<https://www.bilibili.com/read/cv12594929?spm_id_from=333.999.0.0>

note初始化 整数和字符串

chord初始化 字符串（预留字music\database.py） 列表（音符）

## 音符类型（Note）

### 初始化

* 可变长度参数列表，依次代表（音名，音高，时长，音量）
  + a = note(‘C’, 5, 0.5, 100)
  + a = note(‘C’, 5)
  + a = note(‘C’) // 至少传递一个参数（音名），其余可以设为默认值
* 直接传递音名字符串
  + a = N(‘C5’)
* 跟据整型音高创建
  + a = N(60)

ps：这个地方可以强制声明时必须初始化？

### 运算

* 更改音符变量参数，可变长度参数列表，依次代表（音名，音高，时长，音量）
  + a % (‘C’, 5, 0.5, 100)
  + a % (‘C’, 5)
  + a % (‘C’) // 至少传递一个参数（音名），其余可以设为默认值
* 升（减）半音
  + +a // 升一个半音
  + ++a // 升两个半音
  + ---a // 减三个半音
* 升（减）n个半音
  + a + n
  + a – n
* 音名升降号相互转换
  + ~a // 例如从D#变成Db
* 多个音符连接组成和弦
  + A = a + b + c // A是Chord类型

### 音乐分析相关

* 访问一些属性，这个有利于用户分析语言，或者有利于ai分析乐曲，或许可以print出来？
  + 得到音符音名：a.name
  + 得到音符音高：a.degree
  + 得到音符时长：a.duration
  + 得到音符音量：a.volume

## 和弦类型（Chord）

### 初始化

* 跟据根音与和弦名创建和弦（和弦名是预留字？）
  + Am7 = chord(‘A’, ‘m7’) // [A5, C6, E6, G6] with interval [0, 0, 0, 0] 默认根音音高是5
  + Am7 = chord(‘A6’, ‘m7’) // [A6, C7, E7, G7] with interval [0, 0, 0, 0] 可以写明根音音高
* 跟据音符**字符串**列表创建和弦
  + a = chord(['C5','E5', 'G5', 'B5'])

ps：我合计音符间隔和诗长这些就不在初始化里加了，单独设置吧

### 运算

* 对和弦进行音符长度和音符间隔的设置
  + Am7 % (0.5, 1) // 长度均为0.5，间隔均为1
  + Am7 % ([0.5, 0.5, 1, 1], [1, 1, 2, 2]) // 对长度和间隔分别进行设置
  + Am7 % ([0.5, 0.5, 1, 1], 1)
  + Am7 % (0.5, [1, 1, 2, 2])
* 对和弦音符进行音量设置
  + Am7.setVolume([80, 80, 100, 100])
* 和弦的转位
  + Am7 / 1 // 第一转位
  + Am7 / ‘E’ // 只给音名，根音为E6（相当于第二转位）
  + Am7 / ‘E6’ // 给音名和音高名
* 另一种转位（把要转的音放到最低音）
  + A / 1
  + A / ‘E’
* 往一个和弦上加一个最低音
  + A / ‘C5’ // 这地方/右边就按音符初始化那套来就可以
* 对和弦进行排序
  + A / [1, 2, 0, 3] // 比如有一个A的小七和弦，对于它的4个组成音A,C,E,G按照CEAG的顺序排列
* 和弦的拼接
  + A + B // 可以得到音乐片段B追加到音乐片段A之后的新的音乐片段，不过需要注意的是，这个模式是直接把B的音符列表追加到A的音符列表之后， 然后把B的音符间隔列表也追加到A的音符间隔列表之后，如果音乐片段A的最后几个音符间隔是0的话，那么可能会出现音乐片段B的开始几个音与A的最后几个音重叠在一起的情况，如果很确定A的最后几个音符间隔不是0，那么这个模式可以放心地使用
  + A & B // 可以得到合并A和B两个音乐片段后的新的音乐片段，B的开头对齐到A的开头，也就是A和B同时从头开始进行演奏的音乐片段。合并的机制是重新计算合并过后的音的间隔，然后把A和B的音符按照计算的间隔，调整好顺序进行重新的合并排列。
    - A & (B, start) // 参数start，可以用来设定B要从A的什么地方开始进行合并，也就是B的开头对齐到A的什么地方，单位为小节，换句话说，就是B延后A开始多少小节开始进行演奏
  + A | B // 与tail模式的区别在于，这个模式会特别计算A与B之间是否需要多一些音符间隔，以避免tail模式里面某些情况下A的结尾与B的开头会重叠的情况。所以当不确定A的最后几个音符间隔是否为0的时候，使用这个模式最好
* 往一个和弦后加n小节休止符
  + A | n
* 和弦内增加音符
  + A + x
* 和弦内删去音符
  + A – x // A – ‘B’ or A – ‘B5’
* 重复和弦n次
  + A \* n // 如果和弦A是intervals全部为0的和弦（全部的音同时开始一起弹），那么A \* n会让重复的A的音符重叠在一起，在这种情况下，如果想要让多次和弦A的重复不重叠，那么需要写A % n
  + A % n // 这个语法会让A在不与前一个和弦重叠的情况下进行重复
* 对一个和弦进行倒序排列
  + ~A
* 和弦的索引切片（和python完全一样）
  + A[0]
  + A[-1]
  + A[:6]
  + ……
* 对和弦进行升调和降调
  + +A // 整体升半音
  + -A // 整体降半音
  + ++A
  + A + n // 整体升高n个半音
  + A – n
* 和弦内第k个音升高/降低x
  + A + (x, k)
  + A – (x, k)
* 对每个音不同的升降
  + A+ [1, 2, 0, -1]
* 对和弦内音符进行修改
  + A[2] = ‘F5’
  + A[2] = a
  + … // 这地方=右边就按音符初始化那套来就可以
* 对一个和弦内的音符按照索引值提取，包括高八度和低八度的变换
  + A @ [1, 3, 2, 3, 1.1, 3, 2, 3] // A是C大三和弦 // [C5, G5, E5, G5, C6, G5, E5, G5] with interval [0, 0, 0, 0, 0, 0, 0, 0]
* 构造一套和弦处理规则，适用于任何和弦类型
  + rules = exp('(x / 1) % ([1/4, 1/2, 1/2], [1/4, 1/4, 1/2])') // rules(C('Dmaj7'))

### 音乐分析相关

* 获得和弦内音程关系
  + chd('C', 'maj').intervalof() // [4, 7]
  + chd('C', 'maj').intervalof(translate = True) // ['major third', 'perfect fifth']
  + chd('C', 'maj').intervalof(translate = True, cummulative=False) // ['major third', 'minor third']
* 得到和弦音符列表
  + A.notes // [A5, C6, E6, G6]
* 得到和弦音名列表
  + A.names // [‘A’, ‘C’, ‘E’, ‘G’]
* 查看和弦音符长度列表
  + A.get\_duration()
* 查看和弦音符间隔列表
  + A.get\_interval()
* 查看和弦音量列表
  + A.get\_volume()

## 乐曲类型（Piece）



C1 = chord('G4, D5, B5, F#5') % (1, 1/8) % 4

C2 = (chord('C2, C2, G1, G1') % (1,1)) % 2

C3 = (chord('F#6, G6') % (1/8, 1/8) % 8 | chord('A5, B5') % (1/8, 1/8) % 8) % 2

C4 = chord('G3, G3, G3, G3') % ([3/8,1/8,1/4,1/4], [3/8,1/8,1/4,1/4]) % 4

new\_piece = piece(tracks=[C1, C2, C3, C4],

instruments\_list=['Acoustic Grand Piano', 'Electric Bass (finger)', 'Orchestral Harp', 'Synth Drum'],

bpm=120,

start\_times=[0, 2, 2, 6],

track\_names=['piano', 'bass', 'harp', 'drum'])

>>> new\_piece

[piece]

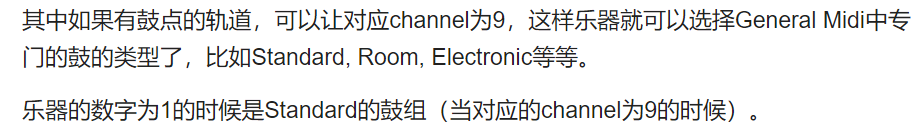
BPM: 120

track 1 piano | instrument: Acoustic Grand Piano | start time: 0 | [G4, D5, B5, F#5, G4, D5, B5, F#5, G4, D5, B5, F#5, G4, D5, B5, F#5] with interval [0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125]

track 2 bass | instrument: Electric Bass (finger) | start time: 2 | [C2, C2, G1, G1, C2, C2, G1, G1] with interval [1, 1, 1, 1, 1, 1, 1, 1]

track 3 harp | instrument: Orchestral Harp | start time: 2 | [F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5, F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, F#6, G6, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5, A5, B5] with interval [0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125]

track 4 drum | instrument: Synth Drum | start time: 6 | [G3, G3, G3, G3, G3, G3, G3, G3, G3, G3, G3, G3, G3, G3, G3, G3] with interval [0.375, 0.125, 0.25, 0.25, 0.375, 0.125, 0.25, 0.25, 0.375, 0.125, 0.25, 0.25, 0.375, 0.125, 0.25, 0.25]



另一种参考的构造方式



## 其他

* 播放
  + / c // 默认速度
  + / (c, 100) // 按速度播放
  + / (c, 100, instrument=‘violin’) // 按乐器播放
    - 对这种需要按照参数名传参的是否可以直接穿个字典进去？

## todo

