()`
      - `IndexedLyric.absoluteEnd`
      - `IndexedLyric.absoluteStart`
      - `IndexedLyric.el`
      - `IndexedLyric.end`
      - `IndexedLyric.identifier`
      - `IndexedLyric.lyric`
      - `IndexedLyric.measure`
      - `IndexedLyric.start`
      - `IndexedLyric.text`
  - SearchMatch
    - `SearchMatch`
      - `SearchMatch.els`
      - `SearchMatch.identifier`
      - `SearchMatch.indices`
      - `SearchMatch.mEnd`
      - `SearchMatch.mStart`
      - `SearchMatch.matchText`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation

- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.midi

Next topic

music21.midi.realtime

Table of Contents

- music21.midi.percussion
  - PercussionMapper
    - `PercussionMapper`
      - `PercussionMapper.midiInstrumentToPitch()`
      - `PercussionMapper.midiPitchToInstrument()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.roman

Next topic

music21.romanText.rtObjects

Table of Contents

- music21.romanText.clercqTemperley
  - CTSong
    - `CTSong`
      - `CTSong.comments`
      - `CTSong.homeKey`
      - `CTSong.homeTimeSig`
      - `CTSong.rules`
      - `CTSong.title`
      - `CTSong.parse()`
      - `CTSong.toPart()`
      - `CTSong.toScore()`
      - `CTSong.year`
  - CTRule
    - `CTRule`
      - `CTRule.comment`
      - `CTRule.sectionName`
      - `CTRule.LHS`
      - `CTRule.musicText`
      - `CTRule.parent`
      - `CTRule.addOptionalTieAndLyrics()`
      - `CTRule.expand()`
      - `CTRule.expandExpansionContent()`
      - `CTRule.expandSimpleContent()`
      - `CTRule.fixupChordAtom()`
      - `CTRule.insertKsTs()`
      - `CTRule.isSame()`
      - `CTRule.text`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation

- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.romanText.clercqTemperley

Next topic

music21.romanText.translate

Table of Contents

- music21.romanText.rtObjects
  - RTAnalyticKey
    - `RTAnalyticKey`
  - RTAtom
    - `RTAtom`
      - `RTAtom.isAtom()`
  - RTBeat
    - `RTBeat`
      - `RTBeat.getBeatFloatOrFrac()`
      - `RTBeat.getOffset()`
  - RTChord
    - `RTChord`
  - RTCloseParens
    - `RTCloseParens`
  - RTEllisonStart
    - `RTEllisonStart`
  - RTEllisonStop
    - `RTEllisonStop`
  - RTFile
    - `RTFile`
      - `RTFile.close()`
      - `RTFile.open()`
      - `RTFile.openFileLike()`
      - `RTFile.read()`
      - `RTFile.readstr()`
  - RTHandler
    - `RTHandler`
      - `RTHandler.tokens`
      - `RTHandler.definesMovement()`
      - `RTHandler.definesMovements()`
      - `RTHandler.process()`
      - `RTHandler.splitAtHeader()`
      - `RTHandler.splitByMovement()`
      - `RTHandler.tokenize()`
      - `RTHandler.tokenizeAtoms()`
      - `RTHandler.tokenizeBody()`
      - `RTHandler.tokenizeHeader()`

- RTKey
    - RTKey
- RTKeySignature
    - RTKeySignature
        - RTKeySignature.getKeySignature()
- RTKeyTypeAtom
    - RTKeyTypeAtom
        - RTKeyTypeAtom.getKey()
        - RTKeyTypeAtom.getKeySignature()
- RTMeasure
    - RTMeasure
        - RTMeasure.getCopyTarget()
        - RTMeasure.isMeasure()
- RTNoChord
    - RTNoChord
- RTOpenParens
    - RTOpenParens
- RTOptionalKeyClose
    - RTOptionalKeyClose
        - RTOptionalKeyClose.getKey()
- RTOptionalKeyOpen
    - RTOptionalKeyOpen
        - RTOptionalKeyOpen.getKey()
- RTPhraseBoundary
    - RTPhraseBoundary
- RTPhraseMarker
    - RTPhraseMarker
- RTRepeat
    - RTRepeat
- RTRepeatStart
    - RTRepeatStart
- RTRepeatStop
    - RTRepeatStop
- RTTagged
    - RTTagged
        - RTTagged.isAnalyst()
        - RTTagged.isComposer()
        - RTTagged.isForm()
        - RTTagged.isKeySignature()
        - RTTagged.isMovement()
        - RTTagged.isNote()
        - RTTagged.isPedal()
        - RTTagged.isPiece()
        - RTTagged.isProofreader()
        - RTTagged.isSeventhMinor()
        - RTTagged.isSixthMinor()
        - RTTagged.isTimeSignature()
        - RTTagged.isTitle()
        - RTTagged.isVersion()
        - RTTagged.isWork()
- RTToken
    - RTToken
        - RTToken.isAnalyst()
        - RTToken.isAtom()
        - RTToken.isComposer()
        - RTToken.isForm()
        - RTToken.isKeySignature()
        - RTToken.isMeasure()

    - `RTToken.isMovement()`
    - `RTToken.isNote()`
    - `RTToken.isPedal()`
    - `RTToken.isPiece()`
    - `RTToken.isProofreader()`
    - `RTToken.isTimeSignature()`
    - `RTToken.isTitle()`
    - `RTToken.isVersion()`
    - `RTToken.isWork()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.serial

Previous topic

music21.search.serial

Next topic

music21.sieve

Table of Contents

- music21.serial
  - ToneRow
    - `ToneRow`
      - `ToneRow.findOriginalCenteredTransformations()`
      - `ToneRow.findZeroCenteredTransformations()`
      - `ToneRow.getIntervalsAsString()`
      - `ToneRow.isSameRow()`
      - `ToneRow.isTwelveToneRow()`
      - `ToneRow.makeTwelveToneRow()`
      - `ToneRow.noteNames()`
      - `ToneRow.originalCenteredTransformation()`
      - `ToneRow.pitchClasses()`
      - `ToneRow.zeroCenteredTransformation()`
      - `ToneRow.row`
  - TwelveToneRow
    - `TwelveToneRow`
      - `TwelveToneRow.areCombinatorial()`
      - `TwelveToneRow.findHistorical()`
      - `TwelveToneRow.findTransformedHistorical()`
      - `TwelveToneRow.getLinkClassification()`
      - `TwelveToneRow.isAllInterval()`
      - `TwelveToneRow.isLinkChord()`
      - `TwelveToneRow.matrix()`
  - HistoricalTwelveToneRow
    - `HistoricalTwelveToneRow`
      - `HistoricalTwelveToneRow.mergeAttributes()`
      - `HistoricalTwelveToneRow.composer`
      - `HistoricalTwelveToneRow.opus`
      - `HistoricalTwelveToneRow.title`
  - TwelveToneMatrix
    - `TwelveToneMatrix`
  - Functions
    - `pcToToneRow()`
    - `rowToMatrix()`
    - `getHistoricalRowByName()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.prebase

Next topic

music21.roman

Table of Contents

- music21.repeat
  - RepeatExpression
    - `RepeatExpression`
      - `RepeatExpression.applyTextFormatting()`
      - `RepeatExpression.getText()`
      - `RepeatExpression.getTextExpression()`
      - `RepeatExpression.isValidText()`
      - `RepeatExpression.setText()`
      - `RepeatExpression.setTextExpression()`
  - RepeatExpressionMarker
    - `RepeatExpressionMarker`
  - Coda
    - `Coda`
  - Segno
    - `Segno`
  - Fine
    - `Fine`
  - RepeatExpressionCommand
    - `RepeatExpressionCommand`
  - DaCapo
    - `DaCapo`
  - DaCapoAlFine
    - `DaCapoAlFine`
  - DaCapoAlCoda
    - `DaCapoAlCoda`
  - AlSegno
    - `AlSegno`
  - DalSegno
    - `DalSegno`
  - DalSegnoAlFine
    - `DalSegnoAlFine`
  - DalSegnoAlCoda
    - `DalSegnoAlCoda`
  - RepeatFinder
    - `RepeatFinder`
      - `RepeatFinder.getMeasureSimilarityList()`
      - `RepeatFinder.getQuarterLengthOfPickupMeasure()`

      - RepeatFinder.getSimilarMeasureGroups()
      - RepeatFinder.hasPickup()
      - RepeatFinder.simplify()
      - RepeatFinder.correctMeasureNumbers
      - RepeatFinder.defaultHash
      - RepeatFinder.s
  - Expander
    - Expander
      - Expander.findInnermostRepeatIndices()
      - Expander.getRepeatExpressionIndex()
      - Expander.isExpandable()
      - Expander.measureMap()
      - Expander.process()
      - Expander.processInnermostRepeatBars()
      - Expander.repeatBarsAreCoherent()
  - RepeatMark
    - RepeatMark
  - Functions
    - deleteMeasures()
    - insertRepeat()
    - insertRepeatEnding()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.musedata

Next topic

music21.musedata.translate

Table of Contents

- music21.musedata.base40
  - BaseN
    - `BaseN`
      - `BaseN.generateLetters()`
  - Functions
    - `base40ActualInterval()`
    - `base40DeltaToInterval()`
    - `base40Interval()`
    - `base40ToPitch()`
    - `pitchToBase40()`
    - `quickEnharmonicString()`
    - `quickHigherEnharmonicString()`
    - `quickLowerEnharmonicString()`

Table of Contents

- About *music21*
- User's Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Quick search

This Page

- Show Source

music21.musedata.base40¶

Implementation of Walter Hewlett's base40 system for musical intervals. (See Hewlett and Ann K. Blombach 1989 article)

BaseN¶

class music21.musedata.base40.BaseN(*order=2*)¶

`BaseN` methods

BaseN.generateLetters()¶

Functions¶

music21.musedata.base40.base40ActualInterval(*base40NumA*, *base40NumB*)¶

> Calculates a music21 Interval between two Base40 pitch numbers, as calculated using the music21.interval module.
>
> Raises a Base40 Exception if (a) Either of the Base40 pitch numbers does not correspond to a pitch name or (b) If an unusual interval is encountered that can't be handled by music21.

```
>>> musedata.base40.base40ActualInterval(163, 191)
<music21.interval.Interval m6>
>>> musedata.base40.base40ActualInterval(186, 174)  # Descending M3
<music21.interval.Interval M-3>
>>> musedata.base40.base40ActualInterval(1, 5)
<music21.interval.Interval AAAA1>
>>> musedata.base40.base40ActualInterval(1, 3)
<music21.interval.Interval AA1>
>>> musedata.base40.base40ActualInterval(2, 6)
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Pitch name not assigned to this Base40 number
```

music21.musedata.base40.base40DeltaToInterval(*delta*)¶

> Returns a music21 Interval between two Base40 pitch numbers given the delta (difference) between them.

Raises a Base40 Exception if the interval is not handled by Base40. Base40 can only handle major, minor, perfect, augmented, and diminished intervals. Although not for certain, it seems that the engineers that designed this system assumed that other intervals (doubly augmented intervals, for instance) would be of a very rare occurrence, and extreme intervals which would trigger an incorrect answer (C– to C##, for instance, would return a diminished second, even though it’s a quadruplely augmented unison) just would not occur.

```
>>> musedata.base40.base40DeltaToInterval(4)
<music21.interval.Interval d2>
>>> musedata.base40.base40DeltaToInterval(11)
<music21.interval.Interval m3>
>>> musedata.base40.base40DeltaToInterval(23)
<music21.interval.Interval P5>
>>> musedata.base40.base40DeltaToInterval(-23)
<music21.interval.Interval P-5>
>>> musedata.base40.base40DeltaToInterval(52)
<music21.interval.Interval M10>
>>> musedata.base40.base40DeltaToInterval(-52)
<music21.interval.Interval M-10>
>>> musedata.base40.base40DeltaToInterval(77)
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Interval not handled by Base40 37
```

music21.musedata.base40.base40Interval(*base40NumA*, *base40NumB*)¶

Returns a music21 Interval between two base40 pitch numbers, using their delta (difference) as defined in Base40. The interval provided is without direction.

Raises a Base40 Exception if the delta doesn’t correspond to an interval in Base40, or if either base40 pitch number doesn’t correspond to a pitch name.

```
>>> musedata.base40.base40Interval(163, 191)
<music21.interval.Interval m6>

>>> musedata.base40.base40Interval(186, 174)      # Descending M3
<music21.interval.Interval M-3>
```

Base40 has limitations for intervals smaller than diminished or bigger than augmented.

```
>>> musedata.base40.base40Interval(1, 5)
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Base40 cannot compute interval between 1 and

>>> musedata.base40.base40Interval(1, 3)
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Interval not handled by Base40 2

>>> musedata.base40.base40Interval(2, 6)
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Pitch name not assigned to this Base40 number
    Interval does not exist

>>> musedata.base40.base40Interval(12, 6)
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Pitch name not assigned to these Base40 numbe
    12 and 6 Interval does not exist
```

music21.musedata.base40.base40ToPitch(*base40Num*)¶

Converts a Base40 pitch number into a music21 Pitch. The Base40 number is octave specific.

Raises a Base40 Exception if the Base40 pitch number given doesn’t have an associated pitch name. There is one unassigned number each time the interval between two letters is a whole step.

```
>>> musedata.base40.base40ToPitch(1)
<music21.pitch.Pitch C--1>
```

```
>>> musedata.base40.base40ToPitch(40)
<music21.pitch.Pitch B##1>
>>> musedata.base40.base40ToPitch(23)
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Pitch name not assigned to this Base40 number
>>> musedata.base40.base40ToPitch(186)
<music21.pitch.Pitch G5>
```

music21.musedata.base40.pitchToBase40(*pitchToConvert*)¶

Converts a pitch string or a music21 Pitch into a Base40 pitch number. The Base40 number is octave specific.

Raises a Base40 Exception if the pitch to convert is outside the set of pitches that Base40 can handle; for example, half flats and half sharps or triple flats and triple sharps.

```
>>> musedata.base40.pitchToBase40(pitch.Pitch('C--5'))
161
>>> musedata.base40.pitchToBase40('F##4')
142
>>> musedata.base40.pitchToBase40('F###4')
Traceback (most recent call last):
music21.musedata.base40.Base40Exception: Base40 cannot handle this pitch F###4
```

music21.musedata.base40.quickEnharmonicString(*nameStr*, *allowDoubleAccidentals=True*)¶

Takes a name of a string and returns a list of the quick lower and higher enharmonics, limited to double sharps and double flats (or to single sharps and single flats if allowDoubleAccidentals is False)

```
>>> musedata.base40.quickEnharmonicString('C')
['B#', 'D--']
>>> musedata.base40.quickEnharmonicString('C', allowDoubleAccidentals=False)
['B#']
>>> musedata.base40.quickEnharmonicString('G')
['F##', 'A--']
```

music21.musedata.base40.quickHigherEnharmonicString(*nameStr*, *allowDoubleAccidentals=True*)¶

Takes a name of a string and returns a list of the quick higher enharmonics, limited to double sharps and double flats (or to single sharps and single flats if allowDoubleAccidentals is False)

```
>>> musedata.base40.quickHigherEnharmonicString('F#')
['G-']
>>> musedata.base40.quickHigherEnharmonicString('C##')
['D', 'E--']
>>> musedata.base40.quickHigherEnharmonicString('C##', allowDoubleAccidentals=False)
['D']
>>> musedata.base40.quickHigherEnharmonicString('B#')
['C', 'D--']
```

music21.musedata.base40.quickLowerEnharmonicString(*nameStr*, *allowDoubleAccidentals=True*)¶

Takes a name of a string and returns a list of the quick lower enharmonics, limited to double sharps and double flats (or to single sharps and single flats if allowDoubleAccidentals is False)

```
>>> musedata.base40.quickLowerEnharmonicString('B-')
['A#']
>>> musedata.base40.quickLowerEnharmonicString('G-')
['F#', 'E##']
>>> musedata.base40.quickLowerEnharmonicString('G-', allowDoubleAccidentals=False)
['F#']
>>> musedata.base40.quickLowerEnharmonicString('C-')
['B', 'A##']
```

Navigation

© Copyright 2006-2023 Michael Scott Asato Cuthbert. Last updated on Jun 12, 2023.

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.tree.timespanTree

Previous topic

music21.tree.spans

Next topic

music21.tree.toStream

Table of Contents

- music21.tree.timespanTree
    - TimespanTree
        - `TimespanTree`
            - `TimespanTree.offset`
            - `TimespanTree.element`
            - `TimespanTree.allParts()`
            - `TimespanTree.elementEndTime()`
            - `TimespanTree.findNextPitchedTimespanInSameStreamByClass()`
            - `TimespanTree.findPreviousPitchedTimespanInSameStreamByClass()`
            - `TimespanTree.getVerticalityAtOrBefore()`
            - `TimespanTree.index()`
            - `TimespanTree.iterateConsonanceBoundedVerticalities()`
            - `TimespanTree.iterateVerticalities()`
            - `TimespanTree.iterateVerticalitiesNwise()`
            - `TimespanTree.maximumOverlap()`
            - `TimespanTree.removeTimespan()`
            - `TimespanTree.removeTimespanList()`
            - `TimespanTree.splitAt()`
            - `TimespanTree.toPartwiseTimespanTrees()`
            - `TimespanTree.unwrapVerticalities()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.metadata.bundles

Next topic

music21.metadata.primitives

Table of Contents

- music21.metadata.caching
  - JobProcessor
    - `JobProcessor`
      - `JobProcessor.process_parallel()`
      - `JobProcessor.process_serial()`
  - MetadataCachingJob
    - `MetadataCachingJob`
      - `MetadataCachingJob.cleanFilePath`
      - `MetadataCachingJob.getErrors()`
      - `MetadataCachingJob.getResults()`
      - `MetadataCachingJob.parseFilePath()`
      - `MetadataCachingJob.parseNonOpus()`
      - `MetadataCachingJob.parseOpus()`
      - `MetadataCachingJob.parseScoreInsideOpus()`
      - `MetadataCachingJob.run()`
  - WorkerProcess
    - `WorkerProcess`
      - `WorkerProcess.run()`
  - Functions
    - `cacheMetadata()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.tree.trees

Next topic

music21.variant

Table of Contents

- music21.tree.verticality
  - Verticality
    - `Verticality`
      - `Verticality.bassTimespan`
      - `Verticality.beatStrength`
      - `Verticality.measureNumber`
      - `Verticality.nextStartOffset`
      - `Verticality.nextVerticality`
      - `Verticality.pitchClassSet`
      - `Verticality.pitchSet`
      - `Verticality.previousVerticality`
      - `Verticality.startAndOverlapTimespans`
      - `Verticality.timeToNextEvent`
      - `Verticality.getAllVoiceLeadingQuartets()`
      - `Verticality.getPairedMotion()`
      - `Verticality.makeElement()`
      - `Verticality.toChord()`
      - `Verticality.offset`
      - `Verticality.overlapTimespans`
      - `Verticality.startTimespans`
      - `Verticality.stopTimespans`
      - `Verticality.timespanTree`
  - VerticalitySequence
    - `VerticalitySequence`
      - `VerticalitySequence.__getitem__()`
      - `VerticalitySequence.unwrap()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation

- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.musicxml.lilypondTestSuite

Previous topic

music21.musicxml.helpers

Next topic

music21.musicxml.m21ToXml

Table of Contents

- music21.musicxml.lilypondTestSuite
  - Functions
    - `allFiles()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.languageExcerpts.naturalLanguageObjects

Next topic

music21.lily.lilyObjects

Table of Contents

- music21.layout
  - ScoreLayout
    - `ScoreLayout`
      - `ScoreLayout.tenthsToMillimeters()`
  - PageLayout
    - `PageLayout`
  - SystemLayout
    - `SystemLayout`
  - StaffLayout
    - `StaffLayout`
      - `StaffLayout.staffType`
  - LayoutBase
    - `LayoutBase`
  - LayoutScore
    - `LayoutScore`
      - `LayoutScore.pages`
      - `LayoutScore.getAllMeasurePositionsInDocument()`
      - `LayoutScore.getMarginsAndSizeForPageId()`
      - `LayoutScore.getPageAndSystemNumberFromMeasureNumber()`
      - `LayoutScore.getPositionForStaff()`
      - `LayoutScore.getPositionForStaffMeasure()`
      - `LayoutScore.getPositionForSystem()`
      - `LayoutScore.getStaffDistanceFromPrevious()`
      - `LayoutScore.getStaffHiddenAttribute()`
      - `LayoutScore.getStaffSizeFromLayout()`
      - `LayoutScore.getSystemBeforeThis()`
      - `LayoutScore.measurePositionWithinSystem()`
      - `LayoutScore.show()`
  - Page
    - `Page`
      - `Page.systems`
      - `Page.show()`
  - System
    - `System`
      - `System.staves`
  - Staff
    - `Staff`

- PageSize
  - `PageSize`
- StaffGroup
  - `StaffGroup`
    - `StaffGroup.barTogether`
    - `StaffGroup.symbol`
- SystemSize
  - `SystemSize`
- Functions
  - `divideByPages()`
  - `getPageRegionMeasureNumbers()`
  - `getRegionMeasureNumbers()`
  - `getSystemRegionMeasureNumbers()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.sieve

Next topic

music21.sorting

Table of Contents

- music21.sites
  - SiteRef
    - `SiteRef`
      - `SiteRef.site`
  - Sites
    - `Sites`
      - `Sites.add()`
      - `Sites.clear()`
      - `Sites.get()`
      - `Sites.getAttrByName()`
      - `Sites.getById()`
      - `Sites.getObjByClass()`
      - `Sites.getSiteCount()`
      - `Sites.getSiteIds()`
      - `Sites.getSitesByClass()`
      - `Sites.hasSiteId()`
      - `Sites.hasSpannerSite()`
      - `Sites.hasVariantSite()`
      - `Sites.purgeLocations()`
      - `Sites.remove()`
      - `Sites.removeById()`
      - `Sites.setAttrByName()`
      - `Sites.yieldSites()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.stream.makeNotation

Next topic

music21.stream.tools

Table of Contents

- music21.stream.streamStatus
  - StreamStatus
    - `StreamStatus`
      - `StreamStatus.accidentals`
      - `StreamStatus.beams`
      - `StreamStatus.tuplets`
      - `StreamStatus.haveAccidentalsBeenMade()`
      - `StreamStatus.haveBeamsBeenMade()`
      - `StreamStatus.haveTupletBracketsBeenMade()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.musicxml.xmlToM21

Next topic

music21.noteworthy.binaryTranslate

Table of Contents

- music21.note
  - Note
    - `Note`
      - `Note.fullName`
      - `Note.name`
      - `Note.nameWithOctave`
      - `Note.octave`
      - `Note.pitches`
      - `Note.step`
      - `Note.pitchChanged()`
      - `Note.transpose()`
      - `Note.isNote`
      - `Note.isRest`
      - `Note.pitch`
  - Rest
    - `Rest`
      - `Rest.fullName`
      - `Rest.fullMeasure`
      - `Rest.isNote`
      - `Rest.isRest`
      - `Rest.name`
      - `Rest.stepShift`
  - Unpitched
    - `Unpitched`
      - `Unpitched.displayName`
      - `Unpitched.storedInstrument`
      - `Unpitched.displayPitch()`
  - NotRest
    - `NotRest`
      - `NotRest.notehead`
      - `NotRest.noteheadFill`
      - `NotRest.noteheadParenthesis`
      - `NotRest.stemDirection`
      - `NotRest.storedInstrument`
      - `NotRest.volume`
      - `NotRest.getInstrument()`
      - `NotRest.hasVolumeInformation()`

        - NotRest.beams
  - GeneralNote
    - GeneralNote
        - GeneralNote.fullName
        - GeneralNote.lyric
        - GeneralNote.pitches
        - GeneralNote.tie
        - GeneralNote.__eq__()
        - GeneralNote.addLyric()
        - GeneralNote.augmentOrDiminish()
        - GeneralNote.getGrace()
        - GeneralNote.insertLyric()
        - GeneralNote.articulations
        - GeneralNote.expressions
        - GeneralNote.isChord
        - GeneralNote.lyrics
  - Lyric
    - Lyric
        - Lyric.isComposite
        - Lyric.identifier
        - Lyric.number
        - Lyric.rawText
        - Lyric.syllabic
        - Lyric.text
        - Lyric.setTextAndSyllabic()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Previous topic

music21.tree.verticality

Next topic

music21.vexflow.toMusic21j

Table of Contents

- music21.variant
  - Variant
    - `Variant`
      - `Variant.containedHighestOffset`
      - `Variant.containedHighestTime`
      - `Variant.containedSite`
      - `Variant.highestOffset`
      - `Variant.highestTime`
      - `Variant.lengthType`
      - `Variant.replacementDuration`
      - `Variant.__getitem__()`
      - `Variant.getElementIds()`
      - `Variant.purgeLocations()`
      - `Variant.purgeOrphans()`
      - `Variant.removeReplacedElementsFromStream()`
      - `Variant.replaceElement()`
      - `Variant.replacedElements()`
      - `Variant.show()`
  - Functions
    - `addVariant()`
    - `getMeasureHashes()`
    - `makeAllVariantsReplacements()`
    - `makeVariantBlocks()`
    - `mergePartAsOssia()`
    - `mergeVariantMeasureStreams()`
    - `mergeVariantScores()`
    - `mergeVariants()`
    - `mergeVariantsEqualDuration()`
    - `refineVariant()`

Table of Contents

- About *music21*

- User’s Guide

- **Module Reference**

- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.vexflow.toMusic21j

Previous topic

music21.variant

Next topic

music21.voiceLeading

Table of Contents

- music21.vexflow.toMusic21j
  - VexflowPickler
    - `VexflowPickler`
      - `VexflowPickler.fromObject()`
      - `VexflowPickler.fromStream()`
      - `VexflowPickler.getHTML()`
      - `VexflowPickler.getJSBody()`
      - `VexflowPickler.getJSBodyScript()`
      - `VexflowPickler.getLoadTemplate()`
      - `VexflowPickler.splitLongJSON()`
  - Functions
    - `fromObject()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.sorting

Next topic

music21.stream.base

Table of Contents

- music21.spanner
  - Spanner
    - `Spanner`
      - `Spanner.__getitem__()`
      - `Spanner.addSpannedElements()`
      - `Spanner.fill()`
      - `Spanner.getFirst()`
      - `Spanner.getLast()`
      - `Spanner.getSpannedElementIds()`
      - `Spanner.getSpannedElements()`
      - `Spanner.getSpannedElementsByClass()`
      - `Spanner.hasSpannedElement()`
      - `Spanner.isFirst()`
      - `Spanner.isLast()`
      - `Spanner.purgeLocations()`
      - `Spanner.purgeOrphans()`
      - `Spanner.replaceSpannedElement()`
  - Glissando
    - `Glissando`
      - `Glissando.lineType`
      - `Glissando.slideType`
  - Line
    - `Line`
      - `Line.endHeight`
      - `Line.endTick`
      - `Line.lineType`
      - `Line.startHeight`
      - `Line.startTick`
      - `Line.tick`
  - MultiMeasureRest
    - `MultiMeasureRest`
      - `MultiMeasureRest.numRests`
      - `MultiMeasureRest.maxSymbols`
      - `MultiMeasureRest.useSymbols`
  - Ottava
    - `Ottava`
      - `Ottava.type`

        - `Ottava.interval()`
        - `Ottava.performTransposition()`
        - `Ottava.shiftDirection()`
        - `Ottava.shiftMagnitude()`
        - `Ottava.undoTransposition()`
- RepeatBracket
    - `RepeatBracket`
        - `RepeatBracket.number`
        - `RepeatBracket.getNumberList()`
        - `RepeatBracket.numberRange`
        - `RepeatBracket.overrideDisplay`
- Slur
    - `Slur`
- SpannerAnchor
    - `SpannerAnchor`
- SpannerBundle
    - `SpannerBundle`
        - `SpannerBundle.__getitem__()`
        - `SpannerBundle.append()`
        - `SpannerBundle.freePendingSpannedElementAssignment()`
        - `SpannerBundle.getByClass()`
        - `SpannerBundle.getByClassIdLocalComplete()`
        - `SpannerBundle.getByCompleteStatus()`
        - `SpannerBundle.getByIdLocal()`
        - `SpannerBundle.getBySpannedElement()`
        - `SpannerBundle.getSpannerStorageIds()`
        - `SpannerBundle.remove()`
        - `SpannerBundle.replaceSpannedElement()`
        - `SpannerBundle.setIdLocalByClass()`
        - `SpannerBundle.setIdLocals()`
        - `SpannerBundle.setPendingSpannedElementAssignment()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.romanText.rtObjects

Next topic

music21.romanText.tsvConverter

Table of Contents

- music21.romanText.translate
  - PartTranslator
    - `PartTranslator`
      - `PartTranslator.fillMeasureFromPreviousRn()`
      - `PartTranslator.fillToMeasureToken()`
      - `PartTranslator.parseKeySignatureTag()`
      - `PartTranslator.processRTChord()`
      - `PartTranslator.setAnalyticKey()`
      - `PartTranslator.setMinorRootParse()`
      - `PartTranslator.translateMeasureLineToken()`
      - `PartTranslator.translateOneLineToken()`
      - `PartTranslator.translateSingleMeasure()`
      - `PartTranslator.translateSingleMeasureAtom()`
      - `PartTranslator.translateTokens()`
  - RomanTextUnprocessedMetadata
    - `RomanTextUnprocessedMetadata`
  - RomanTextUnprocessedToken
    - `RomanTextUnprocessedToken`
  - Functions
    - `appendMeasureToRepeatEndingsDict()`
    - `fixPickupMeasure()`
    - `romanTextToStreamOpus()`
    - `romanTextToStreamScore()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.vexflow.toMusic21j

Next topic

music21.volpiano

Table of Contents

- music21.voiceLeading
  - VoiceLeadingQuartet
    - `VoiceLeadingQuartet`
      - `VoiceLeadingQuartet.key`
      - `VoiceLeadingQuartet.v1n1`
      - `VoiceLeadingQuartet.v1n2`
      - `VoiceLeadingQuartet.v2n1`
      - `VoiceLeadingQuartet.v2n2`
      - `VoiceLeadingQuartet.antiParallelMotion()`
      - `VoiceLeadingQuartet.closesIncorrectly()`
      - `VoiceLeadingQuartet.contraryMotion()`
      - `VoiceLeadingQuartet.hiddenFifth()`
      - `VoiceLeadingQuartet.hiddenInterval()`
      - `VoiceLeadingQuartet.hiddenOctave()`
      - `VoiceLeadingQuartet.inwardContraryMotion()`
      - `VoiceLeadingQuartet.isProperResolution()`
      - `VoiceLeadingQuartet.leapNotSetWithStep()`
      - `VoiceLeadingQuartet.motionType()`
      - `VoiceLeadingQuartet.noMotion()`
      - `VoiceLeadingQuartet.obliqueMotion()`
      - `VoiceLeadingQuartet.opensIncorrectly()`
      - `VoiceLeadingQuartet.outwardContraryMotion()`
      - `VoiceLeadingQuartet.parallelFifth()`
      - `VoiceLeadingQuartet.parallelInterval()`
      - `VoiceLeadingQuartet.parallelMotion()`
      - `VoiceLeadingQuartet.parallelOctave()`
      - `VoiceLeadingQuartet.parallelUnison()`
      - `VoiceLeadingQuartet.parallelUnisonOrOctave()`
      - `VoiceLeadingQuartet.similarMotion()`
      - `VoiceLeadingQuartet.voiceCrossing()`
      - `VoiceLeadingQuartet.voiceOverlap()`
      - `VoiceLeadingQuartet.hIntervals`
      - `VoiceLeadingQuartet.vIntervals`
  - ThreeNoteLinearSegment
    - `ThreeNoteLinearSegment`
      - `ThreeNoteLinearSegment.iLeft`
      - `ThreeNoteLinearSegment.iLeftToRight`

        - ThreeNoteLinearSegment.iRight
        - ThreeNoteLinearSegment.n1
        - ThreeNoteLinearSegment.n2
        - ThreeNoteLinearSegment.n3
        - ThreeNoteLinearSegment.couldBeChromaticNeighborTone()
        - ThreeNoteLinearSegment.couldBeChromaticPassingTone()
        - ThreeNoteLinearSegment.couldBeDiatonicNeighborTone()
        - ThreeNoteLinearSegment.couldBeDiatonicPassingTone()
        - ThreeNoteLinearSegment.couldBeNeighborTone()
        - ThreeNoteLinearSegment.couldBePassingTone()
- Verticality
    - Verticality
        - Verticality.objects
        - Verticality.color
        - Verticality.lyric
        - Verticality.changeDurationOfAllObjects()
        - Verticality.getChord()
        - Verticality.getLongestDuration()
        - Verticality.getObjectsByClass()
        - Verticality.getObjectsByPart()
        - Verticality.getShortestDuration()
        - Verticality.getStream()
        - Verticality.getVerticalityOffset()
        - Verticality.isConsonant()
        - Verticality.makeAllLargestDuration()
        - Verticality.makeAllSmallestDuration()
        - Verticality.contentDict
- VerticalityNTuplet
    - VerticalityNTuplet
- MotionType
    - MotionType
- NChordLinearSegment
    - NChordLinearSegment
        - NChordLinearSegment.chordList
- NNoteLinearSegment
    - NNoteLinearSegment
        - NNoteLinearSegment.melodicIntervals
        - NNoteLinearSegment.noteList
- NObjectLinearSegment
    - NObjectLinearSegment
- TwoChordLinearSegment
    - TwoChordLinearSegment
        - TwoChordLinearSegment.bassInterval()
        - TwoChordLinearSegment.rootInterval()
- VerticalityTriplet
    - VerticalityTriplet
        - VerticalityTriplet.hasNeighborTone()
        - VerticalityTriplet.hasPassingTone()
- Functions
    - getVerticalityFromObject()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**

- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.key

Next topic

music21.layout

Table of Contents

- music21.languageExcerpts.naturalLanguageObjects
  - Functions
    - `generateLanguageDictionary()`
    - `toChord()`
    - `toNote()`
    - `toPitch()`

Table of Contents

- About *music21*
- User's Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.stream.core

Next topic

music21.stream.filters

Table of Contents

- music21.stream.enums
  - GivenElementsBehavior
    - `GivenElementsBehavior`
  - RecursionType
    - `RecursionType`
  - ShowNumber
    - `ShowNumber`
  - StaffType
    - `StaffType`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.tree.node

Next topic

music21.tree.timespanTree

Table of Contents

- music21.tree.spans
  - ElementTimespan
    - `ElementTimespan`
      - `ElementTimespan.measureNumber`
      - `ElementTimespan.part`
      - `ElementTimespan.quarterLength`
      - `ElementTimespan.__eq__()`
      - `ElementTimespan.getParentageByClass()`
      - `ElementTimespan.makeElement()`
      - `ElementTimespan.new()`
      - `ElementTimespan.parentage`
  - PitchedTimespan
    - `PitchedTimespan`
      - `PitchedTimespan.pitches`
      - `PitchedTimespan.canMerge()`
  - Timespan
    - `Timespan`
      - `Timespan.endTime`
      - `Timespan.offset`
      - `Timespan.__eq__()`
      - `Timespan.canMerge()`
      - `Timespan.mergeWith()`
      - `Timespan.new()`
      - `Timespan.splitAt()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference

- Documentation and tests in progress

Navigation

Previous topic

music21.tree.fromStream

Next topic

music21.tree.spans

Table of Contents

- music21.tree.node
  - ElementNode
    - `ElementNode`
      - `ElementNode.highestPosition`
      - `ElementNode.lowestPosition`
      - `ElementNode.updateEndTimes()`
      - `ElementNode.updateIndices()`
      - `ElementNode.endTimeHigh`
      - `ElementNode.endTimeLow`
      - `ElementNode.payloadElementIndex`
      - `ElementNode.subtreeElementsStartIndex`
      - `ElementNode.subtreeElementsStopIndex`
  - OffsetNode
    - `OffsetNode`
      - `OffsetNode.payloadEndTimes()`
      - `OffsetNode.updateEndTimes()`
      - `OffsetNode.updateIndices()`
      - `OffsetNode.payload`
      - `OffsetNode.payloadElementsStartIndex`
      - `OffsetNode.payloadElementsStopIndex`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.ipython21.converters

Next topic

music21.languageExcerpts.naturalLanguageObjects

Table of Contents

- music21.key
  - KeySignature
    - `KeySignature`
      - `KeySignature.isNonTraditional`
      - `KeySignature.alteredPitches`
      - `KeySignature.sharps`
      - `KeySignature.accidentalByStep()`
      - `KeySignature.asKey()`
      - `KeySignature.getScale()`
      - `KeySignature.transpose()`
      - `KeySignature.transposePitchFromC()`
  - Key
    - `Key`
      - `Key.parallel`
      - `Key.relative`
      - `Key.tonicPitchNameWithCase`
      - `Key.__eq__()`
      - `Key.deriveByDegree()`
      - `Key.tonalCertainty()`
      - `Key.transpose()`
  - Functions
    - `convertKeyStringToMusic21KeyString()`
    - `pitchToSharps()`
    - `sharpsToPitch()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference

- Documentation and tests in progress

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.omr.correctors

Previous topic

music21.noteworthy.translate

Next topic

music21.omr.evaluators

Table of Contents

- music21.omr.correctors
  - MeasureHash
    - `MeasureHash`
      - `MeasureHash.differenceProbabilityForOneOpCode()`
      - `MeasureHash.getHashString()`
      - `MeasureHash.getMeasureDifference()`
      - `MeasureHash.getOpCodes()`
      - `MeasureHash.getProbabilityBasedOnChanges()`
      - `MeasureHash.getProbabilityFromOneCharSub()`
      - `MeasureHash.getProbabilityOnAddition()`
      - `MeasureHash.getProbabilityOnEquality()`
      - `MeasureHash.getProbabilityOnOmission()`
      - `MeasureHash.getProbabilityOnSubstitute()`
      - `MeasureHash.hashGrace()`
      - `MeasureHash.hashNote()`
      - `MeasureHash.hashQuarterLength()`
      - `MeasureHash.hashRest()`
      - `MeasureHash.setSequenceMatcher()`
  - MeasureRelationship
    - `MeasureRelationship`
  - MeasureSlice
    - `MeasureSlice`
      - `MeasureSlice.getSliceHashes()`
      - `MeasureSlice.runSliceSearch()`
  - PriorsIntegrationScore
    - `PriorsIntegrationScore`
  - ScoreCorrector
    - `ScoreCorrector`
      - `ScoreCorrector.generateCorrectedScore()`
      - `ScoreCorrector.getAllHashes()`
      - `ScoreCorrector.getAllIncorrectMeasures()`
      - `ScoreCorrector.getMeasureSlice()`
      - `ScoreCorrector.getSinglePart()`
      - `ScoreCorrector.getVerticalProbabilityDistributionSinglePart()`
      - `ScoreCorrector.getVerticalProbabilityDistributionSinglePartSingleMeasu`
      - `ScoreCorrector.run()`
      - `ScoreCorrector.runHorizontalCorrectionModel()`

            - ScoreCorrector.runPriorModel()
            - ScoreCorrector.runVerticalCorrectionModel()
            - ScoreCorrector.runVerticalSearch()
            - ScoreCorrector.substituteOneMeasureContentsForAnother()
            - ScoreCorrector.verticalProbabilityDist()
    - SinglePart
        - SinglePart
            - SinglePart.getIncorrectMeasureIndices()
            - SinglePart.getMeasures()
            - SinglePart.getProbabilityDistribution()
            - SinglePart.getSequenceHashesFromMeasureStream()
            - SinglePart.horizontalProbabilityDist()
            - SinglePart.runHorizontalCorrectionModel()
            - SinglePart.runHorizontalSearch()

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Quick search

This Page

- Show Source

music21.omr.correctors¶

MeasureHash¶

class music21.omr.correctors.MeasureHash(*measureObject=None*)¶

Able to do a number of matching, substitution and hashing operations on a given measure object

`MeasureHash` methods

MeasureHash.differenceProbabilityForOneOpCode(*opCodeTuple*, *source*, *destination=None*)¶

Given an opCodeTuple and a source, differenceProbabilityForOneOpCode returns the difference probability for one type of op-code (replace, insert, delete, or equal). Here, the destination is in the set F of flagged measures and the source is in the set C of correcting measures. Source and destination are both hashStrings

```
>>> source = 'PFPFFF'
>>> destination = 'PFPFGF'
>>> ops = ('equal', 0, 4, 0, 4)
>>> mh = omr.correctors.MeasureHash()
>>> mh.differenceProbabilityForOneOpCode(ops, source, destination)
0.8762013031640626
```

Omission

```
>>> ops2 = ('insert', 4, 4, 4, 5)
>>> mh2 = omr.correctors.MeasureHash()
>>> mh2.differenceProbabilityForOneOpCode(ops2, source, destination)
0.009
```

```
>>> ops3 = ('replace', 2, 4, 2, 4)
>>> mh3 = omr.correctors.MeasureHash()
>>> mh3.differenceProbabilityForOneOpCode(ops3, 'PPPPP', 'PPVZP')
0.0001485
```

Five deletes in a row:

```
>>> ops4 = ('delete', 0, 5, 0, 0)
>>> mh3 = omr.correctors.MeasureHash()
>>> mh3.differenceProbabilityForOneOpCode(ops4, 'e', 'GFPGF')
1.024e-12
```

Example of Violin II vs. Viola in K525 I, m. 17

```
>>> vlnII = converter.parse('tinynotation: 4/4 e4 e8. e8 c4 c8 c8').flatten().notes.st
>>> viola = converter.parse('tinynotation: 4/4 c4 c8  c8 A4 A8 A8').flatten().notes.st
>>> vlnIIMH = omr.correctors.MeasureHash(vlnII)
>>> violaMH = omr.correctors.MeasureHash(viola)
>>> vlnIIMH.hashString
'PLFPFF'
>>> violaMH.hashString
'PFFPFF'
>>> opCodes = vlnIIMH.getOpCodes(violaMH.hashString)
>>> for oc in opCodes:
...    print('%30r : %.3f' %
...           (oc, vlnIIMH.differenceProbabilityForOneOpCode(oc, violaMH.hashString)))
          ('equal', 0, 1, 0, 1) : 0.968
        ('replace', 1, 2, 1, 2) : 0.009
          ('equal', 2, 6, 2, 6) : 0.876
```

MeasureHash.getHashString()¶

takes a stream and returns a hashed string for searching on and stores it in self.hashString

If a measure object has multiple voices, use the first voice.

```
>>> m = stream.Measure()
>>> m.append(note.Note('C', quarterLength=1.5))
>>> m.append(note.Note('C', quarterLength=0.5))
>>> m.append(note.Rest(quarterLength=1.5))
>>> m.append(note.Note('B', quarterLength=0.5))

>>> hasher = omr.correctors.MeasureHash(m)
>>> hasher.getHashString()
'VFUF'
>>> hasher.hashString == 'VFUF'
True
```

MeasureHash.getMeasureDifference(*hashString*)¶

Returns the difference ratio between two measures b is the “correct” measure that we want to replace the flagged measure with

Takes a hashString

```
>>> m = stream.Measure()
>>> m.append(note.Note('C', quarterLength=1.5))
>>> m.append(note.Note('C', quarterLength=0.5))
>>> m.append(note.Rest(quarterLength=1.5))
>>> m.append(note.Note('B', quarterLength=0.5))

>>> hasher = omr.correctors.MeasureHash(m)
>>> hasher.setSequenceMatcher()
>>> hasher.getMeasureDifference('VGUF')
0.25

>>> m = stream.Measure()
>>> m.append(note.Note('C', quarterLength=1.5))
>>> m.append(note.Note('C', quarterLength=0.5))
>>> m.append(note.Rest(quarterLength=1.5))
>>> m.append(note.Note('B', quarterLength=0.5))

>>> hasher = omr.correctors.MeasureHash(m)
>>> hasher.setSequenceMatcher()
>>> hasher.getMeasureDifference('VFUF')
1.0
```

MeasureHash.getOpCodes(*otherHash=None*)¶

Gets the opcodes from a simple sequenceMatcher for the current measureHash

Example of Violin II vs. Viola and Cello in K525 I, m. 17

```
>>> vlnII = converter.parse('tinynotation: 4/4 e4 e8. e8 c4 c8 c8').flatten().notes.st
>>> viola = converter.parse('tinynotation: 4/4 c4 c8  c8 A4 A8 A8').flatten().notes.st
>>> cello = converter.parse('tinynotation: 4/4 C4 C4     D4 D4   ').flatten().notes.st
>>> vlnII_MH = omr.correctors.MeasureHash(vlnII)
>>> viola_MH = omr.correctors.MeasureHash(viola)
>>> cello_MH = omr.correctors.MeasureHash(cello)
>>> vlnII_MH.getOpCodes(viola_MH.hashString)
[('equal', 0, 1, 0, 1), ('replace', 1, 2, 1, 2), ('equal', 2, 6, 2, 6)]
>>> vlnII_MH.getOpCodes(cello_MH.hashString)
[('equal', 0, 1, 0, 1), ('delete', 1, 3, 1, 1),
 ('equal', 3, 4, 1, 2), ('replace', 4, 6, 2, 4)]
```

MeasureHash.getProbabilityBasedOnChanges(*otherHash*)¶

Takes a hash string and gets the probability based on changes.

```
>>> otherHash = 'e'
>>> hashString = 'GFPGF'
>>> mh = omr.correctors.MeasureHash()
>>> mh.hashString = hashString
>>> mh.getProbabilityBasedOnChanges(otherHash)
2.9472832125e-14
```

Example of Violin II vs. Viola and Cello in K525 I, m. 17

```
>>> vlnII = converter.parse('tinynotation: 4/4 e4 e8. e8 c4 c8 c8').flatten().notes.st
>>> viola = converter.parse('tinynotation: 4/4 c4 c8  c8 A4 A8 A8').flatten().notes.st
>>> cello = converter.parse('tinynotation: 4/4 C4 C4     D4 D4   ').flatten().notes.st
>>> vlnII_MH = omr.correctors.MeasureHash(vlnII)
>>> viola_MH = omr.correctors.MeasureHash(viola)
>>> cello_MH = omr.correctors.MeasureHash(cello)
>>> vlnII_MH.getProbabilityBasedOnChanges(viola_MH.hashString)
0.0076295...
>>> vlnII_MH.getProbabilityBasedOnChanges(cello_MH.hashString)
4.077...e-09
```

MeasureHash.getProbabilityFromOneCharSub(*source*, *destination*)¶

Source and destination are strings of one character

```
>>> mh = omr.correctors.MeasureHash()
```

Eighth note to eighth rest:

```
>>> mh.getProbabilityFromOneCharSub('F', 'G')
0.003
```

Eighth note to quarter note:

```
>>> mh.getProbabilityFromOneCharSub('F', 'P')
0.0165
```

Eighth note to half note:

```
>>> mh.getProbabilityFromOneCharSub('F', 'Z')
0.0002722...
```

Quarter note to dotted quarter note:

```
>>> mh.getProbabilityFromOneCharSub('P', 'V')
0.009
```

Dotted quarter note to quarter note:

```
>>> mh.getProbabilityFromOneCharSub('V', 'P')
0.004
```

```
>>> mh.getProbabilityFromOneCharSub('A', 'Y')
3.6e-05
```

MeasureHash.getProbabilityOnAddition()¶

In order for the source to be correct, the destination added a symbol Associated with type ‘insert’

```
>>> omr.correctors.MeasureHash().getProbabilityOnAddition()
0.004
```

MeasureHash.getProbabilityOnEquality()¶

Parts or the whole of a string were equal.

```
>>> omr.correctors.MeasureHash().getProbabilityOnEquality()
0.9675
```

MeasureHash.getProbabilityOnOmission()¶

In order for the source to be correct, the destination omitted a symbol. Associated with type ‘delete’ and in the case of replacement of a dotted version of a note with an undotted version (or double dot with dotted, etc.)

```
>>> omr.correctors.MeasureHash().getProbabilityOnOmission()
0.009
```

MeasureHash.getProbabilityOnSubstitute(*source*, *destination*)¶

Source and destination are measureHash strings Source is in set C of correcting measures. Destination is in set F of flagged measures.

(Rossant & Bloch)

- value change: 50.77% of all errors (inverse: 0.0197)
- confusions: 9.23% of all errors (inverse: 0.108)

  Note: these get the most probability, because they are the rarest
- omission: 27.69% of all errors (inverse: 0.0361)
- addition: 12.31% of all errors (inverse: 0.08125)

```
>>> mh = omr.correctors.MeasureHash()
```

Replacement of eighth note (F) for quarter note (P) = shift of one value:

```
>>> mh.getProbabilityOnSubstitute('F', 'P')
0.0165
```

Replacement of eighth note (F) for eighth rest (G) = shift of one type:

```
>>> mh.getProbabilityOnSubstitute('F', 'G')
0.003
```

Omission of any symbol, less common so costs more The proposed correction assumes that the incorrect measure omitted a symbol

```
>>> mh.getProbabilityOnSubstitute('', 'P')
0.009
```

Addition of any symbol, less common so costs more The proposed correction assumes that the incorrect measure added a symbol

```
>>> mh.getProbabilityOnSubstitute('P', '')
0.004
```

Combination of value shift and an addition:

```
>>> mh.getProbabilityOnSubstitute('F', 'PP')
0.0001485
```

Take minimum length. Compare index to index. Any additional letters in the flagged measure get graded as additions. Any additional letters in the comparison measure get graded as omissions.

MeasureHash.hashGrace(*n*)¶

Gives a Grace Note a duration of a 128th note

MeasureHash.hashNote(*n*)¶

Encodes a note

```
>>> hasher = omr.correctors.MeasureHash()

>>> n = note.Note('C')
>>> n.duration.type = 'quarter'
>>> hasher.hashNote(n)
'P'
>>> n2 = note.Note('C')
>>> n2.duration.type = 'half'
>>> hasher.hashNote(n2)
'Z'
>>> n3 = note.Note('C', quarterLength=1.5)
>>> hasher.hashNote(n3)
'V'
```

MeasureHash.hashQuarterLength(*ql*)¶

Turns a QuarterLength duration into an integer from 1 to 127

```
>>> hasher = omr.correctors.MeasureHash()
>>> hasher.hashQuarterLength(1.0)
80

>>> hasher.hashQuarterLength(2.0)
90
```

MeasureHash.hashRest(*r*)¶

Encodes a rest

```
>>> r = note.Rest(1.0)
>>> hasher = omr.correctors.MeasureHash()
>>> hasher.hashRest(r)
'Q'
```

MeasureHash.setSequenceMatcher(*hashes=None*)¶

MeasureRelationship¶

class music21.omr.correctors.MeasureRelationship(*flaggedMeasurePart*, *flaggedMeasureIndex*, *correctMeasurePart*, *correctMeasureIndex*, *correctionProbability*)¶

MeasureSlice¶

class music21.omr.correctors.MeasureSlice(*score*, *i*)¶

represents a single measure from all parts

`MeasureSlice` methods

MeasureSlice.getSliceHashes()¶

```
>>> omrPath = omr.correctors.K525omrShortPath
>>> omrScore = converter.parse(omrPath)
>>> ssOMR = omr.correctors.ScoreCorrector(omrScore)
>>> ssOMR
<music21.omr.correctors.ScoreCorrector object at 0x...>
>>> measureSlice = ssOMR.getMeasureSlice(2)
>>> measureSlice
<music21.omr.correctors.MeasureSlice object at 0x...>
```

MeasureSlice.runSliceSearch(*incorrectPartIndex*)¶

Takes in an incorrectPartIndex and returns an array of the measure indices within the slice that have the maximum probability to correct a given flagged measures.

Returns a namedtuple (MeasureRelationship)

```
>>> omrPath = omr.correctors.K525omrShortPath
>>> omrScore = converter.parse(omrPath)
>>> ssOMR = omr.correctors.ScoreCorrector(omrScore)
>>> measureSlice = ssOMR.getMeasureSlice(2)
>>> measureSlice
<music21.omr.correctors.MeasureSlice object at 0x...>
>>> measureSlice.runSliceSearch(1)
MeasureRelationship(flaggedMeasurePart=1, flaggedMeasureIndex=2,
    correctMeasurePart=3, correctMeasureIndex=2, correctionProbability=0.0054...)

>>> measureSlice = ssOMR.getMeasureSlice(3)
>>> measureSlice.runSliceSearch(0)
MeasureRelationship(flaggedMeasurePart=0,
    flaggedMeasureIndex=3, correctMeasurePart=1, correctMeasureIndex=3,
    correctionProbability=2.41...e-14)
```

PriorsIntegrationScore¶

class music21.omr.correctors.PriorsIntegrationScore(*total*, *horizontal*, *vertical*, *ignored*)¶

ScoreCorrector¶

class music21.omr.correctors.ScoreCorrector(*score=None*)¶

takes in a music21.stream.Score object and runs OMR correction on it.

`ScoreCorrector` methods

ScoreCorrector.generateCorrectedScore(*horizontalArray*, *verticalArray*)¶

Given two correcting arrays (one from the horizontal model and one from the vertical model), which offer source measures for each flagged measure in each part, this method compares the probabilities of proposed source measures for each flagged measure, and replaces the flagged measures contents with the more probable source measure using substituteOneMeasureContentsForAnother. It then rehashes the score so that a new difference comparison can be run.

Returns a collections.namedtuple of the total number of flagged measures, the total number corrected by the horizontal (Prior based on Distance) and the vertical (Prior based on Parts) methods.

ScoreCorrector.getAllHashes()¶

Returns an array of arrays, each of which is the hashed notes for a part

```
>>> p1 = stream.Part()
>>> p1.insert(0, meter.TimeSignature('4/4'))
>>> p1.append(note.Note('C', type = 'half'))
>>> p1.append(note.Rest(type='half'))
>>> p1.append(note.Note('C', type = 'half'))
>>> p1.append(note.Rest(type='half'))
>>> p1.makeMeasures(inPlace=True)
>>> p2 = stream.Part()
>>> p2.insert(0, meter.TimeSignature('4/4'))
>>> p2.repeatAppend(note.Note('C', type='quarter'), 8)
>>> p2.makeMeasures(inPlace=True)
>>> s = stream.Score()
>>> s.insert(0, p1)
>>> s.insert(0, p2)
>>> ss = omr.correctors.ScoreCorrector(s)
>>> ss.getAllHashes()
[['Z[', 'Z['], ['PPPP', 'PPPP']]
```

ScoreCorrector.getAllIncorrectMeasures()¶

Returns an array of the incorrect measure indices arrays for each part. This is used in the MeasureSlice object to make sure we’re not comparing a flagged measure to other flagged measures in its slice

```
>>> omrPath = omr.correctors.K525omrShortPath
>>> omrScore = converter.parse(omrPath)
>>> ssOMR = omr.correctors.ScoreCorrector(omrScore)
>>> ssOMR
<music21.omr.correctors.ScoreCorrector object at 0x...>
>>> ssOMR.getAllIncorrectMeasures()
[[1, 3, 9, 10, 12, 17, 20], [2, 12, 14, 17], [1, 9], []]
```

ScoreCorrector.getMeasureSlice(*i*)¶

Given an index, i, returns a MeasureSlice object at that index

```
>>> omrPath = omr.correctors.K525omrShortPath
>>> omrScore = converter.parse(omrPath)
>>> ssOMR = omr.correctors.ScoreCorrector(omrScore)
>>> ssOMR.getMeasureSlice(4)
<music21.omr.correctors.MeasureSlice object at 0x...>
```

ScoreCorrector.getSinglePart(*pn*)¶

returns a NEW SinglePart object for part number pn from the score

ScoreCorrector.getVerticalProbabilityDistributionSinglePart(*pn*)¶

Returns the Vertical Probability Distribution (PrP) for a single part.

Get the Priors for the Violin II part (first 20 measures only)

```
>>> omrPath = omr.correctors.K525omrShortPath
>>> omrScore = converter.parse(omrPath)
>>> ssOMR = omr.correctors.ScoreCorrector(omrScore)
>>> allDists = ssOMR.getVerticalProbabilityDistributionSinglePart(1)
>>> ['%0.3f' % p for p in allDists]
['0.571', '1.000', '0.667', '0.714']
```

ScoreCorrector.getVerticalProbabilityDistributionSinglePartSingleMeasure(*pn*, *measureIndex*)¶

ScoreCorrector.run()¶

Run all known models for OMR correction on this score

ScoreCorrector.runHorizontalCorrectionModel()¶

runs for sp in self.singleParts:

> sp.runHorizontalCorrectionModel()

returns correctingArrayAllParts

ScoreCorrector.runPriorModel()¶

run the horizontal and vertical correction models on the score. Returns the new self.score object.

ScoreCorrector.runVerticalCorrectionModel()¶

Runs a basic vertical correction model on a ScoreCorrector object. That is, for each flagged measure, this method replaces the rhythm in that flagged measure with the rhythm of a measure with the least difference.

ScoreCorrector.runVerticalSearch(*i*, *pn*)¶

Returns an array of the minimum distance measure indices given a measure (with index i) within a part pn to compare to

ScoreCorrector.substituteOneMeasureContentsForAnother(*sourceHorizontalIndex*, *sourceVerticalIndex*, *destinationHorizontalIndex*, *destinationVerticalIndex*)¶

Takes a destination measure, deletes its contents, and replaces them with the contents of a source measure but retains as many pitches as possible

The destination measure would normally be in the set F of flagged measures (having an incorrect number of beats) while the source measure is in the set C of correcting measures.

```
>>> s = corpus.parse('bwv66.6').measures(1, 2)
>>> s.show('text')
{0.0} <music21.stream.Part Soprano>
    ...
    {0.0} <music21.stream.Measure 1 offset=0.0>
        {0.0} <music21.note.Note A>
        {1.0} <music21.note.Note B>
        {2.0} <music21.note.Note C#>
        {3.0} <music21.note.Note E>
    {4.0} <music21.stream.Measure 2 offset=4.0>
        {0.0} <music21.note.Note C#>
        {1.0} <music21.note.Note B>
        {2.0} <music21.note.Note A>
        {3.0} <music21.note.Note C#>
{0.0} <music21.stream.Part Alto>
     ...
    {0.0} <music21.stream.Measure 1 offset=0.0>
        {0.0} <music21.note.Note F#>
        {1.0} <music21.note.Note E>
        {2.0} <music21.note.Note E>
        {3.0} <music21.note.Note E>
    {4.0} <music21.stream.Measure 2 offset=4.0>
        {0.0} <music21.note.Note E>
        {0.5} <music21.note.Note A>
        {1.0} <music21.note.Note G#>
        {2.0} <music21.note.Note E>
        {3.0} <music21.note.Note G#>
...
```

Replace part 1, measure 2 (index 1) with part 0, measure 1 (index 0) while retaining as many pitches as possible. The eighth-notes will become quarters:

```
>>> scOMR = omr.correctors.ScoreCorrector(s)
>>> scOMR.substituteOneMeasureContentsForAnother(0, 0, 1, 1)
>>> s2 = scOMR.score
>>> s2.show('text')
{0.0} <music21.stream.Part Soprano>
    ...
    {0.0} <music21.stream.Measure 1 offset=0.0>
        {0.0} <music21.note.Note A>
        {1.0} <music21.note.Note B>
        {2.0} <music21.note.Note C#>
        {3.0} <music21.note.Note E>
    {4.0} <music21.stream.Measure 2 offset=4.0>
        {0.0} <music21.note.Note C#>
        {1.0} <music21.note.Note B>
        {2.0} <music21.note.Note A>
        {3.0} <music21.note.Note C#>
{0.0} <music21.stream.Part Alto>
```

```
          ...
        {0.0} <music21.stream.Measure 1 offset=0.0>
            {0.0} <music21.note.Note F#>
            {1.0} <music21.note.Note E>
            {2.0} <music21.note.Note E>
            {3.0} <music21.note.Note E>
        {4.0} <music21.stream.Measure 2 offset=4.0>
            {0.0} <music21.note.Note E>
            {1.0} <music21.note.Note A>
            {2.0} <music21.note.Note G#>
            {3.0} <music21.note.Note E>
   ...
```

ScoreCorrector.verticalProbabilityDist()¶

Uses a score and returns an array of probabilities. For n in the array, n is the probability that the nth part

SinglePart¶

class music21.omr.correctors.SinglePart(*part=None*, *pn=None*)¶

`SinglePart` methods

SinglePart.getIncorrectMeasureIndices(*runFast=False*)¶

Returns an array of all the measures that OMR software would flag - that is, measures that do not have the correct number of beats given the current time signature

if runFast is True (by default), assumes that the initial TimeSignature is the TimeSignature for the entire piece.

```
>>> p = stream.Part()
>>> ts = meter.TimeSignature('6/8')
>>> m1 = stream.Measure()
>>> m1.number = 1
>>> m1.append(ts)
>>> m1.append(note.Note('C4', quarterLength = 3.0))
>>> p.append(m1)
>>> m2 = stream.Measure()
>>> m2.number = 2
>>> m2.append(note.Note('C4', quarterLength = 1.5))
>>> p.append(m2)

>>> sp = omr.correctors.SinglePart(p, pn = 0)
>>> sp.getIncorrectMeasureIndices()
[1]

>>> p[1]
<music21.stream.Measure 2 offset=3.0>
>>> p[1].insert(0, meter.TimeSignature('3/8'))
>>> sp.getIncorrectMeasureIndices(runFast=False)
[]
```

SinglePart.getMeasures()¶

SinglePart.getProbabilityDistribution(*sourceIndex*, *destinationIndex*)¶

SinglePart.getSequenceHashesFromMeasureStream()¶

takes in a measure stream of a part returns an array of hashed strings

SinglePart.horizontalProbabilityDist(*regenerate=False*)¶

Uses (takes?) an array of hashed measures and returns an array of probabilities. For n in the array, n is

Uses (takes?) an array of flagged measures and returns an array of probabilities. For n in the array, n is the probability that the measure (n-(length of score)) away from a flagged measure will offer a rhythmic solution.

These are the probabilities that, within a part, a measure offers a solution, given its distance from a flagged measure.

SinglePart.runHorizontalCorrectionModel()¶

Runs a basic horizontal correction model on a score. That is, for each flagged measure, this method replaces the rhythm in that flagged measure with the rhythm of a measure with the least difference.

SinglePart.runHorizontalSearch(*i*)¶

Returns an array of the indices of the minimum distance measures given a measure (with index i) to compare to.

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.omr.correctors

© Copyright 2006-2023 Michael Scott Asato Cuthbert. Last updated on Jun 12, 2023.

Navigation

- index
- modules |
- next |
- previous |
- music21 »
- **Module Reference** »
- music21.lily.lilyObjects

Previous topic

music21.layout

Next topic

music21.lily.translate

Table of Contents

- music21.lily.lilyObjects
  - LyAlternativeMusic
    - `LyAlternativeMusic`
      - `LyAlternativeMusic.stringOutput()`
  - LyAssignment
    - `LyAssignment`
      - `LyAssignment.stringOutput()`
  - LyAssignmentId
    - `LyAssignmentId`
      - `LyAssignmentId.stringOutput()`
  - LyBookBlock
    - `LyBookBlock`
      - `LyBookBlock.stringOutput()`
  - LyBookBody
    - `LyBookBody`
      - `LyBookBody.stringOutput()`
  - LyBookpartBlock
    - `LyBookpartBlock`
      - `LyBookpartBlock.stringOutput()`
  - LyBookpartBody
    - `LyBookpartBody`
      - `LyBookpartBody.stringOutput()`
  - LyChordBody
    - `LyChordBody`
      - `LyChordBody.stringOutput()`
  - LyChordBodyElement
    - `LyChordBodyElement`
      - `LyChordBodyElement.stringOutput()`
  - LyCommandElement
    - `LyCommandElement`
      - `LyCommandElement.stringOutput()`
  - LyCommandEvent
    - `LyCommandEvent`
      - `LyCommandEvent.stringOutput()`
  - LyCompositeMusic
    - `LyCompositeMusic`
      - `LyCompositeMusic.contents`

- LyCompositeMusic.stringOutput()

- LyContextChange
  - LyContextChange
    - LyContextChange.stringOutput()
- LyContextDefMod
  - LyContextDefMod
    - LyContextDefMod.stringOutput()
- LyContextDefSpecBlock
  - LyContextDefSpecBlock
    - LyContextDefSpecBlock.stringOutput()
- LyContextDefSpecBody
  - LyContextDefSpecBody
    - LyContextDefSpecBody.stringOutput()
- LyContextMod
  - LyContextMod
    - LyContextMod.stringOutput()
- LyContextModList
  - LyContextModList
    - LyContextModList.stringOutput()
- LyContextModification
  - LyContextModification
    - LyContextModification.stringOutput()
- LyDirectionLessEvent
  - LyDirectionLessEvent
    - LyDirectionLessEvent.stringOutput()
- LyDirectionReqdEvent
  - LyDirectionReqdEvent
    - LyDirectionReqdEvent.stringOutput()
- LyEmbeddedScm
  - LyEmbeddedScm
    - LyEmbeddedScm.stringOutput()
- LyEventChord
  - LyEventChord
    - LyEventChord.stringOutput()
- LyFullMarkup
  - LyFullMarkup
    - LyFullMarkup.stringOutput()
- LyFullMarkupList
  - LyFullMarkupList
    - LyFullMarkupList.stringOutput()
- LyGenTextDef
  - LyGenTextDef
    - LyGenTextDef.stringOutput()
- LyGroupedMusicList
  - LyGroupedMusicList
    - LyGroupedMusicList.stringOutput()
- LyIdentifierInit
  - LyIdentifierInit
    - LyIdentifierInit.stringOutput()
- LyLayout
  - LyLayout
    - LyLayout.stringOutput()
- LyLilypondHeader
  - LyLilypondHeader
    - LyLilypondHeader.stringOutput()
- LyLilypondHeaderBody
  - LyLilypondHeaderBody
    - LyLilypondHeaderBody.stringOutput()

- LyLilypondTop
  - LyLilypondTop
    - LyLilypondTop.stringOutput()
- LyLyricElement
  - LyLyricElement
    - LyLyricElement.stringOutput()
- LyLyricMarkup
  - LyLyricMarkup
    - LyLyricMarkup.stringOutput()
- LyMarkup
  - LyMarkup
    - LyMarkup.stringOutput()
- LyMarkupBracedList
  - LyMarkupBracedList
    - LyMarkupBracedList.stringOutput()
- LyMarkupBracedListBody
  - LyMarkupBracedListBody
    - LyMarkupBracedListBody.stringOutput()
- LyMarkupComposedList
  - LyMarkupComposedList
    - LyMarkupComposedList.stringOutput()
- LyMarkupList
  - LyMarkupList
    - LyMarkupList.stringOutput()
- LyMarkupTop
  - LyMarkupTop
    - LyMarkupTop.stringOutput()
- LyMock
  - LyMock
- LyModeChangingHead
  - LyModeChangingHead
    - LyModeChangingHead.stringOutput()
- LyMultipliedDuration
  - LyMultipliedDuration
    - LyMultipliedDuration.stringOutput()
- LyMusic
  - LyMusic
    - LyMusic.stringOutput()
- LyMusicList
  - LyMusicList
    - LyMusicList.stringOutput()
- LyMusicPropertyDef
  - LyMusicPropertyDef
    - LyMusicPropertyDef.stringOutput()
- LyNewLyrics
  - LyNewLyrics
    - LyNewLyrics.stringOutput()
- LyNoteChordElement
  - LyNoteChordElement
    - LyNoteChordElement.stringOutput()
- LyNumberExpression
  - LyNumberExpression
    - LyNumberExpression.stringOutput()
- LyNumberTerm
  - LyNumberTerm
    - LyNumberTerm.stringOutput()
- LyObject
  - LyObject

    - LyObject.newlineIndent
    - LyObject.ancestorList()
    - LyObject.comment()
    - LyObject.encloseCurly()
    - LyObject.getAncestorByClass()
    - LyObject.getFirstNonNoneAttribute()
    - LyObject.getParent()
    - LyObject.newlineSeparateStringOutputIfNotNone()
    - LyObject.quoteString()
    - LyObject.setAttributes()
    - LyObject.setAttributesFromClassObject()
    - LyObject.setParent()
    - LyObject.stringOutput()
- LyOctaveCheck
  - LyOctaveCheck
    - LyOctaveCheck.stringOutput()
- LyOptionalId
  - LyOptionalId
    - LyOptionalId.stringOutput()
- LyOptionalRest
  - LyOptionalRest
    - LyOptionalRest.stringOutput()
- LyOssiaMusic
  - LyOssiaMusic
    - LyOssiaMusic.stringOutput()
- LyOutputDef
  - LyOutputDef
    - LyOutputDef.stringOutput()
- LyOutputDefBody
  - LyOutputDefBody
    - LyOutputDefBody.stringOutput()
- LyOutputDefHead
  - LyOutputDefHead
    - LyOutputDefHead.stringOutput()
- LyPaperBlock
  - LyPaperBlock
    - LyPaperBlock.stringOutput()
- LyPitch
  - LyPitch
    - LyPitch.stringOutput()
- LyPostEvent
  - LyPostEvent
    - LyPostEvent.stringOutput()
- LyPostEvents
  - LyPostEvents
    - LyPostEvents.stringOutput()
- LyPrefixCompositeMusic
  - LyPrefixCompositeMusic
    - LyPrefixCompositeMusic.stringOutput()
- LyPropertyOperation
  - LyPropertyOperation
    - LyPropertyOperation.stringOutput()
- LyPropertyPath
  - LyPropertyPath
    - LyPropertyPath.stringOutput()
- LyReRhythmedMusic
  - LyReRhythmedMusic
    - LyReRhythmedMusic.stringOutput()

- LyRelativeMusic
  - LyRelativeMusic
    - LyRelativeMusic.stringOutput()
- LyRepeatedMusic
  - LyRepeatedMusic
    - LyRepeatedMusic.stringOutput()
- LySchemeFunction
  - LySchemeFunction
    - LySchemeFunction.stringOutput()
- LyScoreBlock
  - LyScoreBlock
    - LyScoreBlock.stringOutput()
- LyScoreBody
  - LyScoreBody
    - LyScoreBody.stringOutput()
- LyScriptAbbreviation
  - LyScriptAbbreviation
    - LyScriptAbbreviation.stringOutput()
- LyScriptDir
  - LyScriptDir
    - LyScriptDir.stringOutput()
- LySequentialMusic
  - LySequentialMusic
    - LySequentialMusic.stringOutput()
- LySimpleElement
  - LySimpleElement
    - LySimpleElement.stringOutput()
- LySimpleMarkup
  - LySimpleMarkup
    - LySimpleMarkup.stringOutput()
- LySimpleMusic
  - LySimpleMusic
    - LySimpleMusic.stringOutput()
- LySimultaneousMusic
  - LySimultaneousMusic
    - LySimultaneousMusic.stringOutput()
- LyStenoDuration
  - LyStenoDuration
    - LyStenoDuration.stringOutput()
- LyTempoEvent
  - LyTempoEvent
    - LyTempoEvent.stringOutput()
- LyTempoRange
  - LyTempoRange
    - LyTempoRange.stringOutput()
- LyTopLevelExpression
  - LyTopLevelExpression
    - LyTopLevelExpression.stringOutput()
- LyTremoloType
  - LyTremoloType
    - LyTremoloType.stringOutput()

Table of Contents

- About *music21*
- User’s Guide

- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress

Navigation

Previous topic

music21.musicxml.archiveTools

Next topic

music21.musicxml.lilypondTestSuite

Table of Contents

- music21.musicxml.helpers
  - Functions
    - `dump()`
    - `dumpString()`
    - `indent()`
    - `insertBeforeElements()`
    - `isFullMeasureRest()`
    - `measureNumberComesBefore()`

Table of Contents

- About *music21*
- User’s Guide
- **Module Reference**
- Installation
- Developer Reference
- Documentation and tests in progress