frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 22 May 2023
DOI 10.3389/fcomp.2023.1085507

EDITED BY
Carmen Llorente Cejudo,
University of Seville, Spain

REVIEWED BY
Daisuke Saito,
Waseda University, Japan
Radoslava Stankova Kraleva,
South-West University "Neofit Rilski", Bulgaria

*CORRESPONDENCE
Hector Belmar
hector.belmar@hotmail.com

RECEIVED 31 October 2022
ACCEPTED 02 May 2023
PUBLISHED 22 May 2023

CITATION
Belmar H (2023) Teaching computer programming: impact of Brown and Wilson's didactical principles.
Front. Comput. Sci. 5:1085507.
doi: 10.3389/fcomp.2023.1085507

Teaching computer programming: impact of Brown and Wilson's didactical principles

Hector Belmar[1,2]*

[1]School of Computer Engineering, Ñuñoa Campus, INACAP (National Professional Training Institute), Santiago, Chile, [2]UMCE (Metropolitan University of Education Sciences), Santiago, Chile

This research studies the effects of the application of didactics to the teaching of computer programming, focusing on programming skills in the Python computer language. The problem arises from the failure and dropout rates of students in computer programming in computer science careers in INACAP and the consequent interest in promoting better learning. The general objective is to study the effects of an innovative methodology, based on Brown and Wilson's didactic principles, on the teaching process of Python programming in computer science students at INACAP. The theoretical framework is based on the didactics of teaching computer programming and the concepts of computational thinking skills of various theoretical references, and in particular on the didactic principles of Brown and Wilson. This research is carried out with a quantitative methodology of explanatory scope and with a quasi-experimental design, with a purposive sample, for the experimental stage the sample will consist of 100 first year undergraduate students of Computer Science, of which 50 will be the experimental group and 50 will be the control group. The hypothesis proposed is that "The students in the experimental group obtain a higher performance when applying Brown and Wilson's didactic principles than the students in the control group who are taught in a traditional way." The data collection technique used will be a 45-question multiple-choice test. The data analysis will be performed by applying statistical criteria, comparison of means and variances, among others.

KEYWORDS
computer programming, algorithm, didactics, evaluation instrument, didactical principles

1. Introduction

Since its beginnings in the 20th century with the theoretical conceptualization by Alan Turing of a machine capable of making calculations and with memory to store data, computing has led us in a spiral of technological advances, which has evolved exponentially. The Englishman Alan Turing was the founder of Computer Science, he was a mathematician, philosopher and cryptographer, a visionary of his time. Turing lived from (1912 to 1954). It should be noted that, without that initial idea and the construction of the first computers as electromechanical machines, computing would not be what we know today (Kulkarni, 2015).

In this context, computer science has evolved during the second half of the 20th century, starting with computer programming in assembler through procedural languages to focus at the beginning of the 21st century on Object Oriented programming and code generators. Currently there are several initiatives in the world to incorporate programming in schools, one of them is the Bebras challenge, which is aimed at motivating, practicing and evaluating the skill levels of students at an extracurricular level. Along with the Bebras challenge is the initiative called https://code.org/, which provides free material to anyone who requests it, in order to facilitate access to computer programming teaching material. There is also the NGO https://www.ideodigital.cl/, which provides free curriculum materials for schools, including teaching materials and associated assessment tools. But let's see what each of these initiatives is.

The Bebras Challenge (https://www.bebras.org/) is a community computer education network that was born in Lithuania in 2005 and has been consolidated in more than 40 countries to discuss computer science concepts for school computer education. The Bebras algorithm development workshops, which have been organized annually since their inception, bring together representatives from all these countries for rigorous work and decision making on good tasks to promote computer education in schools. On the one hand, the Bebras challenge is an international assembly driven to respond to the needs of computer education worldwide, and on the other hand, almost all activities are nationally based, organized by communities in the participating countries. Bebras is an attractive way to promote computer science learning worldwide. It has a community-based, distributed organization, which makes it a democratic organization, as everything comes from the bottom up. The workshops work as extracurricular activities so participants do it voluntarily and based on their motivation to learn (Dagiene and Stupuriene, 2016; Araujo et al., 2019).

Regarding the NGO Code.org is a non-profit organization dedicated to expanding access to computer programming in schools and increasing the participation of all who need and want to learn to program computers. Code.org's vision (www.code.org) is that every student in every school can learn to program and that this learning is free for those still in school. Code.org, the leading distributor of school computing curriculum to school communities in the United States, also established the annual "Hour of Code" crusade, which has reached more than 15% of all school children worldwide (Kale and Yuan, 2021).

Ideodigital is a national initiative originated thanks to a strategic alliance between the Kodea Foundation (www.kodea.org) and the BHP Foundation (www.bhp.com), which seeks to create the necessary conditions for thousands of children and adolescents to become protagonists of the digital society of the 21st century, incorporating computer science in the Chilean public school system. The purpose is to create awareness in the actors of the educational community about the benefits and feasibility of the application of computer science in the public school system. It also promotes the implementation of computer science in the classroom, for which it develops curricular courses from the first year of primary education to the fourth year of secondary education, which are available as a public good for the entire school system, articulating an ecosystem to train and accompany teachers and develop a network of leading schools in the teaching of computer science, which by July 2022 there were more than 80 schools that have already joined the ideodigital system (www.ideodigital.cl).

At present, there are several problems for the implementation of the teaching of computer programming in schools, which begin in the political discussion of the countries and the sufficient budget allocation to equip the educational system with an adequate number of computers in laboratories suitable for teaching. After overcoming the issue of computer infrastructure, an additional problem arises, which is the availability of a sufficient number of teachers properly trained to teach programming. In this sense, the political discussion will begin when a diagnosis is made about the implications of the advance of technology and how it could displace a significant percentage of workers (due to the automation of work) and with it an increase in unemployment that will imply socio-political instability, which will shake the economy and governments that are not properly prepared for these changes. Once the first barrier is overcome, the implementation will come, which would imply an operational diagnosis, how many trained teachers exist, how they are distributed in the country, also regarding digital infrastructure, laboratories, software, etc. (Belmar, 2022).

1.1. Context of programming teaching

Already at the beginning of the 21st century, computing is ubiquitous, filling every space, ranging from a system that allows programming a microwave oven to a complex Artificial Intelligence system that is capable of driving motor vehicles without endangering the safety of people, animals and property. In this context, in developed countries it has been defined that the XXI century is where work, economy and social relations are and will be guided by a new type of skills of people, which has been called computational thinking, which is constituted by the cognitive skills of; abstraction, decomposition, algorithmic thinking, debugging and problem solving. In addition, it has been proposed by several researchers that these skills are transferable to other areas of knowledge, such as natural sciences, mathematics and history (Wing, 2006).

It should be noted that the present research is oriented to implement a didactic strategy in the teaching process of computer programming, since until now the application of didactic strategies in the teaching of programming is unknown, which occurs for several reasons; on the one hand, the area of teaching programming (in schools) is incipient, so didactic theories for teaching programming have not been developed, or because the teachers who teach programming do not have pedagogical training, which in some cases are professional disciplines such as Computer Engineers or Electronic Engineers or other equivalent professionals with training in some computer programming language, so it has been naturalized that each teacher teaches the way he or she has learned in his or her own professional training. The following are some references on the teaching of programming in computer science courses in the first year of study in higher education.

1.2. A case from the University of TI West, Denmark

Bennedsen and Caspersen (2007) conducted a survey in 63 universities in various countries to estimate the failure rate of the subject "introduction to programming." In this regard, the research states that a common conception of students is that computer programming is a difficult course and that failure rates are high.

The average failure rate for different types of initial programming courses was around 30%. Therefore, it is suggested that ACM Education (Association Computing Machinery Education) and other participants in the study done by TI West University, may conduct further studies with a larger sample in order to postulate more representative results (Bennedsen and Caspersen, 2007).

1.3. A view from Durham University in the UK

Watson and Li (2014) researchers at Durham University in the United Kingdom, calculated a high failure rate in the achievement level of students in the subject of introduction to computer programming. The study describes the results of 161 introductory computer programming courses that were conducted in 15 different countries, in which the average pass rate was found to be 67.7%, therefore, the average failure rate is 32.3%. However, despite several studies conducted on the topic of achievement level in novice first-year computer programming students, which cite a motivation for research on pass/fail rate, in introductory programming courses, most of the available evidence is still not sufficiently representative. In addition, it was found that the pass rate has been maintained over time, and that it does not depend on the programming language taught, nor on the professor, nor on the University Institution that teaches it and that the perception of students is that learning to program is a difficult task (Watson and Li, 2014).

The objective of this study is to study the effects of Brown and Wilson's didactic principles on the teaching process of Python programming in computer science students at INACAP. To achieve this objective, it was necessary to work hand in hand with the institution in order to involve the computer science area to carry out the experimentation, which was carried out during two semesters, first the validation of the test published in Belmar (2023), and then with that validated test, work with the teachers of the control and experimental group, so that they would take part in the research, taking the students the pre and post-test, necessary for the data collection and statistical analysis described in this report.

2. Theoretical framework

2.1. Generalities

While it is true that this research aims to study the effects of a didactic strategy for teaching computer programming, it is also true that the concept of "Computational Thinking" has taken some prominence whenever the topic of computer programming is addressed. What is happening is that the teaching of computer programming is being added for all students in school, with the idea of generating broad skills, which go beyond being educated in programming. Thus, learning computer programming is used as a scaffold for the development of computational thinking skills, which are transferable to other areas of knowledge, since they also radiate to the whole area of science and technology (STEM), including art (STEAM) (Rojas and Garcia, 2020).

According to Wing (2006), Computer Science is the study of computation, what is computable and what is not, and what is computable, how to do it? But to appropriate the concept, we will see other researches that consider studies based on computational sciences, which broaden their meanings based on their application. Some applications of computing can be named, such as; Internet of Things, social networks, big data, Artificial Intelligence, Robotics, Video Games, Communications, smart phones, augmented reality, virtual reality, among others. The truth is that computing is present in everything (Psycharis et al., 2020).

For García (2018), the digitization of information is the beginning in the digital edification that will energize the world of the 21st century. Given this, from which we cannot dissociate, schools must carry out activities with students to act in a virtual world, for which they must learn the language of this century, without which they will be catechized into digital illiterates. Therefore, schools must teach students the skills of computational thinking. In this sense, ICT training is absolutely insufficient, since what this century requires is to get the skills in computation, to adapt to a new way of solving problems and thinking. Therefore, the present challenge is to train young people to succeed in a digitalized world. That is, they must be trained in the new paradigm of computational thinking skills (García, 2018).

Complementing the above, from Lithuania in 2018, researchers Juškevičiene and Dagiene from Vilnius University, published the article "relationship between computational thinking and digital competence," which investigates both concepts, starting with computational thinking. It should be noted, the researchers report, that the European Commission's scientific center has been guiding member countries in the sense that computational thinking is the most important skill of the 21st century. But, despite the interest in implementing computational thinking in schools and public and private investment, there are a number of challenges to the integration of computational thinking into school curricula (Juškevičiene and Dagiene, 2018).

It should be noted that the teaching of programming should consider students' ability to develop algorithms that solve concrete problems. Regarding algorithms, these are finite sequences of instructions written in some language according to its syntax, whose instructions involve: simple step-by-step sequences, as when guiding a student on the steps to follow to solve a second-degree equation; sequences that consider cycles, as when elaborating an algorithm that allows calculating powers by multiplying the base n-times by itself; conditional sequences as when implementing a Taylor series of a trigonometric function such as sine or cosine. Thus when assessing students learning to program, it is key to consider branching instructions (if statement condition then ... else), defined cycles (for {sentence$_1$ ... sentence$_k$}), and conditional cycles (while condition {sentence$_1$ ... sentence$_k$}), being control structures a prominent part of the learning that students must acquire (Mühling et al., 2015).

Of the articles read, the contribution of Belmar (2022) is important in his publication "Review on the teaching of programming and computational thinking in the world," research in which he integrates modeling and simulation as basic skills of computational thinking, so that it incorporates designing problems, categorizing and studying data logically, symbolizing data through models and mechanizing procedures. In this way, taking a look at teaching in schools, these skills enhance qualities such as: freedom to deal with complexity, perseverance in working with complex projects, tolerance of the fuzzy, and readiness for teamwork to achieve a common goal and report on it. Which is complemented by Grgurina (2021) who relates computational thinking in its main concepts, such as: "data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, modeling and simulation."

When confronting the achievements of countries in the teaching of computational thinking in schools, it is convenient to look at the development of nations in public policies that incorporate the subject in the mandatory curriculum, such as the case of England in 2013 and European countries since 2016, or other Asian states such as Japan, South Korea and China, in which computational thinking is understood as the skills of the XXI century and that will move investments in technology. However, in underdeveloped countries, whether in Latin America or Africa, the states have other priorities, so the issue is not the subject of political discussion, in this sense the teaching of programming remains in the hands of the initiative of individuals and lacks institutionalization and involvement in public policies that cement the creation of the necessary conditions for students to incorporate the skills of computational thinking in their knowledge (Belmar, 2022).

2.2. Didactics in programming education

2.2.1. Gamification: teaching through the use of games

The didactics based on the use of games helps in the students' willingness to learn, which in the field of computational thinking covers a larger area, ranging from the conception of the design idea, the construction and use of games. Gamification is also transversal to many areas of knowledge. Thus there are practices of the use of digital games in storytelling, in the practice in mathematics, in history, etc. Thus, de Paula et al. (2018)) in his work "Playing Beowulf: linking computational thinking, arts and literature through the creation of games," reading with that opens a game programmed by two 14-year-old students, "Playing Beowulf," which in collaboration with students from the British Library, a program in which 13–14 year olds from a London school learn, has managed to expand the scope of programming beyond the STEM area. The games are based on their unique understandings of the Anglo-Saxon poem Beowulf (de Paula et al., 2018).

In the USA, researchers from Carnegie Mellon University and Notre Dame, published in 2017 the article entitled "A computer game that promotes mathematics learning more than a conventional approach," which unveils a practical work that generates evidence that a mathematical educational game can help with learning alternatives, which manages to make learning mathematics more entertaining. The program "Decimal Point" is a single-player game that recreates the amusement park and is oriented to young teenagers. The program called "Decimal Point: the fantastic and fabulous world of fractional fun." The program is set up so that youngsters can jump in sequence to different themes (Haunted House, Wild West, Space Adventure, Amusement Park), playing a variety of small games in each theme area predestined to learn decimals. The youngster's progress is made through a sequence of steps in the park and players are graphically shown the next game (McLaren et al., 2017).

Similarly, a paper whose objective was to incorporate the computational thinking board game with robots so that students practice computational thinking skills as they complete the board game activities by controlling the action of the robots. The game teams are joined by two students in one team helping each other, those who played with the other team composed of two students. The work used Robots City board games to allow young people to learn the concept of computational thinking through game-based learning and use cell phone programs to command the actions of the robots (Zhou and Hsu, 2020).

2.2.2. Educational robotics

In teaching, the multimedia environment helps to lower learners' anxiety and provides a stress-free classroom environment. Also, multimedia materials collaborate with English instructors to promote students to perfect their English learning and decrease their language stress (Huang and Hwang, 2013 in Kong et al., 2020). As soon as students feel confident in the foreign language classroom, they lose inhibition and start speaking in the second language, the target language. The experiments reveal that, in the learning process, with the incorporation of robots, students speak English as a matter of course to command the robot. In addition, the research analyzed the behavior among the students and revealed that when the student is practicing oral English, there will always be other students to help him/her in his/her language practice, so if the student makes a mistake, there will always be someone to correct and guide him/her (Kong et al., 2020).

In another work at the Norwegian University of Science and Technology, children's teamwork and their creative disposition in programming games and educational robotics was investigated. The objective of the work was to investigate the teamwork driving qualities of children in programming activities. For this purpose, an experiment was designed with 44 students between 8 and 17 years old, who were programming for a whole day. Their programming work was guided and data was recorded such as; their look and attitudes in relation to their achievements, well-being, collaborative work, with post-activity tests. In the analysis of the data it was revealed that the behavior helps the relationship between willingness to learn, openness to teamwork, compliance - joy and learning achieved (Sharma et al., 2019).

In Taiwan researchers Jen and Hsu (2020), from National Taiwan University, in an investigation entitled "The impact of using mobile block-based programming to control robots with fifth grade students learning computational thinking in Singapore." The robots were configured in Chinese language to act with block-based programming. The experiment revealed that the students achieved significant improvement in both computational thinking learning and also in handling conditional sentences through the knowledge training tasks based on robot games (Jen and Hsu, 2020).

Finally, the authors Lee and Low (2020), implemented a computational thinking curriculum with robot programming tasks. Also, Cheng et al. (2018) worked on an investigation regarding the fundamental applications of the educational robot, through a requirements analysis, seen from the eyes of experts and researchers. On SRA (Sence, Reasoning, Action) coding the authors Fanchamps et al. (2020, 2021), worked regarding the implication of SRA coding on algorithmic thinking. In addition, the implications of using cell phone applications to command robots with elementary school students in school was investigated (Yi and Ting, 2020 in Kong et al., 2020). Finally, researchers Souza et al. (2019), studied the implication of computational thinking in mathematics through robotics.

2.2.3. Metaphors and blocks for teaching programming

Pérez et al. (2018), propose using metaphors, such as; pantry/memory and boxes/variables. Thus, it shows the alternative of employing such metaphors to diverse resources that the teacher has access to. Four step-by-step guidelines are provided on the recommended way to use metaphors in class. In this sense, an opinion study was carried out among students, for which the opinions of 62 students from 9 to 11 years old and their teacher were collected. The study revealed that 65% of the respondents found metaphors useful. The children were proficient in understanding metaphors, and <10% of the children preferred direct language without using metaphors. In a conversation with him as a teacher, he reported, "I think it is correct, because the students are not only working with instructions in the recipes, but they are able to see themselves on the screen or on the board" (Pérez et al., 2018, 2020).

In another case, the AgentSheets program exchanged four alternatives to generate a block programming way. After initially locating on syntax alternatives, it has been tested with perspectives to go to the next step, semantics, to resolve meaning and practical conflicts. Three considerations are reported there: (1) Contextualized explanations to aid understanding, (2) Traditional programming to proactively encourage possible outcomes, and (3) Color palettes to make programming easier. The block programming community has been on the lookout for syntactic coverage of coding (Repenning, 2017).

2.2.4. Learning programming like learning a second language

Learning to code computers is undoubtedly a linguistic learning, since it is a way of communicating to the computer what to do and how to do it, and once it has finished how to elaborate the results and where to send them. Thus the lack of grammatical rules to get started in coding leaves students with a pedagogical fissure to achieve by their own means, although teachers expect to solve the exercises using concepts of logic mediated by a language in which many complicate him to say the elementary. So much so, that some more talented as to get their codings to work correctly stagnate in their learning during their first year. The exact complications of learning a programming language are similar with the complexity of speaking a second language (Portnoff, 2018).

Portnoff (2018) says that the 10 years he has been a teacher in an introductory computer programming course, he has used a diversity of programming languages, such as; Scratch, Alice, minecraft and others. However, the promising significance of software tools, what everyone remembered was the explicit language model. In recent years, it has become increasingly clear that implicit language strategies are critical. This is not to say that the explicit language model is ineffective: there are evidently talented learners who go on to train as expert programmers (Portnoff, 2018).

2.2.5. Other didactic initiatives in computer programming

In Germany in 2018, the book "Content and Skills of Computer Science" was published by the University Press of the University of Potsdam. In the first paper that is part of the publication, entitled "What everyone should learn about computer science: an analysis of University courses for students of other disciplines," professors Stefan Seegerer and Ralf Romeike from Friedrich-Alexander University propose a didactic way to be prepared for life in the digital society, this is not only because computer science is present in our lives in the productive area and in education, but also in all aspects of our daily lives. In their work they analyzed 70 modules on computer education for students from other disciplines; syllabi and lesson plans, identifying some key issues and types of tools used (Bergner et al., 2018).

2.2.6. Pedagogical models and didactics

For the purposes of this work, didactics is a science, so it has an object of study, since its purpose is to transpose academic knowledge into sufficient actions and nurtured holistic components coming from teachers, so that it makes sense to learners and allows students to appropriate new knowledge. There are didactics generated by areas and sub-areas of subjects; for example, there is a didactics of language, a didactics of geometry, a didactics of algebra, a didactics of chemistry, a didactics of physics, etc., Thus, what best generalizes didactics is what Shulman proposes about the didactic knowledge of content, as a methodology capable of making the teacher an architect of himself in his creations and practices rich in ways of teaching that allow students to move toward new knowledge in a pleasant, motivated and participatory way. Thus, the teaching of computer programming, as an emerging area still navigates on some postulates of didactics that have not been experimented and proof of this is that those who teach programming, are disciplinary professionals without pedagogical studies (Belmar, 2022).

For Ortiz et al. (2015) and his team, all researchers from the University of Magdalena in Colombia, published a research entitled "Pedagogical models from a psychological-spiritual dimension," in which they analyze pedagogical models, making a brief description of the behaviorist pedagogical model (Skinner), the constructivist pedagogical model (Piaget) and the sociohistorical-cultural theory (Vygotsky) as a pedagogical model. In addition, various holistic and ecological pedagogical proposals derived from new epistemologies are analyzed, such as Maturana's bio-pedagogical model, which proposes a biology of love as the ontological and epistemological basis of pedagogy. Every educational process has a method, an axis, which directs and contributes to the development of the exercise of training, which we call pedagogical model (Ortiz et al., 2015).

Precisely, pedagogy is a science whose boundaries show its dynamics between society and human thought. A pedagogical model is a theoretical and practical plan of strategies that teachers and educational institutions possess to develop the training process of their students. The pedagogical model is characterized by the articulation of notions such as: curriculum, pedagogy, didactics, training, education, teaching, learning and evaluation; but it also contributes to the configuration of practice and theory. For the configuration and identification of a pedagogical model it is important, according to Coll (1991 in Ortiz et al., 2015), to answer five essential questions: why teach, what to teach, how to teach, when to teach, and what, how and when to evaluate (Ortiz et al., 2015).

For his part, Zipitría (2018), a research scholar at the Computer Science Institute of the University of the Republic of Montevideo in Uruguay published a research entitled "Piaget and computational thinking," a perspective that points in the same direction of the construction of computational thinking skills, both as a cognitive skill and the means by which this skill is built, which is computational programming. The publication discusses the concept of "computational thinking." Computational thinking is in a higher category than algorithmic thinking, however, the latter is the key piece from where computational thinking skills begin to be acquired, since the way of thinking when solving algorithmic problems and representing their solutions as algorithms, are characteristic of Computer Science (Zipitría, 2018).

The most relevant aspects of Piaget's theory is to look at and analyze the construction of knowledge as a process and to unveil how the transition from a lower level of knowledge to a more complex level takes place. In this expanded sense, the learning of computational thinking can be explained. The research methodology is supported by Piaget's application of Piaget to have students solve problems by sorting, counting and searching elements of data and reflecting on the method they employ and the reasons for their success (or failure), as a first step toward the conceptualization of algorithms and data structures. From this perspective, the author developed the extension of Piaget's postulates as the need to describe cases where the subject must instruct an action to a computer (Zipitría, 2018).

Researchers from Japan, investigated regarding pedagogical transformation based on student-led design and computational thinking. In current times that technology is everywhere it is appropriate to discuss transformative teaching where technology is part of who we are. We do not believe, say the authors, that we have any basis for claiming that the unification of technology happens by itself, for students performing complex robot coding activities, it was the redesign of didactics and learning by doing that made the real difference. The authors' current thesis points out that the idea of unification of information technologies, is slowly disappearing, and in its replacement is gradually integrating computational thinking with its associated skills (Vallance and Towndrow, 2016).

Grgurina's (2008) work is oriented toward the teaching of ICT (Information and Communication Technology), and develops experiences in the teaching of computing at a general level without pointing out didactics as the basis of teaching action to achieve student learning, and does not contain a didactic formulation of the teaching of computer programming in initial teacher training. It should be noted that Grgurina (2021), 13 years later, describes computational thinking in terms of its main concepts, such as: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, simulation, and parallelization (Grgurina, 2021).

Also, Chinese researchers inquired about a mixed didactic method and fostering innovative aptitude, whose methodology of action is based on computational thinking. Computational thinking incorporates computational thinking and unification with the natural environment, which considers progressive thinking in computing environments. In the future, non-computer professionals can use computing to create and shape new ideas for multiple professions. In addition, they would be able to support other areas of research into new electronic equipment and software. Computational thinking can significantly collaborate diverse non-computing professionals to bridge the gap between knowledge construction and tool creation (Zhang et al., 2019).

Finally, the teaching of computer programming requires the use of computers, however, when there are no resources for it, it is possible to work in an unplugged way, which in reality is learning through a series of board games that seek to replace the use of a computer (plugged) in the teaching of programming. Thus, for example, the assignment of a variable is represented by the movement of a figure on a board, the conditional if is also represented by some different figure, and so on for the different operations and control statements of a programming language, such as the conditional *while loop {condition}*, the *repeat until {condition}* or the unconditional *for loop*. Once novices have achieved some learning in a unplagged way (without computer) then they are already moderately prepared to start learning programming on the computer, i.e., plugged (Grgurina, 2021).

3. Materials and methodology

3.1. Materials

The materials to carry out the present work consider having an office equipped with a desk with a personal computer with Internet connection, which is at least equipped with Office software and SPSS software (Statistical Package for the Social Sciences).

3.2. Methodology

The methodology of the study corresponds to a quasi-experimental design with pretest, posttest, and control group, and comprises a didactic intervention in 12 classes of Unit III of the subject "Introduction to Programming." The test data collection, data analysis and presentation of the results. The test consists of 45 multiple-choice questions, scored 0 or 1, depending on whether the answer is correct or incorrect, is attached in Supplementary Annex 1. The test was taken during class time, so it was carried out synchronously. See results of the pre- and post-test in Supplementary Annex 2.

3.3. Design guidelines and principles

- Objective: The objective is to determine the effects of applying the didactic principles of Brown and Wilson (2018) on the teaching-learning process of Python programming in computer science students of INACAP (National Professional Training Institute) in Chile.
- Construct definition: Programming in Python, implies the ability to solve problems based on computer science concepts and using the logical syntax of the programming language: basic sequences, loops, iteration, conditionals, functions, variables, and data structures such as arrays, tuples and dictionaries.
- Population: In the year 2021, 32,802 first year students of the computer science degree program at INACAP.
- Sample: The experiment was carried out with 100 students of the course "Introduction to Programming" in professional technical education, INACAP Ñuñoa branch.
- Type of test: multiple-choice test of 45 items with four answer options (only one correct) (Belmar, 2023).
- Estimated completion time: 90 min.

Didactics: Ten principles for teaching programming by Brown and Wilson.

Principle 1: Remember that there is no such thing as a programming gene. The first and most important tip is that there is no innate knowledge in computer programming. Computer programming skills are not innate, but rather a learned skill that can be acquired and improved with practice.

Principle 2: Use Peer Support. One-on-one tutoring is perhaps the ideal form of teaching; a teacher's full attention can be focused on one student, and they can fully customize their teaching for that person and tailor individual feedback and corrections based on a two-way dialogue with them.

Principle 3: Use live coding. Instead of using slides, instructors should create programs in front of their students.

Principle 4: Have students make predictions. When instructors use live coding, they usually run the program several times during its development to show what it does. The key to making demonstrations more effective is to have students predict the outcome of the demonstration before running it.

Principle 5: Use pair programming. Pair programming is a software development practice in which two programmers share a computer.

Principle 6: Use solved examples with labeled subgoals. Learning to program involves learning the syntax and semantics of a programming language, but it also involves learning to build programs. A good way to guide students in building programs is to use solved examples: step-by-step guides that show how to solve an existing problem.

Principle 7: Stick to one language. A principle that applies in all areas of education is that transfer only comes with mastery of a programming language. Therefore, courses should stick to one language until students have made enough progress with it to be able to distinguish the forest from the trees.

Principle 8: Use authentic tasks. Learners find authentic tasks more engaging than abstract examples. One caveat about the choice of context is that the chosen topic may inadvertently exclude some people while appealing to others.

Principle 9: Remember that novices are not experts. This principle is tautological, but it is easily forgotten. Novices program differently from experts and need different approaches or tools.

Principle 10: Don't just code. Design before coding.

Finally, for details on the 10 principles, go to the source found at Brown and Wilson's publication (Brown and Wilson, 2018).

4. Discussion

4.1. Global context of computer programming education

The teaching of computer programming involves resources ranging from the implementation of digital infrastructure (computers and software), the availability of sufficient human resources to cover the entire primary and secondary education, to the lack of didactics to guide the delivery of computer programming knowledge. In this context, the questions arise: what to teach and at what level? Which programming languages to teach; Python, Scratch, Alice or Java? In which subjects to incorporate programming? Only to teach in the STEM area or to extend to the humanistic area? In addition, there is the aspect of how to measure learning, how to know that what is done pays off in new learning for students and that this learning truly constitutes the acquisition of computational thinking skills (Belmar, 2022).

At the 21st edition of the "International Conference on Interactive Collaborative Learning" and the "47th edition of the International Conference on Engineering-Pedagogy" held in 2018 at the "Aristotle University of Thessaloniki of Greece," they published a text with the papers presented entitled "The Challenges of the Digital Transformation in Education," in which among the many papers presented, a section on research conducted on preschool, primary and secondary education stands out. Some of the titles observed are: "Cyber and Internet Module Using Python in Junior-High School," "Children's Reflection-in-Action During Collaborative Design-Based Learning," "Internet Addiction and Anxiety Among Greek Adolescents: An Online Survey," "Intelligent Robotics in High School: An Educational Paradigm for the Industry 4.0 Era," "Design and Use of Digitally Controlled Electric Motors for Purpose of Engineering Education," among others. It should be noted that, from the observed titles, didactic teaching strategies do not appear and neither are observed validations of tests to measure learning (Auer and Tsiatsos, 2019).

For its part, UNESCO proposes a master plan for the development of digital skills, in which in one of the sections it highlights some important points in which it points out that the plan should have described aspects on the technological infrastructure in the school, as a necessary prerequisite, teachers trained in the area of digital technologies, and the integration of digital technology within the curriculum, not only in specific courses but within the goals or objectives (Fau and Moreau, 2018). These aspects must be supported by public policies that guide in teaching methodologies and didactics, beyond providing digital infrastructure and human resources. It should be noted that implementing the teaching of computational thinking is an enormous task, which should start in universities by preparing teachers for such a gigantic task, who after the curricular policies have been dictated by governments and the economic resources have been allocated, will be able to implement the new teaching in schools (Law et al., 2018).

This is complemented by the European Commission, which published in 2016 a paper entitled "Developing Computational Thinking in Compulsory Education," in which it makes various diagnoses and the state of progress in this area in the member countries. The research points out, among other things, that the states have the obligation to train the next generations to operate the new digitalized world that is approaching, however, the states that have implemented the changes in education have revealed the lack of teachers trained in Computer Science, and where they have them, although few, there is the problem that there are no pedagogical and didactic models for teaching the new skills. The report analyzes the most significant of the development of computational thinking for formal education in Europe and in sum shows the implications in the classroom and in the academy (Bocconi et al., 2016).

4.2. Results

A statistical test that validates the analysis of the results is the repeated measures factorial ANOVA (pre- and post-test), with the pre- and post-test treatment as an intra-subject factor and teaching method (traditional and innovated) as between-group factors. The pre- and post-test, being evaluated in the same subjects, is a repeated measure so it should be treated as an intra-subject factor, since these observations are not independent. Statistical analysis was performed with SPSS (Statistical Package for the Social Sciences) software version 29.0.1.1 under license from IBM (International Business Machines).

The repeated measures ANOVA procedure performs the analysis of groups of related dependent variables corresponding to different measurements of the same property at different times. In such an experimental design, the dependent variables correspond to measurements of more than one variable for different levels of within-subjects factors. For example, the score achievement and the time taken to respond may have been measured for each subject at two or three different times. The repeated measures ANOVA procedure provides multivariate analyses for repeated measures data. In addition to testing hypotheses, repeated measures ANOVA generates parameter estimates (Camacho, 2019).

In the tables below it can be seen that the tests of inter-subject effects give us a significance of 0.01, a value that is <0.05, a threshold that confirms that the results are statistically significant. In addition, the multivariate tests ratify the significance of the experiment and show a statistical power of 100%, that is, an observed power of 1.00. As shown in Table 1.

According to Camacho (2019) the statistics for multivariate tests with repeated factors are mainly; the Pillai trace which is a test statistic produced by a Multivariate Analysis of Variance (MANOVA), which is a value ranging from 0 to 1. The meaning is that when the Pillai trace approaches 1, the more significant is the evidence that the explanatory variable has a statistically significant effect on the values of the response variables. Another indicator is Wilks' Lambda which is the product of the unexplained variances in each of the discriminant variants, and the lower its value, the greater the disparity between the groups being compared, and the greater Roy's root which corresponds to the increasing values of the statistic indicating increasing contributions of the effects to the model in question. For the case under study, the Pillai trace is 0.623 value closer to 1, thus showing that the explanatory variable has a significant effect on the values of the response variables, which is the score obtained by the students, in addition the Wilks' Lambda index resulted in 0, 377, which the lower its value, the greater the disparity between the groups being compared, i.e., it makes a significant difference between the control group with the experimental group, and finally the greater Roy's root (1.654) which indicates increasing contributions of the effects to the experiment. See table Multivariate tests. As shown in Table 2.

In addition, we have the graph of marginal measures, which compares the average scores obtained in each test, both for the control group and for the experimental group, in which we can appreciate and ratify what the table of statistical indexes indicates; means and variances with their maximum and minimum scores. Here it is shown that the control and experimental group that started with similar averages in the pretest, in the experimental group is slightly higher than the control group, but in the post-test, the experimental group is observed to be much higher than the control group, which makes it clear that the hypothesis is confirmed.

Finally, when observing the ANOVA table, it is seen that for the pretest score there is no significant difference between means between the control group and the experimental group since $p = 0.541$, a value >0.05, so the researcher's hypothesis is rejected, which is evident, since the new teaching method has not yet been applied, However, at the post-test level, $p = 0.001$, a value much lower than 0.05, so the researcher's hypothesis is accepted that the students who receive the teaching with the Brown and Wilson methodology have higher achievements than the students of the control group who receive the teaching in the traditional way.

5. Conclusions

Undoubtedly, the teaching of computer programming has become an essential skill for today's society in constant technological evolution. Programming not only involves learning a new language, but also developing problem-solving skills, logical thinking and creativity. That's why programming is becoming an essential skill in every field, from data science to web development to artificial intelligence. One of the best ways to teach programming is through the use of hands-on examples and projects. Students learn best when they are involved in hands-on projects and applications, and this helps them understand how programming can be applied in real-world situations. In addition, projects also allow them to work in teams and collaborate on solutions, important skills for any career in technology.

It is worth remembering that the teaching of programming aims to develop computational thinking skills in children and young people, which has become an essential skill for today's digital society, as it involves problem-solving and logical thinking skills that are necessary for the development of innovative technologies and solutions. Computational thinking can also help people

TABLE 1 Inter-subject effects test.

Origen	Sum of squares	df	Root mean square	*F*	Sig.	Partial Eta squared	Non-centrality parameter	Observed power
Intersection	105,248.40	1	105,248.4	3,147.288	<0.001	0.972	3,147.288	1.000
Group	614.20	1	614.203	18.367	<0.001	0.171	18.367	0.989
Error	2,976.24	89	33.441					

Has been calculated using alpha = 0.05.

TABLE 2 Multivariate tests.

	Value	*F*	Df of hypothesis	Error de	Sig.	Partial Eta squared	Non-centrality parameter	Observed power
Pillai trace	0.623	147.2	1.000	89.000	<0.001	0.623	147.203	1.000
Wilks lambda	0.377	147.2	1.000	89.000	<0.001	0.623	147.203	1.000
Hotelling trace	1.654	147.2	1.000	89.000	<0.001	0.623	147.203	1.000
Roy's mayor root	1.654	147.2	1.000	89.000	<0.001	0.623	147.203	1.000

Each F-tests the effect of TEST. These tests are based on linearly independent pairwise comparisons between the estimated marginal measures.
Has been calculated using alpha = 0.05.

understand how technologies work and how they can use them to improve their daily lives. Countries that have implemented computational thinking in their educational programs have a competitive advantage in the digital age, as their citizens have advanced skills and knowledge in technology and programming. These countries can develop advanced technological solutions and be at the forefront of innovation, allowing them to be more competitive in the global marketplace. In addition, computational thinking can also have a positive impact on a country's economy and employment. Jobs in technology and programming are in high demand around the world, and companies are increasingly looking for employees with advanced programming and technology skills.

In the study that is the subject of this report, there was undoubtedly an improvement in the students' learning, which is ratified in the increase of the mean and average, but even more significant was that the experimental group that was dispersed in the results tended to become more uniform in their learning, as shown by the results of the post-test. There is a positive effect of greater camaraderie and collaboration, which explains the certain uniformity of the post-test results of the experimental group, which did not occur in the control group, where the dispersion of results increased significantly. In the following graph, the pretest of the control group (PRETEST-COTROL) corresponds to the first scheme, followed by the pretest of the experimental group (PRETEST-EXPERIMENTAL), and in third position is the post-test of the control group (POSTEST-CONTROL) in which the increase in dispersion can be seen, Finally, further to the right is the scheme of the experimental group (POSTEST-EXPERIMENTAL), in which a greater uniformity in the results is observed.

Table 3, which shows the indicators of the control and experimental groups, clearly shows that the experimental group achieved better results than the control group, which are more significant in terms of the minimum value observed in the post-test and the uniformity of achievements, which tend to level out, which means that all students learned, approaching the total achievement score, however, none managed to reach 100% of the test.

Undoubtedly, having experimented with the didactic principles of Brown and Wilson, the lack of didactic tools for teaching programming and the lack of instruments to measure the skills achieved by the students are discussed. This report intends to be a pioneer contribution in this matter, and as the culmination of a project that took 2 and a half years of work, in which works from all over the world were analyzed, from the most diverse countries, such as China, South Korea, Japan, several European countries, Brazil and Argentina, among others, which allowed placing computational thinking as an essential skill for the 21st century, and which is projected as one of the educational tasks with the greatest impact in this century that is beginning.

6. Projections and limitations of the study

6.1. Limitations

The limitations of the study are in the experimental part in which it would be convenient to include a larger sample and from several institutions, which could show in a reliable way the goodness of the didactic principles of Brown and Wilson for the teaching of computer programming. The research worked with one course as the experimental group and another course as the control group, and each course with different professors, which could already pose a possible bias in the application of the methodology. One way in which the Brown and Wilson methodology could be applied that would allow eliminating one of the teacher biases would be to have the same teacher teach classes in the experimental group and in the control group, which could be replicated with several pairs of courses as control and experimental with the same teacher.

The general objective was to determine the effects of applying the didactic principles of Brown and Wilson (2018) on the teaching—learning process of programming in Python

TABLE 3 Indicators and results.

Index / Group	Pretest control	Pretest experimental	Post-test control	Post-test experimental
Median	16.5	16	31	34
Average	17.02	17.80	27.62	34.22
Minimum	8	8	9	28
Maximum	27	39	39	42
Standard Deviation	4.98	7.19	9.10	3.90
Variance	28.80	51.7	82.81	15.21
Confidence interval (95%)	15.60–18.44	15.53–20.08	25.04–30.2	32.99–35.45

Source: Own elaboration.

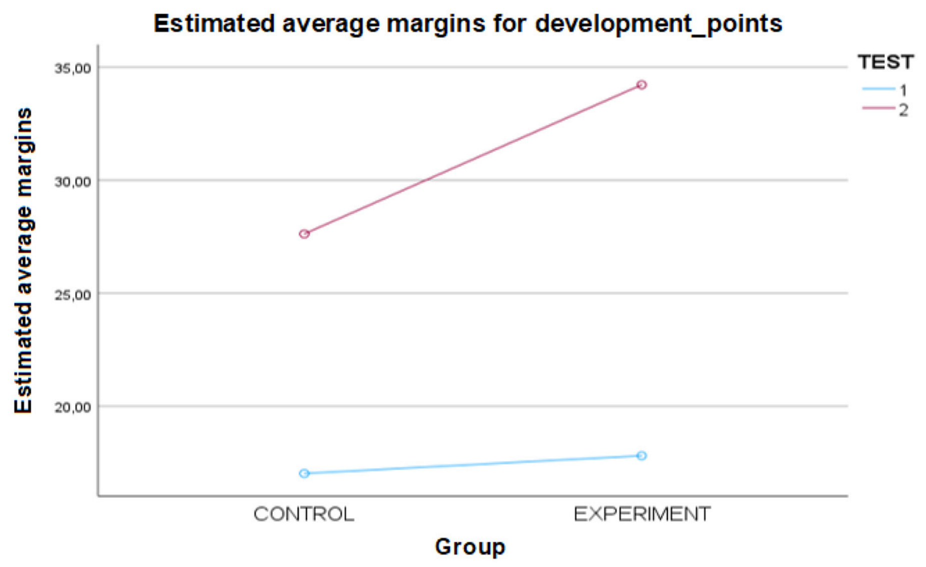

FIGURE 1
Estimated marginal averages by group.

with students of computer science careers of INACAP, an objective that is observed to be fully met, since the results were satisfactory, and project new scenarios to conduct new studies where the set of didactic principles of Brown and Wilson can be tested again (see Figure 1).

6.2. Projections

In the future there should be many investigations that test different didactic strategies for teaching programming. It should be noted that the didactics for primary and secondary students should be different from the one applied in tertiary education, since the latter works with adult students who have a specific purpose for which they carry out the study and not general as when they are in primary and secondary education. Thus emerges the concept of andragogy, which is the analog of pedagogy, but for adult education. Moreover, in the future it will not only be necessary to evaluate algorithms with conceptual or quantitative results, but it will be necessary to evaluate actions such as those performed by a robot or an automated machine (Carrillo, 2018).

The educational systems of the future, should incorporate not only the teaching of computational thinking, but also elements of neuroscience, so that once the basis of knowledge in computer programming, educational robotics and gamification is laid, it can go further, making the connection between computational processes and brain processes, in order to understand and create applications that are able to take advantage of brain waves in the activation of electronic devices that perform actions as an extension of the body, as there are already developments in the war industry, but with peaceful motivations and focused on the physical disabilities of human beings and also on brain disabilities in order to correct diseases such as deafness, blindness or go beyond, correcting Alzheimer's disease.

The teaching of computational thinking should permeate all knowledge formation in the educational system, integrating STEM and non-STEM areas or better known as STEAM. Currently there are lost subjects such as technology education in Latin American countries when students are made to do crafts related to handicrafts or issues that contribute very little to the formation of knowledge. The subject of technology education should change its content from craft to technology, and should be composed of electronics, integrating electronic devices with the management of programs that allow students to be true generators of new digital technologies.

The formation of knowledge in schools should cut across all areas of knowledge, starting with mathematics and natural sciences, where one could experiment with the creation of virtual reality and/or augmented reality applications for chemistry and biology, passing through history and geography where one could teach through the creation of games and stories located in certain territories and eras, such as the Age of Empires game where different versions of the game show the ancient civilizations of Europe until about 1,700, and could create applications for schoolchildren on other continents, such as Africa, Latin America, Asia and Oceania. In language, tales and stories could be recreated to give life to letters, and in art, it could be integrated with virtual reality developments, so that students can navigate within technology, building the different educational knowledge, all of which would be done in a progressive and interactive way, training students in programming and gradually incorporating the skills of computational thinking.

Some of the computer science topics that could be studied and evaluated in the future would be computer security, computer programming in education, disconnected activities in learning computational thinking, didactic strategies in the teaching and learning process of programming, internet of things, data science and big data, artificial intelligence, students with special needs and studies on psychological aspects of technology and humans, in addition to deepening on gamification and educational robotics in primary school and robotics and industrial automation in secondary school, so that learning by projects, learning by doing, is practiced. It should be noted that all this should be properly distributed in the 12 years of primary and secondary education.

In addition to the above, there are emerging technologies such as nanotechnology and quantum computing, topics that should be part of the educational system in research phases and emerging technology topics. It should be noted that companies such as IBM already have prototypes of quantum computers that are fully operational. It is important to keep in mind that a quantum computer can decode all existing computer security systems in the world in just minutes, which will leave governments and organizations around the world unprotected. This happens because of the processing speed of this type of computers, since they are based on the parallelism of the binary system. While in the current electronic system, the bits (1 and 0) manifest themselves sequentially, in the quantum system they do so in parallel.

Finally, I would like to make a reflection on the difference in priorities between countries, which occurs on multiple levels; cultural, economic, technological, political and social, which leads us to think, where should we start from, should we promote the teaching of computational thinking as a way to transfer knowledge and thus in the medium and long term countries solve their social problems, or should we categorize where to start from, in addition to the fact that each country is independent. For example, in Haiti in Central America, with more than 200 years of independence, they still have not managed to have a stable political system that allows them to overcome extreme poverty and hunger. In this situation there are several countries in Africa and Latin America, while the world, represented by the developed countries, is faced with the dilemma of climate change, which is just around the corner and which will affect us all. I firmly believe that, in order of priority, the teaching of computational thinking is second only to climate change and world hunger.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material, further inquiries can be directed to the corresponding author.

Ethics statement

The studies involving human participants were reviewed and approved by Institutional Ethics Committee - Universidad de Santiago de Chile. The patients/participants provided their written informed consent to participate in this study.

Author contributions

The author confirms being the sole contributor of this work and has approved it for publication.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2023.1085507/full#supplementary-material

References

Araujo, A. L. S. O., Andrade, W. L., Guerrero, D. D. S., and Melo, M. R. A. (2019). "How many abilities can we measure in computational thinking? A study on Bebras challenge," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Universidad Federal de Paraíba, Paraíba, Brasil), 545–551. doi: 10.1145/3287324.3287405

Auer, M. E., and Tsiatsos, T. (eds) (2019). "The challenges of the digital transformation in education," in *Proceedings of the 21st International Conference on Interactive Collaborative Learning (ICL2018)-Volume 1* (Vol. 916) (Cham: Springer), 215–226. doi: 10.1007/978-3-030-11935-5

Belmar, H. M. (2022), Review on the teaching of programming and computational thinking in the world. *Front. Comput. Sci.* 4, 997222. doi: 10.3389/fcomp.2022.997222

Belmar, H. M. (2023), Expert validation of a Python test, reliability, difficulty and discrimination indices. *J. Educ. Dev.* 7, 52. doi: 10.20849/jed.v7i1.1320

Bennedsen, J., and Caspersen, M. (2007). Failure rates in introductory programming. *AcM SIGcSE Bull.* 39, 32–36. doi: 10.1145/1272848.1272879

Bergner, N., Röpke, R., Schroeder, U., and Krömker, D. (2018). *Hochschuldidaktik der Informatik HDI.* Berlin: University Press.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). *Developing Computational Thinking in compulsory education – implications for policy and practice*; EUR 28295 EN. Luxembourg: Publications Office of the European Union. doi: 10.2791/792158

Brown, N., and Wilson, G. (2018). Ten quick tips for teaching programming. *PLoS Comput. Biol.* 14, e1006023. doi: 10.1371/journal.pcbi.1006023

Camacho, C. (2019). *Análisis de la Varianza Para Medidas Repetidas.* Sevilla: Universidad de Sevilla.

Carrillo, F. (2018). *Formación de Competencias para el Trabajo en Chile.* Tech. Rep., Comisión Nacional de Productividad. Universidad de Chile, Santiago, Chile.

Cheng, Y., Sun, P., and Chen, N. (2018). The essential applications of educational robot: requirement analysis from the perspectives of experts, researchers and instructors. *Comput. Educ.* 126, 399–416. doi: 10.1016/j.compedu.2018.07.020

Coll, C. (1991). *Reforma Al Sistema Educativo Español, la.* Libresa. Universidad de Chile, Santiago, Chile.

Dagiene, V., and Stupuriene, G. (2016). Bebras–a sustainable community building model for the concept based learning of informatics and computational thinking. *Inform. Educ.* 15, 25–44. doi: 10.15388/infedu.2016.02

de Paula, B., Burn, A., Noss, R., and Valente, J. (2018). Playing Beowulf: Bridging computational thinking, arts and literature through game-making. *Int. J. Child-Comput. Interact.* 16, 39–46. doi: 10.1016/j.ijcci.2017.11.003

Fanchamps, N., Slangen, L., Hennissen, P., and Specht, M. (2021). The influence of SRA programming on algorithmic thinking and self-efficacy using Lego robotics in two types of instruction. *Int. J. Technol. Des. Educ.* 31, 203–222. doi: 10.1007/s10798-019-09559-9

Fanchamps, N., Specht, M., Hennissen, P., and Slangen, L. (2020). *The Effect of Teacher Interventions and SRA Robot Programming on the Development of Computational Thinking.* Fontys University of Applied Science, Eindhoven, Netherlands.

Fau, S., and Moreau, Y. (2018). *Managing Tomorrow's Digital Skills-What Conclusions can we Draw from International Comparative Indicators?* Education 2030 – UNESCO. Ministerio de Educación, Lima, Perú.

García, F. (2018). Editorial computational thinking. *IEEE Ibero-Am. J. Learn. Technol.* 13, 17–19. doi: 10.1109/RITA.2018.2809939

Grgurina, N. (2008). "Computer science teacher training at the University of Groningen," in *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (Berlin: Springer), 272–281. doi: 10.1007/978-3-540-69924-8_25

Grgurina, N. (2021). *Getting the Picture: Modeling and Simulation in Secondary Computer Science Education.* Naples: University Press.

Huang, P., and Hwang, Y. (2013). An exploration of EFL learners' anxiety and e-learning environments. *J. Lang. Teach. Res.* 4, 27–35. doi: 10.4304/jltr.4.1.27-35

Jen, T., and Hsu, T. (2020). *The Impact of Using Mobile Block-based Programming to Control Robots on the Performance of the Fifth Grader Students Learning Computational Thinking in Singapore.* National Taiwan Normal University, Taipei, Taiwan.

Juškevičiene, A., and Dagiene, V. (2018). Computational thinking relationship with digital competence. *Infor. Educ.* 17, 265–284. doi: 10.15388/infedu.2018.14

Kale, U., and Yuan, J. (2021). Still a new kid on the block? Computational thinking as problem solving in Code. org. *J. Educ. Comput. Res.* 59, 620–644. doi: 10.1177/0735633120972050

Kong, S., Hoppe, H., Hsu, T., Huang, R., Kuo, B., Li, K., et al. (eds) (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Kulkarni, V. (2015). *Looking Back: Alan Turing-The Father of Computer Science.* CSI Communications, India.

Law, N., Woo, D., de la Torre, J., and Wong, G. (2018). *A Global Framework of Reference on Digital Literacy Skills for Indicator 4.4. 2.* Montreal, QC: UNESCO - Institute for Statistics.

Lee, P., and Low, C. (2020). Implementing a Computational Thinking Curriculum with Robotic Coding Activities through Non-formal Learning. *CoolThink@ JC*, 150. Bukit View Secondary School, Singapore.

McLaren, B., Adams, D., Mayer, R., and Forlizzi, J. (2017). A computer-based game that promotes mathematics learning more than a conventional approach. *Int. J. Game-Based Learn.* 7, 36–56. doi: 10.4018/IJGBL.2017010103

Mühling, A., Ruf, A., and Hubwieser, P. (2015). "Design and first results of a psychometric test for measuring basic programming abilities," in *Proceedings of the Workshop in Primary and Secondary Computing Education,* TUM School Universität München, Germany. 2–10. doi: 10.1145/2818314.2818320

Ortiz, A., Sánchez, J., and Sánchez, I. (2015). Los modelos pedagógicos desde una dimensión psicológica-espiritual. *Rev. Cient. Gen. José María Córdova* 13, 183–194. doi: 10.21830/19006586.22

Pérez, D., Hijón, R., Bacelo, A., and Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?. *Comput. Hum. Behav.* 105, 105849. doi: 10.1016/j.chb.2018.12.027

Pérez, D., Hijón, R., and Martín, M. (2018). A methodology proposal based on metaphors to teach programming to children. *IEEE Rev. Iberoam. Tecnol. Aprendiz.* 13, 46–53. doi: 10.1109/RITA.2018.2809944

Portnoff, S. (2018). The introductory computer programming course is first and foremost a language course. *ACM Inroads* 9, 34–52. doi: 10.1145/3152433

Psycharis, S., Kalovrektis, K., and Xenakis, A. (2020). A conceptual framework for computational pedagogy in STEAM education: determinants and perspectives. *Hell. J. STEM Educ.* 1, 17–32. doi: 10.51724/hjstemed.v1i1.4

Repenning, A. (2017). Moving beyond syntax: lessons from 20 years of blocks programing in AgentSheets. *J. Vis. Lang. Sentient Syst.* 3, 68–91. doi: 10.18293/VLSS2017-010

Rojas, A., and Garcia, F. J. (2020). Evaluation of computational thinking for learning computer programming in higher education. *Distance Education Magazine (RED)*, 20. Technological University of Puebla Puebla, México.

Sharma, K., Papavlasopoulou, S., and Giannakos, M. (2019). Coding games and robots to enhance computational thinking: how collaboration and engagement moderate children's attitudes? *Int. J. Child-Comput. Interact.* 21, 65–76. doi: 10.1016/j.ijcci.2019.04.004

Souza, I., Andrade, W., and Sampaio, M. (2019). "Analyzing the effect of computational thinking on mathematics through educational robotics," in *2019 IEEE Frontiers in Education Conference (FIE)* (Covington, KY: IEEE), 1–7.

Vallance, M., and Towndrow, P. (2016). Pedagogic transformation, student-directed design and computational thinking. *Pedagog. Int. J.* 11, 218–234. doi: 10.1080/1554480X.2016.1182437

Watson, C., and Li, F. (2014). "Failure rates in introductory programming revisited," in *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education* (Durham University Library, United Kingdom), 39–44. doi: 10.1145/2591708.2591749

Wing, J. (2006). Computational thinking. *Commun. ACM* 49, 33–35. doi: 10.1145/1118178.1118215

Yi, L., and Ting, H. (2020). "Effects of using mobile phone programs to control educational robots on the programming self-efficacy of the third grade students," in *Proceedings of International Conference on Computational Thinking Education* (National Taiwan Normal University, Taiwan), 31–35.

Zhang, K., Chen, X., and Wang, H. (2019). "Research on the mixed-learning model and the innovative talent cultivation mechanism based on computational thinking," in *Recent Developments in Intelligent Computing, Communication and Devices* (Singapore: Springer), 59–65. doi: 10.1007/978-981-10-8944-2_8

Zhou, T., and Hsu, T. (2020). *Learning Behaviors Analysis of the Six Grader Students Integrating Educational Robots with the Computational Thinking Board Game.* National Taiwan Normal University, Taipei, Taiwan.

Zipitría, S. (2018). "Piaget and computational thinking," in *Proceedings of the 7th Computer Science Education Research Conference* (Universidad de la República Montevideo, Uruguay), 44–50. doi: 10.1145/3289406.3289412