Creativity and Reflection for Teaching Humanistic Data Programming

Suh Young Choi
schoi101@jh.edu
Johns Hopkins University
Baltimore, Maryland, USA

Abstract

This paper describes a set of pedagogical interventions introduced in an intermediate data programming course to promote humanistic thinking, critical reflection, and creativity. Through redesigned lectures, reading assignments, creative homework components, a portfolio option, and a structured final reflection, the course aimed to move beyond technical skills alone and empower students to engage meaningfully with the ethical, social, and communicative dimensions of data work. The ultimate goal of these interventions is to foster student agency in data programming and equip students to be not just competent programmers, but thoughtful practitioners.

ACM Reference Format:
Suh Young Choi. 2026. Creativity and Reflection for Teaching Humanistic Data Programming. In *Proceedings of The First Reflection in Creative Experience (RiCE) Workshop (RiCE W1).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

This paper describes a course redesign of an intermediate data programming course at a large research university, undertaken with the goal of integrating humanistic thinking and creative expression alongside technical skill. The redesign introduced several new elements: new lecture content on data literacy, algorithmic fairness, privacy, and ethics; reading assignments drawn from computing and non-computing fields; creative components in take-home assignments; a final portfolio option; and a structured written reflection in place of a traditional final exam. Together, these elements were intended to build students' capacity for critical thought, written communication, and meaningful self-reflection.

2 Prior Work

2.1 Humanistic and Ethical Dimensions of Computing Education

There is a growing body of work arguing that computing education must grapple with the epistemological assumptions embedded in computational tools and methods. Sollazzo and Cutts develop a working definition of humanistic computing as a practice that keeps the human at the forefront, attends to plurality and complexity, and is grounded in a constantly critical stance supported by humanistic theory [9].

Barkhuff's ongoing research finds strong support for computing ethics education (CEE) across computing educators, teaching assistants, and program administrators, and documents that computing TAs are particularly invested in the "humanistic" elements of their roles: inspiring students, building relationships, and attending to learning as a whole-person activity [1, 3]. Further, prior work has found that integrated ethics curricula are more effective than isolated interventions [2, 6].

More broadly, humanistic perspectives on computing have entered adjacent fields as well. Benabdallah et al.'s work on the politics of imaginaries in HCI calls for humanistic methods that center power, positionality, and the social constitution of technology [4].

2.2 Creativity in Computing Education

Groeneveld, Becker, and Vennekens's systematic literature review of creativity in computing education finds that while creativity is frequently mentioned positively in course experience reports, it is rarely taught or assessed with explicit theoretical grounding [7]. Fewer than 5% of computing courses explicitly name creativity as a learning outcome, and depth papers identify a consistent set of enabling conditions: open-ended assignments, intrinsic motivation, and a separation of creative expression from correctness evaluation. By grading Creative Components on completion with demonstrated effort rather than against a correctness rubric, the course aimed to create the conditions for what Groeneveld et al. describe as genuine creative engagement—as opposed to the "responsive creativity" that closed, externally-driven prompts tend to produce.

2.3 Soft Skills, Reflection, and Long-Term Learning

Motschnig et al.'s mixed-methods study of long-term outcomes from a master's-level course on communication and soft skills in computing provides important evidence that student-centered, experiential approaches leave lasting traces [8]. Surveying graduates up to ten years after completing the course, they find that active listening skills, constructive feedback practices, and people-related takeaways are remembered far more frequently than specific knowledge items. Critically, they argue that these effects are not incidental but are a product of a deliberate pedagogical stance: student choice in course content, peer feedback, self-evaluation, and a constructive open-error climate.

Reflective writing has similarly been shown to encourage students to synthesize knowledge and develop a sense of ownership over their learning [5].

RiCE W1, London, UK

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

The interventions described here build on this prior work, adapting them for an intermediate-level data programming course serving a large, diverse, and non-major population.

3 Course Context

The course described in this paper is an intermediate data programming course in Python at the University of Washington's Paul G. Allen School of Computer Science & Engineering. It is the second course in a two-course introductory sequence and enrolls approximately 150–200 students per offering. The course serves as a prerequisite for students in engineering, informatics, and applied mathematics and statistics, and functions as an elective for other STEM majors and data science students. Many students enroll primarily to fulfill a graduation requirement or prerequisite for another course.

The author of this paper and instructor is a veteran teaching assistant and three-time instructor of the course, with additional teaching experience in Classics, Textual and Digital Studies, and other computer science courses. This background informed the humanistic orientation of the redesign. The course offering described here took place in Winter 2026.

Previous iterations of the course integrated oral code interviews as an assessment mechanism; however, constraints on staff time and course scale made this difficult to sustain. Exploring alternative assignment formats—particularly creative ones—was in part motivated by a desire to assess student understanding of course concepts without relying on resource-intensive oral exams.

4 Design Choices

The redesign introduced five categories of change: new and revised lecture content, reading assignments, creative components in take-home assessments, a final portfolio option, and a structured final reflection. Each is described below.

4.1 New Lectures

Lectures in the course have traditionally followed a flipped-classroom model. Readings and/or short videos were released prior to each class meeting; the first 10 to 15 minutes of class were devoted to review and answering student questions; the remainder of class time was used for active learning through coding exercises, polls, and think-pair-share activities.

Seven new or significantly revised lectures were introduced:

Data Literacy and Communication. Originally written by the instructor and piloted as a bonus lesson in Summer 2023, this lecture was piloted as a graded lesson in Summer 2025 and reinstated in Winter 2026. Students learned data visualization and communication principles, data storytelling, and the foundations of technical writing.

Humanistic Computing. New material written by the instructor and launched for the first time in Winter 2026. Students examined how context, ethics, and interpretation shape computational work; recognized that technical choices embed values and perspectives rather than being neutral; and applied critical thinking to data collection and analysis.

Inheritance. New material written by the instructor, also launched in Winter 2026. Following two lessons on object-oriented programming, this lecture prepared students for an upcoming homework assignment, introduced applications of graphs and social network analysis, and provided additional object-oriented programming examples.

Statistical Testing. Adapted from existing teaching assistant section material, this lecture was elevated to a full lesson to free a section slot for student project and portfolio presentations. Students learned exploratory data analysis and hypothesis testing in preparation for their final projects, and surveyed different statistical tests and when to apply them.

Algorithmic Fairness. Adapted and updated from a previous instructor's content, this lecture introduced students to competing definitions of algorithmic fairness, the tradeoff between fairness and accuracy, and two case studies illustrating algorithmic bias in real systems.

Data and Privacy. Similarly adapted and updated, this lecture covered differential privacy, jittering, and k-anonymity, illustrated with two case studies on location tracking.

Ethics Case Studies. Building on the two preceding lectures, this class session examined four additional case studies involving location tracking, facial recognition, and the intersection of artificial intelligence and labor. Class time was devoted primarily to guided discussion.

To make room for this new content, lectures on time-series data, trees, geospatial indexing, and large language models were removed from the syllabus.

4.2 Reading Assignments

Reading assignments were administered through Hypothesis, a social annotation platform integrated with the Canvas learning management system. Students were organized by section so that their annotations were visible to their section peers. Each student was required to make two original annotations and two replies to classmates' annotations. Guided reading questions were provided as optional scaffolding, and students were encouraged to write annotations of at least two to three sentences, though the content was open-ended.

Five readings were assigned over the quarter:

Syllabus. The first reading asked students to annotate the course syllabus, familiarizing them with course logistics and surfacing questions early in the quarter, before misunderstandings could compound.

"The Origins of Python" by Lambert Meertens.[1] This reading introduced students to the history of the Python programming language and invited them to reflect on the design philosophy and value judgments embedded in a language they were already using. It also situated the creation of Python within a broader story about the demand for accessible, user-friendly programming languages.

Chapter 1 of *Data Visualization: A Practical Introduction* by Kieran Healy.[2] Timed alongside the data literacy and visualization lectures, this reading asked students to critically examine

[1] https://inference-review.com/article/the-origins-of-python
[2] https://socviz.co/01-look-at-data.html

figures using visualization principles encountered in both the reading and in class. It was also referenced in a homework assignment.

"An Introduction to the Ethics of Artificial Intelligence" by Matthew J. Gaudet,[3] originally published in the *Journal of Moral Theology*. Including a text from theology and ethics (i.e., a non-computing field) helped students appreciate how technology is understood and experienced across disciplines, and introduced them to frameworks for moral reasoning not typically encountered in CS curricula. This reading preceded the in-class discussions of algorithmic fairness, privacy, and ethics.

"On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" by Bender et al.[4] This widely-cited paper prompted students to reconsider how they interact with AI tools, examine the semantics of meaning in language models, and consider how they might engage ethically with AI technology going forward.

4.3 Creative Components

The course included five take-home assessments (THAs) over the quarter. Each THA comprised two parts: a *Technical Component* and a *Creative Component*.

The Technical Component consisted of a staff-written specification of four to five coding tasks with well-defined outcomes. These were graded for correctness via an autograder, and manually for code style, documentation, and testing completeness.

The Creative Component prompted students to define their own problem statement, design and implement a code solution, and write accompanying documentation, according to a theme or prompt. Creative Components were graded on completion with demonstrated effort against minimum requirements; there was no single correct answer.

After submission, Creative Components were exchanged anonymously between students for peer review using the "I Like, I Wish, What If" framework. Students were not grading each other's work; they were providing constructive feedback. Credit was awarded for completing a peer review, but grades were not affected by the feedback received.

Resubmissions were permitted for the Technical Components of THAs, since these could be assessed consistently by staff. Resubmissions of Creative Components were not offered during the quarter due to the logistical complexity of re-running peer review cycles.

4.4 Final Portfolio Option

Previous offerings of the course concluded with a final project in which students selected a dataset, formulated research questions and challenge goals, and answered them using the skills learned in the course. In Winter 2026, students were given the option to complete either the traditional final project or a final portfolio.

The portfolio allowed students to revise creative work submitted during the quarter, providing a meaningful opportunity for reflection and revision that the one-time submission model of the THAs did not afford. The portfolio was completed individually. Its components were designed to parallel those of the final project, as shown in Table 1.

[3]https://jmt.scholasticahq.com/article/34121-an-introduction-to-the-ethics-of-artificial-intelligence

[4]https://dl.acm.org/doi/10.1145/3442188.3445922

Table 1: Analogous components of the final project and final portfolio.

Final Project	Final Portfolio
Proposal	Vision Statement
Exploratory Data Analysis	Milestone
Report	Report
Presentation	Presentation
Peer Feedback	Peer Feedback

There was a roughly even split between students who chose the final project and students who chose the final portfolio.

4.5 Final Reflection

The course has historically not held a traditional final exam; however, a two-hour final exam period is assigned by the university. Previous offerings used this time for a "science fair"-style project showcase or for makeup oral exams. In Winter 2026, the exam period was used for a structured written reflection.

Students received five reflection prompts approximately one month before the final period, allowing time to think through responses in advance. Students were required to respond to two prompts and chose one of three additional prompts. The prompts were:

(1) *(Required)* Choose two concepts, skills, or ideas from this course and reflect on how you might apply them in your life or future work. Be specific: rather than saying "I learned X and I'll use it for Y," dig into the how and why. What would it actually look like to use these things? What problems might they help you solve, and why are you the kind of person who might encounter those problems?
(2) *(Required)* This quarter, we piloted several new elements: reading assignments, creative components, peer feedback, a portfolio option, and this final reflection. Choose one and reflect honestly on how it affected your experience and learning. Critical feedback is as valuable as positive feedback.
(3) *(Choice)* Which homework assignment from this quarter was your favorite, and why? You might write about what you learned, what you would do differently now, what surprised you, or something else entirely.
(4) *(Choice)* Describe a moment this quarter when an interaction with a classmate or teaching assistant shifted how you thought about a concept, an assignment, or your approach to programming more broadly. What happened, and why did it stick with you?
(5) *(Choice)* Think back to why you enrolled in this course, even if the answer is simply "it was a requirement." Did the course meet your expectations? Did you get more out of it than you anticipated, or less?

Releasing the prompts in advance served two purposes: it reduced anxiety by giving students time to prepare their thoughts, and it encouraged students to pay closer attention to the rest of the quarter's work as they considered what they might say.

5 Takeaways

The observations reported here are drawn from anecdotal data: student responses to reading assignments, peer feedback on creative components, a mid-quarter survey, and the final reflections themselves. Results are presented as general trends rather than individual responses, and no formal study was conducted.

Overall, students responded positively to the course changes and offered constructive feedback on specific elements. Several themes emerged consistently.

Valuing "soft skills" alongside technical skills. Students frequently cited skills like critical reading, written communication, analysis, and presentation as valuable outcomes of the course—not just proficiency with Python, data structures, or machine learning. Many expressed surprise that a programming course had given them opportunities to develop these capacities.

Appreciation for completion-based grading of creative work. Students reported that the completion-with-effort grading model for Creative Components reduced anxiety and gave them permission to take risks and be genuinely creative, rather than optimizing for a correct answer.

Reading assignments were more manageable than expected. A common theme in the mid-quarter survey and final reflections was that students had anticipated the reading assignments would be burdensome, but found them more accessible and engaging than expected. The annotation format in particular helped students engage actively rather than passively.

Exceeded expectations for a requirement course. Many students enrolled in the course solely to satisfy a graduation requirement or prerequisite. These students frequently reported leaving the course with more than they had anticipated, citing the humanistic and reflective elements as unexpected sources of value.

The final reflection was a welcome alternative to an exam. Students appreciated that the final reflection required no studying in the traditional sense, and that it came after all other coursework had been submitted. From the instructor's perspective, the structured session produced more thoughtful and substantive responses than reflective prompts embedded in other assignments, where engagement tended to be more perfunctory.

Portfolio students demonstrated meaningful growth. Students who chose the portfolio option frequently noted in their reflections that they had been surprised by their own progress over the quarter. Many demonstrated a genuine ability to engage with and incorporate peer feedback in their revisions, suggesting that the portfolio structure supported a more integrated and reflective learning experience.

6 Discussion and Future Work

Integrating humanistic content into a technical course inevitably involves tradeoffs. Several topics present in prior offerings were removed to make room for the new lecture material. Similarly, one teaching assistant section was repurposed for project and portfolio presentations, and the exam period was redirected from a showcase to a reflection session. These are genuine tradeoffs, and future work should examine whether students who complete the redesigned course are meaningfully disadvantaged in downstream courses that expect familiarity with removed content.

That said, the instructor's view is that these tradeoffs are worth making. The course has historically attracted students for whom this will be their last or only formal exposure to computing, and equipping them with frameworks for thinking critically about data and technology may be more durable than additional coverage of specific data structures.

One of the clearest takeaways from this offering is that giving students structured time and advance notice for reflection produces qualitatively richer responses than ad-hoc reflective prompts embedded in assignments. This suggests that treating reflection as a first-class activity rather than an afterthought yields meaningfully different outcomes.

Further, the final reflection did not succeed in isolation. Students' capacity to engage thoughtfully with the reflection prompts was supported by the reading assignments, which practiced critical annotation; the creative components and peer review, which practiced giving and receiving substantive feedback; and the portfolio, which practiced structured self-assessment. Embedding these smaller acts of reflection throughout the quarter appeared to scaffold the more extended reflection at the end.

Several directions for future development stand out. First, examining how different reading selections and annotation tasks affect the depth and character of student engagement would help refine the reading assignment component. Second, allowing resubmission of Creative Components (with or without another round of peer review) would provide evidence on whether revision opportunities meaningfully improve creative work, and whether the logistical overhead is manageable at scale. Third, developing more systematic training for teaching staff on how to assess creative and humanistic student work would help ensure consistent and equitable evaluation across sections. Finally, a more formal evaluation of the interventions, such as using pre- and post-course surveys, analysis of reflection artifacts, and comparison with prior offerings, would strengthen the evidence base for these design choices.

References

[1] Grace Barkhuff. 2025. Ethical Computing Education in the Age of Generative AI. (Aug. 2025), 46–47. doi:10.1145/3702653.3744305

[2] Grace Barkhuff, Jason Borenstein, Daniel Schiff, Judith Uchidiuno, and Ellen Zegura. 2024. Considerations for Improving Comprehensive Undergraduate Computing Ethics Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024)*. ACM, Portland, OR, USA, 1560–1561. doi:10.1145/3626253.3635557

[3] Grace Barkhuff, Ian Pruitt, Vyshnavi Namani, William Gregory Johnson, Rodrigo Borela, Ellen Zegura, Anu G. Bourgeois, and Ben Rydal Shapiro. 2025. Exploring the Humanistic Role of Computer Science Teaching Assistants across Diverse Institutions. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSETS 2025)*. ACM, Pittsburgh, PA, USA, 67–73. doi:10.1145/3641554.3701861

[4] Gabrielle Benabdallah, Michael W. Beach, Nathanael Elias Mengist, Daniela Rosner, Kavita Philip, and Lucy Suchman. 2023. The Politics of Imaginaries: Probing Humanistic Inquiry in HCI. In *Designing Interactive Systems Conference (DIS Companion '23)*. ACM, Pittsburgh, PA, USA, 131–134. doi:10.1145/3563703.3591457

[5] David Boud, Rosemary Keogh, and David Walker. 2013. *Enhancing Learning Through Self-Assessment*. Routledge.

[6] Noelle Brown, Benjamin Xie, Ella Sarder, Casey Fiesler, and Eliane S. Wiese. 2024. Teaching Ethics in Computing: A Systematic Literature Review of ACM Computer Science Education Publications. *ACM Transactions on Computing Education* 24, 1, Article 6 (Jan. 2024), 36 pages. doi:10.1145/3634685

[7] Wouter Groeneveld, Brett A. Becker, and Joost Vennekens. 2022. How Creatively Are We Teaching and Assessing Creativity in Computing Education: A Systematic Literature Review. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022)*. ACM, Providence, RI, USA, 934–940. doi:10.1145/3478431.3499360

[8] Renate Motschnig, Dominik Dolezal, Valdemar Švábenský, Katarína Palubová, and Michael Silber. 2026. Long-term Effects of Promoting Communication and Soft Skills in Higher Computing Education. *ACM Transactions on Computing Education* 26, 3, Article 53 (April 2026), 30 pages. doi:10.1145/3800569

[9] Anna Sollazzo and Quintin Cutts. 2024. Towards a Theory of Humanistic Computing and How to Teach It. In *24th Koli Calling International Conference on Computing Education Research (Koli Calling '24)*. ACM, Koli, Finland, 1–11. doi:10.1145/3699538.3699559