Players (3) Each player has a set of attributes (these attributes can be changed).

Patterns (however many you want) each note in the pattern has 3 attributes

<table>
<thead>
<tr>
<th>pitch (offset from base pitch)</th>
<th>duration (multiple of global)</th>
<th>MIDI velocity</th>
<th>tick</th>
<th>note</th>
<th>basenote</th>
<th>outlet</th>
<th>volume (% of global)</th>
<th>MIDI Channel (not used)</th>
<th>which pattern to use (list)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 50, 3, 0.6, 1, 6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Players = [
[0, 0, 50, 3, 0.6, 1, 6],
[0, 0, 50, 0, 1.0, 10, 2],
[0, 0, 50, 0, 1.0, 10, 1],
]

ImprovPlayer = [
[0, 0, 50, 6, 1.0, 1, -1]]

Improv = [
[ [5, 3, 100],
[3, 1, 100],
[0, 1, 80] ] ],
e tc...

WhichImprov = [-1] tells which improv pattern to use when improvising (-1 means don't play)
def setPattern( playerid, patternid ):
    global whichPattern
    whichPattern[playerid] = patternid

Sets the pattern for a player to use
playerid will be indexed value from players list
patternid will be an indexed pattern from whichpattern list

def SetImprovPattern( playerid, patternid ):
    global whichImprov
    whichImprov[playerid] = patternid

Sets the pitch on which all players’ pitch
will be calculated

def setBasePitch( playerid, pitch ):
    global baseNote
    baseNote[playerid] = pitch

def PlayNote( player, pattern, unit ):
    # if we're at the beginning of a note (ticks == 0)
    if player[0] == 0:
        # get the values we need from the player
        note = player[1]
        base = player[2]
        outletID = player[3]
        volume = player[4]
        channel = player[5]
        # calculate pitch, duration and velocity
        pitch = int(pattern[note][0] + base)
        duration = int(pattern[note][1] * unit) - 5
        velocity = int(pattern[note][2] * volume)
        maxObject.outlet( outletID + 1, velocity )
        maxObject.outlet( outletID + 2, duration )
        maxObject.outlet( outletID + 3, channel )
        maxObject.outlet( outletID, pitch )
        # increment the tick for this player
        player[0] += 1

    # test if we're at the end of the current note (i.e. the ticks match the length of the note)
    if player[0] == pattern[player[1]][1]:
        # reset the tick counter
        player[0] = 0

        # move the note counter forward and modulo length of the pattern
        player[1] = (player[1] + 1) % len(pattern)

    # return which tick and note will be played next (returns 0 if it's the start of the pattern)
    return ( player[0], player[1] )
def play(unit):

    global whichPattern
    global patterns
    global players

    # for each player
    for i in range(len(players)):
        curPattern = players[i][6]
        if curPattern >= 0:
            (tick, note) = PlayNote(players[i], patterns[curPattern], unit)

    # update the base pitch and patterns only on bar lines
    if tick == 0 and note == 0:
        players[i][2] = baseNote[i]
        players[i][6] = whichPattern[i]
        if players[i][6] == 6:
            random.shuffle(patterns[6])

    # handle the improv players, who can enter on any tick
    # but they have to complete their pattern before changing
    for i in range(len(improvPlayer)):
        curPattern = improvPlayer[i][6]
        tick = 0
        note = 0

        # if the improve pattern has been set (from Max) to 0 or higher, play from the improv pattern in addition to the other stuff that's going on.
        if curPattern >= 0:
            (tick, note) = PlayNote(improvPlayer[i], improv[curPattern], unit)

        if tick == 0 and note == 0:
            improvPlayer[i][2] = baseNote[i]
            improvPlayer[i][6] = whichImprov[i]
        # only let the improv play once without user input
        whichImprov[i] = -1

    # return control to max
    return

Play is the function that’s called repeatedly from Max (each time the metro ticks) It in turn calls the PlayNote function (which cycles through the players list to see if it’s time for a player to output another note.  If not, PlayNote merely increments the tick counter for that player and Play moves on to the next player.)