Journal of Inclusive Methodology and Technology in Learning and Teaching
ISSN 2785-5104
Anno 6 n. 4 (2026)

EDIZIONI UNIVERSITARIE ROMANE
editoria scientifica dal 1952

When the "Easy" Tool Is the Hard One: Task Authenticity and the Block-to-Text Transition

Katiuscia Mannaro [1,*] and Roberto Tonelli [1]

[1] Department of Mathematics and Computer Science, University of Cagliari, Italy; katiuscia.mannaro@unica.it, roberto.tonelli@unica.it
***** Correspondence: katiuscia.mannaro@unica.it

Abstract: Block-based environments such as Scratch are commonly recommended as a first step into programming because they reduce syntactic demands. In a Communication Sciences degree, however, students who moved from Scratch to Python spontaneously described the text-based language as more engaging and more useful. This study examines whether that reversal holds when concurrent and retrospective evaluations are separated. Reporting the first round of data collection within a design-based research cycle, we compared first-year students rating Scratch while attending it (n = 26) with second-year students rating Python concurrently and Scratch retrospectively (n = 12), using adapted measures of cognitive load, task value and intention to abandon. Students who had experienced both languages attributed higher task value to Python than to Scratch almost unanimously, regardless of their Scratch instructor, and reported greater learning benefit. First-year students, who could not yet compare, rated Scratch as favourably as second-year students rated Python, and better than Scratch was remembered. Task value was negatively associated with intention to abandon Scratch. The advantage of text-based programming thus became visible once students had a point of comparison. What distinguished Python was the value attributed to the effort it required rather than a lower cognitive load, consistent with an account based on task authenticity. Given the small groups from a single institution, these findings are exploratory.

Keywords: task authenticity; cognitive load; adult non-STEM learners; block-based programming; task value

Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Block-based environments such as Scratch are widely used as an entry point to programming. Their rationale is well established: by replacing typed syntax with visual blocks that fit together only in valid ways, they remove a class of errors that absorbs novices' attention without contributing to computational reasoning (Bau et al., 2017). In terms of cognitive load theory, which originated in the study of problem solving (Sweller, 1988), blocks reduce *extraneous load*, the processing imposed by how a task is presented rather than by what has to be learned (Sweller et al., 2019). The transition to a text-based language is then treated as the difficult step that block-first instruction prepares for (Strong et al., 2025). In secondary schools, students working in blocks found programming easier, although they also perceived block-based tools as less authentic and less powerful than text (Weintrop & Wilensky, 2015, 2017).

EDIZIONI UNIVERSITARIE ROMANE
editoria scientifica dal 1952

Most of this evidence concerns school-age learners or prospective computer scientists. Adult learners whose disciplinary identity is not technical have received less attention, although introductory programming is increasingly part of their curricula. For these students, the question of *why* a programming activity matters may be as consequential as how difficult it is. Research on non-majors has shown that introductory courses tailored to contexts students recognise as relevant can improve their success and attitudes towards computing (Forte & Guzdial, 2005), a principle developed in the tradition of contextualized computing education (Cooper & Cunningham, 2010; Guzdial, 2010). Expectancy-value theory offers a complementary lens: alongside expectations of success, the value students attach to a task, including its relevance to their identity, its usefulness and its interest, shapes their choices, effort and persistence (Eccles & Wigfield, 2020).

Cognitive load theory has itself been reformulated. After criticism that germane load could not be empirically distinguished from intrinsic load (de Jong, 2010; Kalyuga, 2011), germane processing has been redefined as the share of working-memory resources that learners devote to the intrinsic demands of a task (Sweller, 2010). Taken together, these perspectives suggest that designing engaging programming activities for non-STEM learners is a matter of task authenticity as much as of modality. This study originates from an observation in a Communication Sciences degree at an Italian university. Students who moved from a first-year Scratch course to an optional second-year Python course spontaneously described Python as more engaging and more useful, and some students carried an unpassed Scratch examination into later years. Such impressions are retrospective and comparative: students judge the earlier tool after experiencing the later one. This can be read in two ways. Retrospective judgments may be distorted by memory and by contrast with the later experience; but they are also the only judgments informed by a point of comparison. Some attributes are difficult to evaluate in isolation and become evaluable only when alternatives are considered jointly (Hsee, 1996; Hsee et al., 1999). For a first-year student, the relevance of Scratch to communication practice may be hard to assess without knowing what else programming can offer.

This study was designed to test this observation and weigh its possible explanations: perceived task authenticity, generative AI support, selection and instructor continuity. Reporting the first round of data collection, it addresses three exploratory questions. First, do students rate Scratch and Python differently in terms of cognitive load and task value, and does the answer depend on whether ratings are concurrent or retrospective? Second, is task value associated with the intention to abandon each course? Third, to which factors do students who experienced both languages attribute the difference?

2. Materials and Methods

2.1. Context and Participants

The study took place in the undergraduate Communication Sciences programme during the 2025/26 academic year. Scratch is a compulsory first-year course delivered in two parallel sections under a shared syllabus, with students allocated to sections by an administrative criterion unrelated to their characteristics; approximately 400 students enrol each year across the two sections. The same instructor teaches one of the Scratch sections, regularly attended by about 60–70 students, and the optional second-year Python course, attended by at most 25–30 students drawn from both

EDIZIONI UNIVERSITARIE ROMANE
editoria scientifica dal 1952

Scratch sections. Python activities focused on tasks related to communication practice, such as text processing, content manipulation and editorial automation, and were carried out in Google Colab, where students could use an AI assistant to explain errors and concepts. The content and assessment of the Scratch course remained substantially unchanged across the two academic years considered.

2.2. Design

The design is cross-sectional and was conceived as the initial inquiry phase of a design-based research cycle (Anderson & Shattuck, 2012). Profile A comprises first-year students attending Scratch for the first time, who rated it at the end of the course, either shortly before or after taking the examination, providing a concurrent assessment of the block-based environment, made without experience of a later programming course. Profile B comprises second-year students who had passed Scratch and rated Python concurrently and Scratch retrospectively. Profile C comprises students taking Python without having passed Scratch, a group negatively selected with respect to Scratch and self-selected with respect to Python. Because the Scratch questionnaire was administered only in the section taught by the Python instructor, Profile A and the Profile B students from that section had the same Scratch instructor, who also taught Python. Two robustness checks (R1 and R2) were planned, addressing the role of the rating context (first research question) and of instructor continuity. R1 compares concurrent Scratch ratings (Profile A) with retrospective ones (Profile B); since the two groups belong to different cohorts, it is a diagnostic comparison rather than an estimate, within the same individuals, of how ratings change once students have experienced Python. R2 compares students who had attended Scratch (Profiles B and C) by Scratch section, examining whether the individual difference between Python and recalled Scratch is larger among those taught by the same instructor in both courses; given the subgroup sizes, R2 is also diagnostic.

2.3. Instruments

Two anonymous end-of-course questionnaires were administered through Moodle. All attitudinal items used a six-point agreement scale (1 = strongly disagree, 6 = strongly agree). Cognitive load items were adapted from the scale developed by Leppink et al. (2013): two items for intrinsic load, one for extraneous load (time and effort lost to the environment rather than to the content), and two for germane processing (effort perceived as well spent and as improving understanding of programming). Following Sweller (2010), the latter are interpreted as perceived learning benefit rather than as a distinct load. Task value was measured with three items on relevance to Communication Sciences, usefulness beyond the course and interest, drawing on expectancy-value theory (Eccles & Wigfield, 2020). Further items addressed engagement dimensions (Fredricks et al., 2004), including frustration, persistence and intention to abandon. At times I wanted to stop using it [the language] and give up. The questionnaires also included an ad hoc item on the effort of understanding what to do and an identity-fit item ("it seems designed for someone different from me"). The Python questionnaire repeated the load and value items for Scratch in retrospective form and added items on the comparison between the two

EDIZIONI UNIVERSITARIE ROMANE
editoria scientifica dal 1952

languages and on AI use. Open-ended responses, frustration, the ad hoc and identity-fit items and detailed AI-use items are not analysed in this paper.

2.4. Procedure and Ethics

Questionnaires were completed at the end of each course. Responses were anonymous. A final item asked whether respondents agreed to the use of their anonymous answers for research purposes; only respondents who agreed were included.

2.5. Analysis

Given the small and unequal groups, analyses relied mainly on non-parametric tests and emphasised effect sizes. Between-group comparisons used Mann–Whitney tests with rank-biserial correlation (r_{rb}); within-person comparisons used Wilcoxon signed-rank tests, excluding zero differences, with matched-pairs rank-biserial correlation (r_{rb}); associations used Spearman's ϱ. To assess whether intrinsic load was equivalent across languages, two one-sided t-tests (Welch's test between groups, paired test within Profile B) were applied with a margin of ± 0.5 scale points, set after data collection (Lakens, 2017). All other tests are two-sided with $a = .05$. As the analysis is exploratory, p-values are not corrected for multiplicity; however, the main within-group and R1 results remain significant after Holm correction within each set of four constructs. Data quality was examined through internal consistency checks, straight-lining and adherence to conditional instructions (Meade & Craig, 2012). Two departures from the planned design are reported: the planned use of examination grades was abandoned; the comparison within Profile C could not be carried out as an inferential test because of its size. The anonymised dataset and analysis script are available from the authors on request.

3. Results

3.1. Sample, Reliability and Data Quality

The Scratch questionnaire received 50 responses and the Python questionnaire 20, corresponding to response rates of roughly 70–80% of attending students; 42 and 17 respondents, respectively, consented to research use. Among consenting Scratch respondents, 26 were first-year students attending the course for the first time (Profile A), 8 had attended it in a previous year and had not yet passed the examination (re-attending students), and 8 selected an unspecified 'other situation' and were not assigned to a profile. When all consenting Scratch respondents were included in R1 ($n = 42$), differences were slightly smaller but remained large and significant. Among consenting Python respondents, 12 had passed Scratch (Profile B), 3 had not passed it or were re-attending it alongside Python (Profile C), and 2 had never attended Scratch; the latter's retrospective Scratch ratings were discarded. Reliability was adequate for task value ($a = .80$–$.92$ across the three rating contexts) and for perceived learning benefit (Spearman–Brown coefficients .79–.98). Intrinsic load was reliable for Scratch (.87–.93) but not for Python (.54), probably because the second Python item explicitly mentioned syntax and error handling. Responses were internally consistent (persistence and intention to abandon: $\varrho = -.55$ in both questionnaires). One straight-lining respondent belonged to the re-attending group, reported descriptively only, and three respondents who answered a conditional question without meeting its condition were excluded from that analysis.

EDIZIONI UNIVERSITARIE ROMANE
editoria scientifica dal 1952

3.2. Concurrent and Retrospective Comparisons

Table 1 reports the main comparisons. The clearest result concerns students who had experienced both languages. Compared with Scratch, Profile B rated Python higher in task value (11 of 12 students higher, none lower; r_{rb} = 1.00) and in perceived learning benefit (8 higher, 1 lower; r_{rb} = .93), and lower in extraneous load, although the latter difference did not reach significance (p = .078). The pattern held when the three Profile C students were included (task value: 12 of 15 higher, none lower, p = .002).

Table 1. Mean ratings (*SD*) and comparisons of Scratch and Python. Scale 1–6.

Construct	A: Scratch	B: Python	B: Scratch (recalled)	A vs B Python, r_{rb}	B: Python vs recalled, r_{rb} (+/−)	R1: recalled vs A, r_{rb}
Intrinsic load	3.31 (1.20)	3.83 (1.21)	3.46 (1.27)	−.27	.36 (6/3)	.09
Extraneous load	2.73 (1.25)	3.25 (1.54)	4.33 (1.15)	−.21	−.69 (2/7)	.65**
Perceived learning benefit	4.56 (1.14)	4.17 (0.81)	3.00 (1.33)	.30	.93* (8/1)	−.70**
Task value	4.17 (1.19)	3.83 (1.17)	2.69 (1.32)	.18	1.00** (11/0)	−.59**
n	26	12	12			

Note. A = first-year students rating Scratch concurrently; B = second-year students who passed Scratch, rating Python concurrently and Scratch retrospectively. R1 = comparison of Profile B's recalled Scratch ratings with Profile A's concurrent ones. r_{rb} = rank-biserial correlation (positive values: higher in the first-named rating). Counts show students rating Python higher/lower than recalled Scratch. * p < .05, ** p < .01.

Intrinsic load did not differ, but equivalence within ±0.5 points could not be established either within Profile B (90% CI of the difference −0.41 to 1.16) or between groups (−0.20 to 1.25). When each language was rated without a point of comparison, the picture was different: first-year students rated Scratch as favourably as Profile B rated Python, and no construct differed significantly. Restricting Profile B to the seven students who had attended the Scratch section taught by the Python instructor, so that the instructor was the same for both courses, the differences from Profile A's Scratch ratings were close to zero (extraneous load −0.02, task value −0.12 scale points).

3.3. Robustness Checks

R1 showed that the advantage of Python was matched by a gap between concurrent and comparative ratings of Scratch. Profile B judged Scratch to have had higher extraneous load, lower perceived learning benefit and lower task value than current first-year students reported (Table 1). The gap persisted, and for extraneous load widened (+2.27 points, p = .001, r_{rb} = .84), when Profile B was restricted to students taught by the same instructor in the same course one year earlier.

R2 gave a mixed result. Including all students who had attended Scratch, the gain in task value from recalled Scratch to Python appeared in both sections (7 of 9 students higher and none lower in the section taught by the Python instructor; 5 of 6 higher and none lower in the other section). The reduction in extraneous load, by contrast, appeared only among students who had the same instructor in both courses (−1.78 points, 7 of 9 lower, none higher, p = .016), and not in the other section (+0.33). In that section, taught by the Python instructor, recalled Scratch was rated as

more demanding (4.89 vs 3.83) and Python as less demanding (3.11 vs 4.17) than in the other section.

3.4. Task Value and Intention to Abandon

Task value was negatively associated with intention to abandon in Scratch, both in Profile A ($\varrho = -.50, p = .009$) and across all Scratch respondents ($\varrho = -.54, p < .001$, n = 42), whereas the association in Python was weak and non-significant ($\varrho = -.19$, $n = 17$). Intention to abandon was not lower for Python in Profile B (M = 4.00) than for Scratch in Profile A ($M = 3.69$). Among students rating Scratch concurrently, re-attending students reported the highest intrinsic and extraneous load ($M = 4.44$ and 3.75) and the lowest task value ($M = 3.50$).

3.5. Attributions

Asked which language had been harder, Python respondents split almost evenly: 9 indicated Scratch and 8 Python. Among the 9 who found Python more manageable, 4 attributed this mainly to the authenticity of Python tasks, 4 to AI support and 1 to no particular factor; 14 of the 17 respondents reported using the AI assistant at least rarely. Two of the three Profile C students rated both languages unfavourably, suggesting generalised rather than Scratch-specific difficulty.

4. Discussion

The classroom impression that motivated this study was confirmed among the students in a position to make the comparison. Those who had experienced both languages attributed more value to Python almost unanimously, independently of their Scratch instructor. First-year students, who knew only Scratch, judged it favourably. The "easy" tool thus proved to be the hard one for two groups: students re-attending the course, who reported higher load and lower value than first-year students, and students who, having moved on to Python, looked back on Scratch with a point of comparison.

The gap between first-year and comparative ratings of Scratch admits several explanations. The interpretation most consistent with the study's framework is evaluability: the relevance of a block-based environment to communication practice may become assessable only once students know an alternative, by analogy with the finding that some attributes are judged differently in joint rather than separate evaluation (Hsee et al., 1999). On this reading, comparative judgments are not a distortion but the informed ones. Retrospective ratings may be shaped by memory, although known memory biases point, if anything, in the opposite direction: recollections of past experiences tend to be more positive than the experience itself (Mitchell et al., 1997), which makes a more negative recollection of Scratch harder to attribute to memory alone; concurrent ratings may have been inflated by social desirability, as first-year students evaluated their current course, some before the examination (Podsakoff et al., 2003); and cohort or selection effects cannot be excluded. The stability of the Scratch course and the persistence of the gap within the same instructor's section make a change in the course itself unlikely. In any case, the gap carries a methodological implication: concurrent and comparative judgments of programming environments capture different things and should be reported separately.

The results also qualify the authenticity account. Authenticity did not appear to lower cognitive load: Python was perceived as intrinsically at least as demanding, and

intention to abandon was not lower. What distinguished Python was the value students attributed to the effort it required. In expectancy-value terms, this effort represents a cost, which for these students appears to have been outweighed by the usefulness and relevance of the tasks (Eccles & Wigfield, 2020). Students' own attributions, however, gave AI support the same weight as task authenticity, and the present design cannot separate the two.

The gain in task value was the most consistent result, present in almost every student and in both Scratch sections, and therefore not attributable to instructor continuity. This pattern is consistent with the principle of contextualized computing education (Guzdial, 2010): when students recognise the purpose of a programming task, they may accept its difficulty as worthwhile.

The association between task value and intention to abandon was clear in Scratch and weak in Python. Possibly, in an optional course chosen by students who already value programming, value no longer discriminates between those who persist and those who do not.

Several limitations apply. Groups are small, reflecting a single round of data collection, the limited enrolment of an optional course and the restriction of the Scratch questionnaire to one section; they are drawn from a single institution, and Python students are self-selected. The concurrent and retrospective ratings come from different cohorts, so R1 cannot estimate how ratings change within individuals. The Scratch questionnaire covered only one section, and some items require revision, clearer wording for the purpose-related item and a question on whether first-year respondents had already taken the examination. Task authenticity was not measured directly but inferred from task value and students' attributions.

5. Conclusions

For Communication Sciences students in this study, the advantage of text-based over block-based programming became visible once students could compare the two languages: those who had experienced both languages valued Python more, almost without exception. What Python offered was not a lighter cognitive load but, plausibly, a clearer sense of why the effort was worth making. For instructional design, task authenticity emerges as a candidate design principle. A companion analysis of the same data will draw on students' open-ended responses and on the items concerning the effort of understanding what to do, identity fit and AI use, to examine how students made sense of that effort and to disentangle task authenticity from AI support. The next cycle of this research will examine whether framing block-based activities around communication-related tasks increases their perceived value, while revising the instruments and collecting concurrent ratings from both sections after the examination.

References

Anderson, T., & Shattuck, J. (2012). Design-based research: A decade of progress in education research? Educational Researcher, 41(1), 16–25. https://doi.org/10.3102/0013189X11428813

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. Communications of the ACM, 60(6), 72–80. https://doi.org/10.1145/3015455

Cooper, S., & Cunningham, S. (2010). Teaching computer science in context. ACM Inroads, 1(1), 5–8. https://doi.org/10.1145/1721933.1721934

de Jong, T. (2010). Cognitive load theory, educational research, and instructional design: Some food for thought. Instructional Science, 38(2), 105–134. https://doi.org/10.1007/s11251-009-9110-0

Eccles, J. S., & Wigfield, A. (2020). From expectancy-value theory to situated expectancy-value theory: A developmental, social cognitive, and sociocultural perspective on motivation. Contemporary Educational Psychology, 61, 101859. https://doi.org/10.1016/j.cedpsych.2020.101859

Forte, A., & Guzdial, M. (2005). Motivation and nonmajors in computer science: Identifying discrete audiences for introductory courses. IEEE Transactions on Education, 48(2), 248–253. https://doi.org/10.1109/TE.2004.842924

Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. Review of Educational Research, 74(1), 59–109. https://doi.org/10.3102/00346543074001059

Guzdial, M. (2010). Does contextualized computing education help? ACM Inroads, 1(4), 4–6. https://doi.org/10.1145/1869746.1869747

Hsee, C. K. (1996). The evaluability hypothesis: An explanation for preference reversals between joint and separate evaluations of alternatives. Organizational Behavior and Human Decision Processes, 67(3), 247–257. https://doi.org/10.1006/obhd.1996.0077

Hsee, C. K., Loewenstein, G. F., Blount, S., & Bazerman, M. H. (1999). Preference reversals between joint and separate evaluations of options: A review and theoretical analysis. Psychological Bulletin, 125(5), 576–590. https://doi.org/10.1037/0033-2909.125.5.576

Kalyuga, S. (2011). Cognitive load theory: How many types of load does it really need? Educational Psychology Review, 23(1), 1–19. https://doi.org/10.1007/s10648-010-9150-7

Lakens, D. (2017). Equivalence tests: A practical primer for t tests, correlations, and meta-analyses. Social Psychological and Personality Science, 8(4), 355–362. https://doi.org/10.1177/1948550617697177

Leppink, J., Paas, F., Van der Vleuten, C. P. M., Van Gog, T., & Van Merriënboer, J. J. G. (2013). Development of an instrument for measuring different types of cognitive load. Behavior Research Methods, 45(4), 1058–1072. https://doi.org/10.3758/s13428-013-0334-1

Meade, A. W., & Craig, S. B. (2012). Identifying careless responses in survey data. Psychological Methods, 17(3), 437–455. https://doi.org/10.1037/a0028085

Mitchell, T. R., Thompson, L., Peterson, E., & Cronk, R. (1997). Temporal adjustments in the evaluation of events: The "rosy view". Journal of Experimental Social Psychology, 33(4), 421–448. https://doi.org/10.1006/jesp.1997.1333

Podsakoff, P. M., MacKenzie, S. B., Lee, J.-Y., & Podsakoff, N. P. (2003). Common method biases in behavioral research: A critical review of the literature and recommended remedies. Journal of Applied Psychology, 88(5), 879–903. https://doi.org/10.1037/0021-9010.88.5.879

Strong, G., Bresnihan, N., & Tangney, B. (2025). Supporting learners in the transition from block-based to text-based programming: A systematic review. Journal of Computer Languages, 84, Article 101342. https://doi.org/10.1016/j.cola.2025.101342

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. Educational Psychology Review, 22(2), 123–138. https://doi.org/10.1007/s10648-010-9128-5

Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. Educational Psychology Review, 31(2), 261–292. https://doi.org/10.1007/s10648-019-09465-5

Weintrop, D., & Wilensky, U. (2015, June). To block or not to block, that is the question: students' perceptions of blocks-based programming. In Proceedings of the 14th international conference on interaction design and children (pp. 199-208). https://doi.org/10.1145/2771839.2771860

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. ACM Transactions on Computing Education, 18(1), Article 3. https://doi.org/10.1145/3089799