```
# -*- coding: utf-8 -*-
"""
Created on Thu Dec 10 15:00:27 2015

@author: henryschoeller
"""
from __future__ import division
import threading, time
from sequencer import *
import midi

obertoene=[1,2,3]
gewichte=[0.5,0.25,0.25]
obertoene2=[1,2,3,4,5,6,7]
gewichte2=[1,1/2.,1/3.,1/4,1/5.,1/6.,1/7.]
gewichte2=[1/2.343 *g for g in gewichte2] # damit die Summe 1 ist
viertelnote=0.25
grundton=220

class Komposition (object):
    def __init__(self, tonart, akkordschema):
        self.tonart=tonart
        self.akkordschema=akkordschema
        self.stimmen=[]

    def add_stimme(self, stimme):
        self.stimmen.append(stimme)

    def remove_stimme(self, stimme):
        self.stimmen.remove(stimme)

    def spiele(self, art="standard"):
        sound=Sound()  #  das Objekt, das sich um die Tonausgabe kümme
        sound.klanggen=Sequencer(6000,len(self.stimmen))
        kompende=0.0
        for stimme in self.stimmen:
            for ton in stimme.toene:
                if art=="standard":
                    sound.klanggen.add_note(stimme.stimmennummer, grun
                elif art=="dreieck":
                    sound.klanggen.add_note_dreieck(stimme.stimmennumm
            if stimme.ende>kompende:
                kompende=stimme.ende
        threading.Thread(target=sound.play).start() # Tonausgabe gesta
        time.sleep(1)
        jetzt=time.time()
        while jetzt+kompende*viertelnote+1>time.time():
            pass
        sound.outStream.close()
```

```
class Akkord(object):
    def __init__(self, toene):
        self.toene=toene

class Stimme(object):
    def __init__(self, stimmennummer):
        self.stimmennummer=stimmennummer
        self.ende=0.0
        self.toene=[]   #Eine Stimme ist im Grunde eine Liste aus Töne

    def add_ton(self, ton):
        neuer_ton=Ton_in_Stimme(ton.dauer, ton.hoehe, ton.lautstaerke,
        self.toene.append(neuer_ton)
        self.ende=self.ende+ton.dauer

    def add_pause(self, dauer):
        pause=Ton_in_Stimme(dauer, 1, 0, self.ende, self.ende+dauer)
        self.toene.append(pause)
        self.ende=self.ende+dauer

    def spiele(self, art="standard"):
        sound=Sound()  #  das Objekt, das sich um die Tonausgabe kümme
        sound.klanggen=Sequencer(6000,1)
        for ton in self.toene:
            if art=="standard":
                sound.klanggen.add_note(0, grundton*HALBTON**ton.hoehe
            elif art=="dreieck":
                sound.klanggen.add_note_dreieck(0,grundton*HALBTON**to
        threading.Thread(target=sound.play).start() # Tonausgabe gesta
        time.sleep(1)
        jetzt=time.time()
        while jetzt+self.ende*viertelnote+1>time.time():
            pass
        sound.outStream.close()

    def add_melodie(self, datei):
        '''Diese Methode fügt der Stimme die Melodie hinzu, welche in
        in .mid Form gespeichert ist.'''
        melodie=midi.read_midifile(datei)
        letzter_ton_tick=0
        akt_ton_tick=0
        akt_ton=0
        i=0
        akt_tick=0
        while i<len(melodie[0]):
            try:
                akt_tick=akt_tick+melodie[0][i]
            except:
                pass
```

```
            if isinstance(melodie[0][i], midi.NoteOnEvent):
                if letzter_ton_tick!=0 and letzter_ton_tick!=akt_tick:
                    self.add_pause(akt_tick-letzter_ton_tick)
                akt_ton_tick=akt_tick
                akt_ton=i
            elif isinstance(melodie[0][i], midi.NoteOffEvent):
                neuer_ton=Ton((akt_tick-akt_ton_tick)/200, melodie[0][
                self.add_ton(neuer_ton)
                letzter_ton_tick=akt_tick

            i+=1

class Ton(object):
    '''Ein Ton. Dauer in Viertelnoten, Hoehe in Halbtonschritten über
    def __init__(self, dauer, hoehe, lautstaerke):
        self.dauer=dauer
        self.hoehe=hoehe
        self.lautstaerke=lautstaerke

    def spiele(self, art="standard"):
        sound=Sound()  #  das Objekt, das sich um die Tonausgabe kümme
        sound.klanggen=Sequencer(6000,1)
        if art=="standard":
            sound.klanggen.add_note(0, grundton*HALBTON**self.hoehe, 0
        elif art=="dreieck":
            sound.klanggen.add_note_dreieck(0,grundton*HALBTON**self.h
        threading.Thread(target=sound.play).start() # Tonausgabe gesta
        time.sleep(1)
        jetzt=time.time()
        while jetzt+self.dauer*viertelnote+1>time.time():
            pass
        sound.outStream.close()


class Ton_in_Stimme(Ton):
    '''Objekte dieser Klasse sollen nur beim Einpflegen in eine Stimme
    Da die Klasse allerdings von der Klasse Ton erbt, können auch Obje
    hörbar gemacht werden'''
    def __init__(self, dauer, hoehe, lautstaerke, anfang, ende):
        self.dauer=dauer
        self.hoehe=hoehe
        self.lautstaerke=lautstaerke
        self.anfang=anfang
        self.ende=ende

s=Stimme(0)
t=Stimme(1)
a=Ton(8,2,0.4)
b=Ton(2,6,0.6)
c=Ton(2,3,0.4)
```

```
d=Ton(6,8,0.4)
s.add_ton(a)
s.add_ton(b)
t.add_ton(c)
t.add_ton(d)
k=Komposition(a,a)
k.add_stimme(s)
k.add_stimme(t)
```