Incorporation of Computational Thinking Practices to Enhance Learning in a Programming Course

Leticia Laura-Ochoa, Norka Bedregal-Alpaca
Universidad Nacional de San Agustin de Arequipa, Arequipa, Peru

***Abstract*—The development of computational thinking skills is essential for information management, problem-solving, and understanding human behavior. Thus, the aim of the experience described here was to incorporate computational thinking practices to improve learning in a first Python programming course using programming tools such as PSeInt, CodingBat, and the turtle graphic library. A quasi-experimental methodological design was used in which the experimental and control groups are in different academic semesters. Exploratory mixed research was carried out. The control and experimental group consisted of 41 and 36 students, respectively. The results show that with the use of support programming tools, such as PSeInt, CodingBat, Python turtle graphic library, and the incorporation of computational thinking practices, the experimental group students obtained better learning results. It is concluded that student performance and motivation in university programming courses can be improved by using proper tools that help the understanding of programming concepts and the skills development related to computational thinking, such as abstraction and algorithmic thinking.**

***Keywords*—Programming tools; computational thinking; algorithmic thinking; motivation; abstraction**

I. Introduction

Computational Thinking (CT) is a fundamental skill for all students [1]. In [2], CT has been found to involve abstraction, algorithmic thinking, automation, decomposition, debugging, and generalization. In addition, the formation and development of algorithmic thinking in higher education students is a requirement of the information society, as it provides them with instruments to solve problems of everyday life [3] and get a solution through a series of steps [4]. It is a fundamental skill that students develop when they learn to program [5]. Also, computer programming involves other skills like logical reasoning and creativity in problem-solving.

However, learning computer programming for novice students is considered a challenge for educators, since a decrease in students' interest and motivation to learn programming courses has been noted [6]. The learning process can be complicated and demanding, difficult to master for novice programmers [6][7]. Computer programming courses are considered the most difficult courses in which undergraduate students do not usually succeed [8]; in [9] explain that the content of an introductory programming course emphasizes more on learning the syntax and semantics of the programming language. In addition, programming courses should not only focus on teaching students to write code but should also include the development of skills related to computational thinking, such as algorithmic thinking, logic, and problem-solving [4].

According to [10], programming courses introduce a programming language and the computer science thinking way. Furthermore, programming exposes students to computational thinking, because it requires problem-solving using computer science concepts such as abstraction and decomposition [11]. For [12][13], CT has begun to influence various disciplines and professions, in addition to all science and engineering disciplines, making it necessary to include it in general education.

On the other hand, high dropout rates are found in introductory programming courses [14], one of the main reasons being the lack of students' motivation [15]. Since computer programming requires constant effort and practice, it is important to keep students motivated [16], to get their predisposition to continue learning and improve their learning.

Consequently, the problem we found is that traditional methods used to teach programming courses to novice students, based on syntax and semantic content of the programming language can demotivate students to continue learning programming courses, causing low performance in their learning and even dropout.

In this context, the aim of the experience described here was to select the proper programming language, tools, and teaching strategies to teach introductory programming courses, so that students improve their learning outcomes, develop skills related to computational thinking, learn to program, and increase their motivation towards the subject of programming.

The rest of the paper is organized as follows: Section II provides some related works proposed in the literature. Section III describes the conceptual framework on algorithmic thinking, abstraction, and Python. Section IV explains the overview of the methodology. Section V presents a detailed description of the experience of incorporating computational thinking practices in the programming course. Section VI shows the results of applying programming tools and computational thinking practices to improve student performance. Section VII discusses the results obtained. Section VIII presents the conclusions and future work.

II. Related Work

In the work of [17], they present the use of the ADRI (Approach, Deployment, Result, Improvement) approach in the teaching and learning process of an introductory programming course, for which they redesigned their course materials and

developed an editor so that students can complete the required stages of the approach, managing to improve student learning outcomes compared to previous semesters, focusing on problem-solving strategies as well as programming knowledge. Additionally, ADRI's approach and editor reduced failure and dropout rates.

In [18], they present a teaching approach based on four components: The use of the Python programming language, project-oriented and problem-based learning methodologies, multimedia resources available on virtual platforms, and evaluation rubrics. The approach used improved the academic performance of the students, which is evidenced in the grades obtained, and the dropout rates were reduced. The results obtained suggest Python as a proper programming language for students of a first introductory programming course, due to its simplicity in syntax and code debugging, in addition to the use of other pedagogical strategies that support the learning process.

In the work of [3], they carry out an analysis of the scientific literature considering definitions, main properties, and characteristics of algorithmic thinking. They then present a universal sequence of algorithm development, involving different types of thinking such as abstract, conceptual, logical, constructive, and figurative. They carried out a survey in which the participants demonstrated a low level of understanding about algorithms, algorithmic thinking, and its usefulness in daily life and professional activity, so they end that algorithmic thinking is important for any higher education subject, not only in information and communication technologies (ICT) area and consider it as a new dimension of learning in higher education.

In [9], they introduce a new teaching approach focusing on algorithmic thinking skills besides the knowledge of the syntax and semantics of a programming language in an introductory programming course, using techniques of flowchart and pseudocode. Their results show that the ADRI approach promotes the three-step approach (Problem statement → Solution plans → Code) to solve a problem, fosters programming knowledge, as well as problem-solving strategies, promoting algorithmic thinking.

In the work of [19], they describe the design and implementation of an introductory computational thinking course to teach programming to high school students with activities that take place in a web-based programming environment that uses a variant of the Haskell language, promoting higher-order thinking. They address the need for computational thinking courses geared toward all students, not just future software developers, by making connections between learning programming with science and math. Most of the students who participated in the course considered it difficult; but there was an overall positive reception from the students, who learned the language and the general principles of programming, logic, and modeling. They find that courses like Python typically do not focus on computational thinking and follow traditional syntax-oriented approaches to teaching programming, with little connection to science and math.

III. Conceptual Framework

A. Algorithmic Thinking

Algorithmic thinking is essential in comprehensive general education and programming is a way to teach the basic principles of algorithmic thinking from the beginning [10], it is important in higher education, to develop algorithms in the context of the future profession and everyday life in the modern information society [3]. Also, it is considered a significant component of the cognitive competencies of the future engineer because the algorithmic activity allows forming adequate algorithmic skills, through which students develop techniques of mental actions such as generalization, classification, analogy, the establishment of patterns and logical reasoning, which are the main components of algorithmic thinking [20]. Therefore, it is advisable to promote the development of algorithmic thinking skills through programming in the different disciplines and professions, besides careers related to computing.

According to [5][4][21], algorithmic thinking consists of a clear definition of the steps to reach a solution, thinking in terms of instruction sequences and rules that lead to problem-solving or understanding of situations. It is an important aspect of computational thinking [22], its main properties include discretion, abstraction, formality, integrity, and effectiveness [3].

B. Abstraction

Abstraction, efficiency, and algorithms are considered vital "mental tools" for computational thinking [23]. According to [13], abstraction is the most important and high-level thought process in computational thinking.

Abstraction is a key skill for computing, fundamental for mathematics and engineering in general [24], it involves reducing unnecessary details, eliminating complexity, choosing the correct detail to hide, and thus the problem is easier and understandable without missing anything important [21][5]. Therefore, it allows developing a potential solution by eliminating details of the problem [23]. For [25], abstract thinking is the ability to abstract the properties of objects that are relevant to a study.

Furthermore, abstraction allows defining patterns, generalizing by capturing common essential properties from instances, and parameterization [13]. Without abstraction, students tend to get overwhelmed with details and feel frustrated with the programming process [23], so the development of this skill is necessary, applicable in programming, mathematics, and the different disciplines.

C. Python

The main professional programming languages are based on text such as C, Python, Java [26]. Among these, the use of Python makes it easier for novice students to engage in the main features of computational thinking, mainly due to its basic syntax, dynamic typing (declaring variables is not required), structured and indented writing [4]. Its use is suitable because it includes turtle graphics libraries that allow a smooth transition from Logo to Python [10], which allows focusing on

concepts without a long introduction to the syntactic details of the language [27].

In addition, Python is a high-level programming language, easy to learn, free, and with documentation available on the Web [4].

IV. Methodology

The methodological design used was quasi-experimental with experimental and control groups located in different semesters, so the selection of its members was not random. Exploratory mixed research was carried out.

In the experience, the experimental group consisted of 36 students enrolled in the Programming course, group A, of the 2019-A academic period of the Professional School of Mechanical Engineering of the Universidad Nacional de San Agustin de Arequipa (Peru). In this group, there were 34 male students (94%) and 2 female students (6%). The control group was the students of group A who completed the Programming course in the 2018-A academic period, made up of 41 male students (100%).

The Programming course at the Professional School of Mechanical Engineering of the Universidad Nacional de San Agustin de Arequipa - Peru, is given in the third academic semester and it is developed for 17 weeks. It has 3 hours a week (1 theoretical hour and 2 laboratory hours), it is equivalent to 2 credits and the Python programming language is used.

In the control group, tools such as DFD were used to create data flow diagrams and PSeInt for pseudocode, before the use of the Python programming language; CodingBat and turtle graphic library were not used. A greater preference was also observed for the use of the PSeInt tool concerning the DFD tool, so in the experimental group only PSeInt was used and the use of the CodingBat tool and the Python turtle graphic library were incorporated with an approach oriented to computational thinking practices, in addition to the Python programming language.

Data collection was done from the students' grades obtained in their evaluations of the programming course, before (control group) and after the experiment (experimental group). Direct observations were also made during the programming activities.

To measure success, a comparison of the grades obtained by the students of the control and experimental groups was made, to check if there is an improvement in the students' performance of the experimental group. The results were validated by statistical analysis using SPSS Statistic V25 software, to determine if there is a statistically significant improvement.

V. Description of the Experience

This work describes our experience in the use of PSeInt, CodingBat, and Python turtle graphical library to motivate and reinforce students' learning of programming concepts and develop skills related to computational thinking in a first programming course with Python.

In the programming course, students learned topics such as sequential statements, conditionals, loops, functions, structured types, object-oriented programming.

In the 2019-A academic period, students began learning to create algorithms to solve problems using the PSeInt tool (Fig. 1), with which they developed algorithmic thinking skills, logic, and problem-solving strategies using pseudocode.

With PSeInt, the students were able to execute algorithms in an automated way to test their solution proposals and verify results, analyzing the errors in the logic, which allowed the student to practice automation and debugging.

Then, the Python programming language was taught. Initially, they were asked to perform the same exercises developed in PSeInt, to pass their created algorithms to a computer program using the Python programming language.

Students learned the syntax and semantics of the Python programming language, practiced coding, running programs, checking results, parsing, and fixing syntax errors. Automation and debugging were also present.

To improve students' programming skills, the online code practice tool called CodingBat [28] was used, which presents some examples with solutions available for students to practice coding and executing programs in Python, allowing them to check your answer or see other ways to solve the same problem, plus there are several exercises to solve with hints available, using conditionals, loops, strings, and lists. Fig. 2 shows some exercises that were solved by the students using Boolean logic and conditionals in CodingBat, which provided them with more opportunities to practice their programming constructions, as well as reinforce computational concepts such as sequential and conditional instructions.

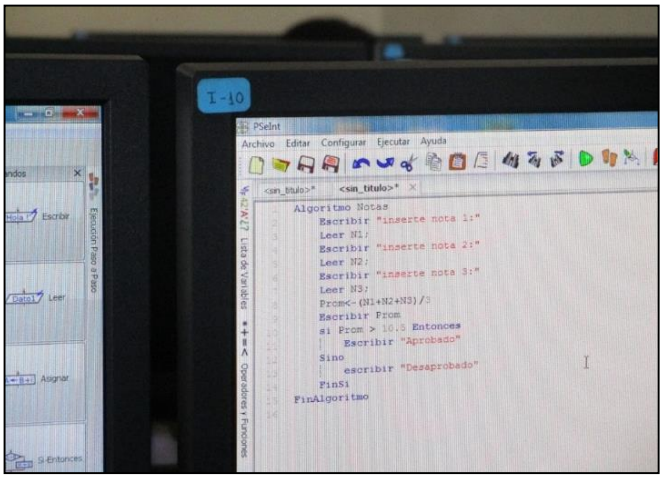

Fig. 1. Creation of Algorithms through Pseudocode using the PSeInt Tool.

Fig. 2. Carrying out Python Logic-1 Exercises in CodingBat.

Students learned to create graphics using Python turtle graphical library, they began by drawing geometric figures using sequential instructions and functions, then drawing different shapes and patterns (Fig. 3, 4, 5) incorporating repetitive instructions.

Fig. 3 shows geometric exercises performed by the students, where they apply iterations and functions from squares and rectangles to create different graphics with repetitive patterns.

Fig. 4 shows additional geometric exercises created from parallelograms and circles, where students apply their creativity and logic with the help of iterations and functions.

Fig. 5 shows an example, in which the problem is first decomposed using the functions parallelograms (to draw small rhombus by tracing lines of 65 pixels) and parallelogram (to draw large rhombus by tracing lines of 100 pixels). Through abstraction and generalization, repetitive patterns were identified to create the graphics, and then their abstractions were improved by proposing new solution strategies using parameters, which allowed reducing the two functions that drew the rhombuses of different sizes into a single function called parallelograms(n). The function parallelograms(n) groups instruction patterns to draw shapes by drawing lines according to the number of pixels specified in the parameter n, this function is reused from the main function called main.

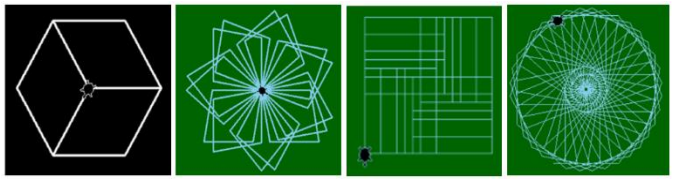

Fig. 3. Drawing Geometric figures such as Squares, Rectangles with different Angles of Rotation using Iterations and Functions.

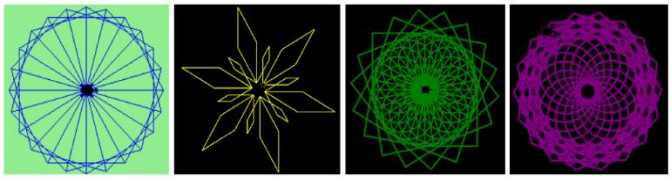

Fig. 4. Drawing Geometric figures Like Parallelograms, Circles using Iterations and Functions.

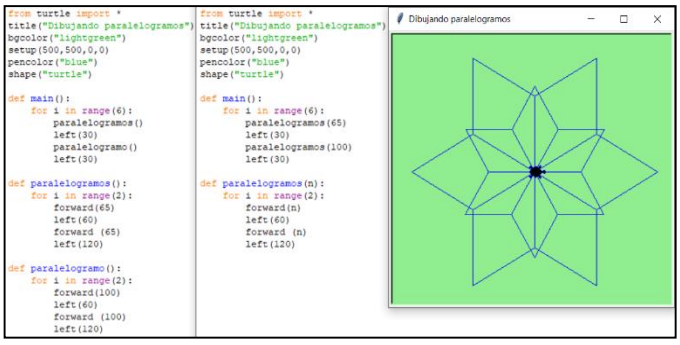

Fig. 5. Using Functions, Loops, Parallelograms of different Sizes (65 and 100 Pixels) with Rotation Angles of 60 and 120 Degrees.

In addition to reinforcing programming concepts such as sequential instructions, loops, and functions, students gained computational thinking practices such as decomposition, iteration, and abstraction that enabled them to recognize repeating patterns.

VI. Results

In the academic period 2019-A, students obtained an average grade of 16.75 in their first exam. Then, in the evaluation with Python, they obtained an average grade of 13.86 in their second exam. Finally, CodingBat and the turtle graphic library were used to reinforce and motivate them in their learning process, obtaining an average grade of 14.97, which improved their grade using only PSeInt for algorithms creation and the Python programming language. Table I shows the average of the grades obtained in the first, second and third exams of group A of the Programming course taught in the academic periods 2018-A and 2019-A, which range from 0 to 20.

Fig. 6 shows the average grades evolution for Exam 1, 2, and 3 in the academic periods 2018-A and 2019-A, showing an improvement for Exam 1 and Exam 3 in 2019-A.

The use of PSeInt, CodingBat, and turtle graphic library has shown an improvement in the students' grades in their average grades. Table II shows the global grade average of group A in 2018-A and 2019-A, which range from 0 to 20.

Fig. 7 shows the global grade average for the Programming course in the academic periods 2018-A and 2019-A, showing an improvement in 2019-A.

TABLE I. Average Grades for Exam 1, 2 and 3

Academic period	*Exam 1*	*Exam 2*	*Exam 3*
2018-A	14.34	13.66	13.37
2019-A	16.75	13.86	14.97

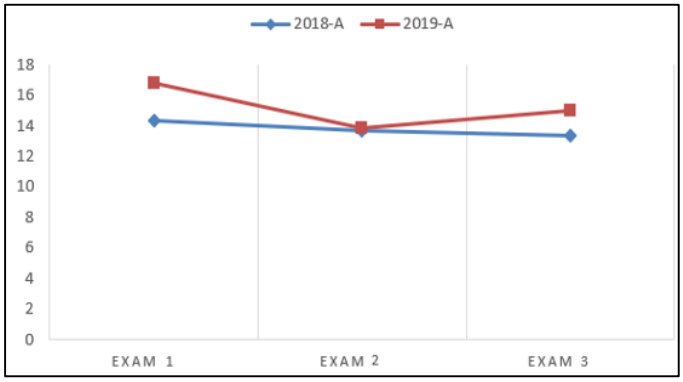

Fig. 6. Average Grade Result for Exam 1, 2, and 3 of the Programming Course by Academic Year.

TABLE II. Global Grade Average

Academic period	*Global average*
2018-A	14.3
2019-A	15.08

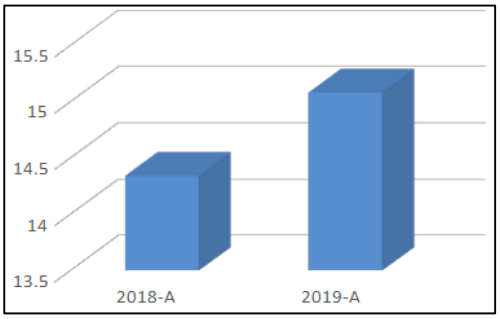

Fig. 7. Global Grade Average for the Programming Course by Academic Year.

In our experience, we have observed that novice students are motivated by using Python turtle graphic library to create their drawings and improve their abstractions, as well as develop skills related to computational thinking. CodingBat allowed them to improve their programming skills by practicing their coding in Python, thereby improving their final grades in the programming course.

SPSS Statistic V25 software was used for the statistical analysis of the results. Table III has some descriptive measures for the grades obtained by students in the years 2018 and 2019. The average of the grades for the year 2018 is 14.3 and with a 95% confidence level, it can be stated that range between 13.86 and 14.82, while the average of the grades for the year 2019 is 15.08 and with the same 95% confidence level it can be stated that range between 14.31 and 15.85. Consequently, it could be assumed that the grades obtained in 2019 were better than those of 2018.

To determine if the difference was statistically significant, as a requirement, it was necessary to verify that the conditions of normality and heteroskedasticity were met. With a significance level of 5% (0.05), the Kolmogorov-Smirnov test was applied, obtaining the results of Table IV, as the p-value (Sig.) is greater than the significance, and the grades distribution normality is accepted.

When applying the t-test for independent samples, the results shown in Table V were obtained.

In Levene's test, as the p-value of 0.27 is greater than the significance, then it was possible to affirm that the assumptions of normality and heteroskedasticity were met, therefore, it was possible to apply the t-test for the means difference. Table V shows that the bilateral p-value is 0.093, so the unilateral value is 0.046, which is less than the significance; consequently, the means equality hypothesis is rejected.

TABLE III. DESCRIPTIVE MEASURES FOR THE GRADES OBTAINED IN THE YEARS 2018 AND 2019

Grade	2018	2019
Mean	14.3415	15.0833
Standard deviation	1.52659	2.27251
Standard error of the mean	0.23841	0.37875
N	41	36
CI 95% lower limit	13.86	14.31
CI 95% upper limit	14.82	15.85

TABLE IV. RESULTS OF THE KOLMOGOROV-SMIRNOV NORMALITY TEST

Year	Kolmogorov-Smirnov[a]		
	Statistic	*df*	*Sig.*
2018	,149	41	,052
2019	,126	36	,163

Lilliefors Significance Correction

TABLE V. INDEPENDENT SAMPLES T-TEST

	Levene's test for equality of variances		t-test for equality of means	
	F	*Sig.*	*t*	*Sig. (2-tailed)*
Equal variances assumed	5.120	0.27	-1,699	,093
Equal variances not assumed			-1,658	,103

VII. DISCUSSION

Statistically, it is possible to state that the difference found between the grades obtained in 2018 and 2019 were statistically different and, with a confidence level of 95% it can be stated that the grades obtained by the students in 2019 were better than those obtained in 2018.

The use of support tools such as PSeInt, CodingBat, and Python turtle graphic library have increased students' motivation and performance in a first programming course with Python, because an approach oriented to computational thinking practices was also followed. As indicated by [29], computer programming is the main demonstration of computational thinking skills. However, they tend to follow syntax-oriented programming teaching approaches, without focusing on computational thinking, with few connections to mathematics and science [19]. Furthermore, in the work of [18], they indicate that the Python programming language is suitable for introductory programming courses because of its simple syntax and ease for code debugging, they also point out that it is necessary to consider other aspects such as pedagogical strategies that allow improving the programming teaching-learning process.

According to [5], algorithmic thinking is a fundamental skill that students acquire when they learn to program, developing the ability to think in terms of sequences and rules to solve problems. In the work of [9], they used flowcharts and pseudocode for novice students to propose solutions to problem statements, which promote algorithmic thinking skills. Similarly, in our work, we use the PSeInt tool for students to develop algorithmic thinking skills, logic, and problem-solving strategies by creating algorithms with pseudocode. In addition, because it is a tool that allows algorithms execution in an automated way, the students were able to test their solution proposals, find and resolve errors in logic, practicing automation and debugging, which are part of the computational thinking skills found in the work of [2].

We consider that the use of support tools such as CodingBat is essential so that students can practice their programming constructs and improve their problem-solving skills in the programming language used in the course.

Likewise, in the work of [17], they express the importance of practicing programming skills by dedicating more time and focusing on problem-solving strategies.

According to the experience described, the Python turtle graphic library allowed the acquisition of computational practices such as abstraction, decomposition, iteration, and debugging, which correspond to the computational thinking practices defined by [30] and adapted in the work of [31]. In addition, in the programming course, students developed skills such as algorithmic thinking, automation, and generalization, which are part of the computational thinking skills identified in five articles highlighted in the work of [2], which are abstraction, algorithmic thinking, automation, decomposition, debugging, and generalization. On the other hand, in the programming course, concepts such as sequential instructions, loops, conditionals were learned, which correspond to the concepts of computational thinking considered in the work of [31].

We agree with [32], in the sense that programming environments with graphic components allow the acquisition of computational thinking practices through programs creation, which is attractive to them, helping students develop a positive attitude towards programming.

Currently, technology-mediated training processes are becoming increasingly flexible and collaborative; therefore, problem-solving activities can be involved [33] that involve cooperative learning techniques [34], such as the programming of a robotic hand. Consequently, the student would not only be developing computational thinking but also critical spirit, creativity, and collaborative work.

VIII. Conclusion

In this study, we considered two semesters with different students. Semester 2018-A with 41 students from group A, served as a control group, where we used the DFD and PSeInt tools earlier to teach programming using Python, while in semester 2019-A, with 36 students from group A, we started with the PSeInt tool before teaching Python, then the students' learning of Python programming was reinforced with the CodingBat tool and turtle graphic library. We examined the grades obtained in the midterm exams and the global average of the programming course, where an improvement in the grades in the second experimental group with the support tools used in the course is evidenced, indicating that they acquired better programming skills and therefore better performance. In addition, we observed that students are motivated by using the Python turtle graphic library that reinforces their learning of sequential instructions, loops, functions, and allows the development of skills related to computational thinking such as algorithmic thinking, decomposition, iteration, and abstraction. The experience of this work can serve as a reference for educators interested in approaches oriented to computational thinking practices in programming teaching.

As future work, we consider investigating computational thinking measurement evaluations to be applied after following a programming teaching approach oriented to computational thinking practices.

Acknowledgment

The authors' thanks are expressed to the National University of San Agustin de Arequipa for the support received in the realization of the proposal and the results are expected to benefit the institution.

References

[1] Q. Li, “Computational thinking and teacher education: An expert interview study,” Human Behavior and Emerging Technologies, vol. 3, no. 2, pp. 324-338, 2021.

[2] S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari and K. Engelhardt, “Developing computational thinking in compulsory education - Implications for policy and practice,” in JRC Science for Policy Report, 2016.

[3] M. F. Byrka, A. V. Sushchenko, A. V. Svatiev, V. M. Mazin and O. I. Veritov, “A New Dimension of Learning in Higher Education: Algorithmic Thinking,” Propósitos y Representaciones, vol. 9, no. SPE2, pp. 990, 2021.

[4] F. Buitrago Flórez, R. Casallas, M. Hernández, A. Reyes, S. Restrepo and G. Danies, “Changing a generation’s way of thinking: Teaching computational thinking through programming,” Review of Educational Research, vol. 87, no. 4, pp. 834-860, 2017.

[5] A. Csizmadia, P. Curzon, M. Dorling, S. Humphreys, T. Ng, C. Selby and J. Woollard, “Computational thinking-A guide for teachers,” Computing at School, 2015.

[6] M. Piteira and C. Costa, “Computer programming and novice programmers,” in Proceedings of the Workshop on Information Systems and Design of Communication pp. 51-53, 2012.

[7] M. Karaliopoulou, I. Apostolakis and E. Kanidis, “Perceptions of Informatics Teachers Regarding the Use of Block and Text Programming Environments,” European Journal of Engineering Research and Science, pp. 11-18, 2018.

[8] B. Özmen and A. Altun, “Undergraduate students' experiences in programming: difficulties and obstacles,” Turkish Online Journal of Qualitative Inquiry, vol. 5, no. 3, pp. 1-27, 2014.

[9] S. I. Malik, M. Shakir, A. Eldow and M. W. Ashfaque, “Promoting Algorithmic Thinking in an Introductory Programming Course,” International Journal of Emerging Technologies in Learning, vol. 14, no. 1, 2019.

[10] J. Hromkovic, T. Kohn, D. Komm and G. Serafini, “Algorithmic thinking from the start,” Bulletin of EATCS, vol. 1, no. 121, 2017.

[11] S. Y. Lye and J. H. L. Koh, “Review on teaching and learning of computational thinking through programming: What is next for K-12?,” Computers in Human Behavior, vol. 41, pp. 51-61, 2014.

[12] J. M. Wing, “Computational thinking,” Communications of the ACM, vol. 49, no. 3, pp. 33-35, 2006.

[13] J. M. Wing, “Computational thinking: What and why. The Link,” News from the School of Computer Science at Carnegie Mellon University, 2011.

[14] C. Chen, P. Haduong, K. Brennan, G. Sonnert and P. Sadler, “The effects of first programming language on college students’ computing attitude and achievement: a comparison of graphical and textual languages,” Computer Science Education, vol. 29, no. 1, pp. 23-48, 2019.

[15] P. Kinnunen and L. Malmi, “Why students drop out CS1 course?,” in Proceedings of the second international workshop on Computing education research, pp. 97-108, 2006.

[16] A. Settle, A. Vihavainen and J. Sorva, “Three views on motivation and programming,” in Proceedings of the 2014 conference on Innovation & technology in computer science education, pp. 321-322, 2014.

[17] S. Iqbal Malik and J. Coldwell-Neilson, “Impact of a new teaching and learning approach in an introductory programming course,” Journal of Educational Computing Research, vol. 55, no. 6, pp. 789-819, 2017.

[18] O. Solarte Pabón and L. E. Machuca Villegas, “Fostering Motivation and Improving Student Performance in an Introductory Programming Course: An Integrated Teaching Approach,” Revista EIA, vol. 16, no. 31, pp. 65-76, 2019.

[19] F. Alegre, J. Underwoood, J. Moreno and M. Alegre, "Introduction to Computational Thinking: a new high school curriculum using CodeWorld," in Proceedings of the 51st ACM Technical Symposium on Computer Science Education, pp. 992-998, 2020.

[20] M. Kovalchuk, A. Voievoda and E. Prozor, "Algorithmic Thinking as the Meaningful Component of Cognitive Competencies of the Future Engineer," Universal Journal of Educational Research, vol. 8 (11B), pp. 6248-6255, 2020.

[21] P. Curzon, M. Dorling, T. Ng, C. Selby and J. Woollard, "Developing computational thinking in the classroom: a framework," Computing at School, 2014.

[22] M. Romero, A. Lepage and B. Lille, "Computational thinking development through creative programming in higher education," International Journal of Educational Technology in Higher Education, vol. 14, no. 1, pp. 1-15, 2017.

[23] J. A. Qualls, M. M. Grant and L. B. Sherrell, "CS1 students' understanding of computational thinking concepts," Journal of Computing Sciences in Colleges, vol. 26, no. 5, pp. 62-71, 2011.

[24] J. Kramer, "Is abstraction the key to computing?," Communications of the ACM, vol. 50, no. 4, pp. 36-42, 2007.

[25] M. Zapata-Ros, "Pensamiento computacional: Una nueva alfabetización digital," Revista de Educación a Distancia (RED), vol. 46, no. 4, 2015.

[26] M. Kölling, N. C. Brown and A. Altadmri, "Frame-based editing: Easing the transition from blocks to text-based programming," in Proceedings of the Workshop in Primary and Secondary Computing Education, pp. 29-38, 2015.

[27] J. Hromkovič, T. Kohn, D. Komm and G. Serafini, "Combining the power of python with the simplicity of logo for a sustainable computer science education," in International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer, Cham, pp. 155-166, 2016.

[28] N. Parlante. (2017). CodingBat code practice [Online]. Available: https://codingbat.com/python

[29] M. Román-González, J. C. Pérez-González, J. Moreno-León and G. Robles, "Can computational talent be detected? Predictive validity of the Computational Thinking Test," International Journal of Child-Computer Interaction, vol. 18, pp. 47-58, 2018.

[30] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in Proceedings of the 2012 annual meeting of the American Educational Research Association, 2012.

[31] F. Luo, P. D. Antonenko and E. C. Davis, "Exploring the evolution of two girls' conceptions and practices in computational thinking in science," Computers & Education, vol. 146, pp. 103759, 2020.

[32] L. Laura-Ochoa and N. Bedregal-Alpaca, "Análisis de entornos de programación para el desarrollo de habilidades del pensamiento computacional y enseñanza de programación a principiantes," Revista Ibérica de Sistemas e Tecnologias de Informaçao, no. E43, pp. 533-548, 2021.

[33] V. Cornejo-Aparicio, S. Flores-Silva, N. Bedregal-Alpaca and D. Tupacyupanqui-Jaén, "Capstone courses under the PBL methodology approach, for engineering," 2019 IEEE World Conference on Engineering Education (EDUNINE), 2019, DOI: 10.1109/EDUNINE.2019.8875803.

[34] N. Bedregal-Alpaca, V. Cornejo-Aparicio, A. Padron-Alvarez and E. Castañeda-Huaman, "Design of cooperative activities in teaching-learning university subjects: Elaboration of a proposal," International Journal of Advanced Computer Science and Applications, vol. 11, no. 4, 2020, DOI: 10.14569/IJACSA.2020.0110445.