1. 아이디어의 시작

음악 작곡을 취미로 하는 나는 음악 작곡에 있어서 내가 만들었던 음악과 비슷한 음악을 만들지 않는 것이 가장 힘든 부분 중 하나라고 느껴졌다. 아무리 랜덤한 경우를 만들어내려고 해도 내 머릿속에 한번 박힌 멜로디와 요소들을 벗어나 아예 새로운 것을 만들어내는 것은 쉽지 않았고 이는 작업활동에 크나큰 한계로 느껴졌다. 그러나 이번에 파이썬을 배우게 되면서 ‘음악 스케일은 수치상으로도 표현이 가능한 요소인데 이를 이용하여 자동으로 조합을 해줘 멜로디라인을 만들어주는 프로그램을 만들면 작업활동에 도움이 될 수 있지 않을까’라는 생각을 하게 되었다. 따라서 파이썬으로 음악적인 부분을 표현할 수 있는지에 대해서 먼저 알아보기 시작하였다.

1. 간단한 실현화 (basepj.py)

검색을 통해 알아본 결과 파이썬에는 midiutil이라는 라이브러리가 존재하는데 이를 통해서 멜로디를 생성하고 midi 파일로 저장까지 하는 프로그램을 만들 수 있다는 것을 알게 되었다. 그래서 처음에는 가장 기본적인 C 메이저 스케일을 조건으로 둔 ‘무작위 멜로디 생성 프로그램’을 만들어보았다.

pip install MIDIUtil

from midiutil import MIDIFile

import numpy as np

# midi 파일 생성

def create\_melody():

    midi = MIDIFile(1) # 한 트랙으로 구성된 midi 파일

    track = 0

    time = 0

    midi.addTrackName(track, time, "Sample Track")

    midi.addTempo(track, time, 120)

    scale = [60, 62, 64, 65, 67, 69, 71, 72] # c major scale

    duration = 1 # 한 박자

    volume = 100 # 기본 음량 설정

    for i in range(16):

        pitch = np.random.choice(scale)

        midi.addNote(track, 0, pitch, time, duration, volume)

        time += duration

# midi 파일 저장

    with open("melody.mid", "wb") as output\_file:

        midi.writeFile(output\_file)

if \_\_name\_\_ == "\_\_main\_\_":

    create\_melody()

1. 다양한 스케일 구현 (scalemaker.py)

우리는 음악을 만들 때 하나의 스케일만 사용하지 않는다. 따라서 위에서 C 메이저 스케일을 기본으로 둔 형태와 다르게 스케일들을 미리 입력해두고 랜덤하게 나올 수 있는 경우들을 뽑아낼 수 있는 프로그램을 새로 작성해 보았다.

from midiutil import MIDIFile

import numpy as np

def get\_scale(scale\_name):

    scales = {

        "C\_major": [60, 62, 64, 65, 67, 69, 71, 72],

        "G\_major": [67, 69, 71, 72, 74, 76, 78, 79],

        "D\_major": [62, 64, 66, 67, 69, 71, 73, 74],

        "A\_minor": [69, 71, 72, 74, 76, 77, 79, 81],

        "E\_minor": [64, 66, 67, 69, 71, 72, 74, 76],

        "C\_dorian": [60, 62, 63, 65, 67, 69, 70, 72],

        "G\_dorian": [67, 69, 70, 72, 74, 76, 77, 79],

        "D\_phrygian": [62, 63, 65, 67, 69, 70, 72, 74],

        "A\_phrygian": [69, 70, 72, 74, 76, 77, 79, 81],

        "C\_lydian": [60, 62, 64, 66, 67, 69, 71, 72],

        "G\_lydian": [67, 69, 71, 73, 74, 76, 78, 79],

        "D\_mixolydian": [62, 64, 66, 67, 69, 71, 72, 74],

        "A\_mixolydian": [69, 71, 73, 74, 76, 77, 79, 81],

        "C\_locrian": [60, 61, 63, 65, 67, 68, 70, 72],

        "G\_locrian": [67, 68, 70, 72, 74, 75, 77, 79]

    }

    return scales.get(scale\_name, scales["C\_major"])

def create\_melody(scale\_name="C\_major"):

    scale = get\_scale(scale\_name)

    midi = MIDIFile(1)

    track = 0

    time = 0

    midi.addTrackName(track, time, "Sample Track")

    midi.addTempo(track, time, 120)

    duration = 1

    volume = 100

    for i in range(16):

        pitch = np.random.choice(scale)

        midi.addNote(track, 0, pitch, time, duration, volume)

        time += duration

    filename = f"melody\_{scale\_name}.mid"

    with open(filename, "wb") as output\_file:

        midi.writeFile(output\_file)

    print(f"MIDI file saved as {filename}")

if \_\_name\_\_ == "\_\_main\_\_":

    available\_scales = [

        "C\_major", "G\_major", "D\_major", "A\_minor", "E\_minor",

        "C\_dorian", "G\_dorian", "D\_phrygian", "A\_phrygian",

        "C\_lydian", "G\_lydian", "D\_mixolydian", "A\_mixolydian",

        "C\_locrian", "G\_locrian"

    ]

    print("Available scales:", ", ".join(available\_scales))

    scale\_name = input("Enter the scale name: ")

    create\_melody(scale\_name)

최대한 많은 스케일을 입력해 놓은 형태이고 부연설명을 덧붙이자면

1. get\_scale : 사용자가 입력한 스케일 이름에 해당하는 노트 리스트
2. create\_melody : 입력한 스케일 이름을 사용하여 멜로디 생성
3. \_\_main\_\_ : 블록에서 사용 가능한 스케일 목록을 출력

이며 또 필요한 스케일이 있다면 코드에 추가해서 생성할 수 있다.

프로그램 실행 시 예시로 Enter the scale name에 C\_major이라고 입력하면 파일 이름이 ‘melody\_C\_major.mid’ 인 MIDI 파일이 생성되면 노트 길이는 1박자, 템포는 120 BPM이다.

1. 기능 확장 (rhythmmaker.py)

음악은 단순히 스케일로 이루어져 있는 것이 아니다. 박자의 변수 또한 노래의 중요한 요소로서 이런 부분 또한 프로그램이 랜덤으로 정해주는 것이 아니라면 빠른 시일 내에 같은 느낌의 노래를 반복해서 만들게 될 것이다. 따라서 이번에는 기본 마디 수인 ‘16박자’ 안에서 다양한 변주를 랜덤하게 생성할 수 있도록 프로그램을 만들어 보았다.

from midiutil import MIDIFile

import numpy as np

def get\_scale(scale\_name):

    scales = {

        "C\_major": [60, 62, 64, 65, 67, 69, 71, 72],

        "G\_major": [67, 69, 71, 72, 74, 76, 78, 79],

        "D\_major": [62, 64, 66, 67, 69, 71, 73, 74],

        "A\_minor": [69, 71, 72, 74, 76, 77, 79, 81],

        "E\_minor": [64, 66, 67, 69, 71, 72, 74, 76],

        "C\_dorian": [60, 62, 63, 65, 67, 69, 70, 72],

        "G\_dorian": [67, 69, 70, 72, 74, 76, 77, 79],

        "D\_phrygian": [62, 63, 65, 67, 69, 70, 72, 74],

        "A\_phrygian": [69, 70, 72, 74, 76, 77, 79, 81],

        "C\_lydian": [60, 62, 64, 66, 67, 69, 71, 72],

        "G\_lydian": [67, 69, 71, 73, 74, 76, 78, 79],

        "D\_mixolydian": [62, 64, 66, 67, 69, 71, 72, 74],

        "A\_mixolydian": [69, 71, 73, 74, 76, 77, 79, 81],

        "C\_locrian": [60, 61, 63, 65, 67, 68, 70, 72],

        "G\_locrian": [67, 68, 70, 72, 74, 75, 77, 79]

    }

    return scales.get(scale\_name, scales["C\_major"])

def create\_variation(scale\_name="C\_major"):

    scale = get\_scale(scale\_name)

    midi = MIDIFile(1)

    track = 0

    time = 0

    midi.addTrackName(track, time, "Sample Track")

    midi.addTempo(track, time, 120)

    durations = [0.25, 0.5, 1, 2] # 16분음표, 8분음표, 4분음표, 2분음표

    total\_time = 0

    volume = 100

    while total\_time < 16:

        duration = np.random.choice(durations)

        if total\_time + duration > 16:  # 16박자 초과 막는 기능

            duration = 16 - total\_time

        pitch = np.random.choice(scale)

        midi.addNote(track, 0, pitch, time, duration, volume)

        time += duration

        total\_time += duration

    filename = f"variation\_{scale\_name}.mid"

    with open(filename, "wb") as output\_file:

        midi.writeFile(output\_file)

    print(f"MIDI file saved as {filename}")

if \_\_name\_\_ == "\_\_main\_\_":

    available\_scales = [

        "C\_major", "G\_major", "D\_major", "A\_minor", "E\_minor",

        "C\_dorian", "G\_dorian", "D\_phrygian", "A\_phrygian",

        "C\_lydian", "G\_lydian", "D\_mixolydian", "A\_mixolydian",

        "C\_locrian", "G\_locrian"

    ]

    print("Available scales:", ", ".join(available\_scales))

    scale\_name = input("Enter the scale name: ")

    create\_variation(scale\_name)

여기에서 create\_variation 함수는 입력한 스케일에 따라 랜덤하게 변주를 생성해준다.

16박자를 기준으로 삼았기 때문에 16박자를 초과하지 않도록 프로그램을 생성했다.

1. 부가적인 편의성 (midimaker.py)

MIDI 파일을 직접 MIDI 프로그램에 넣어보며 사용해보니 몇 가지 입력 기능을 더 넣는다면 훨씬 편리하게 사용할 수 있을 거라는 생각이 들어서 다시 한 번 프로그램을 확장하기로 하였다. 지금까지는 스케일만 입력하였지만 여기에 BPM 입력과 옥타브 입력까지 추가해 보았다.

from midiutil import MIDIFile

import numpy as np

def get\_scale(scale\_name, octave):

    scales = {

        "C\_major": [0, 2, 4, 5, 7, 9, 11, 12],

        "G\_major": [7, 9, 11, 12, 14, 16, 18, 19],

        "D\_major": [2, 4, 6, 7, 9, 11, 13, 14],

        "A\_minor": [9, 11, 12, 14, 16, 17, 19, 21],

        "E\_minor": [4, 6, 7, 9, 11, 12, 14, 16],

        "C\_dorian": [0, 2, 3, 5, 7, 9, 10, 12],

        "G\_dorian": [7, 9, 10, 12, 14, 16, 17, 19],

        "D\_phrygian": [2, 3, 5, 7, 9, 10, 12, 14],

        "A\_phrygian": [9, 10, 12, 14, 16, 17, 19, 21],

        "C\_lydian": [0, 2, 4, 6, 7, 9, 11, 12],

        "G\_lydian": [7, 9, 11, 13, 14, 16, 18, 19],

        "D\_mixolydian": [2, 4, 6, 7, 9, 11, 12, 14],

        "A\_mixolydian": [9, 11, 13, 14, 16, 17, 19, 21],

        "C\_locrian": [0, 1, 3, 5, 7, 8, 10, 12],

        "G\_locrian": [7, 8, 10, 12, 14, 15, 17, 19]

    }

    base\_octave = 12 \* octave

    return [note + base\_octave for note in scales.get(scale\_name, scales["C\_major"])]

def create\_variation(scale\_name="C\_major", bpm=120, octave=4):

    scale = get\_scale(scale\_name, octave)

    midi = MIDIFile(1)

    track = 0

    time = 0

    midi.addTrackName(track, time, "Sample Track")

    midi.addTempo(track, time, bpm)

    durations = [0.25, 0.5, 1, 2]

    total\_time = 0

    volume = 100

    while total\_time < 16:

        duration = np.random.choice(durations)

        if total\_time + duration > 16:

            duration = 16 - total\_time

        pitch = np.random.choice(scale)

        midi.addNote(track, 0, pitch, time, duration, volume)

        time += duration

        total\_time += duration

    filename = f"variation\_{scale\_name}\_{bpm}bpm\_octave{octave}.mid"

    with open(filename, "wb") as output\_file:

        midi.writeFile(output\_file)

    print(f"MIDI file saved as {filename}")

if \_\_name\_\_ == "\_\_main\_\_":

    available\_scales = [

        "C\_major", "G\_major", "D\_major", "A\_minor", "E\_minor",

        "C\_dorian", "G\_dorian", "D\_phrygian", "A\_phrygian",

        "C\_lydian", "G\_lydian", "D\_mixolydian", "A\_mixolydian",

        "C\_locrian", "G\_locrian"

    ]

    print("Available scales:", ", ".join(available\_scales))

    scale\_name = input("Enter the scale name: ")

    bpm = int(input("Enter the BPM (e.g., 120): "))

    octave = int(input("Enter the octave (e.g., 4): "))

    create\_variation(scale\_name, bpm, octave)

여러 옥타브 값을 다룰 수 있도록 옥타브 4의 스케일 값을 옥타브 1의 스케일 값으로 변환 후 곲을 이용해 입력한 옥타브 값에 따라 알맞은 결과를 도출할 수 있도록 하였고 create\_variation 함수에 bpm 파라미터를 추가하여 MIDI BPM을 직접 입력해 파일을 생성할 수 있도록 하였다.

프로그램 실행 시 예시로 Enter the scale name에 C\_major이라고 입력하고, Enter the BPM에 140이라 입력하고 Enter the octave에 5라고 입력하면 파일 이름이 ‘variation\_C\_major\_140bpm\_octave5.mid’ 인 MIDI 파일이 생성된다.

1. 마지막 문제 (nobadsong.py)

이제 모든 것을 자동으로 해주는 프로그램을 만들었지만 마지막 문제가 남았다. 바로 음계 간의 불협화음에 의해 조합 시 듣기 거북한 결과물들이 나올 수 있다는 것이다. 불협화음을 완전히 제거하기 위해 몇 가지 시도를 해봤지만 불협화음을 제거하다 보면 불협화음이 아닌 음계 규칙들 또한 함께 제거가 될 수 있다는 것이 문제가 되었고, 불협화음의 리스크를 가져갈지 최소한의 결과물을 가져갈지 고민한 끝에 일단 불협화음을 제거한 프로그램을 만들어 보기로 결정하였다. 불협화음을 없애기 위해서 음계 내에 이동할 때 노트 간의 간격을 조정하여 노트 간의 소리가 조화를 이루도록 프로그램을 조정하였다. 예를 들면 연속된 두 노트가 나오지 않도록 제한하는 식이다.

from midiutil import MIDIFile

import numpy as np

def get\_scale(scale\_name, octave):

    scales = {

        "C\_major": [0, 2, 4, 5, 7, 9, 11, 12],

        "G\_major": [7, 9, 11, 12, 14, 16, 18, 19],

        "D\_major": [2, 4, 6, 7, 9, 11, 13, 14],

        "A\_minor": [9, 11, 12, 14, 16, 17, 19, 21],

        "E\_minor": [4, 6, 7, 9, 11, 12, 14, 16],

        "C\_dorian": [0, 2, 3, 5, 7, 9, 10, 12],

        "G\_dorian": [7, 9, 10, 12, 14, 16, 17, 19],

        "D\_phrygian": [2, 3, 5, 7, 9, 10, 12, 14],

        "A\_phrygian": [9, 10, 12, 14, 16, 17, 19, 21],

        "C\_lydian": [0, 2, 4, 6, 7, 9, 11, 12],

        "G\_lydian": [7, 9, 11, 13, 14, 16, 18, 19],

        "D\_mixolydian": [2, 4, 6, 7, 9, 11, 12, 14],

        "A\_mixolydian": [9, 11, 13, 14, 16, 17, 19, 21],

        "C\_locrian": [0, 1, 3, 5, 7, 8, 10, 12],

        "G\_locrian": [7, 8, 10, 12, 14, 15, 17, 19]

    }

    base\_octave = 12 \* octave

    return [note + base\_octave for note in scales.get(scale\_name, scales["C\_major"])]

def create\_variation(scale\_name="C\_major", bpm=120, octave=4):

    scale = get\_scale(scale\_name, octave)

    midi = MIDIFile(1)

    track = 0

    time = 0

    midi.addTrackName(track, time, "Sample Track")

    midi.addTempo(track, time, bpm)

    durations = [0.25, 0.5, 1, 2]

    total\_time = 0

    volume = 100

    last\_note = None  # 이전 노트 저장

    while total\_time < 16:

        duration = np.random.choice(durations)

        if total\_time + duration > 16:

            duration = 16 - total\_time

        if last\_note is not None:

            while True:

                pitch = np.random.choice(scale)

                if abs(pitch - last\_note) <= 7:  # 두 노트 사이의 간격 처리 위한 설정

                    break

        else:

            pitch = np.random.choice(scale)

        midi.addNote(track, 0, pitch, time, duration, volume)

        time += duration

        total\_time += duration

        last\_note = pitch

    filename = f"variation\_{scale\_name}\_{bpm}bpm\_octave{octave}.mid"

    with open(filename, "wb") as output\_file:

        midi.writeFile(output\_file)

    print(f"MIDI file saved as {filename}")

if \_\_name\_\_ == "\_\_main\_\_":

    available\_scales = [

        "C\_major", "G\_major", "D\_major", "A\_minor", "E\_minor",

        "C\_dorian", "G\_dorian", "D\_phrygian", "A\_phrygian",

        "C\_lydian", "G\_lydian", "D\_mixolydian", "A\_mixolydian",

        "C\_locrian", "G\_locrian"

    ]

    print("Available scales:", ", ".join(available\_scales))

    scale\_name = input("Enter the scale name: ")

    bpm = int(input("Enter the BPM (e.g., 120): "))

    octave = int(input("Enter the octave (e.g., 4): "))

    create\_variation(scale\_name, bpm, octave)

1. 결론

연속된 두 노트 사이의 간격을 7음 이하로는 제한하여 불협화음이 일어나는 경우를 줄였지만 여전히 프로그램의 실용성에 대한 아쉬움이 남는 결과물이다. 이보다 더 확실하게 불협화음을 제거하면서 나머지 경우들을 살리기 위해서는 직접 하나씩 제거해서 프로그램에 입력해 놓는 방법 말고는 없다는 것을 깨닫게 되었다. 이 방법을 사용하게 되면 오히려 직접 되는 경우를 입력해놓고 프로그램을 통해 개수대로 뽑아내는 방법이 효율적일 것 같아 이에 대한 프로그램은 작성하지 않기로 하였다. 아쉬움이 남는 결과물이지만 분명 앞으로의 창작에 있어서 참고용으로는 충분히 기능할 수 있는 프로그램을 만들었다고 생각한다.