

The Development of Module for Improving Computational Thinking Skills on Science Education based on Google Colaboratory

Silvi Putri Ayu Ningsih^{1*}, Zainur Rasyid Ridlo¹

¹Department of Science Education-University of Jember, Jember, Indonesia

[*silviputriayuningsih@gmail.com](mailto:silviputriayuningsih@gmail.com)

Abstract

The purpose of this study was to analyze the module development process to improve skills for students in junior high school and then conduct a limited test on prospective teacher students. The development method used in this research is the ADDIE method by using analyzing the results of validation and limited testing using samples from prospective science teacher students, the purpose of using prospective teacher students as research samples because this module was developed for junior high school students, therefore prospective teacher students must understand the module developed before it is implemented to junior high school students. The results of module development resulted in a content validity value of 88.1%, science concepts 87.8%, Construct 94.7, Language 89.3%, layout 88.9%, on average the validity value of the module that has been developed reaches 89.7%, this indicates that the module developed is categorized as very valid and ready to use. Implementation of the module in a limited class resulted in an increase in the average value of posttest from 27.9 to 87.5 and resulted in an N-gain value of 0.82. based on these results it can be concluded that the developed module is able to direct and improve the ability to think computationally.

Keywords : Computational thinking, Science education, ADDIE, Junior High School, Google Colaboratory.

Background

In the 21st century, learners must develop computational thinking skill to succeed in an increasingly digital world (Pérez-Marín et al., 2020). Computational thinking is a process of problem-solving with computational concepts, enabling students to gain deeper knowledge and expertise in this field (Angeli & Giannakos, 2020). Computational thinking skill goes beyond coding and programming and is related to the students' higher-order thinking skills, these skills involve understanding how computers work, problem modelling, data analysis, and solving problems, as well as abstract and logical information (Ridlo et al., 2024).

In the context of the accelerated evolution of the 21st-century, the integration of technology into the educational sphere is of greater consequence than ever before. Among these skills, computational thinking is particularly noteworthy as a foundational competency that fosters digitalization and prepares students for future challenges. The computational thinking skill plays an instrumental role in 21st-century science education, providing a powerful foundation for students to cultivate profound scientific comprehension and sophisticated problem-solving abilities. Computational thinking represents an effective means of developing 21st-century skills (Lye and Koh, 2014; Agbo et al., 2019). In 1980, Seymour Papert first employed the term "computational thinking" and proposed that computers might facilitate cognitive processes and alter patterns of knowledge accessibility (Papert, 1980). Further years, Papert (1996) additionally underscored the necessity of providing all children with access to computers as a means of influencing their learning and facilitating the expression of their ideas. Subsequently, Jeannette Wing (2006) posits that computational thinking is a universally applicable attitude and skill set that all individuals would be eager to learn and utilize. Wing asserts that computational thinking should be integrated into the curriculum alongside reading, writing, and arithmetic, as part of a comprehensive approach to cognitive development.

Computational thinking is a skill that is typically associated with professionals in fields such as computer science, engineering, mathematics and others who do programming. However, recent research suggests that programming is a valuable skill that can benefit students from a wide range of academic disciplines (Lockwood & Mooney, 2017; Castro et al., 2021). Recent research highlights the importance of teaching basic computer programming to junior high school students for science education. Studies show that Python training can effectively improve students' understanding of programming concepts and computational skills (S. Supian et al., 2023). In the contemporary era, the expectation is that individuals across the lifespan will possess computational thinking abilities, in conjunction with the ongoing technological advancements (Kalelioglu et al., 2016). The ability to think computationally is a crucial skill for students to acquire in order to adapt to the advent of artificial intelligence in the future (Sun et al., 2021). Computational thinking skill is identified as a crucial skill for science education, with specific stages like task decomposition, abstraction, and generalization optimizing computational thinking construction for science students (Y. I. Tanjung et al., 2023). Overall, integrating programming into junior high school curricula enhances digital literacy and prepares students for future technological challenges. It is therefore imperative to integrate computational thinking into the primary and secondary school curriculum as a means of preparing students to compete in a global and digital environment (Lyon & Magana, 2020).

A new curriculum focusing on developing computational thinking has been introduced in developed countries for grades 7-12 (Iris Zur Bargury, 2012). To teach computational thinking in the classroom there needs to be consideration about what pedagogical approach and learning tools. While computational thinking can be applied to a wide range of problems that may not involve programming tasks directly (Román-González et al., 2022). Furthermore, it is convinced that computer programming is the most effective way to develop computational thinking abilities (Lye and Koh, 2014). Students must be guided to connect between concepts and activities from students by creating computer programming (Ridlo et al., 2022). Some educational software tools like Scratch, Logo, and ToolKid have proven useful in making programming more attractive and accessible to young learners (Metodija Jancheski, 2017). These tools offer features such as immediate feedback, visual programming, and object-oriented concepts. But, it has some limitations because these software only focused on block based programming.

For that reason, there is a need for technology integrated as learning media to improve computational thinking skill, one of the learning media to improve Computational thinking skill that is integrated with programming and the internet, namely Google Colaboratory. Google Colaboratory was chosen as a learning integration because students do not need to install any programs and can be easily used to open and edit programming to solve problems (Duda et al., 2021). The ease of accessibility can be utilized by teachers to improve students' Computational thinking skill at the junior high school level in science learning.

Nevertheless, there is a need for a learning model to teach and guide students about Google Colaboratory and improve computational thinking skill, hence using a guided inquiry teaching model is appropriate to use. Regarding the structuring of learning processes and the enrichment of such processes with computational media, inquiry learning in science education and computational education are closely related (Hoppe & Manske, 2022). Also, It is essential to provide students with effective learning materials that can guide them through activities and the learning process, particularly learning modules. In light of this, this study aims to develop a learning module integrated with Google Colaboratory to improve junior high school students' computational thinking skill in science education.

Research Method

2.1 Research Design

This research was conducted by Research and Development (R&D) with the ADDIE development model (Branch, 2009). ADDIE model involves Analyze, Design, Development, Implement, and Evaluate. The stages of the research method are explained in Table 1.

Table 1. The ADDIE research design

Stage	Description
Analysis	At this stage, information is collected about learning needs, learning objectives to be achieved, and students' characteristics.
Design	This stage, the module will be designed based on the results of the data analysis
Development	The development of teaching materials is based on the design prepared at the preceding stage.
Implement	The module was implemented with students to improve computational thinking skill.
Evaluate	The evaluation constitutes a form of reflection on the research conducted

At the stage of development, there is a validity test to obtain validation of the module developed with the suitability of the material. Validators at the validation stage involved lecturers and science teachers at junior high school. Validation results can be measured by the following formula. The calculation results of the validity are aligned to the criteria that can be observed in Table 2.

$$V = \frac{T_{sv}}{T_{sm}} \times 100\%$$

Description : V : validation percentage ; Tsv : validation result ; Tsm : maximum result.

Table 2. Criteria of Validity Score

Range score (%)	Criteria
85 - 100	Strongly Valid
69 - 84	Valid
53 - 68	Netral
37 - 52	Invalid
20 - 36	Strongly Invalid

(Nyutu *et al.*, 2020).

At the stage of implementation, the module was implemented with students to improve computational thinking skills. Pretest and posttest are used to measure students computational thinking skill, the results of pretest and posttest will be measured using N-gain score with following formula below. N-gain score will be aligned to the criteria that can be observed in Table 3.

$$N\text{-gain} = \frac{X_{\text{post}} - X_{\text{pre}}}{X_{\text{max}} - X_{\text{pre}}}$$

Description : N-gain = Average gain score ; Xpre = Average posttest score ; Xpost = Average pretest score ; Xmax = Maximum value

Table 3. Criteria of N-gain Score

Range Score	Criteria
$0,7 < \text{gain} \leq 1$	High
$0,3 < \text{gain} \leq 0,7$	Moderate
$0 \leq \text{gain} \leq 0,3$	Low

(Hake, 1998).

2.2 Population and Sample

Participants are subjects who are involved in physical activities as part of the response to activities carried out in the teaching and learning process, thus supporting the achievement of learning objectives. In this study, a preliminary test was conducted on science education students, the validated module trials were conducted on preservice science teacher students who will teach in secondary high schools. Additionally, researchers engaged several participants or research subjects, namely expert validators, to validate the products developed. This validation stage was collected with a validation instrument.

2.3 Data Collection Tools

The preliminary tests in this research are pretest and posttest for preservice science teacher students. The pretest will be administered to students before they engage with the module, while the posttest will be administered to students after they have completed the module. The test given is an assessment question aligned with the Computational Thinking Skill indicators.

The instruments utilized in this research are instruments for expert validators to validate learning modules produced and tests for students to evaluate implementation of learning modules. The objective of these instruments is to ascertain the validity and feasibility of the learning modules. The instrument was administered to three expert validators, comprising two science education lecturers and one science teacher in junior high school. The aforementioned aspects serve as the criteria for evaluation by expert validators, namely content, science concept, construct, language, and layout.

2.4 Data Analysis

The data analysis tests for students employed pretest and posttest. The student of this research was conducted in the science education study programme with a total number of participants was 32 science teacher candidate students. These tests aimed to evaluate the use of the learning module before it is widely used and tested by junior high school students.

The data collection procedure employed in this research is that of product validity data. The data obtained from the validation instruments administered to expert validators were analyzed using a Likert scale 1-5 with a response description. This scale ranged from 5 (indicating that the predicate was "very good") to 1 (indicating that the predicate was "very poor"). After obtaining the data from the study, a descriptive data analysis was conducted.

The Paired t-test statistical method is used to compare the means of two measurements taken from the same group or individuals, under two different conditions or times. In this study, this method was used to analyze the pretest and posttest results of 32 student teachers, with the aim of determining whether the application of the module made a significant difference to their computational thinking skills.

The paired t-test begins by calculating the difference between the pretest and posttest scores for each individual, then the average of the differences is analyzed for improvement. In addition, the standard deviation of the difference was also calculated to determine the consistency of change between individuals. Based on the mean difference and variability, the t value is calculated and compared to the critical value to determine statistical significance. In hypothesis testing, if the t-value or p-value shows a significant difference, then the null hypothesis that there is no difference between the pretest and posttest can be rejected. Paired t-test is very effective in detecting significant differences by reducing the variability between individuals because the measurements are made on the same group. Thus, this method allow to statistically verify that the improvement in posttest results is indeed significant and not just a coincidence.

Result and Discussion

This section describes the validation results of the learning module that uses Google Colaboratory to improve students' computational thinking skills. The validated module trials were conducted on preservice science teacher students who will teach in secondary high schools. The five stages of module development follow the ADDIE development model which refers to the characteristics of the material and the characteristics of the students who will use this module. The science material used in this module is Physics material, namely regular straight motion, focusing on deepening the material on the analysis of the formulas used and the representation of answers using two ways of thinking, namely using manual solutions and computer programming. The manual solution uses the Polya system, the method of solving problems in science according to Polya's rules consists of four main steps that help students think systematically. The first step is to understand the problem, where students must identify known information, what is asked, and relevant scientific concepts. After understanding the problem, students move on to the second step, which is developing a plan. At this stage, they have to think of the right method, such as the use of formulas or experimental steps, to solve the problem. Once the plan is organised, students move on to the third step, which is executing the plan, where they apply the planned steps carefully and thoroughly to get an accurate solution. Finally, the fourth step is to re-examine the results that have been obtained. Students should evaluate whether their answer makes sense, whether it fits the question, and whether there is a more efficient method. In the context of assessment, Polya's method emphasises the importance of the scientific thinking process and not just the end result, so students are assessed based on how they follow each step of problem solving.

The first stage in developing this module is the analysis stage. The research analysis includes the analysis of the current curriculum used in general in junior high school. The second stage of analysis is the analysis of student characteristics and material characteristics, the character of student thinking that exists in junior high school is able to think abstractly, the character of the material used in this module is Physics material that uses many mathematical equations/formulas. The results of this analysis will be integrated into the module and indicators of computational thinking skills. After the analysis stage is completed, the second stage is to design the module based on the results of the data analysis obtained. The module design considers the layout of the module preparation, and the suitability of the module content with the indicators of computational thinking skills and the language used in the module. Module development as a teaching material is carried out based on the design that has been prepared at the previous stage. At this stage of development, development is carried

out in the field of content / material which is the most important part of the module, the development in question is a detailed explanation of the concept which includes the proof of the formula used.

In addition to the content of the material, the supporting part was also developed, namely the use of programming languages integrated with Google Colaboratory. After the module development stage is complete, the module will be tested for validity. The purpose of validity testing is to obtain a valid module that can train students' computational thinking skills. The validators used in the validation stage consisted of lecturers and science teachers at the research school.

Modules that have been validated and declared valid by validators are tested on a limited basis on 32 science teacher candidate students before being tested on students in schools. The implementation of electronic modules based on computer programs using Google Colaboratory on the material of Kinematics for junior high school students can be done through several stages. First, learning needs analysis was conducted by identifying materials, basic competencies in accordance with the curriculum, and determining learning objectives that focus on understanding the concept of Kinematics and motion simulation using computer programs. After that, the module design is carried out by compiling a material structure consisting of an introduction to Kinematics, the concept of acceleration, Kinematics equations, visualisation of motion graphs, and the application of Python in Google Colaboratory for simulation. In the development stage, the module begins with an introduction to Google Colaboratory and continues with the preparation of Python code to model the Kinematics equation, including visualisation of velocity-time and position-time graphs using the `numpy` and `matplotlib` libraries. Students can interact directly with the simulation, try to change parameters such as initial velocity and acceleration, and explore the effects on the motion of objects.

Once the module was developed, it was validated and trialled by involving a co-teacher and a small group of students to ensure its suitability and effectiveness. Feedback from students was collected and used to revise the module before its implementation in the classroom. At the implementation stage, students used their respective devices to access Google Colaboratory and run the simulation, with monitoring from the teacher. The module is completed with assignments and quizzes to evaluate students' understanding of the material and the simulations they create. Finally, learning evaluation is conducted through tests and analysis of learning outcomes to ensure that learning objectives are achieved and students have mastered the motion concept as well as the technology skills that support it.

At the evaluation stage in the development of ADDIE-based modules, the effectiveness of computer programming modules that have been implemented to improve the computational thinking skills of prospective science teachers is measured. The evaluation begins with collecting data on learning outcomes through pretests and posttests to measure changes in student abilities before and after using the module. Furthermore, N-gain calculation and analysis using SPSS using the paired sample t-test method were carried out to assess the effectiveness of the improvement, by comparing the difference in pretest and posttest scores. The data collected was then analysed to assess the overall effectiveness of the module and identify aspects that require improvement. Based on the results of the analysis at the evaluation stage, recommendations will be made for improving the module to make it more effective in improving students' computational thinking skills in the future.

The illustration of the Validation result of the learning module shows on Table 4 below.

Table 4. Validation Result of Modul

Aspect	Score
Content	88.1%
Science concept	87.8%
Construct	94,7 %
Language	89,3 %
Layout	88,9 %
Total	89.7 %

The content validity of the module assessed reached 88.1%. This shows that the module has good quality in terms of the suitability of the material presented with the learning objectives to be achieved. The validators assessed that the module material was in accordance with the expected competencies and relevant to the computational thinking skills to be developed. However, there is room for slight improvement in the aspect of material delivery to further support learning outcomes. In the aspect of science concepts, the module received a validity score of 87.8%. This means that the science concepts presented in the module are appropriate and in accordance with applicable scientific principles. This almost perfect validity indicates that the module is able to integrate science materials with computational-based learning effectively. Some suggestions for improvement may be given by the validators related to the presentation of simpler or more applicable concepts. This module has a construct validity of 94.7%, indicating that the structure or framework of the module is very strong and supports successful learning. The validator assessed that the flow and structure of the material presentation is very logical, systematic, and makes it easy for users to follow the learning stages. This score is one of the highest, indicating that the module preparation has been very well designed to facilitate the development of computational thinking skills.

The linguistic aspect of the module received a validity score of 89.3%. This indicates that the use of language in the module is good enough in terms of clarity, suitability to the audience, and appropriate use of terms. Validators may find some parts that could be simplified or explained in more detail to clarify the intent and purpose of the material to make it easier for users to understand. The layout aspect of the module scored 88.9%. This validity reflects that the layout of the module is quite attractive and makes it easier for users to access information. The validators considered that the arrangement of text, images, graphics, and other visual elements was well organised. However, there may be some suggestions related to improving aesthetics or readability to optimise the layout and support more effective learning. Overall, the module you developed demonstrates excellent quality with high validity scores in various aspects. Although there are some areas that could be further improved, these results indicate that the module has met the expected standards and is suitable for use in the learning process to train computational thinking skills.

The following is a complete description of the results of the module trials that have met the valid category. The module you developed was tested on 32 student teachers. This number of participants is representative enough to see the effectiveness of the module in training computational thinking skills. Student teachers are the right target since the module is designed to develop their ability to integrate programming and



computational thinking into learning. Before the application of the module, students underwent a pretest with an average score of 27.9. This pretest score indicates that prior to the use of the module, students' initial level of understanding of the material taught was still relatively low. This is reasonable, because the module is designed to improve computational thinking skills that may still not be well mastered by students before training. After the implementation of the module, students underwent a posttest, and the average result increased significantly to 87.5. This shows that the module you developed successfully improved students' understanding and skills significantly. This posttest value reflects the success of the module in delivering the material and training the skills that are the main objectives of module development. The N-gain value obtained is 0.82 which is included in the high category. N-gain is an indicator that measures the relative increase in learning outcomes, compared to the maximum score that can be achieved. This value of 0.82 indicates that there is a very significant increase in student ability after participating in learning by using the module.

This 'high' category indicates that this module is very effective in improving students' computational thinking skills. Overall, the pilot test data shows that the module that has been developed and validated is able to significantly improve the computational thinking skills of student teachers. The module is highly effective in providing understanding and skills relevant to the learning objectives, with measurable improvement through high N-gain. The illustration of pretest and posttest show in Image 1 below

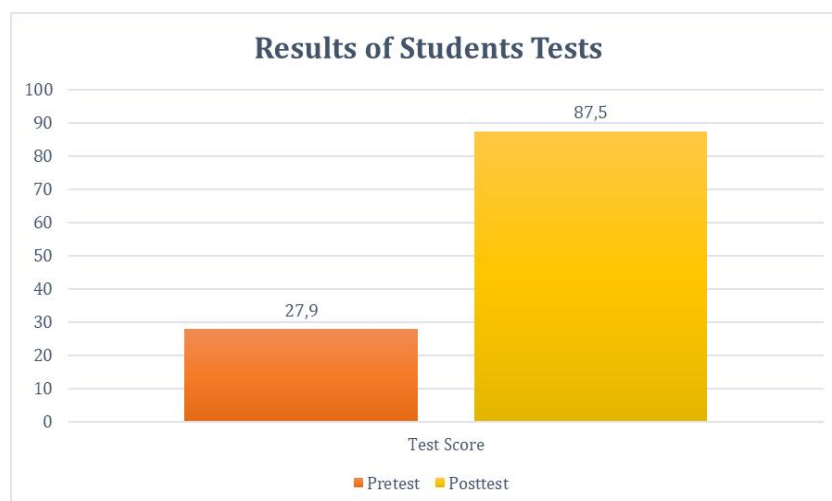


Image 1. Pretest and Posttest Score

Furthermore, a paired sample t-test was employed to ascertain whether a notable distinction existed between the pretest and posttest scores in regard to the utilization of the learning module. Table 5 shows a Sig. (2-tailed) of 0.000, which is smaller than the threshold of 0.05, indicating that the implementation of the learning module had a significant positive effect on the improvement of computational thinking skills.

Table 5. Paired sample test

	Paired Differences					t	df	Sig.(2-tailed)
	Mean	Std, Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
Pair 1 Pretest- Posttest	-59.53125	19.23368	3.40007	-66.46573	-52.59677	-17.509	31	.000

Each indicator of computational thinking skill involves abstraction to debugging is also analyzed on Image 2. Specifically in the pretest, students' abstraction skills were at a moderate level with an average of 57.8. This shows that before the application of the module, students already had a basic understanding of the concept of abstraction, but not optimal yet. Abstraction is the ability to simplify complex problems by focusing on the essential elements, and this score indicates that there is a chance for improvement. After module implementation, the average score of abstraction skills increased significantly to 78.1. This indicates that the module successfully improved students' ability to identify and simplify complex problems, improving their skills in managing and organising relevant information. In the pretest, students obtained an average score of 60.9 on the generalisation indicator, which measures their ability to apply general patterns and principles from specific situations to other situations. This score indicates that they have a fairly good foundation, but still need further improvement. After using the module, students' generalization skills increased sharply to 92.2. This shows that the module is very effective in training their ability to find patterns and apply them across new situations, an important component of computational thinking. In the pretest, the decomposition indicator score was 21.9, indicating that prior to the use of the module, students were not very capable of breaking down large problems into smaller, more manageable parts. This score indicates that students' decomposition skills are still very low at the initial stage.

After the implementation of the module, students' decomposition skills improved very significantly with a score of 96.9. This shows a remarkable improvement in their ability to break down complex problems into smaller and more manageable components. The module was very effective in improving this ability. In the pretest, students' algorithm skills were at zero. This means that before taking the module, they have absolutely no understanding or skills in making systematic steps to solve problems (algorithms). After using the module, the algorithm indicator score jumped dramatically to 96.9.

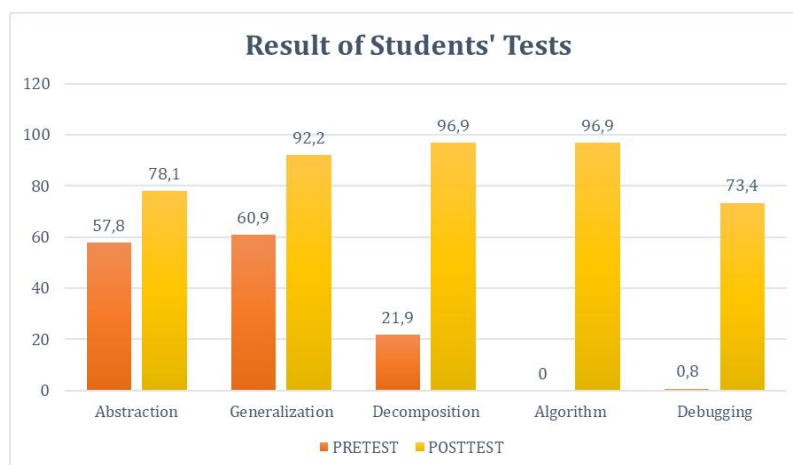


Image 2. Results of students' test

The module we developed successfully provided students with a deep understanding of algorithm design and implementation. They became very capable of organising structured steps to solve problems systematically. Last indicator analysis is debugging. In the pretest, the debugging skill score was almost non-existent, with a score of 0.8. This indicates that students at the initial stage do not have sufficient ability to identify and correct errors in the programming process or solution. After the module was implemented, the debugging score increased significantly to 73.4. Students became much more skilled in recognizing errors and correcting them effectively. This shows that the module has been successful in teaching debugging as part of computational thinking skills. The pretest and posttest results showed a very significant improvement in all indicators of computational thinking skills. The greatest improvement was seen in algorithm, decomposition, and debugging skills, indicating that the module was very effective in developing students' ability to solve problems, create algorithms, apply patterns found in new contexts and also fix the syntax errors. Despite in the pretest the students' abilities were still low in some aspects, especially algorithms and debugging, after the use of the module, all computational thinking skills experienced a huge improvement.

Conclusion and Suggestion

Based on the results of the study, it can be concluded that the module developed to train the computational thinking skills of prospective teacher students proved to be effective. Module validation by experts showed high scores, including content, science concepts, constructs, language, and layout. The pilot test on 32 students showed a significant increase from pretest to posttest, with an average pretest of 27.9 and posttest of 87.5, and N-gain of 0.82 which is in the high category. Further analysis based on computational thinking skills revealed significant improvements in all aspects, such as abstraction, generalization, decomposition, algorithm, and debugging. The result of a statistical test with the Paired t-test method confirmed that the difference between pretest and posttest was statistically significant, indicating that this module is effective in improving computational thinking skills. As a suggestion, this module can be applied on a wider scale, for example in other institutions, to see if similar results can be generated in various contexts. Although there were significant improvements in all skills, the debugging aspect still had relatively lower posttest scores compared to other skills.

Therefore, further development is recommended to focus on improving materials or exercises related to debugging. Integration of more interactive technologies, such as augmented reality or computer simulation, is also suggested to enrich the learning experience. In addition, a project-based collaborative learning approach can help improve students' cooperation and communication skills. Further research can also be conducted to test the effectiveness of this module at other education levels or in different fields of study.

References

- Agbo, F. J., Oyelere, S. S., Suhonen, J., & Adewumi, S. (2019). A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. <https://doi.org/10.1145/3364510.3364521>
- Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, 105, 106185. <https://doi.org/10.1016/j.chb.2019.106185>
- Bargury, I Zur . (2012). A new curriculum for junior-high in computer science. *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, 204–208. <https://doi.org/10.1145/2325296.2325347>
- Branch, R. M. 2009. *Instructional Design: The ADDIE Approach*. New York: Springer US. <https://doi.org/10.1007/978-0-387-09506-6>
- Castro, L. C., Shoaib, H., & Douglas, K. (2021). Computational Thinking Frameworks used in Computational Thinking Assessment in Higher Education. A Systematized Literature Review. 2021 ASEE Virtual Annual Conference Content Access Proceedings. <https://doi.org/10.18260/1-2--36824>
- Duda, M., Sovacool, K., Farzaneh, N., Nguyen, V., Haynes, S., Falk, H., Furman, K., Walker, L., Diao, R., Oneka, M., Drotos, A., Woloshin, A., Dotson, G., Kriebel, A., Meng, L., Thiede, S., Lapp, Z., & Wolford, B. (2021). Teaching Python for Data Science: Collaborative development of a modular interactive curriculum. *Journal of Open Source Education*, 4(46), 138. <https://doi.org/10.21105/jose.00138>
- Guggemos, J. (2021). On the predictors of computational thinking and its growth at the high-school level. *Computers & Education*, 161, 104060. <https://doi.org/10.1016/j.compedu.2020.104060>
- Hake, R. R. (1998). Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 66(1), 64–74. <https://doi.org/10.1119/1.18809>
- Hoppe, H. U., & Manske, S. (2022). Developing Computational Thinking Skills with Multiple Models and Representations. *Computational Thinking Education in K–12*, 49–70. <https://doi.org/10.7551/mitpress/13375.003.0006>
- Jancheski, M. (2017). Improving Teaching and Learning Computer Programming in Schools through Educational Software. *OLYMPIADS IN INFORMATICS*, 11(1), 55–75. <https://doi.org/10.15388/oi.2017.05>
- Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing*, 4(3), 583.
- Lockwood, J., & Mooney, A. (2018). Computational Thinking in Secondary Education: Where does it fit? A systematic literary review. *International Journal of Computer Science Education in Schools*, 2(1), 41–60. <https://doi.org/10.21585/ijcses.v2i1.26>
- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61. <https://doi.org/10.1016/j.chb.2014.09.012>
- Lyon, J. A., & J. Magana, A. (2020). Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education*, 28(5), 1174–1189. Portico. <https://doi.org/10.1002/cae.22295>
- Nyutu, E., Cobern, W. W., & Pleasants, B. A.-S. (2020). Correlational Study of Student Perceptions of their Undergraduate Laboratory Environment with respect to Gender and Major. *International Journal of Education in Mathematics, Science and Technology*, 9(1), 83–102. <https://doi.org/10.46328/ijemst.1182>
- Papert, S. (1980). Computers for children. *Mindstorms: Children, computers, and powerful ideas*, 3-18. <https://dl.acm.org/doi/pdf/10.5555/1095592.C1104753>

- Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1). <https://doi.org/10.1007/bf00191473>
- Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior*, 105, 105849. <https://doi.org/10.1016/j.chb.2018.12.027>
- Ridlo, Z. R., Dafik, Waluyo, J., & Yushardi. (2024). Exploring the impact of makerspace-based learning materials on students' computational thinking skills: Using machine learning to address challenges in smart coffee agroforestry. *Edelweiss Applied Science and Technology*, 8(5), 52–73. <https://doi.org/10.55214/25768484.v8i5.1630>
- Ridlo, Z. R., Supeno, S., Wahyuni, S., Wicaksono, I., & Ulfa, E. M. (2022). Analysis of Implementation Project-Based Learning Model of Teaching Integrated with Computer Programming in Improving Computational Thinking Skills in a Classical Mechanics Course. *Jurnal Penelitian Pendidikan IPA*, 8(4), 2029–2035. <https://doi.org/10.29303/jppipa.v8i4.1789>
- Román-González, M., Moreno-León, J., & Robles, G. (2022). Toward a Theory (and Practice) of Multiple Computational Thinkings. *Computational Thinking Education in K–12*, 71–102. <https://doi.org/10.7551/mitpress/13375.003.0007>
- Sun, L., Hu, L., & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple factors. *Thinking Skills and Creativity*, 42, 100926. <https://doi.org/10.1016/j.tsc.2021.100926>
- Supian, S., Ahmad, M. F., & Subiyanto, S. (2023). Basic Programming Training in Python for Junior High School Students at Al Fitrah Islamic Boarding School. *International Journal of Research in Community Services*, 4(4), 170–173. <https://doi.org/10.46336/ijrcs.v4i4.489>
- Tanjung, Y. I., Azhar, M., Razak, A., Yohandri, Y., Arsih, F., Wulandari, T., Nasution, B., & Lubis, R. H. (2023). State of The Art Review: Building Computational Thinking on Science Education. *Jurnal Pendidikan Fisika Indonesia*, 19(1), 65–75. <https://doi.org/10.15294/jpfi.v19i1.41745>
- Wing, J. M. 2006. Computational thinking. *Communications of the ACM*. 49(3): 33–35. <https://doi.org/10.1145/1118178.1118215>