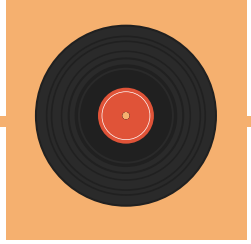


Creative Coding Final Reflection on "Code the Particles"

Jinyi sun
14425096

CONCEPT MAPPING
PROJECT JOURNAL
PEER REVIEW
SOURCE

Introduction

When I first chose “Code the Particles” as my creative coding project, it was not because I thought it would be technically impressive. It was because I genuinely loved the song. “粒子们” (Particles) is a track by Chinese indie artist Yu Zhen that I already knew how to play on the piano, and for me, it carried emotional weight. That emotional connection became the foundation of my project. I think if I am already familiar with its sound and feel on the keyboard, then this song must be my best representative work in this task.

At first, I saw this assignment as an opportunity to simply convert piano sheet music into a TunePad sequence. I began by manually copying each note from the sheet into the code using playNote() functions. Because of my musical familiarity with the song, this part came easily to me. But very quickly, during peer review and class discussion, it became clear that musical replication alone was not enough. My code lacks complexity, has too much repetitive structure, and most importantly, there is no explanation. If the audience has never heard this song, no one can figure out why I wrote it this way or how each section was conceived.

Therefore, this soon turned into a deeper challenge. How could I not only compose melodies, but also design and document a musical system that shows my thought processes, experiments and creative decisions?

Challenge

One of the biggest obstacles I encountered was the interface itself. TunePad does not have an undo function. This means that if I accidentally delete a large chunk of code or insert some invalid content, the entire line will turn red, or the entire page of code will disappear from the view. Especially when adding drum beats or bass lines. Each time I made a change, I had to carefully re-listen to ensure the new elements did not clash with the piano melody. Even a small adjustment often caused the code's indentation to shift, and since TunePad has a very low tolerance for formatting errors, a single misplaced space could crash the entire program. This made the debugging process feel tense and anxiety-inducing, as I constantly feared one small mistake would make all my progress disappear.

Another stage that left me confused and frustrated occurred when I actively sought out more learning resources. I repeatedly searched on YouTube and other programming teaching platforms, hoping to find tutorials or sample videos about the advanced features of TunePad. However, there is hardly any systematic or updated teaching content available on the Internet. Most of the materials only cover the most basic usage methods. This gradually made me realize that my existing musical ability alone was far from enough to complete this project.

Challenge

Gradually, I came to understand that the core of this task was not to "transcribe" a piano piece into code, but to strike a balance among musical logic, structural design and computational thinking. I must demonstrate how to transform emotions into structured creativity. As Romero et al., (2017) pointed out, creative programming in higher education is not merely a technical exercise, but a cognitive process through which learners develop computational thinking through artistic experimentation and reflection. I gradually came to realize that what this project demands is precisely this ability to coordinate between intuition and logic.

Solution

After encountering format errors and debugging difficulties many times, I realized that relying solely on intuition about the original piece and proficiency in the piano to complete the project is far from enough. To enhance the technical complexity and depth of reflection of my project, I began to proactively integrate multiple learning resources.

Firstly, I read the descriptions of each function (like chorusNote, synthPopBass, and offset) in TunePad doc item by item, and attempted to conduct combination experiments on drumbeats, offsets, timbres, etc., while comparing them with the impression of the original piano melody.

In addition, I also watched the video examples embedded in the course materials. In this video, the structure hierarchy of the sample code is very clear and well-organized. I studied the functions and musical logic employed by the speaker. This process reminds me of the method of "reverse engineering our creative landscape" as mentioned by Stephensen (2020). Gain insight into one's own creative process by analyzing existing creative works. Subsequently, I applied this method to this project, breaking down the sample code, commenting on the logic, and adapting the structure into my work. Make the functional usage of this project no longer limited to just the single code playNote().

Solution

In the second round of peer review, I shared the updated version of the code with detailed comments. A classmate（Rong）said, "Wow, this time your annotations are so many. You even wrote down the knowledge of music theory!" This moment is very inspiring.I began to understand that annotations are not only for others to read, but also for myself to clarify my thoughts and record the creative process. Peppler and Kafai（2007）pointed out that in a creative programming environment, learners can externalize their thinking and reflect on the relationship between code and expression intentions, which is a very crucial learning transformation. For me, this is a turning point.

AI Use Statement

I used GenAI tools to look up piano sheet music for “Particles” and translated some technical terms. All coding, arrangement, and reflection writing were done independently.

Summary

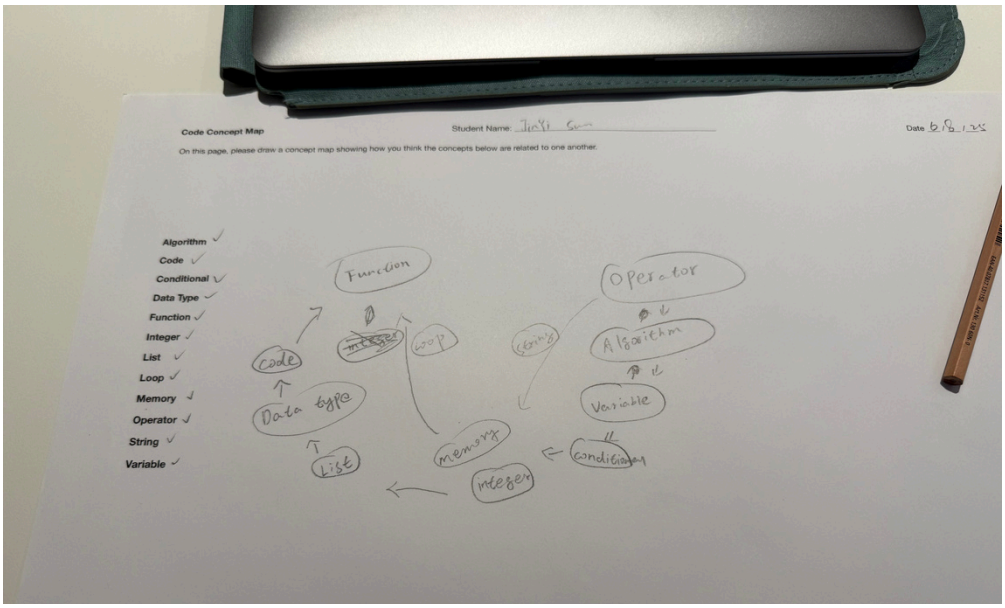
Looking back on this experience, I realize that my success depends not only on my musical skills but also on cultivating a systematic habit of reflection. The setbacks I encountered on the TunePad interface taught me to carefully plan, document and revise my creative process. If I encounter similar projects in the future, I will start with a clearer structural mapping and maintain detailed logs after each iteration. This project ultimately demonstrated to me that reflective practice itself is a creative act that combines sound, structure and thought.

Reference

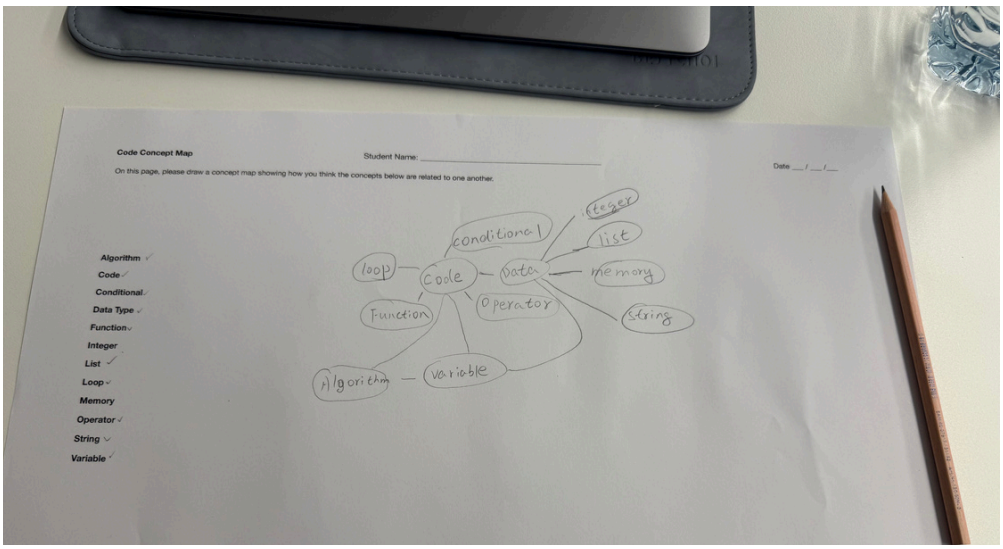
- Peppler, K. A., & Kafai, Y. B. (2007). W hat video game making can teach us about learning and literacy: Alternative pathways into participatory cultures. In Paper to be presented at the Digital International Games Research Association meeting in Tokyo, Japan.
- Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. International Journal of Educational Technology in Higher Education, 14(1), 42.
- Stephensen, J. L. (2020). Post-creativity and AI: Reverse-engineering our Conceptual Landscapes of Creativity. In Proceedings of the 11th International Conference on Computational Creativity (ICCC'20) (pp. 326-333). Association for Computational Creativity.
- TunePad. (n.d.). TunePad documentation. Retrieved October 28, 2025, from https://learn.tunepad.com/docs/

Concept Mapping

WEEK 1

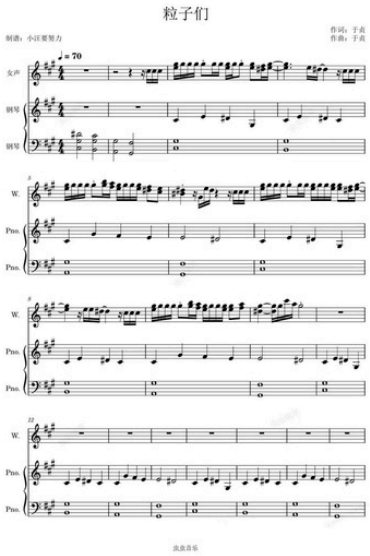


WEEK 12



Journal ——week 6

Sheet music



Work progress

```
1
2 def melody_particles():
3     for i in range(1):
4         playNote(72, beats=0.5)  #do
5         playNote(72, beats=0.5)
6         playNote(72, beats=0.5)
7
8         playNote(79, beats=0.5)  #so
9         playNote(79, beats=0.5)
10        playNote(79, beats=0.5)
11        playNote(79, beats=0.5)
12        playNote(79, beats=1)
13
14        playNote(79, beats=0.5)  #so
15        playNote(79, beats=0.5)
16        playNote(79, beats=0.5)
17        playNote(79, beats=0.5)
18        playNote(79, beats=0.5)
19
20        playNote(79, beats=0.5)  #so
21        playNote(78, beats=0.5)  #fa
22        playNote(79, beats=1)    #so
23
24        playNote(76, beats=0.5)  #mi
25        playNote(75, beats=2)    #re
26
```

```
28 def left_hand_particles():
29     for i in range(2):
30         playNote(72, beats=0.5)  #do
31         playNote(72, beats=0.5)
32         playNote(72, beats=0.5)
33
34         playNote(79, beats=0.5)  #so
35         playNote(79, beats=0.5)
36         playNote(79, beats=0.5)
37         playNote(79, beats=0.5)
38         playNote(79, beats=1)
39
40         playNote(83, beats=0.5)  #si
41         playNote(82, beats=0.5)  #la
42         playNote(79, beats=0.5)  #so
43         playNote(79, beats=0.5)
44         playNote(79, beats=0.5)
45
46         playNote(78, beats=0.5)  #fa
47         playNote(76, beats=0.5)  #mi
48         playNote(75, beats=1)    #re
49
50         playNote(68, beats=0.5)  #so
51         playNote(76, beats=0.5)  #mi
52         playNote(74, beats=0.5)  #re
53
```

Week 6 —— Start writing the right-hand melody, input the notes and beats of the main melody, and add annotations to explain the relationship between the notes and the code.

Journal ——week 7

Work progress

```
def melody_particles():
    for i in range(1):
        playNote(72, beats=0.5)  # do
        playNote(72, beats=0.5)
        playNote(72, beats=0.5)

        playNote(79, beats=0.5)
        playNote(79, beats=0.5)  # so
        playNote(79, beats=0.5)
        playNote(79, beats=0.5)
        playNote(79, beats=1)

        playNote(79, beats=0.5)
        playNote(79, beats=0.5)
        playNote(79, beats=0.5)
        playNote(79, beats=0.5)
        playNote(79, beats=0.5)

        playNote(79, beats=0.7)
        playNote(78, beats=0.5)  # fa
        playNote(79, beats=1)    # so

        playNote(76, beats=0.5)  # mi
        playNote(75, beats=2)    # re

melody_particles()
```

```
def melody_particles_part2():
    # do x3
    for _ in range(3):
        playNote(72, beats=0.5)  # do

    # so x4
    for _ in range(4):
        playNote(79, beats=0.5)  # so
    playNote(79, beats=1)        # so

    # si la so so so
    playNote(83, beats=0.5)  # si
    playNote(82, beats=0.5)  # la

    # so x4
    for _ in range(4):
        playNote(79, beats=0.5)  # so
    playNote(79, beats=1)        # so

    # fa mi re
    playNote(78, beats=0.5)  # fa
    playNote(76, beats=0.5)  # mi
    playNote(75, beats=1.2)  # re

    # so mi re
    playNote(68, beats=0.7)  # so
    playNote(76, beats=0.5)  # mi
    playNote(75, beats=0.5)  # re
melody_particles_part2()
```


```
# 36 = Kick, 38 = Snare, 42 = Closed HiHat, 46 = Open HiHat, 49 = Crash Cymba

def play_kick_beat():
    playNote(36, beats=1)

def play_snare_beat():
    rest(1)
    playNote(38, beats=1)

def play_hihat_beat():
    for i in range(4):
        playNote(42, beats=0.5)

#  hi-hat
def play_advanced_hihat():
    for i in range(3):
        playNote(42, beats=0.5)
    playNote(46, beats=0.5)  # Open HiHat on 4th beat
```

```
# fill-in (climax)
def play_fill_in():
    playNote(38, beats=0.5)
    playNote(38, beats=0.5)
    playNote(42, beats=0.25)
    playNote(42, beats=0.25)
    playNote(38, beats=0.25)
    playNote(49, beats=1.25)

#  Crash
def play_crash_intro():
    playNote(49, beats=1)
    rest(3)

def play_main_groove():
    for i in range(4):
        play_kick_beat()
        play_snare_beat()
        if i < 3:
            play_hihat_beat()
        else:
            play_advanced_hihat()

play_crash_intro()

for i in range(7):
    play_main_groove()

play_fill_in()
```

Week 7 ——I mprove the right-hand melody and implement the melody repetition and variation section structure by using for i in range(). Start studying the code of the drum.

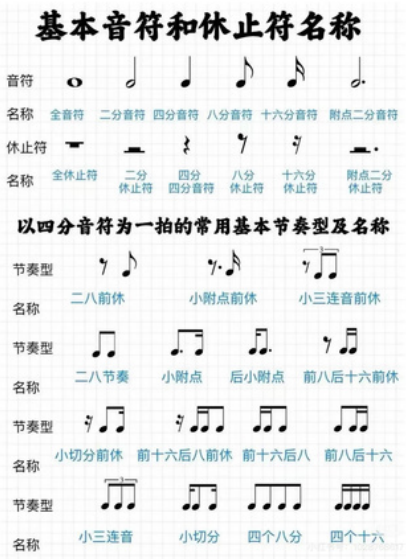
Journal ——week 8

Work progress

```
# Define the full melody of (the particles)
#Part 1 + Part 2, repeated twice

def _melody_particles():
    for i in range(2):   #  Repeat the full melody twice

        # Part 1: Opening melody - part of main theme
        for i in range(1):
            #Triplet notes
            playNote(72, beats=0.5)  # do 1st
            playNote(72, beats=0.5)  # do 2nd
            playNote(72, beats=0.5)  # do 3rd

            # Four sixteenth notes
            playNote(79, beats=0.5)  # so 1st
            playNote(79, beats=0.5)  # so 2nd
            playNote(79, beats=0.5)  # so 3rd
            playNote(79, beats=0.5)  # so 4th
            playNote(79, beats=1)    # so - hold the note longer
```

```
        playNote(79, beats=0.5)  # so
        playNote(79, beats=0.5)  # so
        playNote(79, beats=0.5)  # so
        playNote(79, beats=0.5)  # so
        playNote(79, beats=0.5)  # so

        #Rhythm: Eighth note followed by two sixteenth notes
        playNote(79, beats=0.7)  # so — hold the note longer
        playNote(78, beats=0.5)  # fa  (F sharp)
        playNote(79, beats=1)    # so  (legato) - hold the note longer

        playNote(76, beats=0.5)  # mi
        playNote(75, beats=2)  # re (D sharp),(legato)- hold the note longer

        # one beat rest - ends the first phrase
```

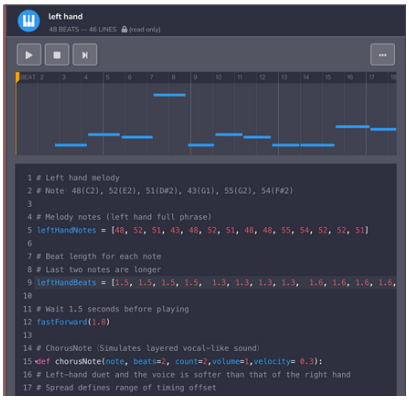
```
        # Part 2: Extended melody section - variation & echo
        for _ in range(3):
            playNote(72, beats=0.5)  # do x3   # (Triplet notes)

        for _ in range(4):
            playNote(79, beats=0.5)  # so x4   # (Four sixteenth notes)
        playNote(79, beats=1)  # hold the last so

        playNote(83, beats=0.5)  # si
        playNote(82, beats=0.5)  # la

        for _ in range(4):
            playNote(79, beats=0.5)  # so x4 again
        playNote(79, beats=1)

        playNote(78, beats=0.5)  # fa  (F sharp)
        playNote(76, beats=0.5)  # mi
        playNote(75, beats=1.2)  # re  (D sharp) - slow finish

        playNote(68, beats=0.7)  #low so
        playNote(76, beats=0.5)  # mi
        playNote(75, beats=1)  # re - hold the note longer

#run the full melody
_melody_particles()
```

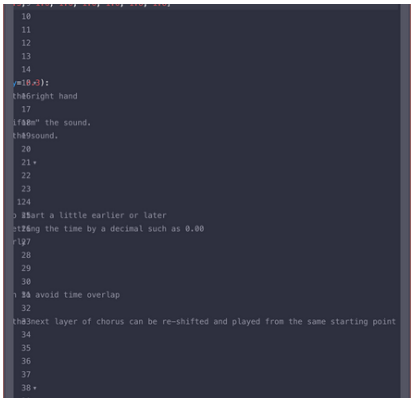


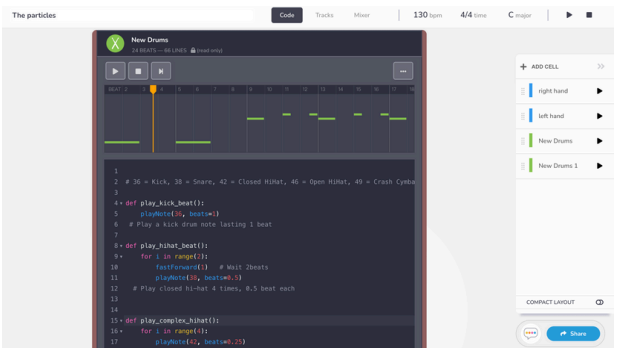
Week 8 —— Based on the peer review, start to add more detailed comments to the perfect right-hand main melody.

Journal ——week 9 ,10

Work progress



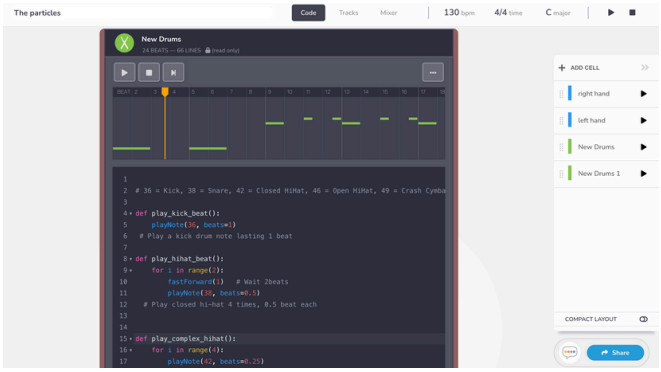


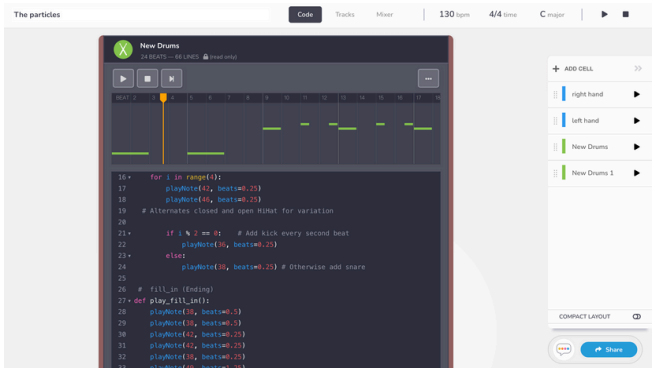


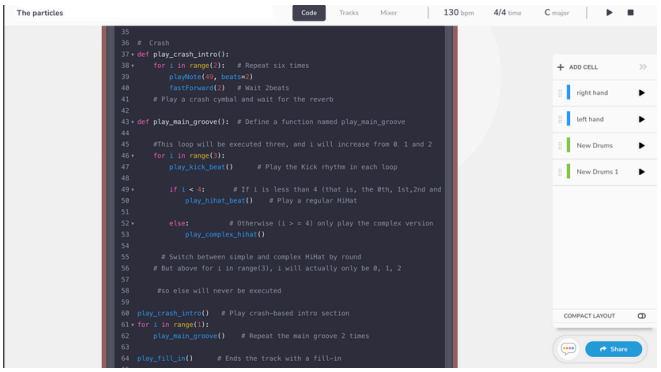
week 9 ,10 —— the accompaniment for the left hand was written using the phrase "the same meaning, but different methods", and annotations were completed, citing complex functions.

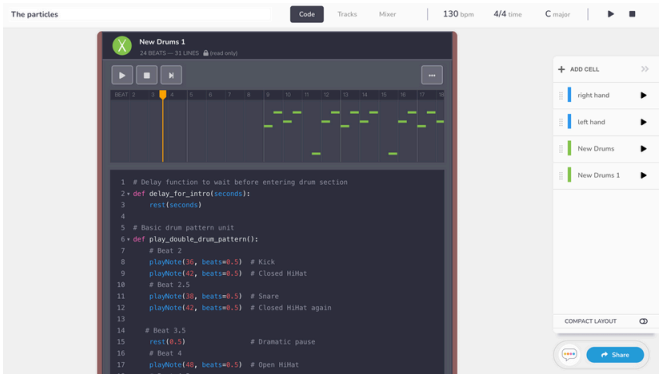
Journal ——week 11

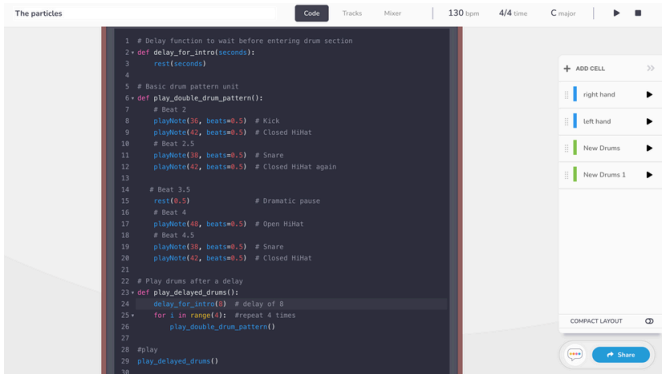
Work progress







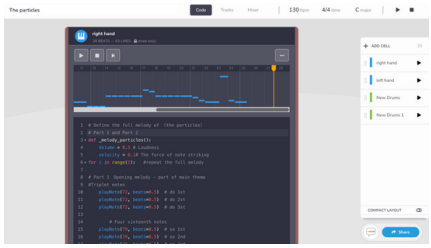
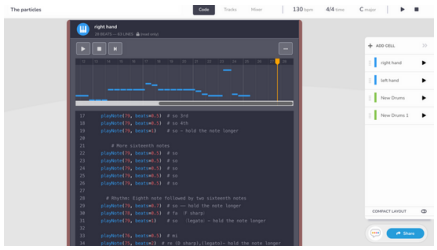
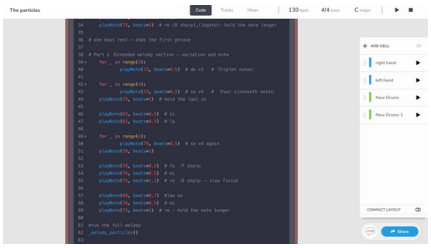
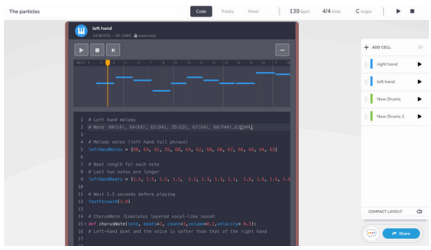
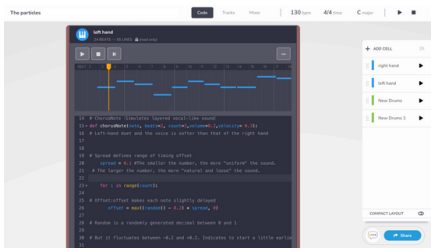
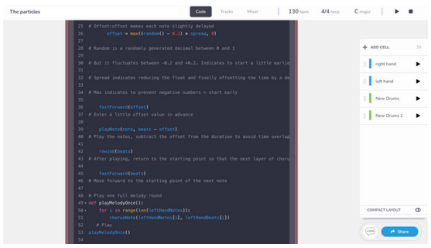
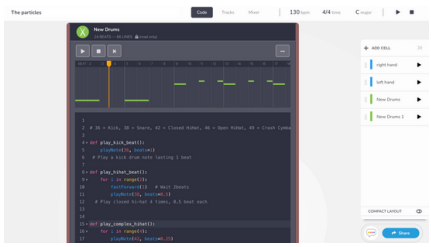
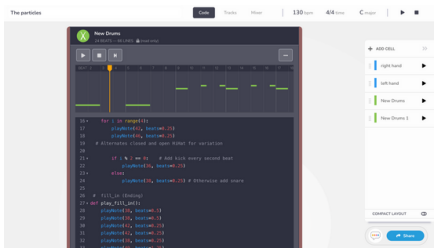
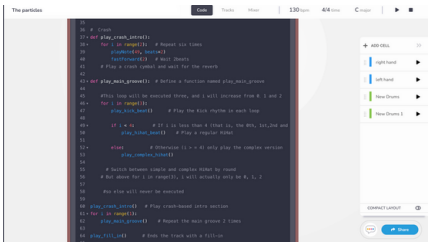
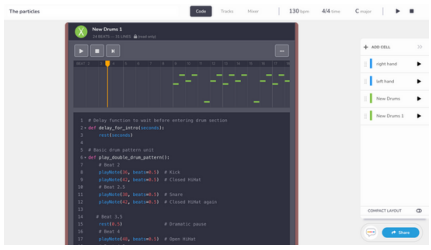
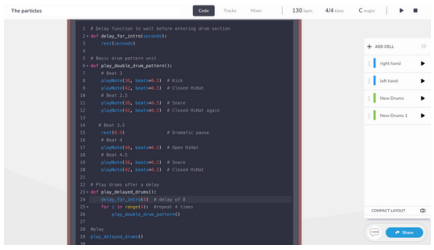




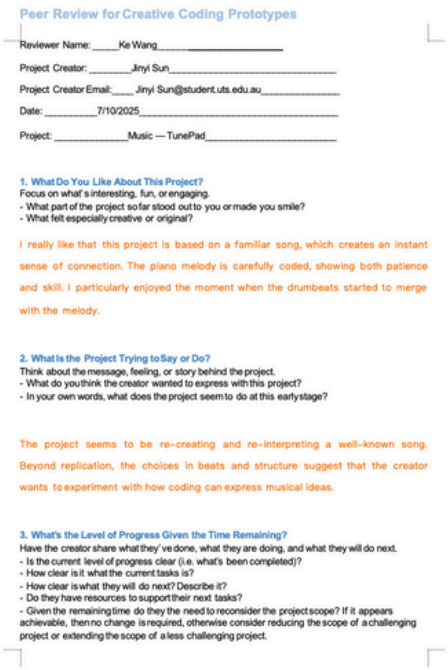
week 11—— The drum beat part was improved, with double drum tracks and drum elements such as Kick, Snare, Hi-hat, Clap, and Tom fill introduced.

Journal ——week 12

Work progress

Peer Review 1



Peer Review for Creative Coding Prototypes

Reviewer Name: Ke Wang

Project Creator: Jinyi Sun

Project Creator Email: Jinyi Sun@student.uts.edu.au

Date: 7/10/2025

Project: Music — TunePad

1. What Do You Like About This Project?
Focus on what's interesting, fun, or engaging.
- What part of the project so far stood out to you or made you smile?
- What felt especially creative or original?

I really like that this project is based on a familiar song, which creates an instant sense of connection. The piano melody is carefully coded, showing both patience and skill. I particularly enjoyed the moment when the drumbeats started to merge with the melody.

2. What Is the Project Trying to Say or Do?
Think about the message, feeling, or story behind the project.
- What do you think the creator wanted to express with this project?
- In your own words, what does the project seem to do at this early stage?

The project seems to be re-creating and re-interpreting a well-known song. Beyond replication, the choices in beats and structure suggest that the creator wants to experiment with how coding can express musical ideas.

3. What's the Level of Progress Given the Time Remaining?
Have the creator share what they've done, what they are doing, and what they will do next.
- Is the current level of progress clear (i.e. what's been completed)?
- How clear is it what the current tasks is?
- How clear is what they will do next? Describe it?
- Do they have resources to support their next tasks?
- Given the remaining time do they the need to reconsider the project scope? If it appears achievable, then no change is required, otherwise consider reducing the scope of a challenging project or extending the scope of a less challenging project.

Peer Review for Creative Coding Prototypes

So far, the main melody is complete and the drum patterns are functioning with multiple layers (kick, snare, hi-hat). This shows about two-thirds of the work has been done. The next steps are clear: refine transitions between sections, balance instrument volumes, and possibly add more harmony or bass. Given the remaining time, the scope looks realistic, and the focus can stay on polishing rather than reducing or expanding.

4. How Well Does It Work?
Look at both the idea and what's been built so far.
- Is there something in the project that worked really well or surprised you?
- Is there anything that didn't work or seemed confusing?

The prototype works well. The piano tones are accurate and recognizable, while the drum track brings energy and rhythm. One thing that could be improved is the transition between melody and drum — sometimes it feels a bit sudden. But overall, the combination of accuracy and creativity.

4. How Clear is the Code?
Have the creator explain their code and what it does.

- How easy or difficult is the code to read? Does it need more line spaces to make it clearer?
- How well has it been commented? Are more comments needed?
- How clear are the comments at explaining the code? Do they need to better describe the code?
- How much sense do the names for variables, functions etc. make in relation to what they do?

The code is well-structured, and function names like play_kick_beat or melody particles make sense. However, the comments are minimal — just short notes. Adding clearer comments about why certain choices were made (e.g., why a loop is repeated or why rests are inserted) would make the code more understandable to others.

Peer Review for Creative Coding Prototypes

5. What's One Idea to Make It Even Better?
Offer one helpful, kind suggestion.
- Can you suggest one way the creator could expand, fix, or improve their project?

Adding a bass line or simple chords could make the music fuller. Another idea is to vary the velocity (volume) of notes, so the music feels more expressive.

6. Ask a Thoughtful Question
Help your classmate think more about their creative or coding process.
- What question can you ask that gets them thinking about how or why they made something?

How do you balance staying true to the original song with adding your own creative interpretation? Would you consider experimenting with tempo shifts, syncopation, or even an improvised section?

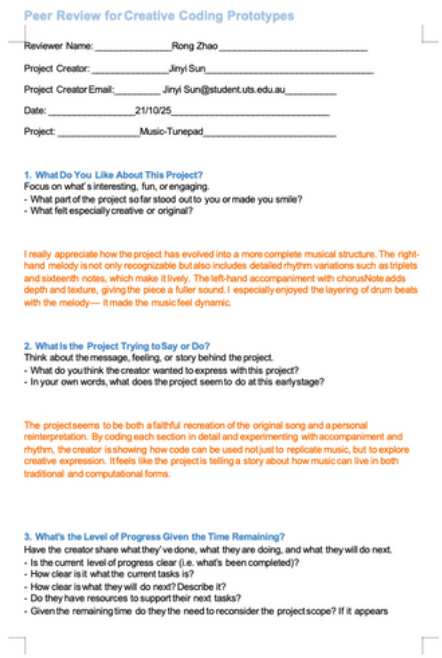
7. Final Thoughts
Share your overall experience with the project.
- In one or two sentences, how would you describe the project and your reaction to it?

This project is enjoyable and impressive. The melody and rhythm are already solid, and with more creative variation, it could become a polished piece that feels both faithful and unique. I look forward to hearing the final version!

*Remember to document your progress, thoughts, reflections and so in in your journal!

https://studentutsedu-my.sharepoint.com/:w:/g/personal/jinyi_sun_student_uts_edu_au/EfRAFBdZOhhIhRvyaF7S5aABeF__Q8Z-8kVHd4725Q0llw?e=jMFrGa

Peer Review 2



Peer Review for Creative Coding Prototypes

Reviewer Name: _____Rong Zhao_____

Project Creator: _____Jinyi Sun_____

Project Creator Email:_____ Jinyi Sun@student.uts.edu.au_____

Date: _____21/10/25_____

Project: _____Music-Tunepad_____

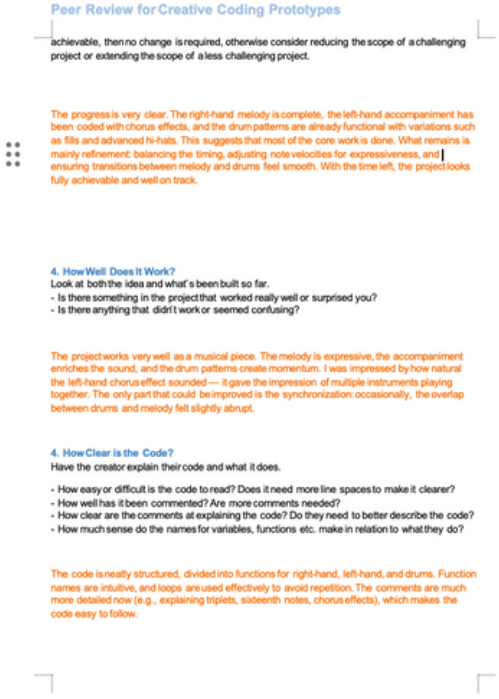
1. What Do You Like About This Project?
Focus on what's interesting, fun, or engaging.
- What part of the project so far stood out to you or made you smile?
- What felt especially creative or original?

I really appreciate how the project has evolved into a more complete musical structure. The right-hand melody is not only recognizable but also includes detailed rhythm variations such as triplets and sixteenth notes, which make it lively. The left-hand accompaniment with chorusNote adds depth and texture, giving the piece a fuller sound. I especially enjoyed the layering of drum beats with the melody— it made the music feel dynamic.

2. What Is the Project Trying to Say or Do?
Think about the message, feeling, or story behind the project.
- What do you think the creator wanted to express with this project?
- In your own words, what does the project seem to do at this early stage?

The project seems to be both a faithful recreation of the original song and a personal reinterpretation. By coding each section in detail and experimenting with accompaniment and rhythm, the creator is showing how code can be used not just to replicate music, but to explore creative expression. It feels like the project is telling a story about how music can live in both traditional and computational forms.

3. What's the Level of Progress Given the Time Remaining?
Have the creator share what they've done, what they are doing, and what they will do next.
- Is the current level of progress clear (i.e. what's been completed)?
- How clear is it what the current tasks is?
- How clear is what they will do next? Describe it?
- Do they have resources to support their next tasks?
- Given the remaining time do they the need to reconsider the project scope? If it appears



Peer Review for Creative Coding Prototypes

achievable, then no change is required, otherwise consider reducing the scope of a challenging project or extending the scope of a less challenging project.

The progress is very clear. The right-hand melody is complete, the left-hand accompaniment has been coded with chorus effects, and the drum patterns are already functional with variations such as fills and advanced hi-hats. This suggests that most of the core work is done. What remains is mainly refinement: balancing the timing, adjusting note velocities for expressiveness, and ensuring transitions between melody and drums feel smooth. With the time left, the project looks fully achievable and well on track.

4. How Well Does It Work?
Look at both the idea and what's been built so far.
- Is there something in the project that worked really well or surprised you?
- Is there anything that didn't work or seemed confusing?

The project works very well as a musical piece. The melody is expressive, the accompaniment enriches the sound, and the drum patterns create momentum. I was impressed by how natural the left-hand chorus effect sounded— it gave the impression of multiple instruments playing together. The only part that could be improved is the synchronization: occasionally, the overlap between drums and melody felt slightly abrupt.

4. How Clear is the Code?
Have the creator explain their code and what it does.

- How easy or difficult is the code to read? Does it need more line spaces to make it clearer?
- How well has it been commented? Are more comments needed?
- How clear are the comments at explaining the code? Do they need to better describe the code?
- How much sense do the names for variables, functions etc. make in relation to what they do?

The code is neatly structured, divided into functions for right-hand, left-hand, and drums. Function names are intuitive, and loops are used effectively to avoid repetition. The comments are much more detailed now (e.g., explaining triplets, sixteenth notes, chorus effects), which makes the code easy to follow.

Peer Review for Creative Coding Prototypes

5. What's One Idea to Make It Even Better?
Offer one helpful, kind suggestion.
- Can you suggest one way the creator could expand, fix, or improve their project?

One idea would be to introduce dynamic changes in volume or tempo to create contrast between sections. For example, making the left hand quieter in some places and louder in others would simulate real musical expression.

6. Ask a Thoughtful Question
Help your classmate think more about their creative or coding process.
- What question can you ask that gets them thinking about how or why they made something?

What inspired you to add the chorus effect in the left hand? Do you see this as a way to move beyond simple reproduction and add your own musical identity?

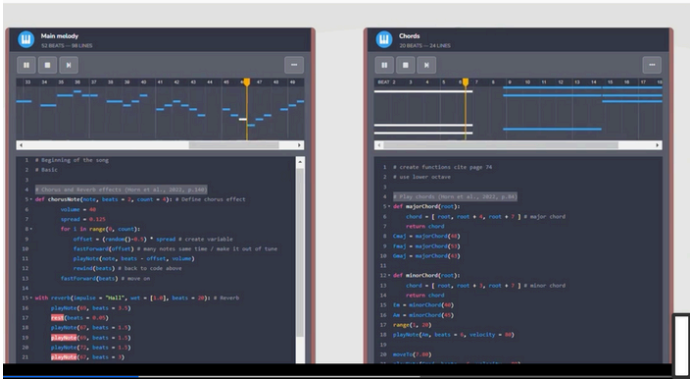
7. Final Thoughts
Share your overall experience with the project.
- In one or two sentences, how would you describe the project and your reaction to it?

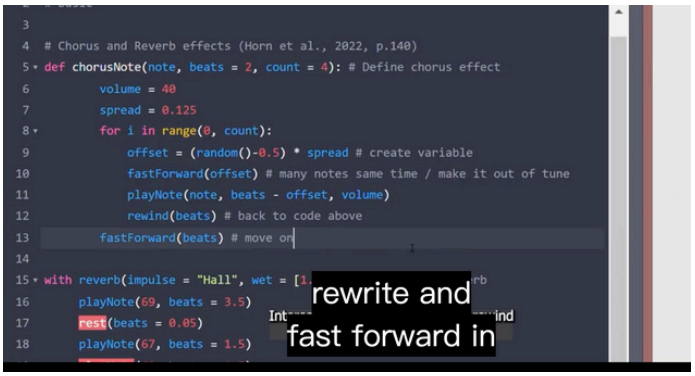
This is a strong project that shows both technical skill and musical creativity. The right-hand melody, left-hand accompaniment, and drum rhythms already create a cohesive and enjoyable piece. With a little more polish, it could sound like a performance rather than just code.

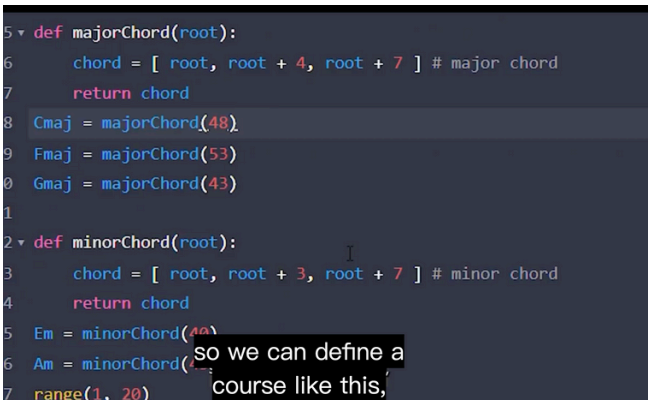
*Remember to document your progress, thoughts, reflections and so in in your journal!

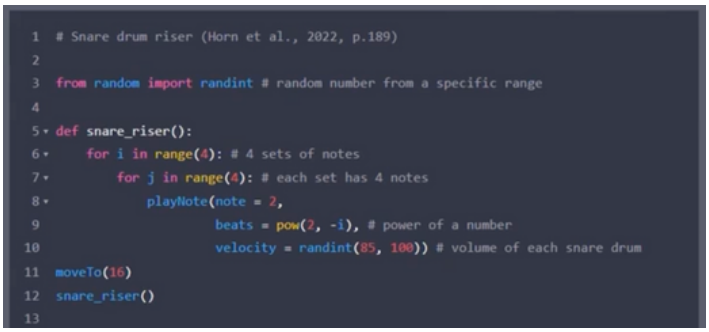
https://studentutsedu-my.sharepoint.com/:w:/g/personal/jinyi_sun_student_uts_edu_au/EdY2Nx2giJ9EuCfKMJylmeIBlF8IqtqVn1TYuQvcqo1Glw?e=vJKPhO

Source 1

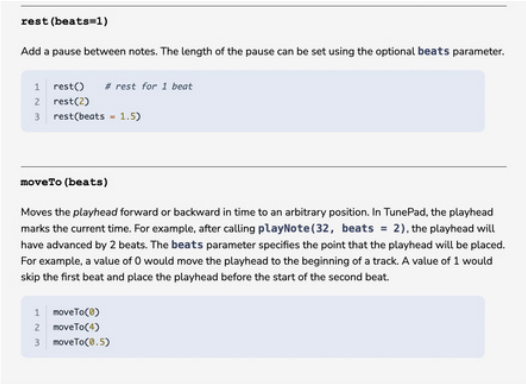

Main melody
Chords


3
4 # Chorus and Reverb effects (Horn et al., 2022, p.140)
5 def chorusNote(note, beats = 2, count = 4): # Define chorus effect
6 volume = 40
7 spread = 0.125
8 for i in range(0, count):
9 offset = (random()-0.5) * spread # create variable
10 fastForward(offset) # many notes same time / make it out of tune
11 playNote(note, beats - offset, volume)
12 rewind(beats) # back to code above
13 fastForward(beats) # move on
14
15 with reverb(impulse = "Hall", wet = [1.
16 playNote(69, beats = 3.5)
17 rest(beats = 0.05)
18 playNote(67, beats = 1.5)
rewrite and
fast forward in


5 def majorChord(root):
6 chord = [root, root + 4, root + 7] # major chord
7 return chord
8 Cmaj = majorChord(48)
9 Fmaj = majorChord(53)
0 Gmaj = majorChord(43)
1
2 def minorChord(root):
3 chord = [root, root + 3, root + 7] # minor chord
4 return chord
5 Em = minorChord(40)
6 Am = minorChord(
7 range(1, 20)
so we can define a
course like this,


1 # Snare drum riser (Horn et al., 2022, p.189)
2
3 from random import randint # random number from a specific range
4
5 def snare_riser():
6 for i in range(4): # 4 sets of notes
7 for j in range(4): # each set has 4 notes
8 playNote(note = 2,
9 beats = pow(2, -i), # power of a number
10 velocity = randint(85, 100)) # volume of each snare drum
11 moveTo(16)
12 snare_riser()
13

Source 2



rest(beats=1)

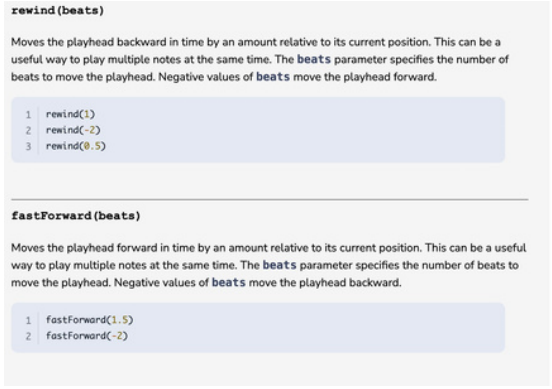
Add a pause between notes. The length of the pause can be set using the optional **beats** parameter.

```
rest()    # rest for 1 beat
rest(2)
rest(beats = 1.5)
```

moveTo(beats)

Moves the *playhead* forward or backward in time to an arbitrary position. In TunePad, the playhead marks the current time. For example, after calling **playNote(32, beats = 2)**, the playhead will have advanced by 2 beats. The **beats** parameter specifies the point that the playhead will be placed. For example, a value of 0 would move the playhead to the beginning of a track. A value of 1 would skip the first beat and place the playhead before the start of the second beat.

```
moveTo(0)
moveTo(4)
moveTo(0.5)
```



rewind(beats)

Moves the playhead backward in time by an amount relative to its current position. This can be a useful way to play multiple notes at the same time. The **beats** parameter specifies the number of beats to move the playhead. Negative values of **beats** move the playhead forward.

```
rewind(1)
rewind(-2)
rewind(0.5)
```

fastForward(beats)

Moves the playhead forward in time by an amount relative to its current position. This can be a useful way to play multiple notes at the same time. The **beats** parameter specifies the number of beats to move the playhead. Negative values of **beats** move the playhead backward.

```
fastForward(1.5)
fastForward(-2)
```

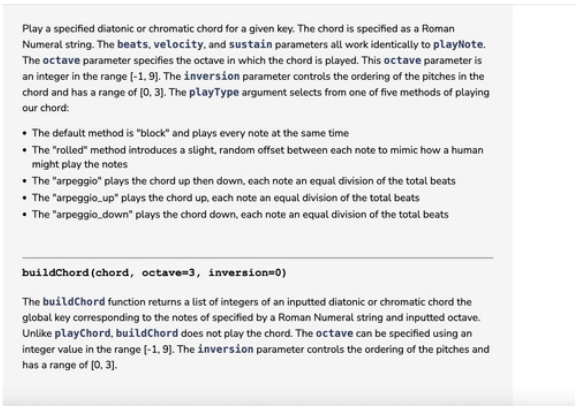
Note durations are represented by the traditional note type (e.g. quarter or half).

Examples:

```
from tunepad.constants import *

print(WHOLE_NOTE)          # prints 4.0
print(QUARTER_NOTE_DOTTED) # prints 1.5
print(TRIPLET_NOTE)        # prints 0.33333

playNote(C5, HALF_NOTE)

for _ in range(6):
    playNote(A4, beats = SEXTUPLET_NOTE)

playSound(123, EIGHTH_NOTE)
```

The full set of values is:

```
DOUBLE_WHOLE_NOTE = 8.0
WHOLE_NOTE = 4.0

HALF_NOTE_DOTTED = 3.0
HALF_NOTE = 2.0

QUARTER_NOTE_DOTTED = 1.5
```

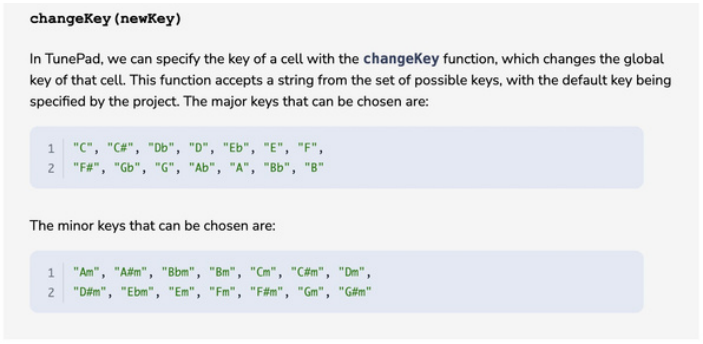


Play a specified diatonic or chromatic chord for a given key. The chord is specified as a Roman Numeral string. The **beats**, **velocity**, and **sustain** parameters all work identically to **playNote**. The **octave** parameter specifies the octave in which the chord is played. This **octave** parameter is an integer in the range [-1, 9]. The **inversion** parameter controls the ordering of the pitches in the chord and has a range of [0, 3]. The **playType** argument selects from one of five methods of playing our chord:

- The default method is "block" and plays every note at the same time
- The "rolled" method introduces a slight, random offset between each note to mimic how a human might play the notes
- The "arpeggio" plays the chord up then down, each note an equal division of the total beats
- The "arpeggio_up" plays the chord up, each note an equal division of the total beats
- The "arpeggio_down" plays the chord down, each note an equal division of the total beats

buildChord(chord, octave=3, inversion=0)

The **buildChord** function returns a list of integers of an inputted diatonic or chromatic chord the global key corresponding to the notes of specified by a Roman Numeral string and inputted octave. Unlike **playChord**, **buildChord** does not play the chord. The **octave** can be specified using an integer value in the range [-1, 9]. The **inversion** parameter controls the ordering of the pitches and has a range of [0, 3].



changeKey(newKey)

In TunePad, we can specify the key of a cell with the **changeKey** function, which changes the global key of that cell. This function accepts a string from the set of possible keys, with the default key being specified by the project. The major keys that can be chosen are:

```
"C", "C#", "Db", "D", "Eb", "E", "F",
"F#", "Gb", "G", "Ab", "A", "Bb", "B"
```

The minor keys that can be chosen are:

```
"Am", "A#m", "Bbm", "Bm", "Cm", "C#m", "Dm",
"D#m", "Ebm", "Em", "Fm", "F#m", "Gm", "G#m"
```

TunePad. (n.d.). TunePad documentation. Retrieved October 28, 2025, from https://learn.tunepad.com/docs/

Source 3