Reflection

Developing Code Literacy Through "Midnight Groove"

Introduction

Over the course of ten weeks, I developed a three-minute Jazz-Pop composition titled *Midnight Groove* using TunePad's Python-based music coding environment. This project was not only about producing a finished piece of music but also about building my code literacy, synthesising theory and practice, and critically reflecting on the creative-technical process. My reflection follows the *What?- So what? - Now what?* framework, connecting practical experience with coursework readings, journal entries and peer feedback.

What?

The project began with a clear scope: a 3:00 track at 120 BPM (360 beats), structured into an intro, jazz-focused middle section (1:00–2:00), pop-focused finale (2:00–3:00), and outro. My success criteria included musicality, exact runtime, reproducibility, and readability of code.

In the early weeks, I experimented with TunePad's environment, first building a drum loop, then layering a bassline with walking motifs and extended jazz chords (Am7–Dm7–G7–Cmaj7). These foundational layers established both the harmonic skeleton and my understanding of bar-beat calculations. As my Week 2 journal entry highlights, I quickly realised the importance of aligning all instrumental cells with drum bars to avoid timing drift.

By Week 3, I created melodic hooks using "blue notes" and experimented with swing feel on hi-hats. These small rhythmic shifts fundamentally changed the track's identity, teaching me that code is not just functional but expressive. Week 4 became a turning point: I recalculated loops to ensure the runtime stopped exactly at 3:00. Writing helper functions to assert bar counts gave me a mathematical confidence in coding, moving me beyond experimentation into precision.

The mid-project introduced genre contrasts. In Week 5, I layered a saxophone improvisation using chord extensions (9ths and 13ths), giving the track a strong jazz identity. By Week 6, I coded a pop bass groove and guitar riffs, producing a radio-style finale. Peer testers later described this section as "catchy" and suggested repeating motifs at the end, feedback I implemented.

Week 7 focused on atmosphere: a 30-second intro with piano arpeggios and a fading outro coded through velocity changes. Debugging in Week 8 became critical: I discovered TunePad rejects volume parameters and requires MIDI velocity values (0–127). Fixing these errors taught me not only about the platform's constraints but about the discipline of debugging itself.

Finally, in Weeks 9–10, I documented my code with comments and created both modular cells and a consolidated "mix" cell using rests and conditionals. I exported the track, tested peer reactions, and finalised a README for GitHub submission.

So what?

Several insights emerged from this iterative process, each reinforcing my growth in code literacy.

Code as Creative Literacy

Initially, I saw coding as a technical requirement. Through this project, I came to understand code as a form of creative expression. Baym (2018) argues that digital media is a site of identity construction, and my use of Python in TunePad reflected this—my composition was not merely "functional code" but an articulation of my personal taste in jazz and pop fusion. The process aligns with Della Porta's (2020) view that digital tools can enable creative autonomy, here expressed through algorithmic improvisation.

Debugging as Literacy Practice

The debugging incidents—particularly resolving velocity errors—were frustrating but transformative. As Ciampa et al. (2023) emphasise, digital literacy involves persistence through ambiguity. Each bug forced me to question my assumptions, test systematically, and consult documentation. My Week 8 journal highlights how this sharpened my technical attention to detail. Debugging thus became more than fixing errors; it was an epistemic practice that deepened my literacy.

Timing as Analytical Thinking

Calculating 360 beats at 120 BPM and designing helper functions for bar counts built a bridge between music theory and computational logic. This demonstrated what Lupton (2016) calls "the quantified self" in digital practice—where personal creativity is translated into measurable data. By controlling timing algorithmically, I learned to integrate precision with artistry, moving beyond intuition toward reproducibility.

Peer Feedback as Co-Creation

Peer testing influenced my final structure, particularly repeating a melodic motif in the last 30 seconds. This illustrated Couldry and Mejias' (2019) argument that digital production is relational: creativity emerges not in isolation but through interaction. My code became more audience-aware, adapting to feedback loops both literal (audio) and social.

Theory–Practice Integration

Across the project, I consistently drew from course readings and tutorials. For example, Meadows and Molnar's discussion of media representation encouraged me to reflect on genre blending—how jazz signifiers (blue notes, extensions) and pop conventions (hooks, riffs) could coexist in coded form. Similarly, Fredericks et al.'s idea of "disrupting the colonial algorithm" resonated with my choice to use improvisational motifs, subverting the stereotype that code must always be rigid or deterministic.

Now what?

Reflecting on this experience, several lessons stand out for future application.

First, **structured planning is critical**. Beginning with clear success criteria (runtime, readability) gave me benchmarks to evaluate progress. In future projects, I will continue to set measurable goals early, ensuring creative exploration remains anchored by technical rigour.

Second, **debugging is not a detour but the work itself**. I now recognise errors as essential learning opportunities. Moving forward, I will embrace debugging logs and systematic testing, reframing bugs as insights into system logic rather than obstacles.

Third, **documentation is a form of collaboration**. By creating modular and commented code, I enabled remixing by peers and future-me. This aligns with open-source practices and has implications for my professional trajectory: as a digital and social media student, I see parallels between code documentation and transparent communication in online communities.

Finally, this project has reinforced the **interdisciplinary nature of code literacy**. Music coding required me to blend artistic creativity with algorithmic reasoning. In the future, whether designing interactive media, analysing platform data, or experimenting with generative tools, I will carry forward this mindset: that code is not simply technical but cultural, expressive, and collaborative.

Conclusion

Through the creation of *Midnight Groove*, I experienced a transformative development in my code literacy. Applying the *What? – So what? – Now What?* framework allowed me to critically reflect on key incidents: early structuring, timing calculations, debugging velocity errors, layering genre sections, and responding to peer feedback. Each stage linked practical experience with theoretical insights from course readings, reinforcing the connection between coding, creativity, and critical reflection.

Ultimately, this project taught me that code literacy is not only about writing functional instructions but about cultivating persistence, reflexivity, and the ability to merge artistic

vision with computational logic. These skills will extend beyond this assignment into my future work in digital and social media, where creative coding increasingly shapes communication, representation, and culture.