Fig A04

Code written during session on 11 Sep 18

```
# STEP 2: MAKE LIST OF POSITIONS
for chord in chordsList:
    # Numbers at which chord notes are (1 = C, 2 = C#...)
    chord_notes = []
    for s in chord:
        try:
            # Try converting the string to a number
            num = int(s)
            # Append the positions
            chord_notes.append(num)
        except:
            # Go to next element if the string can't be converted to a number
            pass
    chordPossList.append(chord_notes)
print(chordPossList)
print(len(chordPossList))
```

```
def writeToChordsFile_2(CL, CPL, output_file="../input_dataset_binaries_2.txt"):
    SPACE_LENGTH = 10
    try:
        with open(output_file, 'w') as chordsFile:
            for chord_type in range(len(CPL)):
                for chord in range(len(CPL[chord_type])): # len: 444
                    # Write the chord name
                    chordsFile.write(CL[chord_type][0] + ((SPACE_LENGTH - len(CL[chord_type][0])) * " "))
                    # Write a vector of 128 0s and 1s, with 1s at the specified indeces in CPL
                    for note in range(128):
                        if note in CPL[chord_type][chord]:
                            chordsFile.write("1 ")
                        else:
                            chordsFile.write("0 ")
                    # Write a newline
                    chordsFile.write("\n")
    except Exception as e:
        print(e)
```

Fig A06

Code written during session on 18 Sep 18

Fig	A05
Code written during session on 16 Sep 18	

```
    # STEP 3: CONVERT NOTE NUMBER DISTANCES TO MIDI NUMBERS
    for chord_iteration in range(len(chordPossList)): # All chord types
        chordMIDINumbersList.append([])
        for oct_num in range(11):                     # All possible MIDI octaves
            temp = []
            for note_num in range(len(chordPossList[chord_iteration])):
                    temp.append((12 * oct_num) + (chordPossList[chord_iteration][note_num] - 1))
            if no_greater_than_127(temp):
                chordMIDINumbersList[chord_iteration].append(temp)
    print(chordMIDINumbersList)
    print(len(chordMIDINumbersList))
    print(str(count3DLayeredList2D(chordMIDINumbersList)))

def no_greater_than_127(l):
    for n in l:
        if n > 127:
            return False
    return True

def count3DLayeredList2D(ll):
    k = 0
    for a in range(len(ll)):
        for b in range(len(ll[a])):
            k += 1
    return k
```