def make\_chord\_sequence(filepath, range\_length=2, way='symmetrical-fifth',   
 show=False, save=True):

    '''將指定的樂譜檔案作範例，依指定和聲樣式循環的生成結果

    Parameters

    ---

    filepath:

        檔案路徑。

    range\_length:

        將旋律分割的四分音符長度的區間單位。

    way:

        指定的循環模式，提供 6 種：

            上五度（'fifth'）

            下五度（'forth'）

            上半音（'halftone'）

            下半音（'halftone-down'）

            對稱五度（'symmetrical-fifth'，預設值）

            逆對稱五度（'anti-symmetrical-fifth'，預設值）

    show:

        設定是否要檢視生成結果。

    save:

        是否要存檔，將會存檔成 mxl。

        存檔路徑與原檔案路徑位置一樣，且在附檔名前加 "\_output" 的字樣。

    Tips

    ---

    本程式不會對音樂記譜法的正確性做檢查。

    '''

    # 讀檔，並擷取資料

    score = music21.converter.parse(filepath)

    part = score.recurse().parts[0]

    notes = part.recurse().notes

    key = music21.analysis.discrete.analyzeStream(score, 'key')

    max\_offset = int(notes[-1].offset)  # 合理假設最後一個音的 offset 最大

    # 設定新 part 資訊

    chord\_part = music21.stream.Part()

    instrument = music21.instrument.Piano()

    chord\_part.insert(0, instrument)

    chord\_part.insert(0, key)

    # 在每個區間單位中判斷適合的和弦

    for i in range(0, max\_offset + 1, range\_length):

        melody = [note for note in notes   
                       if note.offset >= i and note.offset < i + range\_length]

        chord = musicTheory.makeChord(melody, range\_length, key, way)

        chord\_part.insert(i, chord)

    # 添加到舊的 score 裡後顯示（第三方軟體）

    score.insert(0, chord\_part)

    return score

def makeChord(notes, quarterLengt=2, key=None, way='symmetrical-fifth'):

    '''

    依照給定的音符序列找出適當的和弦

    Parameters

    ---

    notes:

        由 music21.note.Note 組成的序列。

    key:

        指定調性，若無設定則自行判斷。

        調性會影響到主音的位置與和弦判斷的順序。

    quarterLengt:

        指定和弦的持續長度，預設為 2 拍

    way:

        巡迴的方式。可選擇上五度（'fifth'）、下五度（'forth'）、上半音（'halftone'）、下半音（'halftone-down'）、對稱五度（'symmetrical-fifth'，預設值）、逆對稱五度（'anti-symmetrical-fifth'）。前四項分別可用 7, 5, 11, 1 表示。其他數字不會判斷到所有和弦。

    Returns

    ---

    newChord:

        依照判斷結果生成出來的和弦。

    '''

    # 計算該片段中每個音出現的次數

    countNote = [0 for \_ in range(12)]

    for note in notes:

        if isinstance(note, music21.note.Note):

            countNote[int(note.pitch.ps)%12] += 1

    # 引入各種 chordPattern

    major = [(i in chord['Maj']) for i in range(12)]

    minor = [(i in chord['Min']) for i in range(12)]

    dim = [(i in chord['Dim']) for i in range(12)]

    arg = [(i in chord['Arg']) for i in range(12)]

    dom7 = [(i in chord['Dom7']) for i in range(12)]

    dim7 = [(i in chord['Dim7']) for i in range(12)]

    maj7 = [(i in chord['Maj7']) for i in range(12)]

    # 判斷的順序，越前面優先權越高

    chordPatternSequrence = [major[:], minor[:], dim[:]]

    if key == None:

        key = music21.analysis.discrete.analyzeStream(music21.stream.Stream(notes), 'key')

    elif isinstance(key, music21.key.Key):

        pass

    else:

        key = music21.key.Key(key)

    # 將 chordPattern 移動到主音的位置

    nextChordGroup = []

    for chordPattern in chordPatternSequrence:

        nextChordGroup.append(moveChordPattern(chordPattern, way=key.tonic.ps))

    chordPatternSequrence = nextChordGroup

    # 若判斷為大調則優先判斷大和弦，否則優先判斷小和弦（原本的順序為 [maj, min, ...]）

    if key.mode == 'minor':

        chordPatternSequrence[0], chordPatternSequrence[1] = chordPatternSequrence[1][:], chordPatternSequrence[0][:]

    currChord = None

    max = 0

    # 利用 chordPattern 做比對，取與片段吻合最多的 chordPattern 作為搭配的和弦

    # chordPattern 循環模式包含上五度（'fifth'）、下五度（'forth'）、上半音（'halftone'）、下半音（'halftone-down'）、對稱五度（'symmetrical-fifth'）

    # 只會以最 general 「其他 case」做說明。

    if way == 'symmetrical-fifth':

        toUp = [moveChordPattern(chordPattern[:]) for chordPattern in   
                chordPatternSequrence]

        toDown = [chordPattern[:] for chordPattern in chordPatternSequrence]

        for \_ in range(6):

            sum = 0

            nextChordGroup = []

            for chordPattern in toDown:

                sum = chordPatternMatch(chordPattern, countNote)

                if sum > max:

                    max = sum

                    currChord = chordPattern[:]

                nextChordGroup.append(moveChordPattern(chordPattern, way='forth'))

            toDown = nextChordGroup[:]

            sum = 0

            nextChordGroup = []

            for chordPattern in toUp:

                sum = chordPatternMatch(chordPattern, countNote)

                if sum > max:

                    max = sum

                    currChord = chordPattern[:]

                nextChordGroup.append(moveChordPattern(chordPattern, way='fifth'))

            toUp = nextChordGroup[:]

    elif way == 'anti-symmetrical-fifth':

        toDown = [moveChordPattern(chordPattern[:]) for chordPattern   
                  in chordPatternSequrence]

        toUp = [chordPattern[:] for chordPattern in chordPatternSequrence]

        for \_ in range(6):

            sum = 0

            nextChordGroup = []

            for chordPattern in toUp:

                sum = chordPatternMatch(chordPattern, countNote)

                if sum > max:

                    max = sum

                    currChord = chordPattern[:]

                nextChordGroup.append(moveChordPattern(chordPattern, way='fifth'))

            toUp = nextChordGroup[:]

            sum = 0

            nextChordGroup = []

            for chordPattern in toDown:

                sum = chordPatternMatch(chordPattern, countNote)

                if sum > max:

                    max = sum

                    currChord = chordPattern[:]

                nextChordGroup.append(moveChordPattern(chordPattern, way='forth'))

            toDown = nextChordGroup[:]

    else:

        for \_ in range(12):

            nextChordGroup = []

            for chordPattern in chordPatternSequrence:

                # 計算吻合分數

                sum = chordPatternMatch(chordPattern, countNote)

                # 比較並更新目前最大的和弦樣式

                if sum > max:

                    max = sum

                    currChord = chordPattern[:]

                nextChordGroup.append(moveChordPattern(chordPattern, way=way))

            chordPatternSequrence = nextChordGroup[:]

    # 以分數最大的和聲樣式當作該區間搭配的和弦

    newChord = music21.chord.Chord([music21.note.Note(i, octave=3)   
                                    for i in range(12) if currChord[i] == True],   
                                    quarterLength=quarterLengt)

    return newChord