

STRENGTHENING COMPUTATIONAL THINKING THROUGH PYTHON TRAINING WITH GOOGLE COLAB

Taslim¹, Susi Handayani², Syahriatna³

^{1,2,3} Faculty of Computer Science, Lancang Kuning University, Pekanbaru, Indonesia
email taslim@unilak.ac.id, susi@unilak.ac.id, syahriatna@unilak.ac.id

Abstract: This community service program was conducted to address students' limited initial exposure to basic programming concepts at SMK Migas Inovasi Riau. A total of 20 students participated in introductory Python training implemented through Google Colab accessed via Android smartphones, enabling the learning process to proceed despite limited device availability and without requiring software installation. The training activities consisted of brief conceptual explanations, live coding demonstrations, guided practice, and assignments. Program effectiveness was evaluated using pre-test and post-test questionnaires. The results indicated an improvement in students' understanding of key topics, particularly basic programming concepts and control structures, as reflected in post-test outcomes that were predominantly categorized as "Very Good" and "Good," with an overall improvement of approximately 70%. These findings suggest that cloud-based, lightweight learning tools can effectively support introductory programming instruction in vocational school contexts.

Keywords: Python; Google Colab; SMK Student; Programming Training; Digital Literacy

1. Introduction

The era of digital transformation requires not only mastery of technological tools but also fundamental competencies in designing structured work procedures, managing information, and implementing simple automation. In the field of informatics education, this need is aligned with the computational thinking (CT) approach, which emphasizes problem solving, system design, and the use of core computer science concepts to formulate solutions that can be executed computationally (Li et al., 2020; Angeli and Giannakos, 2020). A number of studies also show that strengthening CT among students can be effectively facilitated through programming activities that are designed in an applied and gradual manner (Wan et al., 2024; Liu et al., 2024).

However, programming learning at the beginner level is often faced with significant pedagogical challenges. Students tend to become trapped in syntactic complexity, making it difficult for them to maintain logical flow when solving more complex problems (Lappi, Tirronen, and Itkonen, 2023; Chuang and Chang, 2024). On the other hand, the success of novice learners is strongly influenced by the quality and timeliness of feedback during the problem-solving and debugging process (Marwan et al., 2022; Rocha, Tedesco, and Costa, 2023). Systematic literature reviews emphasize the critical role of formative feedback in helping students identify errors, reflect on their thinking processes, and maintain learning motivation (Morris, Perry, and Wardle, 2021; Fisher et al., 2025).

In response to these challenges, this community service activity was conducted at SMK Migas Inovasi Riau. This activity aimed to strengthen students' basic programming competencies through Python-based training that emphasized hands-on practice and intensive mentoring. The initial mapping through the pre-test showed that participants' understanding of core programming concepts was still low, particularly in mastering control structures, namely branching and looping, as well as the ability to translate problems into programming solutions. This finding indicates that participants needed not merely an introduction to a programming language, but also the development of a solid programming

mental model, including the ability to construct solution steps, convert them into code, and correct errors through the debugging process (Heinonen et al., 2023; Eckerdal et al., 2024).

The selection of Python as the training medium was based on its accessibility for beginners and the availability of extensive documentation and learning resources. From the perspective of instructional design, the materials focused on the minimum competencies that determine beginner success, namely data representation through variables and data types, execution flow through input and output, and control structures such as if/else and loops. These competencies were then applied through gradual exercises and simple case studies to effectively manage participants' cognitive load and ensure an adequate foundation before they constructed more complex programming solutions (Ma et al., 2023; Shin et al., 2023).

This article contributes by presenting an evaluation of the training impact through a comparative analysis of pre-test and post-test results, accompanied by a critical discussion of competency aspects that still showed weaknesses, particularly the ability to integrate several concepts to develop a complete case study solution. In addition to reporting an increase in participants' conceptual understanding of approximately 70%, this article emphasizes the implications for the design of similar training activities: conceptual achievement needs to be followed by tiered exercises and continuous mentoring so that problem-solving competence can develop into independent learning.

Based on these problems, this community service activity aimed to strengthen the basic programming and computational thinking skills of students at SMK Migas Inovasi Riau through Python training based on Google Colab. Specifically, this activity was directed toward improving participants' understanding of basic programming concepts, the use of control structures, and the ability to solve simple exercises gradually through hands-on practice and intensive mentoring. In addition, this article aims to describe the implementation of the training, the materials delivered, and the impact of the activity based on pre-test and post-test results as a basis for formulating implications for similar training in the context of vocational education.

2. Method

The pre-test instrument used a dichotomous scale, namely Yes/No, to capture participants' initial experience with programming topics. Meanwhile, the post-test instrument used the categories "Very Good" and "Good" to assess participants' level of understanding of the core indicators. Both instruments were administered to the same participants, allowing changes in understanding to be observed directly.

The improvement analysis was conducted descriptively by comparing the initial condition shown in the pre-test and the final achievement shown in the post-test on the main training indicators. The increase in understanding was reported in the range of approximately 70%, based on the dominant shift from low initial exposure in the pre-test to achievement in the "Very Good" and "Good" categories in the post-test.

During the implementation, the facilitators applied the "example–practice–feedback" strategy for each topic and provided debugging guidance for common errors, such as indentation errors, variable writing errors, and branching or looping logic errors.

The implementation method combined short lectures, demonstrations, guided practice, discussion, question-and-answer sessions, and assignments. The training was organized using a "one concept–one exercise" pattern to ensure that participants understood the relationship between concepts and their implementation. All practices were conducted

through Google Colab, accessed using Android devices, so that participants could focus on programming logic without software installation constraints.

The stages of the activity included: (a) delivery of basic materials, including program logic, variables and data types, input/output, branching, and looping; (b) initial evaluation through a pre-test questionnaire; (c) core practice-based training and simple case studies with mentoring; (d) reinforcement discussion and discussion of obstacles; and (e) final evaluation using a post-test questionnaire.

The subjects of the activity were 20 students. The evaluation indicators included initial exposure to Python, experience in writing code, understanding of basic concepts, understanding of control structures, use of supporting tools or IDEs, and application of programming for problem solving.



Figure 1. Flow of the Community Service Activity Implementation

3. Results and Discussion

This community service activity was carried out through basic Python programming training based on Google Colab for students of SMK Migas Inovasi Riau. The training was designed to gradually introduce basic programming concepts to participants, most of whom had no prior experience in writing code. The materials delivered included an introduction to Google Colab as a practice medium, basic Python concepts, variables and data types, input/output, basic operations, branching using if/else, and looping. Each material was delivered through brief explanations, live coding demonstrations, guided practice, and simple exercises so that participants could understand the relationship between concepts and program implementation.

In addition to concept delivery, the training also emphasized hands-on practice. Participants were trained to run simple code using Google Colab through Android devices, write programs to receive user input, display output, and construct basic logic using branching and looping. One of the learning outputs produced during the activity was the participants' ability to create simple programs, such as determining odd or even numbers, displaying number sequences, and developing input-process-output-based programs. These outputs

indicate that the training not only improved conceptual understanding but also provided initial experience in applying computational logic into simple program forms.

The activity evaluation was conducted by comparing the results of the pre-test and post-test questionnaires. The pre-test was used to describe participants' initial condition before the training, while the post-test was used to assess participants' level of understanding after completing the entire series of activities. The quantitative evaluation results were then strengthened by activity documentation and observations of participants' involvement during the practice process.

Table 1. Pre-Training Recapitulation (Pre-Test)

No	Question Item	Yes	No
1	Has the participant ever heard of the Python programming language?	6	14
2	Has the participant ever written simple program code in any programming language?	4	16
3	Does the participant understand the concepts of variables and data types?	3	17
4	Does the participant understand branching using if/else and looping?	2	18
5	Has the participant ever used a programming support tool, IDE, or practice platform?	5	15
6	Has the participant ever applied programming to solve simple problems?	2	18

The results in Table 1 show that participants' initial abilities were still low, particularly in control structures and the application of programming for problem solving. This finding reinforces the need for intensive mentoring in basic exercises and the habituation of debugging practices.

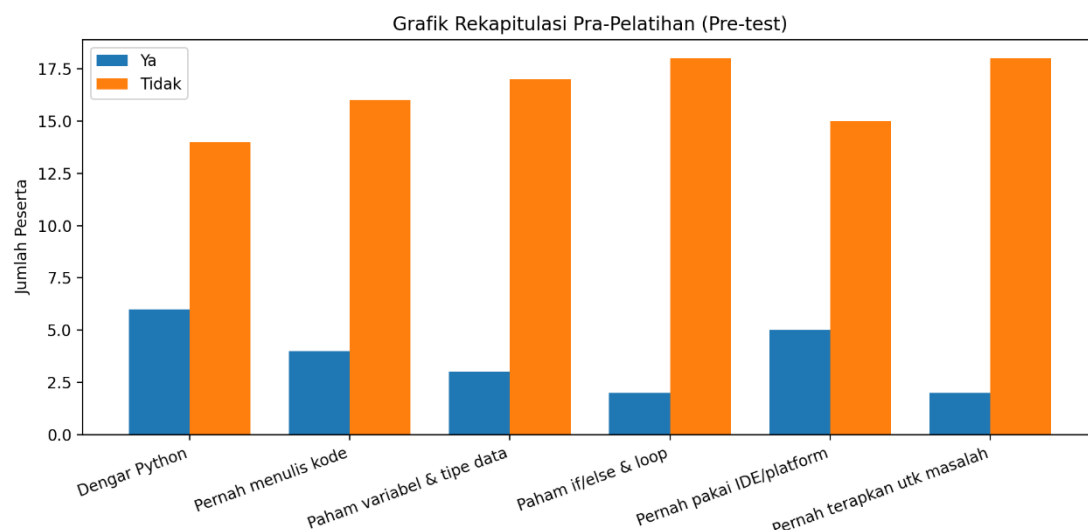


Figure 2. Pre-Training Recapitulation Chart (Pre-Test)

Table 2. Post-Training Recapitulation (Post-Test)

No	Question Item	Very Good	Good
1	Understanding of basic Python concepts, including variables, input/output, and basic operations	15	5
2	Understanding of control structures, including if/else and loops	14	6
3	Understanding of the function of coding in the IT field and its applications	13	7
4	Ability to complete exercises or case studies with guidance	12	6
5	Ease of understanding the material by participants	12	7

Based on Table 2, the post-training achievements were dominated by the “Very Good” and “Good” categories in the core indicators. Compared to the initial condition, the average increase in understanding was approximately $\pm 70\%$. This improvement indicates the effectiveness of the hands-on practice strategy supported by mentoring.

Nevertheless, the indicator related to completing case studies still requires further strengthening. In general, participants already understood the components of programming, such as variables, if statements, and loops, but they were not yet fully stable in integrating these components into a complete solution without guidance. Therefore, follow-up activities are recommended in the form of tiered exercises and contextual mini-projects.

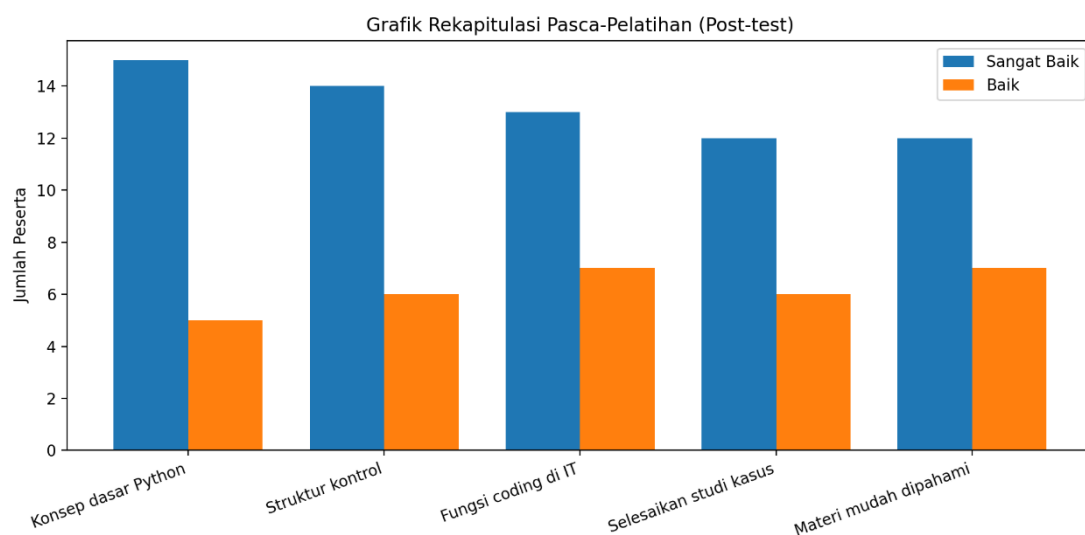


Figure 3. Post-Training Recapitulation Chart (Post-Test)

Activity Documentation



Figure 4. Training Documentation (Photo 1)



Figure 5. Training Documentation (Photo 2)

4. Conclusion

The community service activity through basic Python programming training for students of SMK Migas Inovasi Riau had a positive impact on strengthening digital literacy and computational thinking skills. The evaluation results showed a clear shift from the initial condition in the pre-test, which was still low, to higher post-test results, with the majority of participants falling into the "Very Good" and "Good" categories. Descriptively, participants' understanding increased by approximately $\pm 70\%$. This finding indicates that

practice-based training is effective in strengthening initial programming competencies; however, further reinforcement is still needed, particularly to improve participants' ability to integrate concepts into case study solutions more independently.

Acknowledgment

The authors would like to express their gratitude to the Faculty of Computer Science, Universitas Lancang Kuning, for supporting the implementation of this activity, including facility and funding support, so that this Community Service program could be carried out successfully.

References

- Angeli, Charoula, and Michail Giannakos. 2020. "Computational Thinking Education: Issues and Challenges." *Computers in Human Behavior* 105:106185. doi: <https://doi.org/10.1016/j.chb.2019.106185>.
- Chuang, Yung-Ting, and Hsin-Yu Chang. 2024. "Analyzing Novice and Competent Programmers' Problem-Solving Behaviors Using an Automated Evaluation System." *Science of Computer Programming* 237:103138. doi: <https://doi.org/10.1016/j.scico.2024.103138>.
- Eckerdal, Anna, Anders Berglund, Michael Thuné, Anders Berglund, and Michael Thuné. 2024. "Learning Programming Practice and Programming Theory in the Computer Laboratory Computer Laboratory." *Information and Software Technology* 3797. doi: 10.1080/03043797.2023.2294953.
- Fisher, Darren Paul, Gaelle Brotto, Iris Lim, and Colette Southam. 2025. "The Impact of Timely Formative Feedback on University Student Motivation." *Assessment & Evaluation in Higher Education* 50(4):622–31. doi: 10.1080/02602938.2025.2449891.
- Heinonen, Ava, Bettina Lehtelä, Arto Hellas, and Fabian Fagerholm. 2023. "Synthesizing Research on Programmers' Mental Models of Programs, Tasks and Concepts — A Systematic Literature Review." *Information and Software Technology* 164:107300. doi: <https://doi.org/10.1016/j.infsof.2023.107300>.
- Lappi, Vilma, Ville Tirronen, and Jonne Itkonen. 2023. "A Replication Study on the Intuitiveness of Programming Language Syntax." *Software Quality Journal* 31(4):1211–40. doi: 10.1007/s11219-023-09631-7.
- Li, Yeping, Alan H. Schoenfeld, Andrea A. diSessa, Arthur C. Graesser, Lisa C. Benson, Lyn D. English, and Richard A. Duschl. 2020. "Computational Thinking Is More about Thinking than Computing." *Journal for STEM Education Research* 3(1):1–18. doi: 10.1007/s41979-020-00030-2.
- Liu, Zhichun, Zarina Gearty, Eleanor Richard, Chandra Hawley Orrill, Shakhnoza Kayumova, and Ramprasad Balasubramanian. 2024. "Bringing Computational Thinking into Classrooms: A Systematic Review on Supporting Teachers in Integrating Computational Thinking into K-12 Classrooms." *International Journal of STEM Education* 11(1):51. doi: 10.1186/s40594-024-00510-6.
- Ma, Ning, Jinglong Qian, Kaixin Gong, and Yao Lu. 2023. "Promoting Programming Education of Novice Programmers in Elementary Schools: A Contrasting Cases Approach for Learning Programming." *Education and Information Technologies* 28(7):9211–34. doi: 10.1007/s10639-022-11565-9.
- Marwan, Samiha, Bitak Akram, Tiffany Barnes, and Thomas W. Price. 2022. "Adaptive Immediate Feedback for Block-Based Programming: Design and Evaluation." *IEEE Transactions on Learning Technologies* 15(3):406–20. doi: 10.1109/TLT.2022.3180984.
- Morris, Rebecca, Thomas Perry, and Lindsey Wardle. 2021. "Formative Assessment and Feedback for Learning in Higher Education: A Systematic Review." *Review of Education*

9(3):e3292–e3292. doi: 10.1002/rev3.3292.

Rocha, Hemilis Joyse Barbosa, Patricia Cabral de Azevedo Restelli Tedesco, and Evandro de Barros Costa. 2023. "On the Use of Feedback in Learning Computer Programming by Novices: A Systematic Literature Mapping." *Informatics in Education* 22(2):209–32. doi: 10.15388/infedu.2023.09.

Shin, Yoonhee, Jaewon Jung, Joerg Zumbach, and Eunseon Yi. 2023. "The Effects of Worked-Out Example and Metacognitive Scaffolding on Problem-Solving Programming." *Journal of Educational Computing Research* 61(6):1312–31. doi: 10.1177/07356331231174454.

Wan, Haipeng, Xue Zhang, Xinxue Yang, and Shan Li. 2024. "Which Approach Is Effective: Comparing Problematization-Oriented and Structuring-Oriented Scaffolding in Instructional Videos for Programming Education." *Education and Information Technologies* 29(14):17807–23. doi: 10.1007/s10639-024-12550-0.