*Tianshu* Segment IV.

import abjad

import itertools

import os

import pathlib

import time

import abjadext.rmakers

from MusicMaker import MusicMaker

from AttachmentHandler import AttachmentHandler

from random import random

from random import seed

print('Interpreting file ...')

# Define the time signatures we would like to apply against the timespan structure.

time\_signatures = [

abjad.TimeSignature(pair) for pair in [

(5, 4), (2, 4), (4, 4), (3, 4), (4, 4), (4, 4),

(4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),

(4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),

(2, 4), (3, 4), (4, 4), (3, 4), (4, 4), (3, 4),

(5, 4), (3, 4), (3, 4), (4, 4), (3, 4), (3, 4),

(4, 4), (5, 4), (4, 4), (3, 4), (5, 4), (5, 4),

(5, 4), (5, 4), (4, 4), (4, 4), (5, 4), (5, 4),

(4, 4), (4, 4), (3, 4), (4, 4), (4, 4), (3, 4),

(9, 8),

]

]

bounds = abjad.mathtools.cumulative\_sums([\_.duration for \_ in time\_signatures])

#Define Pitch Material

def reduceMod3(rw):

return [(x % 4) for x in rw]

def reduceMod5(rw):

return [(x % 6) for x in rw]

def reduceMod7(rw):

return [(x % 8) for x in rw]

def reduceMod9(rw):

return [(x % 10) for x in rw]

def reduceMod13(rw):

return [(x % 14) for x in rw]

seed(1)

flute\_random\_walk\_one = []

flute\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = flute\_random\_walk\_one[i-1] + movement

flute\_random\_walk\_one.append(value)

flute\_random\_walk\_one = [abs(x) for x in flute\_random\_walk\_one]

flute\_chord\_one = [2, 12, 18, 20, 25, 20, 18, 12, ]

flute\_notes\_one = [flute\_chord\_one[x] for x in reduceMod7(flute\_random\_walk\_one)]

seed(2)

clarinet\_random\_walk\_one = []

clarinet\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = clarinet\_random\_walk\_one[i-1] + movement

clarinet\_random\_walk\_one.append(value)

clarinet\_random\_walk\_one = [abs(x) for x in clarinet\_random\_walk\_one]

clarinet\_chord\_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]

clarinet\_notes\_one = [clarinet\_chord\_one[x] for x in reduceMod7(clarinet\_random\_walk\_one)]

seed(3)

bassoon\_random\_walk\_one = []

bassoon\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = bassoon\_random\_walk\_one[i-1] + movement

bassoon\_random\_walk\_one.append(value)

bassoon\_random\_walk\_one = [abs(x) for x in bassoon\_random\_walk\_one]

bassoon\_chord\_one = [-19, -8, -5, 2, 12, 2, -5, -8, ]

bassoon\_notes\_one = [bassoon\_chord\_one[x] for x in reduceMod7(bassoon\_random\_walk\_one)]

seed(4)

horn\_random\_walk\_one = []

horn\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = horn\_random\_walk\_one[i-1] + movement

horn\_random\_walk\_one.append(value)

horn\_random\_walk\_one = [abs(x) for x in horn\_random\_walk\_one]

horn\_chord\_one = [-19, -8, -5, 2, 12, 18, 12, 2, -5, -8, ]

horn\_notes\_one = [horn\_chord\_one[x] for x in reduceMod9(horn\_random\_walk\_one)]

seed(5)

trumpet\_random\_walk\_one = []

trumpet\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = trumpet\_random\_walk\_one[i-1] + movement

trumpet\_random\_walk\_one.append(value)

trumpet\_random\_walk\_one = [abs(x) for x in trumpet\_random\_walk\_one]

trumpet\_chord\_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]

trumpet\_notes\_one = [trumpet\_chord\_one[x] for x in reduceMod7(trumpet\_random\_walk\_one)]

seed(6)

trombone\_random\_walk\_one = []

trombone\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = trombone\_random\_walk\_one[i-1] + movement

trombone\_random\_walk\_one.append(value)

trombone\_random\_walk\_one = [abs(x) for x in trombone\_random\_walk\_one]

trombone\_chord\_one = [-19, -8, -5, 2, -5, -8, ]

trombone\_notes\_one = [trombone\_chord\_one[x] for x in reduceMod5(trombone\_random\_walk\_one)]

seed(7)

tuba\_random\_walk\_one = []

tuba\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = tuba\_random\_walk\_one[i-1] + movement

tuba\_random\_walk\_one.append(value)

tuba\_random\_walk\_one = [abs(x) for x in tuba\_random\_walk\_one]

tuba\_chord\_one = [-27, -19, -8, -5, 2, -5, -8, -19, ]

tuba\_notes\_one = [tuba\_chord\_one[x] for x in reduceMod7(tuba\_random\_walk\_one)]

seed(8)

violin1\_random\_walk\_one = []

violin1\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = violin1\_random\_walk\_one[i-1] + movement

violin1\_random\_walk\_one.append(value)

violin1\_random\_walk\_one = [abs(x) for x in violin1\_random\_walk\_one]

violin1\_chord\_one = [-5, 2, 12, 18, 20, 25, 34, 35, 34, 25, 20, 18, 12, 2, ]

violin1\_notes\_one = [violin1\_chord\_one[x] for x in reduceMod13(violin1\_random\_walk\_one)]

seed(9)

violin2\_random\_walk\_one = []

violin2\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = violin2\_random\_walk\_one[i-1] + movement

violin2\_random\_walk\_one.append(value)

violin2\_random\_walk\_one = [abs(x) for x in violin2\_random\_walk\_one]

violin2\_chord\_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]

violin2\_notes\_one = [violin2\_chord\_one[x] for x in reduceMod7(violin2\_random\_walk\_one)]

seed(10)

viola\_random\_walk\_one = []

viola\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = viola\_random\_walk\_one[i-1] + movement

viola\_random\_walk\_one.append(value)

viola\_random\_walk\_one = [abs(x) for x in viola\_random\_walk\_one]

viola\_chord\_one = [-8, -5, 2, 12, 18, 12, 2, -5, ]

viola\_notes\_one = [viola\_chord\_one[x] for x in reduceMod7(viola\_random\_walk\_one)]

seed(11)

cello\_random\_walk\_one = []

cello\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = cello\_random\_walk\_one[i-1] + movement

cello\_random\_walk\_one.append(value)

cello\_random\_walk\_one = [abs(x) for x in cello\_random\_walk\_one]

cello\_chord\_one = [-19, -8, -5, 2, 12, 2, -5, -8]

cello\_notes\_one = [cello\_chord\_one[x] for x in reduceMod7(cello\_random\_walk\_one)]

seed(12)

bass\_random\_walk\_one = []

bass\_random\_walk\_one.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = bass\_random\_walk\_one[i-1] + movement

bass\_random\_walk\_one.append(value)

bass\_random\_walk\_one = [abs(x) for x in bass\_random\_walk\_one]

bass\_chord\_one = [-27, -19, -8, -5, 2, -5, -8, -19, ]

bass\_notes\_one = [bass\_chord\_one[x] for x in reduceMod7(bass\_random\_walk\_one)]

flute\_scale = [39, ]

clarinet\_scale = [18, ]

bassoon\_scale = [12, ]

horn\_scale = [-5, ]

trumpet\_scale = [12, ]

trombone\_scale = [-5, ]

tuba\_scale = [-27, ]

violin1\_scale = [30, 29.5, 29, 28.5, 28, 27.5, 27, 26.5, 26, 25.5, 25, 24.5, 24, 23.5, 23, 22.5, 22, 21.5, 21, 20.5, 20, 19.5, 19, 19.5, 20, 20.5, 21, 21.5, 22, 22.5, 23, 23.5, 24, 24.5, 25, 25.5, 26, 26.5, 27, 27.5, 28, 28.5, 29, 29.5, ]

violin2\_scale = [19, 18.5, 18, 17.5, 17, 16.5, 16, 15.5, 15, 14.5, 14, 13.5, 13, 12.5, 12, 11.5, 11, 10.5, 10, 9.5, 9, 8.5, 8, 8.5, 9, 9.5, 10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5, 16, 16.5, 17, 17.5, 18, 18.5, ]

viola\_scale = [8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5, 1, 0.5, 0, -0.5, -1, -1.5, -2, -2.5, -3, -2.5, -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, ]

cello\_scale = [-3, -3.5, -4, -4.5, -5, -5.5, -6, -6.5, -7, -7.5, -8, -8.5, -9, -9.5, -10, -10.5, -11, -11.5, -12, -12.5, -13, -13.5, -14, -13.5, -13, -12.5, -12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8, -7.5, -7, -6.5, -6, -5.5, -5, -4.5, -4, -3.5, ]

bass\_scale = [-14, -14.5, -15, -15.5, -16, -16.5, -17, -17.5, -18, -18.5, -19, -19.5, -20, -20.5, -21, -21.5, -22, -22.5, -23, -23.5, -24, -24.5, -25, -24.5, -24, -23.5, -23, -22.5, -22, -21.5, -21, -20.5, -20, -19.5, -19, -18.5, -18, -17.5, -17, -16.5, -16, -15.5, -15, -14.5, ]

seed(1)

flute\_random\_walk\_two = []

flute\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = flute\_random\_walk\_two[i-1] + movement

flute\_random\_walk\_two.append(value)

flute\_random\_walk\_two = [abs(x) for x in flute\_random\_walk\_two]

flute\_chord\_two = [4, 12, 18, 22, 23, 22, 18, 12, ]

flute\_notes\_two = [flute\_chord\_two[x] for x in reduceMod7(flute\_random\_walk\_two)]

seed(2)

clarinet\_random\_walk\_two = []

clarinet\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = clarinet\_random\_walk\_two[i-1] + movement

clarinet\_random\_walk\_two.append(value)

clarinet\_random\_walk\_two = [abs(x) for x in clarinet\_random\_walk\_two]

clarinet\_chord\_two = [-10, -7, 4, 12, 18, 22, 18, 12, 4, -7, ]

clarinet\_notes\_two = [clarinet\_chord\_two[x] for x in reduceMod9(clarinet\_random\_walk\_two)]

seed(3)

bassoon\_random\_walk\_two = []

bassoon\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = bassoon\_random\_walk\_two[i-1] + movement

bassoon\_random\_walk\_two.append(value)

bassoon\_random\_walk\_two = [abs(x) for x in bassoon\_random\_walk\_two]

bassoon\_chord\_two = [-17, -10, -7, 4, 12, 4, -7, -10, ]

bassoon\_notes\_two = [bassoon\_chord\_two[x] for x in reduceMod7(bassoon\_random\_walk\_two)]

seed(4)

horn\_random\_walk\_two = []

horn\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = horn\_random\_walk\_two[i-1] + movement

horn\_random\_walk\_two.append(value)

horn\_random\_walk\_two = [abs(x) for x in horn\_random\_walk\_two]

horn\_chord\_two = [-17, -10, -7, 4, -7, -10, ]

horn\_notes\_two = [horn\_chord\_two[x] for x in reduceMod5(horn\_random\_walk\_two)]

seed(5)

trumpet\_random\_walk\_two = []

trumpet\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = trumpet\_random\_walk\_two[i-1] + movement

trumpet\_random\_walk\_two.append(value)

trumpet\_random\_walk\_two = [abs(x) for x in trumpet\_random\_walk\_two]

trumpet\_chord\_two = [4, 12, 18, 12, ]

trumpet\_notes\_two = [trumpet\_chord\_two[x] for x in reduceMod3(trumpet\_random\_walk\_two)]

seed(6)

trombone\_random\_walk\_two = []

trombone\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = trombone\_random\_walk\_two[i-1] + movement

trombone\_random\_walk\_two.append(value)

trombone\_random\_walk\_two = [abs(x) for x in trombone\_random\_walk\_two]

trombone\_chord\_two = [-17, -10, -7, 4, -7, -10, ]

trombone\_notes\_two = [trombone\_chord\_two[x] for x in reduceMod5(trombone\_random\_walk\_two)]

seed(7)

tuba\_random\_walk\_two = []

tuba\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = tuba\_random\_walk\_two[i-1] + movement

tuba\_random\_walk\_two.append(value)

tuba\_random\_walk\_two = [abs(x) for x in tuba\_random\_walk\_two]

tuba\_chord\_two = [-27, -17, -10, -7, 4, -7, -10, -17, ]

tuba\_notes\_two = [tuba\_chord\_two[x] for x in reduceMod7(tuba\_random\_walk\_two)]

seed(8)

violin1\_random\_walk\_two = []

violin1\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = violin1\_random\_walk\_two[i-1] + movement

violin1\_random\_walk\_two.append(value)

violin1\_random\_walk\_two = [abs(x) for x in violin1\_random\_walk\_two]

violin1\_chord\_two = [4, 12, 18, 22, 23, 32, 37, 39, 37, 32, 23, 22, 18, 12, ]

violin1\_notes\_two = [violin1\_chord\_two[x] for x in reduceMod13(violin1\_random\_walk\_two)]

seed(9)

violin2\_random\_walk\_two = []

violin2\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = violin2\_random\_walk\_two[i-1] + movement

violin2\_random\_walk\_two.append(value)

violin2\_random\_walk\_two = [abs(x) for x in violin2\_random\_walk\_two]

violin2\_chord\_two = [4, 12, 18, 22, 23, 32, 23, 22, 18, 12, ]

violin2\_notes\_two = [violin2\_chord\_two[x] for x in reduceMod9(violin2\_random\_walk\_two)]

seed(10)

viola\_random\_walk\_two = []

viola\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = viola\_random\_walk\_two[i-1] + movement

viola\_random\_walk\_two.append(value)

viola\_random\_walk\_two = [abs(x) for x in viola\_random\_walk\_two]

viola\_chord\_two = [-10, -7, 4, 12, 18, 12, 4, -7, ]

viola\_notes\_two = [viola\_chord\_two[x] for x in reduceMod7(viola\_random\_walk\_two)]

seed(11)

cello\_random\_walk\_two = []

cello\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = cello\_random\_walk\_two[i-1] + movement

cello\_random\_walk\_two.append(value)

cello\_random\_walk\_two = [abs(x) for x in cello\_random\_walk\_two]

cello\_chord\_two = [-17, -10, -7, 4, 12, 4, -7, -10, ]

cello\_notes\_two = [cello\_chord\_two[x] for x in reduceMod7(cello\_random\_walk\_two)]

seed(12)

bass\_random\_walk\_two = []

bass\_random\_walk\_two.append(-1 if random() < 0.5 else 1)

for i in range(1, 1000):

movement = -1 if random() < 0.5 else 1

value = bass\_random\_walk\_two[i-1] + movement

bass\_random\_walk\_two.append(value)

bass\_random\_walk\_two = [abs(x) for x in bass\_random\_walk\_two]

bass\_chord\_two = [-27, -17, -10, -7, 4, -7, -10, -17, ]

bass\_notes\_two = [bass\_chord\_two[x] for x in reduceMod7(bass\_random\_walk\_two)]

# Define rhythm-makers: two to be sued by the MusicMaker, one for silence.

rmaker\_one = abjadext.rmakers.TaleaRhythmMaker(

talea=abjadext.rmakers.Talea(

counts=[2, 3, 2, 1, 4, 3, 1, 4, 5, 1],

denominator=8,

),

beam\_specifier=abjadext.rmakers.BeamSpecifier(

beam\_divisions\_together=True,

beam\_rests=False,

),

extra\_counts\_per\_division=[1, 1, 0, -1, 0],

burnish\_specifier=abjadext.rmakers.BurnishSpecifier(

left\_classes=[abjad.Note, abjad.Rest],

left\_counts=[0, 1, 1],

),

tuplet\_specifier=abjadext.rmakers.TupletSpecifier(

trivialize=True,

extract\_trivial=True,

rewrite\_rest\_filled=True,

),

)

rmaker\_two = abjadext.rmakers.TaleaRhythmMaker(

talea=abjadext.rmakers.Talea(

counts=[2, 1, 3, 1, 4, 1, 1, 5],

denominator=16,

),

beam\_specifier=abjadext.rmakers.BeamSpecifier(

beam\_divisions\_together=True,

beam\_rests=False,

),

extra\_counts\_per\_division=[0, 1, 0, -1],

burnish\_specifier=abjadext.rmakers.BurnishSpecifier(

left\_classes=[abjad.Note, abjad.Rest],

left\_counts=[1, 1, 0, 0],

right\_classes=[abjad.Note, abjad.Rest],

right\_counts=[1, 0, 0, 1],

),

tuplet\_specifier=abjadext.rmakers.TupletSpecifier(

trivialize=True,

extract\_trivial=True,

rewrite\_rest\_filled=True,

),

)

rmaker\_three = abjadext.rmakers.TaleaRhythmMaker(

talea=abjadext.rmakers.Talea(

counts=[1, 2, 1, 3, 1, 4, 5, 1, 1],

denominator=16,

),

beam\_specifier=abjadext.rmakers.BeamSpecifier(

beam\_divisions\_together=True,

beam\_rests=False,

),

extra\_counts\_per\_division=[0, 1, 0, -1],

burnish\_specifier=abjadext.rmakers.BurnishSpecifier(

left\_classes=[abjad.Note, abjad.Rest],

left\_counts=[1, 0, 1],

),

tuplet\_specifier=abjadext.rmakers.TupletSpecifier(

trivialize=True,

extract\_trivial=True,

rewrite\_rest\_filled=True,

),

)

# Initialize AttachmentHandler

attachment\_handler\_one = AttachmentHandler(

starting\_dynamic='mf',

ending\_dynamic='ff',

hairpin\_indicator='<',

articulation='',

)

attachment\_handler\_two = AttachmentHandler(

starting\_dynamic='ff',

ending\_dynamic='mf',

hairpin\_indicator='>',

articulation='',

)

attachment\_handler\_three = AttachmentHandler(

starting\_dynamic='p',

ending\_dynamic='pp',

hairpin\_indicator='--',

articulation='tenuto',

)

# Initialize MusicMakers with the rhythm-makers.

#####oboe#####

flutemusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=flute\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

flutemusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=flute\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

flutemusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=flute\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####violin1#####

violin1musicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=violin1\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

violin1musicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=violin1\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

violin1musicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=violin1\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####trumpet#####

trumpetmusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=trumpet\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

trumpetmusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=trumpet\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

trumpetmusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=trumpet\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####clarinet#####

clarinetmusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=clarinet\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

clarinetmusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=clarinet\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

clarinetmusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=clarinet\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####violin2#####

violin2musicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=violin2\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

violin2musicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=violin2\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

violin2musicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=violin2\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####viola#####

violamusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=viola\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

violamusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=viola\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

violamusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=viola\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####bassoon#####

bassoonmusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=bassoon\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

bassoonmusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=bassoon\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

bassoonmusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=bassoon\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####trombone#####

trombonemusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=trombone\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

trombonemusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=trombone\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

trombonemusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=trombone\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####cello#####

cellomusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=cello\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

cellomusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=cello\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

cellomusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=cello\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####horn#####

hornmusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=horn\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

hornmusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=horn\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

hornmusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=horn\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####tuba#####

tubamusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=tuba\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

tubamusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=tuba\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

tubamusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=tuba\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

#####bass#####

bassmusicmaker\_one = MusicMaker(

rmaker=rmaker\_one,

pitches=bass\_scale,

continuous=True,

attachment\_handler=attachment\_handler\_one,

)

bassmusicmaker\_two = MusicMaker(

rmaker=rmaker\_two,

pitches=bass\_notes\_two,

continuous=True,

attachment\_handler=attachment\_handler\_two,

)

bassmusicmaker\_three = MusicMaker(

rmaker=rmaker\_three,

pitches=bass\_notes\_one,

continuous=True,

attachment\_handler=attachment\_handler\_three,

)

silence\_maker = abjadext.rmakers.NoteRhythmMaker(

division\_masks=[

abjadext.rmakers.SilenceMask(

pattern=abjad.index([0], 1),

),

],

)

# Define a small class so that we can annotate timespans with additional

# information:

class MusicSpecifier:

def \_\_init\_\_(self, music\_maker, voice\_name):

self.music\_maker = music\_maker

self.voice\_name = voice\_name

# Define an initial timespan structure, annotated with music specifiers. This

# structure has not been split along meter boundaries. This structure does not

# contain timespans explicitly representing silence. Here I make four, one

# for each voice, using Python's list comprehension syntax to save some

# space.

print('Collecting timespans and rmakers ...')

###group one###

voice\_1\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 1',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(9, 4), (10, 4), flutemusicmaker\_one],

[(15, 4), (18, 4), flutemusicmaker\_two],

[(22, 4), (25, 4), flutemusicmaker\_three],

[(27, 4), (30, 4), flutemusicmaker\_one],

[(30, 4), (32, 4), flutemusicmaker\_one],

[(35, 4), (39, 4), flutemusicmaker\_two],

[(42, 4), (43, 4), flutemusicmaker\_three],

[(43, 4), (44, 4), flutemusicmaker\_three],

[(45, 4), (46, 4), flutemusicmaker\_one],

[(46, 4), (50, 4), flutemusicmaker\_one],

[(54, 4), (57, 4), flutemusicmaker\_two],

[(59, 4), (60, 4), flutemusicmaker\_three],

[(65, 4), (67, 4), flutemusicmaker\_one],

[(67, 4), (69, 4), flutemusicmaker\_one],

[(70, 4), (72, 4), flutemusicmaker\_two],

[(72, 4), (75, 4), flutemusicmaker\_two],

[(76, 4), (78, 4), flutemusicmaker\_three],

[(81, 4), (82, 4), flutemusicmaker\_one],

[(82, 4), (85, 4), flutemusicmaker\_one],

[(90, 4), (91, 4), flutemusicmaker\_two],

[(93, 4), (94, 4), flutemusicmaker\_three],

[(94, 4), (96, 4), flutemusicmaker\_three],

[(100, 4), (104, 4), flutemusicmaker\_one],

[(104, 4), (105, 4), flutemusicmaker\_one],

[(106, 4), (107, 4), flutemusicmaker\_two],

[(107, 4), (108, 4), flutemusicmaker\_two],

[(111, 4), (114, 4), flutemusicmaker\_one],

[(114, 4), (115, 4), flutemusicmaker\_one],

[(116, 4), (119, 4), flutemusicmaker\_one],

[(119, 4), (120, 4), flutemusicmaker\_one],

[(121, 4), (123, 4), flutemusicmaker\_one],

[(123, 4), (125, 4), flutemusicmaker\_one],

[(126, 4), (131, 4), flutemusicmaker\_two],

[(131, 4), (133, 4), flutemusicmaker\_two],

[(136, 4), (141, 4), flutemusicmaker\_two],

[(148, 4), (150, 4), flutemusicmaker\_two],

[(150, 4), (153, 4), flutemusicmaker\_three],

[(155, 4), (159, 4), flutemusicmaker\_three],

[(162, 4), (164, 4), flutemusicmaker\_three],

[(168, 4), (171, 4), flutemusicmaker\_three],

[(173, 4), (175, 4), flutemusicmaker\_three],

[(175, 4), (177, 4), flutemusicmaker\_three],

[(180, 4), (182, 4), flutemusicmaker\_three],

[(186, 4), (190, 4), flutemusicmaker\_three],

[(190, 4), (381, 8), silence\_maker],

]

])

voice\_5\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 5',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(9, 4), (10, 4), trumpetmusicmaker\_one],

[(14, 4), (18, 4), trumpetmusicmaker\_two],

[(23, 4), (25, 4), trumpetmusicmaker\_three],

[(27, 4), (30, 4), trumpetmusicmaker\_one],

[(30, 4), (32, 4), trumpetmusicmaker\_one],

[(35, 4), (39, 4), trumpetmusicmaker\_two],

[(42, 4), (43, 4), trumpetmusicmaker\_three],

[(43, 4), (44, 4), trumpetmusicmaker\_three],

[(45, 4), (46, 4), trumpetmusicmaker\_one],

[(46, 4), (50, 4), trumpetmusicmaker\_one],

[(54, 4), (57, 4), trumpetmusicmaker\_two],

[(59, 4), (60, 4), trumpetmusicmaker\_three],

[(65, 4), (67, 4), trumpetmusicmaker\_one],

[(67, 4), (69, 4), trumpetmusicmaker\_one],

[(70, 4), (72, 4), trumpetmusicmaker\_two],

[(72, 4), (75, 4), trumpetmusicmaker\_two],

[(76, 4), (78, 4), trumpetmusicmaker\_three],

[(81, 4), (82, 4), trumpetmusicmaker\_one],

[(82, 4), (85, 4), trumpetmusicmaker\_one],

[(90, 4), (91, 4), trumpetmusicmaker\_two],

[(93, 4), (94, 4), trumpetmusicmaker\_three],

[(94, 4), (96, 4), trumpetmusicmaker\_three],

[(100, 4), (104, 4), trumpetmusicmaker\_one],

[(104, 4), (105, 4), trumpetmusicmaker\_one],

[(106, 4), (107, 4), trumpetmusicmaker\_two],

[(107, 4), (108, 4), trumpetmusicmaker\_two],

[(111, 4), (114, 4), trumpetmusicmaker\_one],

[(114, 4), (115, 4), trumpetmusicmaker\_one],

[(116, 4), (119, 4), trumpetmusicmaker\_one],

[(119, 4), (120, 4), trumpetmusicmaker\_one],

[(121, 4), (123, 4), trumpetmusicmaker\_one],

[(123, 4), (125, 4), trumpetmusicmaker\_one],

[(126, 4), (131, 4), trumpetmusicmaker\_two],

[(131, 4), (133, 4), trumpetmusicmaker\_two],

[(136, 4), (141, 4), trumpetmusicmaker\_two],

[(148, 4), (150, 4), trumpetmusicmaker\_two],

[(150, 4), (154, 4), trumpetmusicmaker\_three],

[(157, 4), (159, 4), trumpetmusicmaker\_three],

[(163, 4), (164, 4), trumpetmusicmaker\_three],

[(164, 4), (166, 4), trumpetmusicmaker\_three],

[(168, 4), (172, 4), trumpetmusicmaker\_three],

[(175, 4), (177, 4), trumpetmusicmaker\_three],

[(181, 4), (183, 4), trumpetmusicmaker\_three],

[(183, 4), (184, 4), trumpetmusicmaker\_three],

[(186, 4), (190, 4), trumpetmusicmaker\_three],

]

])

voice\_8\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 8',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(9, 4), (10, 4), violin1musicmaker\_one],

[(14, 4), (18, 4), violin1musicmaker\_two],

[(22, 4), (25, 4), violin1musicmaker\_three],

[(27, 4), (30, 4), violin1musicmaker\_one],

[(35, 4), (39, 4), violin1musicmaker\_two],

[(42, 4), (43, 4), violin1musicmaker\_three],

[(43, 4), (44, 4), violin1musicmaker\_three],

[(45, 4), (46, 4), violin1musicmaker\_one],

[(46, 4), (50, 4), violin1musicmaker\_one],

[(54, 4), (57, 4), violin1musicmaker\_two],

[(59, 4), (60, 4), violin1musicmaker\_three],

[(65, 4), (67, 4), violin1musicmaker\_one],

[(67, 4), (69, 4), violin1musicmaker\_one],

[(70, 4), (72, 4), violin1musicmaker\_two],

[(72, 4), (75, 4), violin1musicmaker\_two],

[(76, 4), (78, 4), violin1musicmaker\_three],

[(81, 4), (82, 4), violin1musicmaker\_one],

[(82, 4), (85, 4), violin1musicmaker\_one],

[(90, 4), (91, 4), violin1musicmaker\_two],

[(93, 4), (94, 4), violin1musicmaker\_three],

[(94, 4), (96, 4), violin1musicmaker\_three],

[(100, 4), (104, 4), violin1musicmaker\_one],

[(104, 4), (105, 4), violin1musicmaker\_one],

[(106, 4), (107, 4), violin1musicmaker\_two],

[(107, 4), (108, 4), violin1musicmaker\_two],

[(111, 4), (114, 4), violin1musicmaker\_one],

[(114, 4), (115, 4), violin1musicmaker\_one],

[(116, 4), (119, 4), violin1musicmaker\_one],

[(119, 4), (120, 4), violin1musicmaker\_one],

[(121, 4), (123, 4), violin1musicmaker\_one],

[(123, 4), (125, 4), violin1musicmaker\_one],

[(126, 4), (131, 4), violin1musicmaker\_two],

[(131, 4), (133, 4), violin1musicmaker\_two],

[(136, 4), (141, 4), violin1musicmaker\_two],

[(148, 4), (150, 4), violin1musicmaker\_two],

[(150, 4), (152, 4), violin1musicmaker\_three],

[(156, 4), (159, 4), violin1musicmaker\_three],

[(161, 4), (164, 4), violin1musicmaker\_three],

[(164, 4), (165, 4), violin1musicmaker\_three],

[(168, 4), (170, 4), violin1musicmaker\_three],

[(174, 4), (175, 4), violin1musicmaker\_three],

[(175, 4), (177, 4), violin1musicmaker\_three],

[(179, 4), (183, 4), violin1musicmaker\_three],

[(186, 4), (190, 4), violin1musicmaker\_three],

]

])

###group two###

voice\_2\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 2',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(2, 4), (5, 4), clarinetmusicmaker\_one],

[(10, 4), (11, 4), clarinetmusicmaker\_two],

[(11, 4), (13, 4), clarinetmusicmaker\_two],

[(16, 4), (18, 4), clarinetmusicmaker\_three],

[(21, 4), (22, 4), clarinetmusicmaker\_one],

[(22, 4), (25, 4), clarinetmusicmaker\_one],

[(35, 4), (40, 4), clarinetmusicmaker\_one],

[(44, 4), (46, 4), clarinetmusicmaker\_two],

[(46, 4), (47, 4), clarinetmusicmaker\_two],

[(49, 4), (50, 4), clarinetmusicmaker\_three],

[(55, 4), (59, 4), clarinetmusicmaker\_one],

[(62, 4), (64, 4), clarinetmusicmaker\_two],

[(65, 4), (67, 4), clarinetmusicmaker\_three],

[(67, 4), (70, 4), clarinetmusicmaker\_three],

[(70, 4), (71, 4), clarinetmusicmaker\_three],

[(73, 4), (75, 4), clarinetmusicmaker\_two],

[(75, 4), (76, 4), clarinetmusicmaker\_two],

[(80, 4), (82, 4), clarinetmusicmaker\_one],

[(82, 4), (85, 4), clarinetmusicmaker\_one],

[(86, 4), (88, 4), clarinetmusicmaker\_two],

[(91, 4), (94, 4), clarinetmusicmaker\_three],

[(94, 4), (95, 4), clarinetmusicmaker\_three],

[(100, 4), (101, 4), clarinetmusicmaker\_two],

[(103, 4), (104, 4), clarinetmusicmaker\_one],

[(104, 4), (106, 4), clarinetmusicmaker\_one],

[(110, 4), (114, 4), clarinetmusicmaker\_one],

[(115, 4), (119, 4), clarinetmusicmaker\_one],

[(120, 4), (123, 4), clarinetmusicmaker\_one],

[(123, 4), (124, 4), clarinetmusicmaker\_one],

[(125, 4), (126, 4), clarinetmusicmaker\_two],

[(129, 4), (131, 4), clarinetmusicmaker\_two],

[(131, 4), (134, 4), clarinetmusicmaker\_two],

[(141, 4), (144, 4), clarinetmusicmaker\_two],

[(149, 4), (150, 4), clarinetmusicmaker\_two],

[(155, 4), (159, 4), clarinetmusicmaker\_three],

[(162, 4), (164, 4), clarinetmusicmaker\_three],

[(165, 4), (168, 4), clarinetmusicmaker\_three],

[(168, 4), (170, 4), clarinetmusicmaker\_three],

[(174, 4), (175, 4), clarinetmusicmaker\_three],

[(175, 4), (177, 4), clarinetmusicmaker\_three],

[(179, 4), (180, 4), clarinetmusicmaker\_three],

[(185, 4), (186, 4), clarinetmusicmaker\_three],

[(186, 4), (190, 4), clarinetmusicmaker\_three],

]

])

voice\_9\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 9',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(2, 4), (5, 4), violin2musicmaker\_one],

[(9, 4), (11, 4), violin2musicmaker\_two],

[(11, 4), (13, 4), violin2musicmaker\_two],

[(16, 4), (18, 4), violin2musicmaker\_three],

[(21, 4), (22, 4), violin2musicmaker\_one],

[(22, 4), (23, 4), violin2musicmaker\_one],

[(35, 4), (40, 4), violin2musicmaker\_one],

[(44, 4), (46, 4), violin2musicmaker\_two],

[(46, 4), (47, 4), violin2musicmaker\_two],

[(49, 4), (50, 4), violin2musicmaker\_three],

[(55, 4), (59, 4), violin2musicmaker\_one],

[(62, 4), (64, 4), violin2musicmaker\_two],

[(65, 4), (67, 4), violin2musicmaker\_three],

[(67, 4), (70, 4), violin2musicmaker\_three],

[(70, 4), (71, 4), violin2musicmaker\_three],

[(73, 4), (75, 4), violin2musicmaker\_two],

[(75, 4), (76, 4), violin2musicmaker\_two],

[(80, 4), (82, 4), violin2musicmaker\_one],

[(82, 4), (85, 4), violin2musicmaker\_one],

[(86, 4), (88, 4), violin2musicmaker\_two],

[(91, 4), (94, 4), violin2musicmaker\_three],

[(94, 4), (95, 4), violin2musicmaker\_three],

[(100, 4), (101, 4), violin2musicmaker\_two],

[(103, 4), (104, 4), violin2musicmaker\_one],

[(104, 4), (106, 4), violin2musicmaker\_one],

[(110, 4), (114, 4), violin2musicmaker\_one],

[(115, 4), (119, 4), violin2musicmaker\_one],

[(120, 4), (123, 4), violin2musicmaker\_one],

[(123, 4), (124, 4), violin2musicmaker\_one],

[(125, 4), (126, 4), violin2musicmaker\_two],

[(129, 4), (131, 4), violin2musicmaker\_two],

[(131, 4), (134, 4), violin2musicmaker\_two],

[(141, 4), (144, 4), violin2musicmaker\_two],

[(149, 4), (150, 4), violin2musicmaker\_two],

[(154, 4), (157, 4), violin2musicmaker\_three],

[(159, 4), (160, 4), violin2musicmaker\_three],

[(165, 4), (168, 4), violin2musicmaker\_three],

[(168, 4), (169, 4), violin2musicmaker\_three],

[(172, 4), (174, 4), violin2musicmaker\_three],

[(175, 4), (179, 4), violin2musicmaker\_three],

[(179, 4), (180, 4), violin2musicmaker\_three],

[(184, 4), (186, 4), violin2musicmaker\_three],

[(186, 4), (190, 4), violin2musicmaker\_three],

]

])

voice\_10\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 10',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(2, 4), (5, 4), violamusicmaker\_one],

[(9, 4), (11, 4), violamusicmaker\_two],

[(11, 4), (13, 4), violamusicmaker\_two],

[(17, 4), (18, 4), violamusicmaker\_three],

[(21, 4), (22, 4), violamusicmaker\_one],

[(22, 4), (25, 4), violamusicmaker\_one],

[(29, 4), (30, 4), violamusicmaker\_two],

[(30, 4), (32, 4), violamusicmaker\_two],

[(35, 4), (40, 4), violamusicmaker\_one],

[(44, 4), (46, 4), violamusicmaker\_two],

[(46, 4), (47, 4), violamusicmaker\_two],

[(49, 4), (50, 4), violamusicmaker\_three],

[(55, 4), (59, 4), violamusicmaker\_one],

[(62, 4), (64, 4), violamusicmaker\_two],

[(65, 4), (67, 4), violamusicmaker\_three],

[(67, 4), (70, 4), violamusicmaker\_three],

[(70, 4), (71, 4), violamusicmaker\_three],

[(73, 4), (75, 4), violamusicmaker\_two],

[(75, 4), (76, 4), violamusicmaker\_two],

[(80, 4), (82, 4), violamusicmaker\_one],

[(82, 4), (85, 4), violamusicmaker\_one],

[(86, 4), (88, 4), violamusicmaker\_two],

[(91, 4), (94, 4), violamusicmaker\_three],

[(94, 4), (95, 4), violamusicmaker\_three],

[(100, 4), (101, 4), violamusicmaker\_two],

[(103, 4), (104, 4), violamusicmaker\_one],

[(104, 4), (106, 4), violamusicmaker\_one],

[(110, 4), (114, 4), violamusicmaker\_one],

[(115, 4), (119, 4), violamusicmaker\_one],

[(120, 4), (123, 4), violamusicmaker\_one],

[(123, 4), (124, 4), violamusicmaker\_one],

[(125, 4), (126, 4), violamusicmaker\_two],

[(129, 4), (131, 4), violamusicmaker\_two],

[(131, 4), (134, 4), violamusicmaker\_two],

[(141, 4), (144, 4), violamusicmaker\_two],

[(149, 4), (150, 4), violamusicmaker\_two],

[(153, 4), (154, 4), violamusicmaker\_three],

[(154, 4), (155, 4), violamusicmaker\_three],

[(156, 4), (159, 4), violamusicmaker\_three],

[(159, 4), (161, 4), violamusicmaker\_three],

[(165, 4), (168, 4), violamusicmaker\_three],

[(170, 4), (171, 4), violamusicmaker\_three],

[(176, 4), (179, 4), violamusicmaker\_three],

[(179, 4), (180, 4), violamusicmaker\_three],

[(183, 4), (185, 4), violamusicmaker\_three],

[(186, 4), (190, 4), violamusicmaker\_three],

]

])

###group three###

voice\_3\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 3',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(7, 4), (11, 4), bassoonmusicmaker\_one],

[(15, 4), (16, 4), bassoonmusicmaker\_two],

[(19, 4), (22, 4), bassoonmusicmaker\_three],

[(22, 4), (23, 4), bassoonmusicmaker\_three],

[(27, 4), (30, 4), bassoonmusicmaker\_one],

[(32, 4), (35, 4), bassoonmusicmaker\_two],

[(35, 4), (36, 4), bassoonmusicmaker\_three],

[(37, 4), (40, 4), bassoonmusicmaker\_two],

[(40, 4), (42, 4), bassoonmusicmaker\_two],

[(46, 4), (49, 4), bassoonmusicmaker\_one],

[(51, 4), (52, 4), bassoonmusicmaker\_three],

[(57, 4), (59, 4), bassoonmusicmaker\_two],

[(59, 4), (61, 4), bassoonmusicmaker\_two],

[(64, 4), (66, 4), bassoonmusicmaker\_one],

[(67, 4), (70, 4), bassoonmusicmaker\_three],

[(70, 4), (72, 4), bassoonmusicmaker\_one],

[(72, 4), (73, 4), bassoonmusicmaker\_one],

[(77, 4), (79, 4), bassoonmusicmaker\_two],

[(79, 4), (82, 4), bassoonmusicmaker\_two],

[(83, 4), (85, 4), bassoonmusicmaker\_three],

[(88, 4), (89, 4), bassoonmusicmaker\_two],

[(89, 4), (92, 4), bassoonmusicmaker\_two],

[(97, 4), (98, 4), bassoonmusicmaker\_one],

[(100, 4), (103, 4), bassoonmusicmaker\_two],

[(107, 4), (110, 4), bassoonmusicmaker\_three],

[(110, 4), (112, 4), bassoonmusicmaker\_one],

[(113, 4), (114, 4), bassoonmusicmaker\_one],

[(114, 4), (117, 4), bassoonmusicmaker\_one],

[(118, 4), (119, 4), bassoonmusicmaker\_one],

[(119, 4), (122, 4), bassoonmusicmaker\_one],

[(123, 4), (125, 4), bassoonmusicmaker\_one],

[(126, 4), (131, 4), bassoonmusicmaker\_two],

[(138, 4), (141, 4), bassoonmusicmaker\_two],

[(146, 4), (150, 4), bassoonmusicmaker\_two],

[(150, 4), (154, 4), bassoonmusicmaker\_three],

[(154, 4), (155, 4), bassoonmusicmaker\_three],

[(159, 4), (162, 4), bassoonmusicmaker\_three],

[(164, 4), (165, 4), bassoonmusicmaker\_three],

[(170, 4), (172, 4), bassoonmusicmaker\_three],

[(172, 4), (174, 4), bassoonmusicmaker\_three],

[(177, 4), (179, 4), bassoonmusicmaker\_three],

[(180, 4), (183, 4), bassoonmusicmaker\_three],

[(183, 4), (185, 4), bassoonmusicmaker\_three],

[(186, 4), (190, 4), bassoonmusicmaker\_three],

]

])

voice\_6\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 6',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(7, 4), (11, 4), trombonemusicmaker\_one],

[(14, 4), (16, 4), trombonemusicmaker\_two],

[(19, 4), (22, 4), trombonemusicmaker\_three],

[(22, 4), (23, 4), trombonemusicmaker\_three],

[(27, 4), (29, 4), trombonemusicmaker\_one],

[(35, 4), (36, 4), trombonemusicmaker\_three],

[(37, 4), (40, 4), trombonemusicmaker\_two],

[(40, 4), (42, 4), trombonemusicmaker\_two],

[(46, 4), (49, 4), trombonemusicmaker\_one],

[(51, 4), (52, 4), trombonemusicmaker\_three],

[(57, 4), (59, 4), trombonemusicmaker\_two],

[(59, 4), (61, 4), trombonemusicmaker\_two],

[(64, 4), (66, 4), trombonemusicmaker\_one],

[(67, 4), (70, 4), trombonemusicmaker\_three],

[(70, 4), (72, 4), trombonemusicmaker\_one],

[(72, 4), (73, 4), trombonemusicmaker\_one],

[(77, 4), (79, 4), trombonemusicmaker\_two],

[(79, 4), (82, 4), trombonemusicmaker\_two],

[(83, 4), (85, 4), trombonemusicmaker\_three],

[(88, 4), (89, 4), trombonemusicmaker\_two],

[(89, 4), (92, 4), trombonemusicmaker\_two],

[(97, 4), (98, 4), trombonemusicmaker\_one],

[(100, 4), (103, 4), trombonemusicmaker\_two],

[(107, 4), (110, 4), trombonemusicmaker\_three],

[(110, 4), (112, 4), trombonemusicmaker\_one],

[(113, 4), (114, 4), trombonemusicmaker\_one],

[(114, 4), (117, 4), trombonemusicmaker\_one],

[(118, 4), (119, 4), trombonemusicmaker\_one],

[(119, 4), (122, 4), trombonemusicmaker\_one],

[(123, 4), (125, 4), trombonemusicmaker\_one],

[(126, 4), (131, 4), trombonemusicmaker\_two],

[(138, 4), (141, 4), trombonemusicmaker\_two],

[(146, 4), (150, 4), trombonemusicmaker\_two],

[(150, 4), (154, 4), trombonemusicmaker\_three],

[(157, 4), (159, 4), trombonemusicmaker\_three],

[(160, 4), (164, 4), trombonemusicmaker\_three],

[(164, 4), (165, 4), trombonemusicmaker\_three],

[(169, 4), (172, 4), trombonemusicmaker\_three],

[(174, 4), (175, 4), trombonemusicmaker\_three],

[(180, 4), (183, 4), trombonemusicmaker\_three],

[(183, 4), (184, 4), trombonemusicmaker\_three],

[(186, 4), (190, 4), trombonemusicmaker\_three],

]

])

voice\_11\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 11',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(7, 4), (11, 4), cellomusicmaker\_one],

[(14, 4), (16, 4), cellomusicmaker\_two],

[(21, 4), (22, 4), cellomusicmaker\_three],

[(22, 4), (23, 4), cellomusicmaker\_three],

[(27, 4), (30, 4), cellomusicmaker\_one],

[(35, 4), (36, 4), cellomusicmaker\_three],

[(37, 4), (40, 4), cellomusicmaker\_two],

[(40, 4), (42, 4), cellomusicmaker\_two],

[(46, 4), (49, 4), cellomusicmaker\_one],

[(51, 4), (52, 4), cellomusicmaker\_three],

[(57, 4), (59, 4), cellomusicmaker\_two],

[(59, 4), (61, 4), cellomusicmaker\_two],

[(64, 4), (66, 4), cellomusicmaker\_one],

[(67, 4), (70, 4), cellomusicmaker\_three],

[(70, 4), (72, 4), cellomusicmaker\_one],

[(72, 4), (73, 4), cellomusicmaker\_one],

[(77, 4), (79, 4), cellomusicmaker\_two],

[(79, 4), (82, 4), cellomusicmaker\_two],

[(83, 4), (85, 4), cellomusicmaker\_three],

[(88, 4), (89, 4), cellomusicmaker\_two],

[(89, 4), (92, 4), cellomusicmaker\_two],

[(97, 4), (98, 4), cellomusicmaker\_one],

[(100, 4), (103, 4), cellomusicmaker\_two],

[(107, 4), (110, 4), cellomusicmaker\_three],

[(110, 4), (112, 4), cellomusicmaker\_one],

[(113, 4), (114, 4), cellomusicmaker\_one],

[(114, 4), (117, 4), cellomusicmaker\_one],

[(118, 4), (119, 4), cellomusicmaker\_one],

[(119, 4), (122, 4), cellomusicmaker\_one],

[(123, 4), (125, 4), cellomusicmaker\_one],

[(126, 4), (131, 4), cellomusicmaker\_two],

[(138, 4), (141, 4), cellomusicmaker\_two],

[(146, 4), (150, 4), cellomusicmaker\_two],

[(150, 4), (153, 4), cellomusicmaker\_three],

[(155, 4), (156, 4), cellomusicmaker\_three],

[(161, 4), (164, 4), cellomusicmaker\_three],

[(164, 4), (165, 4), cellomusicmaker\_three],

[(168, 4), (170, 4), cellomusicmaker\_three],

[(171, 4), (172, 4), cellomusicmaker\_three],

[(172, 4), (175, 4), cellomusicmaker\_three],

[(175, 4), (176, 4), cellomusicmaker\_three],

[(180, 4), (183, 4), cellomusicmaker\_three],

[(185, 4), (186, 4), cellomusicmaker\_three],

[(186, 4), (190, 4), cellomusicmaker\_three],

]

])

###group four###

voice\_4\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 4',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(0, 4), (5, 4), hornmusicmaker\_one],

[(8, 4), (10, 4), hornmusicmaker\_two],

[(14, 4), (18, 4), hornmusicmaker\_three],

[(21, 4), (22, 4), hornmusicmaker\_one],

[(22, 4), (23, 4), hornmusicmaker\_one],

[(38, 4), (40, 4), hornmusicmaker\_two],

[(41, 4), (43, 4), hornmusicmaker\_one],

[(43, 4), (46, 4), hornmusicmaker\_one],

[(50, 4), (53, 4), hornmusicmaker\_three],

[(55, 4), (56, 4), hornmusicmaker\_two],

[(61, 4), (64, 4), hornmusicmaker\_one],

[(64, 4), (65, 4), hornmusicmaker\_one],

[(68, 4), (70, 4), hornmusicmaker\_three],

[(70, 4), (72, 4), hornmusicmaker\_two],

[(72, 4), (74, 4), hornmusicmaker\_two],

[(79, 4), (80, 4), hornmusicmaker\_three],

[(82, 4), (85, 4), hornmusicmaker\_two],

[(89, 4), (94, 4), hornmusicmaker\_one],

[(95, 4), (97, 4), hornmusicmaker\_two],

[(100, 4), (104, 4), hornmusicmaker\_three],

[(109, 4), (110, 4), hornmusicmaker\_two],

[(110, 4), (111, 4), hornmusicmaker\_one],

[(112, 4), (114, 4), hornmusicmaker\_one],

[(114, 4), (116, 4), hornmusicmaker\_one],

[(117, 4), (119, 4), hornmusicmaker\_one],

[(119, 4), (121, 4), hornmusicmaker\_one],

[(122, 4), (123, 4), hornmusicmaker\_one],

[(123, 4), (125, 4), hornmusicmaker\_one],

[(133, 4), (136, 4), hornmusicmaker\_two],

[(142, 4), (146, 4), hornmusicmaker\_two],

[(146, 4), (150, 4), hornmusicmaker\_two],

[(153, 4), (154, 4), hornmusicmaker\_three],

[(154, 4), (155, 4), hornmusicmaker\_three],

[(159, 4), (162, 4), hornmusicmaker\_three],

[(164, 4), (168, 4), hornmusicmaker\_three],

[(171, 4), (172, 4), hornmusicmaker\_three],

[(172, 4), (173, 4), hornmusicmaker\_three],

[(177, 4), (179, 4), hornmusicmaker\_three],

[(179, 4), (180, 4), hornmusicmaker\_three],

[(182, 4), (183, 4), hornmusicmaker\_three],

[(183, 4), (186, 4), hornmusicmaker\_three],

[(186, 4), (190, 4), hornmusicmaker\_three],

]

])

voice\_7\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 7',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(0, 4), (5, 4), tubamusicmaker\_one],

[(8, 4), (10, 4), tubamusicmaker\_two],

[(14, 4), (18, 4), tubamusicmaker\_three],

[(21, 4), (22, 4), tubamusicmaker\_one],

[(22, 4), (23, 4), tubamusicmaker\_one],

[(26, 4), (30, 4), tubamusicmaker\_two],

[(38, 4), (40, 4), tubamusicmaker\_two],

[(41, 4), (43, 4), tubamusicmaker\_one],

[(43, 4), (46, 4), tubamusicmaker\_one],

[(50, 4), (53, 4), tubamusicmaker\_three],

[(55, 4), (56, 4), tubamusicmaker\_two],

[(61, 4), (64, 4), tubamusicmaker\_one],

[(64, 4), (65, 4), tubamusicmaker\_one],

[(68, 4), (70, 4), tubamusicmaker\_three],

[(70, 4), (72, 4), tubamusicmaker\_two],

[(72, 4), (74, 4), tubamusicmaker\_two],

[(79, 4), (80, 4), tubamusicmaker\_three],

[(82, 4), (85, 4), tubamusicmaker\_two],

[(89, 4), (94, 4), tubamusicmaker\_one],

[(95, 4), (97, 4), tubamusicmaker\_two],

[(100, 4), (104, 4), tubamusicmaker\_three],

[(109, 4), (110, 4), tubamusicmaker\_two],

[(110, 4), (111, 4), tubamusicmaker\_one],

[(112, 4), (114, 4), tubamusicmaker\_one],

[(114, 4), (116, 4), tubamusicmaker\_one],

[(117, 4), (119, 4), tubamusicmaker\_one],

[(119, 4), (121, 4), tubamusicmaker\_one],

[(122, 4), (123, 4), tubamusicmaker\_one],

[(123, 4), (125, 4), tubamusicmaker\_one],

[(133, 4), (136, 4), tubamusicmaker\_two],

[(142, 4), (146, 4), tubamusicmaker\_two],

[(146, 4), (150, 4), tubamusicmaker\_two],

[(154, 4), (157, 4), tubamusicmaker\_three],

[(159, 4), (163, 4), tubamusicmaker\_three],

[(166, 4), (168, 4), tubamusicmaker\_three],

[(172, 4), (175, 4), tubamusicmaker\_three],

[(177, 4), (179, 4), tubamusicmaker\_three],

[(179, 4), (181, 4), tubamusicmaker\_three],

[(184, 4), (186, 4), tubamusicmaker\_three],

[(186, 4), (190, 4), tubamusicmaker\_three],

]

])

voice\_12\_timespan\_list = abjad.TimespanList([

abjad.AnnotatedTimespan(

start\_offset=start\_offset,

stop\_offset=stop\_offset,

annotation=MusicSpecifier(

music\_maker=music\_maker,

voice\_name='Voice 12',

),

)

for start\_offset, stop\_offset, music\_maker in [

[(0, 4), (5, 4), bassmusicmaker\_one],

[(8, 4), (10, 4), bassmusicmaker\_two],

[(14, 4), (18, 4), bassmusicmaker\_three],

[(21, 4), (22, 4), bassmusicmaker\_one],

[(22, 4), (23, 4), bassmusicmaker\_one],

[(38, 4), (40, 4), bassmusicmaker\_two],

[(41, 4), (43, 4), bassmusicmaker\_one],

[(43, 4), (46, 4), bassmusicmaker\_one],

[(50, 4), (53, 4), bassmusicmaker\_three],

[(55, 4), (56, 4), bassmusicmaker\_two],

[(61, 4), (64, 4), bassmusicmaker\_one],

[(64, 4), (65, 4), bassmusicmaker\_one],

[(68, 4), (70, 4), bassmusicmaker\_three],

[(70, 4), (72, 4), bassmusicmaker\_two],

[(72, 4), (74, 4), bassmusicmaker\_two],

[(79, 4), (80, 4), bassmusicmaker\_three],

[(82, 4), (85, 4), bassmusicmaker\_two],

[(89, 4), (94, 4), bassmusicmaker\_one],

[(95, 4), (97, 4), bassmusicmaker\_two],

[(100, 4), (104, 4), bassmusicmaker\_three],

[(109, 4), (110, 4), bassmusicmaker\_two],

[(110, 4), (111, 4), bassmusicmaker\_one],

[(112, 4), (114, 4), bassmusicmaker\_one],

[(114, 4), (116, 4), bassmusicmaker\_one],

[(117, 4), (119, 4), bassmusicmaker\_one],

[(119, 4), (121, 4), bassmusicmaker\_one],

[(122, 4), (123, 4), bassmusicmaker\_one],

[(123, 4), (125, 4), bassmusicmaker\_one],

[(133, 4), (136, 4), bassmusicmaker\_two],

[(142, 4), (146, 4), bassmusicmaker\_two],

[(146, 4), (150, 4), bassmusicmaker\_two],

[(152, 4), (154, 4), bassmusicmaker\_three],

[(154, 4), (156, 4), bassmusicmaker\_three],

[(159, 4), (161, 4), bassmusicmaker\_three],

[(165, 4), (168, 4), bassmusicmaker\_three],

[(170, 4), (172, 4), bassmusicmaker\_three],

[(172, 4), (174, 4), bassmusicmaker\_three],

[(177, 4), (179, 4), bassmusicmaker\_three],

[(183, 4), (186, 4), bassmusicmaker\_three],

[(186, 4), (190, 4), bassmusicmaker\_three],

]

])

# Create a dictionary mapping voice names to timespan lists so we can

# maintain the association in later operations:

all\_timespan\_lists = {

'Voice 1': voice\_1\_timespan\_list,

'Voice 2': voice\_2\_timespan\_list,

'Voice 3': voice\_3\_timespan\_list,

'Voice 4': voice\_4\_timespan\_list,

'Voice 5': voice\_5\_timespan\_list,

'Voice 6': voice\_6\_timespan\_list,

'Voice 7': voice\_7\_timespan\_list,

'Voice 8': voice\_8\_timespan\_list,

'Voice 9': voice\_9\_timespan\_list,

'Voice 10': voice\_10\_timespan\_list,

'Voice 11': voice\_11\_timespan\_list,

'Voice 12': voice\_12\_timespan\_list,

}

# Determine the "global" timespan of all voices combined:

global\_timespan = abjad.Timespan(

start\_offset=0,

stop\_offset=max(\_.stop\_offset for \_ in all\_timespan\_lists.values())

)

# Using the global timespan, create silence timespans for each timespan list.

# We don't need to create any silences by-hand if we now the global start and

# stop offsets of all voices combined:

for voice\_name, timespan\_list in all\_timespan\_lists.items():

# Here is another technique for finding where the silence timespans are. We

# create a new timespan list consisting of the global timespan and all the

# timespans from our current per-voice timespan list. Then we compute an

# in-place logical XOR. The XOR will replace the contents of the "silences"

# timespan list with a set of timespans representing those periods of time

# where only one timespan from the original was present. This has the

# effect of cutting out holes from the global timespan wherever a per-voice

# timespan was found, but also preserves any silence before the first

# per-voice timespan or after the last per-voice timespan. Then we merge

# the newly-created silences back into the per-voice timespan list.

silences = abjad.TimespanList([global\_timespan])

silences.extend(timespan\_list)

silences.sort()

silences.compute\_logical\_xor()

# Add the silences into the voice timespan list. We create new \*annotated\*

# timespans so we can maintain the voice name information:

for silence\_timespan in silences:

timespan\_list.append(

abjad.AnnotatedTimespan(

start\_offset=silence\_timespan.start\_offset,

stop\_offset=silence\_timespan.stop\_offset,

annotation=MusicSpecifier(

music\_maker=None,

voice\_name=voice\_name,

),

)

)

timespan\_list.sort()

# Split the timespan list via the time signatures and collect the shards into a

# new timespan list

for voice\_name, timespan\_list in all\_timespan\_lists.items():

shards = timespan\_list.split\_at\_offsets(bounds)

split\_timespan\_list = abjad.TimespanList()

for shard in shards:

split\_timespan\_list.extend(shard)

split\_timespan\_list.sort()

# We can replace the original timespan list in the dictionary of

# timespan lists because we know the key it was stored at (its voice

# name):

all\_timespan\_lists[voice\_name] = timespan\_list

# Create a score structure

score = abjad.Score([

abjad.Staff(lilypond\_type='TimeSignatureContext', name='Global Context 1'),

abjad.StaffGroup(

[

abjad.Staff([abjad.Voice(name='Voice 1')],name='Staff 1', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 2')],name='Staff 2', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 3')],name='Staff 3', lilypond\_type='Staff',),

],

name='Staff Group 1',

),

abjad.Staff(lilypond\_type='TimeSignatureContext', name='Global Context 2'),

abjad.StaffGroup(

[

abjad.Staff([abjad.Voice(name='Voice 4')],name='Staff 4', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 5')],name='Staff 5', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 6')],name='Staff 6', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 7')],name='Staff 7', lilypond\_type='Staff',),

],

name='Staff Group 2',

),

abjad.Staff(lilypond\_type='TimeSignatureContext', name='Global Context 3'),

abjad.StaffGroup(

[

abjad.Staff([abjad.Voice(name='Voice 8')],name='Staff 8', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 9')],name='Staff 9', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 10')],name='Staff 10', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 11')],name='Staff 11', lilypond\_type='Staff',),

abjad.Staff([abjad.Voice(name='Voice 12')],name='Staff 12', lilypond\_type='Staff',),

],

name='Staff Group 3',

)

],

)

# Teach each of the staves how to draw analysis brackets

# for staff in score['Staff Group']:

# staff.consists\_commands.append('Horizontal\_bracket\_engraver')

# Add skips and time signatures to the global context

for time\_signature in time\_signatures:

skip = abjad.Skip(1, multiplier=(time\_signature))

abjad.attach(time\_signature, skip)

score['Global Context 1'].append(skip)

for time\_signature in time\_signatures:

skip = abjad.Skip(1, multiplier=(time\_signature))

abjad.attach(time\_signature, skip)

score['Global Context 2'].append(skip)

for time\_signature in time\_signatures:

skip = abjad.Skip(1, multiplier=(time\_signature))

abjad.attach(time\_signature, skip)

score['Global Context 3'].append(skip)

# Define a helper function that takes a rhythm maker and some durations and

# outputs a container. This helper function also adds LilyPond analysis

# brackets to make it clearer where the phrase and sub-phrase boundaries are.

print('Making containers ...')

def make\_container(music\_maker, durations):

selections = music\_maker(durations)

container = abjad.Container([])

container.extend(selections)

return container

# Loop over the timespan list dictionaries, spitting out pairs of voice

# names and per-voice timespan lists. Group timespans into phrases, with

# all timespans in each phrase having an identical rhythm maker. Run the

# rhythm maker against the durations of the timespans in the phrase and

# add the output to the voice with the timespan lists's voice name.

def key\_function(timespan):

"""

Get the timespan's annotation's rhythm-maker.

If the annotation's rhythm-maker is None, return the silence maker.

"""

return timespan.annotation.music\_maker or silence\_maker

for voice\_name, timespan\_list in all\_timespan\_lists.items():

for music\_maker, grouper in itertools.groupby(

timespan\_list,

key=key\_function,

):

durations = [timespan.duration for timespan in grouper]

container = make\_container(music\_maker, durations)

voice = score[voice\_name]

voice.append(container)

print('Splitting and rewriting ...')

# split and rewite meters

for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):

for i , shard in enumerate(abjad.mutate(voice[:]).split(time\_signatures)):

time\_signature = time\_signatures[i]

abjad.mutate(shard).rewrite\_meter(time\_signature)

for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):

for i , shard in enumerate(abjad.mutate(voice[:]).split(time\_signatures)):

time\_signature = time\_signatures[i]

abjad.mutate(shard).rewrite\_meter(time\_signature)

for voice in abjad.iterate(score['Staff Group 3']).components(abjad.Voice):

for i , shard in enumerate(abjad.mutate(voice[:]).split(time\_signatures)):

time\_signature = time\_signatures[i]

abjad.mutate(shard).rewrite\_meter(time\_signature)

print('Beaming runs ...')

for voice in abjad.select(score).components(abjad.Voice):

for run in abjad.select(voice).runs():

if 1 < len(run):

# use a beam\_specifier to remove beam indicators from run

specifier = abjadext.rmakers.BeamSpecifier(

beam\_each\_division=False,

)

specifier(run)

# then attach new indicators at the 0 and -1 of run

abjad.attach(abjad.StartBeam(), run[0])

abjad.attach(abjad.StopBeam(), run[-1])

for leaf in run:

# continue if leaf can't be beamed

if abjad.Duration(1, 4) <= leaf.written\_duration:

continue

previous\_leaf = abjad.inspect(leaf).leaf(-1)

next\_leaf = abjad.inspect(leaf).leaf(1)

# if next leaf is quarter note (or greater) ...

if (isinstance(next\_leaf, (abjad.Chord, abjad.Note)) and

abjad.Duration(1, 4) <= next\_leaf.written\_duration):

left = previous\_leaf.written\_duration.flag\_count

right = leaf.written\_duration.flag\_count # right-pointing nib

beam\_count = abjad.BeamCount(

left=left,

right=right,

)

abjad.attach(beam\_count, leaf)

continue

# if previous leaf is quarter note (or greater) ...

if (isinstance(previous\_leaf, (abjad.Chord, abjad.Note)) and

abjad.Duration(1, 4) <= previous\_leaf.written\_duration):

left = leaf.written\_duration.flag\_count # left-pointing nib

right = next\_leaf.written\_duration.flag\_count

beam\_count = abjad.BeamCount(

left=left,

right=right,

)

abjad.attach(beam\_count, leaf)

print('Beautifying score ...')

# cutaway score

for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):

for selection in abjad.select(staff).components(abjad.Rest).group\_by\_contiguity():

start\_command = abjad.LilyPondLiteral(

r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \startStaff',

format\_slot='before',

)

stop\_command = abjad.LilyPondLiteral(

r'\stopStaff \startStaff',

format\_slot='after',

)

abjad.attach(start\_command, selection[0])

abjad.attach(stop\_command, selection[-1])

for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):

for selection in abjad.select(staff).components(abjad.Rest).group\_by\_contiguity():

start\_command = abjad.LilyPondLiteral(

r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \startStaff',

format\_slot='before',

)

stop\_command = abjad.LilyPondLiteral(

r'\stopStaff \startStaff',

format\_slot='after',

)

abjad.attach(start\_command, selection[0])

abjad.attach(stop\_command, selection[-1])

for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):

for selection in abjad.select(staff).components(abjad.Rest).group\_by\_contiguity():

start\_command = abjad.LilyPondLiteral(

r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \startStaff',

format\_slot='before',

)

stop\_command = abjad.LilyPondLiteral(

r'\stopStaff \startStaff',

format\_slot='after',

)

abjad.attach(start\_command, selection[0])

abjad.attach(stop\_command, selection[-1])

print('Stopping Hairpins ...')

for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):

for rest in abjad.iterate(staff).components(abjad.Rest):

previous\_leaf = abjad.inspect(rest).leaf(-1)

if isinstance(previous\_leaf, abjad.Note):

abjad.attach(abjad.StopHairpin(), rest)

elif isinstance(previous\_leaf, abjad.Chord):

abjad.attach(abjad.StopHairpin(), rest)

elif isinstance(previous\_leaf, abjad.Rest):

pass

for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):

for rest in abjad.iterate(staff).components(abjad.Rest):

previous\_leaf = abjad.inspect(rest).leaf(-1)

if isinstance(previous\_leaf, abjad.Note):

abjad.attach(abjad.StopHairpin(), rest)

elif isinstance(previous\_leaf, abjad.Chord):

abjad.attach(abjad.StopHairpin(), rest)

elif isinstance(previous\_leaf, abjad.Rest):

pass

for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):

for rest in abjad.iterate(staff).components(abjad.Rest):

previous\_leaf = abjad.inspect(rest).leaf(-1)

if isinstance(previous\_leaf, abjad.Note):

abjad.attach(abjad.StopHairpin(), rest)

elif isinstance(previous\_leaf, abjad.Chord):

abjad.attach(abjad.StopHairpin(), rest)

elif isinstance(previous\_leaf, abjad.Rest):

pass

# Make pitches

print('Adding pitch material ...')

def cyc(lst):

count = 0

while True:

yield lst[count%len(lst)]

count += 1

#attach instruments and clefs

print('Adding attachments ...')

bar\_line = abjad.BarLine('|.')

metro = abjad.MetronomeMark((1, 4), 120)

markup1 = abjad.Markup(r'\bold { S }')

markup2 = abjad.Markup(r'\bold { T }')

markup3 = abjad.Markup(r'\bold { U }')

markup4 = abjad.Markup(r'\bold { V }')

markup5 = abjad.Markup(r'\bold { W }')

markup6 = abjad.Markup(r'\bold { X }')

mark1 = abjad.RehearsalMark(markup=markup1)

mark2 = abjad.RehearsalMark(markup=markup2)

mark3 = abjad.RehearsalMark(markup=markup3)

mark4 = abjad.RehearsalMark(markup=markup4)

mark5 = abjad.RehearsalMark(markup=markup5)

mark6 = abjad.RehearsalMark(markup=markup6)

instruments1 = cyc([

abjad.Flute(),

abjad.ClarinetInBFlat(),

abjad.Bassoon(),

])

instruments2 = cyc([

abjad.FrenchHorn(),

abjad.Trumpet(),

abjad.TenorTrombone(),

abjad.Tuba(),

])

instruments3 = cyc([

abjad.Violin(),

abjad.Violin(),

abjad.Viola(),

abjad.Cello(),

abjad.Contrabass(),

])

clefs1 = cyc([

abjad.Clef('treble'),

abjad.Clef('treble'),

abjad.Clef('bass'),

])

clefs2 = cyc([

abjad.Clef('treble'),

abjad.Clef('treble'),

abjad.Clef('bass'),

abjad.Clef('bass'),

])

clefs3 = cyc([

abjad.Clef('treble'),

abjad.Clef('treble'),

abjad.Clef('alto'),

abjad.Clef('bass'),

abjad.Clef('bass'),

])

abbreviations1 = cyc([

abjad.MarginMarkup(markup=abjad.Markup('fl.'),),

abjad.MarginMarkup(markup=abjad.Markup('cl.'),),

abjad.MarginMarkup(markup=abjad.Markup('bssn.'),),

])

abbreviations2 = cyc([

abjad.MarginMarkup(markup=abjad.Markup('hr.'),),

abjad.MarginMarkup(markup=abjad.Markup('trp.'),),

abjad.MarginMarkup(markup=abjad.Markup('trmb.'),),

abjad.MarginMarkup(markup=abjad.Markup('tb.'),),

])

abbreviations3 = cyc([

abjad.MarginMarkup(markup=abjad.Markup('vln.I'),),

abjad.MarginMarkup(markup=abjad.Markup('vln.II'),),

abjad.MarginMarkup(markup=abjad.Markup('vla.'),),

abjad.MarginMarkup(markup=abjad.Markup('vc.'),),

abjad.MarginMarkup(markup=abjad.Markup('cb.'),),

])

names1 = cyc([

abjad.StartMarkup(markup=abjad.Markup('Flute'),),

abjad.StartMarkup(markup=abjad.Markup('Clarinet'),),

abjad.StartMarkup(markup=abjad.Markup('Bassoon'),),

])

names2 = cyc([

abjad.StartMarkup(markup=abjad.Markup('Horn'),),

abjad.StartMarkup(markup=abjad.Markup('Trumpet'),),

abjad.StartMarkup(markup=abjad.Markup('Trombone'),),

abjad.StartMarkup(markup=abjad.Markup('Tuba'),),

])

names3 = cyc([

abjad.StartMarkup(markup=abjad.Markup('Violin I'),),

abjad.StartMarkup(markup=abjad.Markup('Violin II'),),

abjad.StartMarkup(markup=abjad.Markup('Viola'),),

abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),

abjad.StartMarkup(markup=abjad.Markup('Contrabass'),),

])

for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):

leaf1 = abjad.select(staff).leaves()[0]

abjad.attach(next(instruments1), leaf1)

abjad.attach(next(abbreviations1), leaf1)

abjad.attach(next(names1), leaf1)

abjad.attach(next(clefs1), leaf1)

for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):

leaf1 = abjad.select(staff).leaves()[0]

abjad.attach(next(instruments2), leaf1)

abjad.attach(next(abbreviations2), leaf1)

abjad.attach(next(names2), leaf1)

abjad.attach(next(clefs2), leaf1)

for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):

leaf1 = abjad.select(staff).leaves()[0]

abjad.attach(next(instruments3), leaf1)

abjad.attach(next(abbreviations3), leaf1)

abjad.attach(next(names3), leaf1)

abjad.attach(next(clefs3), leaf1)

for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:

leaf1 = abjad.select(staff).leaves()[0]

last\_leaf = abjad.select(staff).leaves()[-1]

abjad.attach(metro, leaf1)

abjad.attach(bar\_line, last\_leaf)

for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff)[0]:

leaf1 = abjad.select(staff).leaves()[0]

last\_leaf = abjad.select(staff).leaves()[-1]

abjad.attach(metro, leaf1)

abjad.attach(bar\_line, last\_leaf)

for staff in abjad.select(score['Staff Group 3']).components(abjad.Staff)[0]:

leaf1 = abjad.select(staff).leaves()[0]

last\_leaf = abjad.select(staff).leaves()[-1]

abjad.attach(metro, leaf1)

abjad.attach(bar\_line, last\_leaf)

for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):

leaf1 = abjad.select(staff).leaves()[7]

abjad.attach(mark1, leaf1)

for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):

leaf1 = abjad.select(staff).leaves()[7]

abjad.attach(mark1, leaf1)

for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):

leaf1 = abjad.select(staff).leaves()[7]

abjad.attach(mark1, leaf1)

for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):

leaf2 = abjad.select(staff).leaves()[16]

abjad.attach(mark2, leaf2)

for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):

leaf2 = abjad.select(staff).leaves()[16]

abjad.attach(mark2, leaf2)

for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):

leaf2 = abjad.select(staff).leaves()[16]

abjad.attach(mark2, leaf2)

for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):

leaf3 = abjad.select(staff).leaves()[22]

abjad.attach(mark3, leaf3)

for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):

leaf3 = abjad.select(staff).leaves()[22]

abjad.attach(mark3, leaf3)

for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):

leaf3 = abjad.select(staff).leaves()[22]

abjad.attach(mark3, leaf3)

for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):

leaf4 = abjad.select(staff).leaves()[29]

abjad.attach(mark4, leaf4)

for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):

leaf4 = abjad.select(staff).leaves()[29]

abjad.attach(mark4, leaf4)

for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):

leaf4 = abjad.select(staff).leaves()[29]

abjad.attach(mark4, leaf4)

for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):

leaf5 = abjad.select(staff).leaves()[34]

abjad.attach(mark5, leaf5)

for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):

leaf5 = abjad.select(staff).leaves()[34]

abjad.attach(mark5, leaf5)

for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):

leaf5 = abjad.select(staff).leaves()[34]

abjad.attach(mark5, leaf5)

for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):

leaf6 = abjad.select(staff).leaves()[39]

abjad.attach(mark6, leaf6)

for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):

leaf6 = abjad.select(staff).leaves()[39]

abjad.attach(mark6, leaf6)

for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):

leaf6 = abjad.select(staff).leaves()[39]

abjad.attach(mark6, leaf6)

for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):

abjad.Instrument.transpose\_from\_sounding\_pitch(staff)

for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):

abjad.Instrument.transpose\_from\_sounding\_pitch(staff)

for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):

abjad.Instrument.transpose\_from\_sounding\_pitch(staff)

# Make a lilypond file and show it:

score\_file = abjad.LilyPondFile.new(

score,

includes=['first\_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/\_stylesheets/abjad.ily'],

)

abjad.SegmentMaker.comment\_measure\_numbers(score)

###################

directory = '/Users/evansdsg2/Scores/tianshu/tianshu/Segments/Segment\_IV'

pdf\_path = f'{directory}/Segment\_IV.pdf'

path = pathlib.Path('Segment\_IV.pdf')

if path.exists():

print(f'Removing {pdf\_path} ...')

path.unlink()

time\_1 = time.time()

print(f'Persisting {pdf\_path} ...')

result = abjad.persist(score\_file).as\_pdf(pdf\_path)

print(result[0])

print(result[1])

print(result[2])

success = result[3]

if success is False:

print('LilyPond failed!')

time\_2 = time.time()

total\_time = time\_2 - time\_1

print(f'Total time: {total\_time} seconds')

if path.exists():

print(f'Opening {pdf\_path} ...')

os.system(f'open {pdf\_path}')

# for staff in abjad.iterate(score['Staff Group']).components(abjad.Staff):

# abjad.show(staff)

# abjad.show(score)

# abjad.play(score)