'''

Author: Ji-Sung Kim, Evan Chow

Project: deepjazz

Purpose: Provide pruning and cleanup functions.

Code adapted from Evan Chow's jazzml, https://github.com/evancchow/jazzml

with express permission.

'''

from itertools import zip\_longest

import random

from music21 import \*

#----------------------------HELPER FUNCTIONS----------------------------------#

''' Helper function to down num to the nearest multiple of mult. '''

def \_\_roundDown(num, mult):

return (float(num) - (float(num) % mult))

''' Helper function to round up num to nearest multiple of mult. '''

def \_\_roundUp(num, mult):

return \_\_roundDown(num, mult) + mult

''' Helper function that, based on if upDown < 0 or upDown >= 0, rounds number

down or up respectively to nearest multiple of mult. '''

def \_\_roundUpDown(num, mult, upDown):

if upDown < 0:

return \_\_roundDown(num, mult)

else:

return \_\_roundUp(num, mult)

''' Helper function, from recipes, to iterate over list in chunks of n

length. '''

def \_\_grouper(iterable, n, fillvalue=None):

args = [iter(iterable)] \* n

return zip\_longest(\*args, fillvalue=fillvalue)

#----------------------------PUBLIC FUNCTIONS----------------------------------#

''' Smooth the measure, ensuring that everything is in standard note lengths

(e.g., 0.125, 0.250, 0.333 ... ). '''

def prune\_grammar(curr\_grammar):

pruned\_grammar = curr\_grammar.split(' ')

for ix, gram in enumerate(pruned\_grammar):

terms = gram.split(',')

terms[1] = str(\_\_roundUpDown(float(terms[1]), 0.250,

random.choice([-1, 1])))

pruned\_grammar[ix] = ','.join(terms)

pruned\_grammar = ' '.join(pruned\_grammar)

return pruned\_grammar

''' Remove repeated notes, and notes that are too close together. '''

def prune\_notes(curr\_notes):

for n1, n2 in \_\_grouper(curr\_notes, n=2):

if n2 == None: # corner case: odd-length list

continue

if isinstance(n1, note.Note) and isinstance(n2, note.Note):

if n1.nameWithOctave == n2.nameWithOctave:

curr\_notes.remove(n2)

return curr\_notes

''' Perform quality assurance on notes '''

def clean\_up\_notes(curr\_notes):

removeIxs = []

for ix, m in enumerate(curr\_notes):

# QA1: ensure nothing is of 0 quarter note len, if so changes its len

if (m.quarterLength == 0.0):

m.quarterLength = 0.250

# QA2: ensure no two melody notes have same offset, i.e. form a chord.

# Sorted, so same offset would be consecutive notes.

if (ix < (len(curr\_notes) - 1)):

if (m.offset == curr\_notes[ix + 1].offset and

isinstance(curr\_notes[ix + 1], note.Note)):

removeIxs.append((ix + 1))

curr\_notes = [i for ix, i in enumerate(curr\_notes) if ix not in removeIxs]

return curr\_notes