Education Tech Research Dev
https://doi.org/10.1007/s11423-024-10406-5

DEVELOPMENT ARTICLE

A computational thinking course for all preservice K-12 teachers: implementing the four pedagogies for developing computational thinking (4P4CT) framework

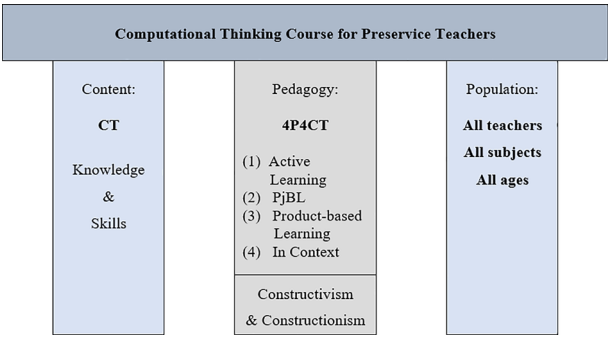
Noa Ragonis[1] · **Rinat B. Rosenberg-Kima**[2] · **Orit Hazzan**[2]

Accepted: 18 July 2024

Abstract

Computational thinking is accepted today as a collection of cognitive and social skills required for functioning in the 21st century. The paper presents a conceptual view at computational thinking that encompass concepts, problem-solving skills, application skills, and social skills. To impart those perceptions and skills the paper proposes the Four Pedagogies for Developing Computational Thinking (4P4CT) framework, which relies on active learning, project-based learning, product-based learning, and context-based learning, and advocates implementing computational thinking across *all* the education system in *all* subject matters at *all* ages by *all* teachers. The framework is presented and its implementation in an academic course for preservice K-12 teachers, taught so far in 16 classes attended by 409 preservice K-12 teachers, is described in detail. To support the effective development of the expected competences among preservice teachers, two types of empirical qualitative evidence, related to student outcomes, are presented: (a) simulations of computational processes, and (b) reflections that indicate a change in preservice teachers' perceptions and the application of computational thinking in their future teaching work.

Graphical abstract



Keywords Computational thinking · Preservice teachers · Teacher education · 4P4CT · 4-pedagogies: active learning, project-based learning, product-based learning, context-based learning

Extended author information available on the last page of the article

Introduction

While the educational community is engaged in the development of a computational thinking (CT) curriculum and its assimilation among students, the importance of teacher training as a key agent in the implementation of CT in the education system has also been raised (Hodhod et al., 2016; Yadav et al., 2017). Although the origins of CT are usually associated with the field of computer science (CS), it is widely acknowledged that CT skills are relevant for all (Lye & Koh, 2014; Wing, 2006, 2011, 2014). CT enables the development of cognitive and social skills required to function in the 21st century (Araujo et al., 2019; Barr et al., 2011; DeSchryver & Yadav, 2015; Günbatar, 2019) and can enhance the learning and teaching of all disciplines (Lee et al., 2020; Pollock et al., 2019; Seegerer & Romeike, 2018). This perspective is reflected in the growing number of countries that incorporate CT in a variety of forms, including elementary and middle schools and even kindergartens (Bocconi et al., 2016a, 2016b, 2022; CSTA, 2017; ISTE & CSTA, 2011; K-12 Computer Science Framework Steering Committee, 2016; Sabitzer et al., 2014). Nevertheless, despite the increasing acknowledgment that CT requires adequate teacher preparation programs, the literature on teacher training is limited, particularly with respect to academic preservice teacher preparation programs (Tang et al., 2020).

This paper presents the Four Pedagogies for Developing Computational Thinking (4P4CT) framework—a holistic approach for teaching CT that can be implemented in teacher preparation programs as well as in schools. Our working assumption is that, although scholars have not yet agreed upon an exact definition of CT, it is commonly agreed that all citizens should acquire CT skills. Accordingly, we suggest 4P4CT, which is geared towards implementing CT across the education system in *all disciplines, for all ages, and by all teachers*, thus promoting this message.

4P4CT is based on constructivism and constructionism learning theories, and it implements four pedagogies: active learning, project-based learning, product-based learning, and context-based learning. This paper describes how the 4P4CT framework is implemented in a course in an academic preservice teacher preparation program, in which it is expressed mainly, but not only, in the main course assignment, when teams of students develop a *computerized simulation* of a *computational process* taken from any discipline. In general, we suggest that when learners of any age implement, in a coding environment, a *computerized simulation* of a computational process taken from any topic they are studying, in any subject matter, they (a) improve their understanding of the specific simulated topic and (b) develop a variety of CT skills–cognitive, programming, and social (Ragonis, 2018; Hazzan et al., 2020).

As educators play a crucial role in developing students' CT skills, they should develop their own CT skills by experiencing this learning process—the development of a computerized simulation of a computational process taken from any discipline. Based on this working assumption, this paper presents an academic course focused on CT. The course aims to equip pre-service teachers, across all school subjects and age groups, with CT skills. Additionally, preliminary research findings are included to demonstrate the course's success in meeting its objectives. The course is offered to preservice K-12 teachers that is already being taught by the first author in different tracks of teacher preparation programs. The course develops preservice teachers in three areas: CT knowledge, CT skills, and technological pedagogy content knowledge (TPACK) (Koehler & Mishra, 2009), all of which contribute to their pedagogical studies and perspectives. The course described in this paper attempts to partially close the gap in relation to the integration of CT into

teacher preparation programs for non-CS teachers by its unique combination of (a) academic framework—an academic course that is part of a degree curriculum; (b) target audience—preservice teachers of all disciplines who are training to teach all ages; and (c) content—learning CT in the context of any field of knowledge.

Background

Computational thinking: an essential 21st century skill

In the last 15 years, since Wing (2006) initiated the discussion of the concept of CT, the CS education community has been widely involved in its assimilation into a variety of educational frameworks. It is now well agreed upon that students who are growing up in today's education system will live their adult lives in multi-tasking, multi-faceted, technology-driven, and highly diverse environments. Therefore, they must acquire the skills needed in this era, many of which are included in CT. In addition, several theoretical and practical foundations of CS are recognized today as essential knowledge for functioning in all domains of life (Boholano, 2017; Harper, 2018; Li et al., 2020a, 2020b), not necessarily in connection to CS or programming (Wing, 2006, 2011, 2014). In this light, CT is recognized nowadays as a set of useful skills, borrowed mainly but not exclusively, from CS, that can be applied in all disciplines (Barr et al., 2011; Cuny et al., 2010). CT is even interpreted as computational literacy, which leverages students' existing literacy skills to improve computational outcomes and conversely fosters students' literacy development through the practice of computing (Jacob & Warschauer, 2018a, 2018b). From this perspective, extending the teaching and learning of CT to literacy instruction increases the types and amounts of exposure to CT that students receive, which in turn better prepares them for success in the new economy.

CT focuses on problem solving but does not have a single definition; rather, a variety of perspectives and interpretations of CT exist. Some approaches connect CT with algorithms and computers (Barr et al., 2011; CSTA, 2017; Denning, 2009; Lye & Koh, 2014; Sabitzer et al., 2014), whereas others highlight the need to detach CT from technology and computers (Hu, 2011; Yadav et al., 2017). CT is also associated with creative thinking (De Schryver & Yadav 2015) and critical thinking (Korkmaz et al., 2017). The approach in which CT is rooted in various subject matter domains is illustrated, among other frameworks, as well as in academia. For example, the University of Delaware, Newark, decided to integrate CT into all areas of knowledge, and a pilot was carried out in three different disciplines: Sociology, Mathematics, and Music (Pollock et al., 2019).

Similar to several conceptual analyses of CT carried out in recent years, we see CT as an important mode of thinking not only in computer science but also in other STEM disciplines (Li et al., 2020a, 2020b). We further conceive of CT from a broader perspective as a set of thinking skills that can and should be applied in *any* discipline, including the humanities and social sciences. This aspect of CT was also addressed recently by Denning and Tedre (2022), who noted that basic CT does not teach us how professional computer scientists see the world; rather, it consists of a set of basic and fundamental ideas for learning central concepts in computing, as well as in other fields. Our emphasis on the application of CT skills in problem-solving processes in all fields of knowledge is also reflected in our discussion of data thinking, which encompasses data thinking in all fields of knowledge (Mike et al., 2022).

Although different views and opinions have been expressed with respect to CT since Papert (1980) planted the seeds into the concept, there seems to be consensus regarding several of its characteristics, with or without computers:

(1) CT is a type of problem-solving skill that involves the ability to design solutions to be implemented by either a person or computer.
(2) CT includes a set of cognitive skills, such as the decomposition of a problem into sub-problems, organization and logical analysis of data, and its representation using abstraction, design, and assessment of several solutions to a given problem, and the examination, implementation, and generalization of the chosen solution (Shute et al., 2017);
(3) Learning and developing CT enable the acquisition of a set of social skills, such as collaborative teamwork and time management, which can be applied in various contexts (Google, 2019; Günbatar, 2019; Hu, 2011; Wing, 2014).

The importance attributed to CT and the need to develop CT in the education system were highlighted by the OECD's inclusion of code-based CT skills in the PISA 2021 mathematics test (OECD, 2019). The importance of integrating CT across the board, and the important role of teachers in those processes were emphasized: "computer science and computational thinking, when taught well, can prepare students to apply problem-solving, creativity, and collaboration in all sorts of domains" (OECD, 2019, p. 1).

Acquiring computational thinking skills in education systems worldwide

The necessity of CT skills for effective citizenship today and certainly in the future is widely agreed upon in education systems worldwide. Furthermore, acquiring CT skills benefits students with low socioeconomic backgrounds, which may help close social gaps and promote social mobility (Bocconi et al.,). In this spirit, considerable work is being done to promote the acquisition of CT skills by learners of all ages, from kindergarten, through elementary, middle, and high schools, to academia.

Many countries incorporate CT into their school curricula, and differences can be seen in its place in the overall curriculum and its status as a mandatory or elective study subject. Two main approaches are typically applied to the implementation of CT in the curriculum. The first is more prevalent and involves introducing CT as a separate subject matter, usually as part of a CS/informatics/information and communication technologies (ICT) curriculum (e.g. US: CSTA, 2017; ISTE & CSTA, 2011; K-12 Computer Science Framework Steering Committee, 2016; in the EU: Bocconi et al., (2016a), (2016b), (2022) in the UK: The Royal Society, (2012); in Japan: Kanemune et al., (2017); and in New Zealand: Kellow, (2018). The second approach is to include CT in the context of STEM subjects, such as mathematics or biology lessons (e.g., Bocconi 2016a, 2016b, 2022; Lee et al., 2020; Peel et al., 2020; Sabitzer et al., 2014).

Some extracurricular programs and competitions are also based on the development and application of CT skills, such as Bebras, an international initiative aimed at promoting Informatics (CS) and CT among school students of all ages (Araujo et al., 2019); CodeMonkey, an educational game-based environment in which children learn to code without any prior programming experience (Israel-Fishelson & Hershkovitz, 2020); and Code.org, an educational nonprofit initiative dedicated to the vision that every student in every school

should have the opportunity to learn computer science as part of his or her core K-12 education (Lambić et al., 2021).

The assimilation of computational thinking in educational systems depends on various factors, including the adoption of new technologies and development of new ways of thinking. It was found that too few innovators and early adopters actually used new technology to enhance existing learning behaviors (Ebner et al., 2007). Respectively, new approaches for the Technology Acceptance Model (TAM) exist, and future directions are offered (Marangunić & Granić, 2015). The adaptation of advanced thinking skills, such as computational thinking, can be interpreted in the context of innovative thinking competencies, which refers to four personal and interpersonal skills (observing, questioning, idea networking, and experimenting) that influence the adoption and/or development of new approaches and technologies in educational settings (Morad et al., 2021a, 2021b).

Teacher training for teaching CT

The role of teachers in teaching CT is of great interest to the education community, and extensive work has been done, especially in the last five years (Armoni, 2019; Bocconi et al., 2016a, 2016b, 2022; DeSchryver & Yadav, 2015; Ragonis, 2018; Yadav et al., 2014a, 2014b; Yadav et al., 2017; Yasar & Veronesi, 2015). Since teachers are recognized as key agents for imparting CT skills (Hughes et al., 2017; Lamprou & Repenning, 2018; Seegerer & Romeike, 2018), the attention given to the development of teachers' CT skills has increased as well. This challenge has been addressed, for example, in Korea, where primary teachers' training is perceived as critical to the success of a new reform introduced to integrate CT, since elementary school teachers teach all subjects and IT/computer teachers are not part of the elementary school staff (Park, 2016).

As part of this effort, the perceptions and knowledge of preservice teachers and in-service teachers regarding CT were studied with the objective of improving the professional development programs offered (Cabrera, 2019; Chang & Peterson, 2018; Ung et al., 2022; Yang et al., 2018a, 2018b; Yilmaz et al., 2018). Specifically, it was found that teachers acknowledge their need to develop their own CT skills to be able to apply them in their classrooms and implement an interdisciplinary approach in their teaching (ibid.).

Different professional development courses for in-service teachers of CT are offered worldwide. The following four cases aim to develop teachers' CT knowledge and skills based on using computerized environments: (a) Hestness et al., (2018) facilitated a professional development workshop series on CT for elementary-level mentor teachers to support preservice teachers in integrating CT into their science classroom practice in early grades; (b) Peel et al. (2020) presented a case study of a science teacher whose secondary school student implemented design-based research projects focusing on integrating CT in computationally enriched biology units; and (c) Kong et al., (2020) presented the design, implementation, and evaluation of a professional development program for primary school teachers that focuses on the essential programming knowledge needed to develop CT, emphasizing the development of teachers' TPACK skills (Mouza et al., 2017). They used general programming assignments (e.g., a dancing cat or a wishing tree), and not problems from any specific subject domain; and (d) Çakır et al. (2021) used basic robotic coding to develop in-service teachers' acceptance of technology, self-development, and CT skills in technology use.

As can be seen, (a) teacher training for CT is introduced mainly in in-service teachers' professional development programs; (b) the integration of CT into many of the programs

is associated with programming-based applications; and (c) increasing attention is given to the integration of CT into the learning of specific subject matters. Our work, presented in the next two sections, contributes to this body of knowledge by focusing on *preservice* teachers' development of CT knowledge and skills in all subject matters, using computational environments, within *a holistic pedagogical framework—the 4P4CT*.

The 4P4CT framework

In this section, we present 4P4CT, which forms the theoretical and pedagogical basis of the CT course described in this paper, for preservice teachers trained to teach all disciplines to school children of all ages.

Theoretical and pedagogical infrastructure

The 4P4CT – Four Pedagogies for Developing Computational Thinking Skills – framework relies on the well-recognized learning theories of constructivism and constructionism and on some of their related pedagogical approaches. *Constructivism* emphasizes the building of learners' knowledge through active learning processes that promote their involvement in and responsibility for constructing new knowledge based on their previous knowledge (Fosnot, 2013; Piaget, 1973; Vygotsky, 1980). *Constructionism* expands the constructivist approach, emphasizing that learners' understanding is more meaningful when they create physical or computational products that represent learned concepts (Harel & Papert, 1991; Papert, 1980). The centrality of the constructionist approach in CT is expressed, for example, in the special issue of Constructivist Foundations Journal, which situates constructionism in the context of CT (Dagienė & Futschek, 2019).

From the variety of pedagogies developed based on these two theories, 4P4CT emphasizes the following:

Active learning is an instructional approach that engages students with the learned material through discussions, problem-solving, case study analysis, role-play, and other methods. In active learning, learners are responsible for their own learning process and gain opportunities to promote higher-order cognitive skills such as knowledge application, analysis, and synthesis (Drew & Mackie, 2011).

Project-based learning (PjBL) is a pedagogical approach in which learners actively develop their knowledge in a teamwork-based process that includes the presentation of an authentic issue/question/problem, search for possible solutions for solving it, selection of criteria for choosing among alternative solutions, and development of a solution that solves the problem and meets the problem requirements and constraints (Reinholz et al., 2018). In addition, learners are guided to apply critical and reflective thinking in relation to the problem-solving process in general and solutions in particular (Capraro & Slough, 2013a, 2013b).

Product-based learning suggests one possible implementation of the PjBL approach that stems from constructionist theory, emphasizing that people construct new knowledge when they are engaged in *constructing*

	meaningful physical products for themselves or for others around them (Ackermann, 1996).
Context-based learning	emphasizes that knowledge and skills must be learned and built up in a relevant context or domain of life, not just as a theory (Bennett et al., 2007a, 2007b; Holbrook, 2014; Prins et al., 2018; Stanisavljević et al., 2016). Moreover, it centers on the belief that both the social context of the learning environment and its concrete context are pivotal to knowledge acquisition and processing (Rose, 2012).

These pedagogies are applied in different ways in teachers' professional development programs and in programs for educators in academia (e.g., PjBl: Rahardjanto, 2019a, 2019b; Roessingh & Chambers, 2011; Product-based learning: Buteau et al., 2019; Context-based learning: Avargil et al., 2012).

Developing simulations of computational process for learning

Papert's constructionist vision (1980) lays out the rationale for using computers in general, and programming, particularly for learning in all disciplines. This vision yields the design of the Logo programming language and NetLogo programming environment for learning and implementing an agent-based modeling approach for science simulations, as well as simulations in many other areas (Tisue & Wilensky, 2004). Papert's approach was continued by his former student, Resnick, who developed Scratch, a block-based visual programming language, to create interactive projects that facilitate the learning of all disciplines and share them worldwide (Resnick et al., 2009). Resnick (2012) emphasizes the role of coding and states, "When you learn to read, you can then read to learn. And it's the same thing with coding. If you learn to code, you can code to learn."

We adopt this approach and take the stand that even though CT is meaningful and can be interpreted and implemented without computers, one of the basic working assumptions of the 4P4CT framework is that it is implemented by *learners' by developing, in a computerized learning environment, a context-based product that is a simulation of a computational process taken from a domain they are studying*. This implements *active learning* through *PjBL*, while developing a *product* in the *context* of learning a subject domain.

Research questions

To demonstrate the realization of the course objectives and pedagogical approach applied in the course, a preliminary examination was conducted to answer the following research questions:

RQ1: How do preservice teachers perceive the importance of computational thinking and its application in schools?

RQ2: How do teachers comprehend computational thinking skills as revealed in the development of simulations of a computational process in their field of knowledge?

Methods

The course design

The Computational Thinking (CT) academic course was designed for preservice K-12 teachers to implement the 4P4CT conceptual framework, enabling them to teach CT to students across all subjects and age groups. The course was developed over five years through an iterative process, and is based on three core foundations: (1) the course content focuses on CT; (2) the pedagogy follows the 4P4CT framework; and (3) the target audience includes all teachers, regardless of subject or age group. See Fig. 1.

Course rationale, objectives, and learning outcomes

Rationale The course rationale is to develop the preservice teachers' CT alongside the development of their understanding and confidence regarding its implementation in their future classes. Since preservice teachers should recognize the importance of all school pupils acquiring CT and be able to integrate it into their future teaching, they should experience CT in a supportive environment that encourages them to think about its implementation in schools. Accordingly, the course had two main interwoven principles. The first focuses on CT as an *object of thought* and includes exposure to and learning about CT skills, familiarity with CT curricula, and reflection on CT and its learning process. The second principle focuses on the *implementation* of CT in schools, and consists of simulation development in a computerized development environment of a computational process, including project development and peer assessment.

Objectives The main objectives of the course are:

(1) To expose preservice teachers to the concept of CT, increase their awareness of its various definitions and enable them to develop their own interpretation of CT.
(2) To develop preservice teachers' CT skills so that they can identify, define, and develop a computational process in their subject matter.
(3) To prepare preservice teachers to integrate CT into their future teaching of their subject matter.

Learning outcomes At the end of the course, the preservice teachers will be able to:

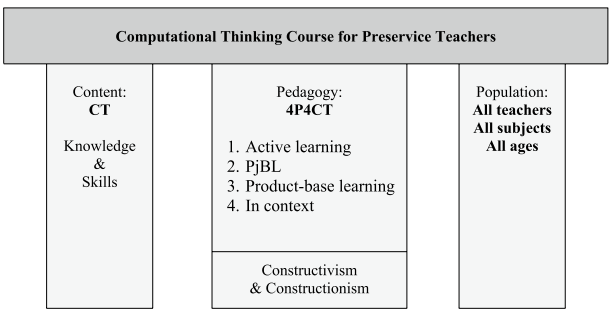


Fig. 1 The three foundations of the course design

(1) Identify their own interpretation of CT;
(2) Define a computational process in the discipline they will be teaching;
(3) Develop a script that simulates the computational process in the discipline they will be teaching.
(4) Analyze the given scenarios, determine whether the scenarios express a computational process, and explain their evaluation.
(5) Realize the importance of developing the CT skills of learners of all ages.

Course structure

The course comprises six units that build on each other and evolve throughout the course. A simulation of a computational process is developed in Unit D, while the preceding units establish the conceptual foundations and knowledge required to develop the simulation that implements the computational process. The units that follow the simulation development foster a holistic perspective of CT and present additional expressions of CT. The course structure is shown in Table 1.

We chose Scratch as the development environment for the course because it is a well-established and community-based environment (Brennan & Resnick, 2012), that enables the achievement of course objectives. This environment is widely known and used by students of all ages (https://scratch.mit.edu/).

Since the course is included in the curriculum of various academic degrees, it focuses not only on practical issues but also on theoretical issues and the investigation of research studies.

Course assignments

The students were assigned five types of assignments in the course. In what follows, we emphasize their pedagogical goals and how they are used to develop preservice teachers' conceptions of CT and their CT skills. Some assignments were given to the students several times during the course in different units and in different contexts, according to their current stage of CT acquisition. After describing the course assignments, we map the assignment types according to the pedagogies that form the basis for the 4P4CT framework and CT dimensions (Sect. Mapping the course assignments).

Assignment I: reverse engineering Reverse engineering processes were originally used in the context of engineering tasks to create a copy or change an existing artifact. The main thinking skill applied in reverse engineering is the analysis to deduce design principles from a given product (Bertoni, 2019a, 2019b). Similarly, a software reverse engineering process investigates a given source code and the underlying thinking processes that guide developers in the software development process.

In the course described in this paper, students are presented with a simulation that they can execute and play in the computational environment, test and explore object behavior, and examine the code that executes it. Students are asked to note, document, and describe every aspect of the given simulation that they discover, such as entity behaviors and dependencies among the behaviors of different characters. Students summarize their conclusions either verbally or graphically. They were also asked to list their thinking skills, assuming that the simulation developer(s) applied during the simulation development process, and to document the strategies they employed in this investigation process.

Table 1 Course structure—units and main objectives

Week	Unit	Main objectives
1	Unit A: exploring CT	This unit exposes the preservice teachers to the concept of CT and guides them to develop personal interpretation and conceptualization of CT The exposure consists of individually locating and viewing YouTube videos about CT, watching simulations implemented in Scratch, and reading articles
2–3	Unit B: CT concepts and skills	This unit is devoted to developing the preservice teachers' understanding of three of the CT foundations: problem decomposition, abstraction, and generalization. This objective is achieved on the conceptual level by practicing working with different charts such as concept maps and state-charts
4–5	Unit C: practicing CT	The objective of this unit is to move from the conceptual level to the implementation level by developing a simulation in a computerized environment (Scratch). The unit leans on the conceptual level attained in Unit B, by implementing the methodologies of decomposing, abstraction, and generalization in Scratch, and at the same time, by deepening students' understanding of the necessity for these methodologies. Basic building structures, both linear and parallel, are illustrated and practiced by solving several basic problems
6–9	Unit D: developing a simulation of a computational process	This unit forms the heart of the course, in which the CT skills learned so far are applied in the development of a simulation of a computational process. The preservice teachers work in teams, choose a subject matter and a respective computational process, and develop a simulation that reflects that computational process. Project development is guided by instructions such as: select a subject matter, analyze its content, choose a computational process from it, decompose the computational process into sub-problems, define relationships between the components, and define the processes for each sub-problem
10	Unit E: advanced exploration of CT skills	This unit returns to the conceptual level after the preservice teachers developed a simulation in Unit D. For example, reverse engineering (see the course assignments in the next section) is applied by introducing the preservice teachers to a given simulation of a computational process and having them identify which CT skills were needed for its development
11–12	Unit F: expanding perspectives of CT	The final unit is dedicated to further experimenting and practicing CT using various resources, such as alternative learning environments to Scratch, solving visual logic puzzles on the web (e.g., Hanoi Towers, the Missionaries and Cannibals problem), exploring international competitions that require the implementation of CT, using CS-unplugged resources, getting familiar with the local school CT curriculum, and more

At the beginning of the course, the students worked on this type of assignment in a given simulation (Unit A). Thus, from the onset of the course, students are exposed to both the applied and conceptual dimensions of CT. Similarly, reverse engineering is implemented in Unit E, towards the end of the course, after students have completed the team project in which they implemented various dimensions of CT. At this late stage of the course, their investigation was based on the knowledge and skills they learned and experienced throughout the course.

Assignment II: writing a position paper This type of assignment was given to the students twice during the course, emphasizing two different dimensions of CT. Both assignments were based on reading materials and critical thinking.

The first assignment, carried out in Unit B, instructs students to write a position paper that reflects their knowledge, attitudes, and position with respect to CT based on what they have learned and have been exposed to so far in the course (up to Lesson 3). The position paper should be based on reading papers, exploring web sources, and watching video clips chosen previously by students. The position paper was limited to 1–2 pages. The students were asked to write about their current interpretation of the concept of CT and to present and elaborate on their opinions with respect to the question "Can CT skills be learned in the context of your teaching subject? If yes—how? if not—why not?".

The second assignment was given to students at the end of the course in Unit F. The main goal of this assignment is to allow students to conceptualize and express the knowledge, insights, and skills that they acquired throughout the entire course, including the completion of the simulation development (presented in Sect. Course assignments). At this stage, the position paper focuses on the implementation of CT in schools. Students are directed to a variety of resources related to CT, including research papers, position papers, school curricula, and extra-curriculum initiatives. Based on their research, they are asked to write a reasoned and justified 2–3-page long position paper that describes their vision of the application of CT in *their* future classrooms. The guidelines explicitly state and encourage students to express independent, supportive, and/or critical positions.

Assignment III: individual perception description Students were asked to write their own interpretation and understanding of the concept of CT three times during the course: at the beginning of the course in Unit A, after Unit C, and at the end of the course in Unit F. The assignments were given through a questionnaire, and students were asked to complete. These assignments invite students to reflect on the challenges they face in entering the current learning stage, to elaborate on what they find interesting, surprising, or confusing, or that aroused their curiosity about CT. They also encourage students to recognize their difficulties, as well as their motivation with respect to learning and implementing CT. The main objective of this task was to foster students' reflective skills in general, and specifically with respect to their own CT skills (Larrivee, 2008; Schön, 1987). Two additional reflection tasks were submitted by the students together with their projects, as described next.

Assignment IV: development of a computerized simulation of a computational process in any subject (PjBL) This assignment was at the heart of the course. Students develop a *computerized simulation of a computational process* that integrates the concepts and skills the course aims to impart. After students have experienced, read, learned, and acquired hands-on practice in the computerized development environment, the goal of this assignment, implemented in Unit D, is to deepen their CT skills in the *context of*

their future teaching discipline. Specifically, the product of this assignment is a script in Scratch that simulates a computational process relevant to the discipline the preservice teachers intend to teach. The assignment was carried out in groups of to 2–3 students, preferably from different disciplines. Any subject matter can serve this purpose, for example, the Trojan War as part of their history studies, the water cycle in geography, or the movement of elements in the solar system in astronomy. The students were given the following guidelines.

(a) Find and define a computational process in your teaching discipline;
(b) Consider a possible visual implementation of the computational process you chose;
(c) Write an outline of a script that describes the computational process;
(d) Develop a simulation of the computational process.

In addition to the project code, two complementary reflective documents were also submitted. The first, submitted by the team, required students to reflect on the simulation development process. Students should address the limitations of the simulation in describing the real computational process, what they wished to achieve and did not succeed, and why. The second, submitted by each student individually, presents the student's reflection on his or her understanding of the various CT dimensions and skills—cognitive, social, and implementation—as expressed in the computational process development. Since most preservice teachers who study the course are not CS majors (see Table 2), their projects are not evaluated by CS standards (such as coding style and simulation structuring); rather, they are evaluated by the level of their analysis of the different CT dimensions of the project.

This learning setup demonstrates the application of 4P4CT theories and pedagogies, as follows: *constructivism* theory is exhibited because students are active and involved in the learning process and are responsible for the construction of their new knowledge. From the *constructionism* perspective, the PjBL and context-based learning theories are exhibited when students create a concrete product—a script that implements the simulation. The implementation of 4P4CT fosters students' various learning processes and developed skills.

(1) *Understanding* the learned topic is refined because the constructed simulation must be accurate to correctly describe the computational process in the learned topic.
(2) *Computational skills* are developed and enhanced while designing the scenario, selecting its entities, developing the entities' behavior, and scheduling their appearances and roles in the simulation (Ragonis, 2018). In particular, *algorithmic and programming skills* in computerized development environments have been strengthened.
(3) C*ognitive skills*, including logical reasoning and real-life problem solving skills, have been developed (Günbatar, 2019). Furthermore, *modeling skills* that require a higher level of abstraction, such as design, are enhanced.
(4) *Social skills*, which include working habits and practices such as teamwork, cooperation, time management and task scheduling, planning, role distribution, and conflict management—all of which are *soft skills* that are required in the twenty-first century—are promoted during the simulation development in teams.

In the Preliminary Findings section, we present examples of student products, specifically simulations of computational processes in various learning subjects.

Table 2 Description of the course populations

Institution	Number of classes	Number of students	Elective/mandatory course	Learning track
Institution1 (anonymized)	1	34	Mandatory course	Humanities and Social Sciences for *high school* (BEd)
Institution1 (anonymized)	5	99[a]	Mandatory course for preservice elementary school teachers Elective course for prospective teachers in the kindergarten track	Various disciplines for *elementary school*. Each student studies the teaching of two disciplines (BEd) *Kindergarten education* (BEd)
Institution1 (anonymized)	4	150[a]	Mandatory course	CS, Mathematics and Physics for *high school* (Teaching certificate)
Institution1 (anonymized)	1	10	Mandatory course	Mathematics for *high school* (M.Teach in Mathematics)
Institution1 (anonymized)	2	60[a]	Mandatory course	Various disciplines for *high school* (M.Teach)
Institution1 (anonymized)	1	7[a]	Mandatory course	M.Ed. in Integrative STEM Education
Institution2 (anonymized)	2	49	Mandatory course for prospective CS teachers Elective for other prospective STEM teachers	Mathematics, Sciences, and Technology for *high school* (BSc or teaching certificate)

[a]Includes a group that studied the course as a new MOOC and also applies the 4P4CT pedagogical framework

Assignment V: simulation evaluation This assignment, which takes place in Unit F, is based on a collection of simulations developed by the students in the course but can also be based on any collection of simulations that represent a variety of computational processes. Students are asked to evaluate the simulations and, specifically, to address the following two questions: (1) To what extent, in your opinion, does each simulation reflect CT principles? (2) In what way are each principle [mentioned in (1)] reflected in each simulation? In addition, for each simulation, students were asked to select one of five descriptive evaluation levels and justify their selection: insufficient CT application, limited CT application, reasonable CT application, good CT application, and excellent CT application. This assignment increases students' analytical skills by asking them to present a holistic assessment to justify their position based on the knowledge they acquired in the course. The student assignments were evaluated based on the justifications provided by the students.

Mapping the course assignments

In this section, we map course assignments according to the 4P4CT framework and the five dimensions of CT.

Mapping the course assignments according to the 4P4CT framework Table 3 maps course assignments according to the pedagogies on which the 4P4CT framework is based. Since students are preservice teachers, two aspects are emphasized with respect to context-based learning: the subject matter context and the teaching context.

As can be seen in Table 3, the constructivist pedagogical approach is expressed in all course assignments because they are all based on *active learning,* which provides students with opportunities to build their knowledge and skills. In a similar manner, since all course assignments are rooted in the context of the preservice teachers' future teaching, the *context-based learning* approach is also exhibited by all course assignments in relation to both the subject matter and teaching processes. Constructionism is reflected in the *PjBL* pedagogy implemented by the simulation development assignment that is anchored in the problem students solve: the identification of a computational process in a field of knowledge, for which students create a computerized product. Product-based learning pedagogy is also exhibited in the position paper-writing assignments.

Mapping the course assignments by five dimensions of CT Although CT has no unified or single acceptable definition, as presented in the introduction and background of this paper, we suggest five dimensions of CT skills that encompass concepts, problem-solving

Table 3 Mapping the course assignments by the four pedagogies

Type of assignment	Active learning	PjBL	Product based learning	Context-based learning	
				In relation to subject matter	In relation to teaching
Reverse engineering	✓			✓	✓
Writing a position paper	✓		✓	✓	✓
Interpreting CT—reflection	✓			✓	✓
Implementing a simulation	✓	✓	✓	✓	✓
Evaluating simulations	✓			✓	✓

skills, application skills, and social skills. Following the presentation of these five dimensions, we mapped the course assignments according to these dimensions. The five dimensions are as follows:

(a) *Cognitive CT skills,* which are problem-solving competencies, such as problem decomposition, abstraction, and generalization.
(b) *Algorithmic CT skills,* which refer to the implementation of the computational process in a programming language and involve computational concepts, such as conditions, iterations, events, and parallelism.
(c) *Programming CT skills,* which include thinking and behavior practices exhibited during code development, such as algorithm/script formulation, debugging, integration, and coordination between simulation components.
(d) *Modeling CT skills,* which address processes such as design, reverse engineering, and connections to real-world problems.
(e) *Social CT skills,* which are behaviors that allow people to function in teams, such as collaboration, communication, and presentation.

Table 4 shows how these CT dimensions are expressed in the course assignments.

Population

The course was taught to diverse populations in two higher education institutions in Israel. The first institution is Institution1 (anonymized), a multidisciplinary academic college focused on training educators of all ages and disciplines. The second institution is Institution2 (anonymized), a leading science and technology research university. This faculty only trains STEM teachers. So far, 409 preservice teachers in 16 classes have attended the course. Table 2 presents these classes and their respective populations.

Tools and procedure

To answer the research questions, we performed the inductive qualitative document analysis (Azungah, 2018) on students' outcomes from two assignments. Among all outcomes, collected from 406 students completed the course (see Table 2), we selected a sample consisted of.

(a) 50 summary guided reflections—the second position paper (Assignment II, "Course assignments" Section), to answer RQ1;
(b) 25 simulations out of 153, developed personally or in teams (Assignment IV, "Course assignments" Section), to answer RQ2.

The sample was selected so as to reflect student demographics: the two educational institutions, the different courses, the different fields of teaching, and the different target age groups. The qualitative analysis of students' outcomes involved identifying central themes that reflected the discourse and provided answers to the research questions (Thornberg & Charmaz, 2014).

To answer RQ1, the outcomes of the second position paper were analyzed in three phases: (1) Each text was divided into sections, each highlighting a different aspect of teaching and learning CT; (2) All excerpts reflecting similar aspects were grouped together,

Table 4 CT dimensions as reflected in the course assignments

Unit	Assignment	Cognitive CT skills	Algorithmic CT skills	Programming CT skills	Modelling CT skills	Social CT skills
A—exploration	Reverse engineering	✓			✓	✓
	Interpreting CT	✓			✓	
B—concepts and skills	Presenting a position paper	✓			✓	
C—practice	Interpreting CT	✓	✓	✓	✓	
D—PjBL	Development of simulation of a computational process	✓	✓	✓	✓	✓
E—advanced exploration of CT skills	Reverse engineering	✓			✓	✓
	Simulations evaluation	✓	✓	✓	✓	
F—expanding perspectives on CT	Presenting a position paper	✓	✓		✓	
	Interpreting CT	✓			✓	

and characteristic category titles were identified for each group. This process revealed five main categories: and (3) the texts were scanned again to select illustrative quotations.

Phases (1) and (2) were first done by each of the three researchers independently, and then a continued discussion was conducted on several disagreements until an agreement was reached by the three researchers.

To answer RQ2, the development of a computerized simulation of a computational process in any subject (see "Course assignments" Section) was analyzed. A sample of simulations was examined at this stage according to two key aspects: a. the definition of a computational process in the preservice teacher's field of knowledge; and b. the use of computational tools in the development of the simulation.

Findings

In what follows, the preliminary findings obtained from analyzing selected student outcomes are presented and discussed.

Preservice teachers' perceptions of the importance of computational thinking and its application in schools

The content analysis of the second position paper assignment revealed five categories. For each category we present its category and interpretation, and illustrate it with representative quote(s) for students' attitudes. The students' initials and subject of teaching are presented in parentheses after each quote; in the case of the MOOC, all quotes are anonymous.

Computational thinking is suitable not only for the sciences

Students testify that based on their own course experience they realized that CT can be applied and is relevant for all fields of knowledge, which is one of the main messages of the course.

> "From the beginning, the task of developing a simulation seemed to me to be more suitable for teachers who teach scientific subjects. In my discipline, which is civil sciences and sociology, the answers to the problems and conflicts that arise are much more diverse and rely on different considerations and comparisons, and I did not see how computational thinking could advance the students' understanding of the subjects within the discipline […]. But later, in the context of applying laws, I found myself saying: "It's very simple, there are clear rules about who is and who is not, and you don't need to invest too much thought here, it's either yes or no" and then I was able to see in my mind how he broke down the problem into small problems whose answers are yes or no and build from that ... a flow chart for questions that arise in relation to the law." [Y.G. Civil Sciences].
>
> "I was afraid of the project, and I was afraid of choosing the topic ... but now that I have finished, I think of how much the simulation can help the students, especially if they develop this type of simulation themselves. There is no doubt that this will make them understand the concepts better because everything is more tangible". [Anonymous]

Applying computational thinking in simulation development promotes understanding of concepts

Students reflected on their insights regarding the importance of the simulation development as a tool that promotes the acquisition of CT concepts and skills.

> "The work we did developing a simulation to demonstrate a process in physics enlightened me to the possibility of attaining a deeper understanding of the scientific process by applying computational thinking." [M.V. Physics].
> "In a software development platform such as Scratch... students build and assemble models according to their wishes and experience a creative experience that teaches them many things such as controlling objects, viewing the result, working and referring to data, weighing parameters, fine tuning, adaptation, and more, thus enabling them to deepen their understanding of the concepts for which the simulation is developed." [E.S. Mathematics]

Computational thinking concepts are an important part of a teacher's toolbox

Students valued the CT skills they gained in the course, considering them a valuable addition to their instructional and pedagogical resources (toolbox).

> "Studying the subject, and understanding it, gave me another tool, in my "toolbox" as a teacher, for transferring and learning some of the content that I will teach within the framework of my disciplines both professionally (assimilation of study material among the students) and from the student's personal point of view (developing the students' personal skills)." [Z.C.B. History]

> "During the simulation development, I learned that every step and order is important in the process, and it is impossible to forego some instruction or place it in the wrong place in the script, as in other processes that happen every day in my work as a teacher and educator. The order of action and planning is significant when I want to achieve a goal." [G.O., Social Sciences].

Concerns regarding the implementation of CT

Beyond the personal insights with respect to the benefits of integration CT in their future classes, the students raised their concerns related to its broad implementation in the education system.

> "After completing the course assignments, my understanding of what computational thinking is and how I can apply it in the discipline of civil sciences sharpened, however, I am not sure that the general student population can apply it, but rather mainly students who are curious, who have high learning skills and analytical skills, as this is a process that requires the students to think independently, ask questions, and research." [O.L. Civil and Social Sciences]

> "To enhance computational thinking, many people need to be involved. This is a message to students worldwide and in all fields of knowledge. Computational thinking and communicating with computers through code are necessary, just as students can now read and write. However, we were at the beginning. I look forward to seeing progress in the coming years." [M.H. Mathematics]

Challenges in developing teachers' computational thinking skills

As part of the mission of a broad implementation of CT in the education system, students addressed the challenges involved in teachers' professional development.

> "Looking today at the subject of computational thinking, I understand more deeply how important it is to develop this ability among the younger generations. In my opinion, this ability is a critical skill for life today and in the future. I will continue to say that the gap and reluctance I saw among my classmates amazed me and emphasized the great challenge and difficulty, which is actually instilling computational thinking among the generation of teachers. In my opinion, it is not just an age gap but also a gap in understanding and the fact that even young teachers who come to teach do not have any realistic or programming background. All of this leads to a great reluctance to enter the field, thus actually harming another generation of students who will not grow up with this ability of computational thinking." [S.M. Civil and Social Sciences].

Teachers' comprehension of computational thinking skills as revealed in the simulations development

We expected that the simulation development will reflect not only prospective teachers' understanding of one of the main concepts emphasized in the course, namely, *computational process* in the field of knowledge, but also, the development of a simulation that reflects it. The examination of a sample of simulations (according to the criteria presented in "Tools and procedure" Section), it was found that they represent computational processes in a wide range of fields of knowledge: Bible, Civil Studies, Computer Science, Environmental Studies; History; Language; Life Sciences; Mathematics; and Physics.

The realization of the computational process in the field of knowledge is demonstrated using four simulations taken from different school subjects learned in various school grades. Figure 2 presents screen snapshots of the simulations in Scratch, followed by a brief description. Additional examples of simulations can be found in Ragonis (2018) and Hazzan et al. (2020, Chapter 4).

Example (a): air pollution, environment studies for high school

One objective of environmental education is to increase students' awareness of environmental factors that affect air pollution. In this example, students simulated a real-world situation that included several factors affecting air pollution (cars, buses, and factories) and several different molecules in the atmosphere. The simulation is dynamic and enables users to add and replace elements (e.g., they can replace several cars with one bus) and observe how each change affects air pollution in the environment. In addition, the background changes its color from red to green, and vice versa, according to the level of pollution (see Fig. 2a).

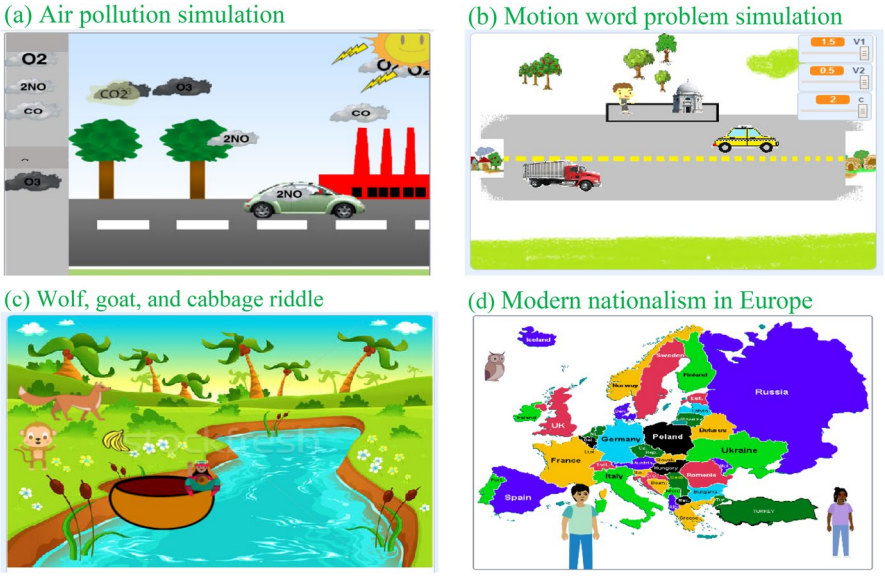


Fig. 2 Screen snapshots of four simulations

Example (b): mathematical word problems, mathematics for middle school

Many learners experience difficulties when trying to solve mathematical word problems. The development of a simulation that reflects the process described in the problem may improve learners' understanding of the behaviors of the entities described in the problem, as well as the mutual relationships among them. In this example, students simulated a story, described as a word problem, about a taxi that drives from City A to City B and picks up a passenger, and a truck that drives from City B to City A. Vehicle speeds are controlled by the users by means of sliders, and the question they are asked is: How many times will the taxi and truck pass each other on their way back and forth within a certain time limit? (see Fig. 2b).

Example (c): the wolf, goat, and cabbage riddle, logic for elementary school

This simulation enabled the students to have a dynamic experience in solving the famous "Wolf, goat, and cabbage" riddle. The riddle conditions were as follows: a farmer wishes to cross a river with a wolf, goat, and cabbage. Each time the boat crosses the river, it can carry the farmer, wolf, goat, or cabbage. If the wolf and goat are left alone on one shore, the wolf will devour the goat; if the goat (or wolf) and cabbage are left alone on the shore, the goat (or wolf) will eat the cabbage. The riddle is: "How can the farmer cross the river so that the wolf, the goat, and the cabbage all get to the other side intact and alive?" The students chose to change the characters but retained the rules. The simulation is dynamic: the user can choose the character who will board the boat with the farmer by clicking on it, then the boat moves from one side of the river to the other side, and the simulation checks the correctness of the new state (see Fig. 2c).

Example (d): the rise of modern nationalism in Europe, history for high school

The simulation shows the national changes in Europe throughout history. The European nation-states were created as a result of several factors that brought man to a new understanding of the place of the individual in society. The departure from the framework of sovereigns and vassals to the new world resulted from processes such as urbanization, secularization, industrialization, education, and romance. The simulation demonstrated changes according to various factors (see Fig. 2d).

Discussion and conclusions

In this paper, we present a computational thinking course designed to be integrated into preservice K-12 teacher preparation programs for all ages (K-12) and disciplines. The course implements 4P4CT–Four Pedagogies for Developing Computational Thinking—a framework that builds on constructivism and constructionism, implementing four main pedagogies: active learning, project-based learning, product-based learning, and context-based learning.

In this section, we first answer and discuss the research questions and then present the main conclusions and contributions of our research initiative.

With respect to *RQ1: How do preservice teachers perceive the importance of computational thinking and its application in schools?* We presented five categories of teachers' conceptions, reflecting the three foundations of the course design (see Fig. 1): (1) content (category b: "*Applying computational thinking in simulation development promotes understanding of concepts"* and category e: "*The challenge of developing teachers' knowledge"*), (2) pedagogy (category c "*Computational thinking concepts are an important part of a teacher's toolbox"* and category d: "*Pedagogical aspects in thinking about the future of the application of CT"*) and (3) population (category a: "*Computational thinking is suitable not only for the sciences*").

Specifically, an analysis of students' reflections in their position paper revealed that the main course objectives were fulfilled. They understand that CT is important for all teachers (category c), they understand the contribution to any discipline and not only to the sciences (category a), and they appreciate the active experience of CT as a promoter of the understanding of concepts in any field of knowledge (category b). Additionally, they considered methods of implementation in schools (category d). Nevertheless, the learning path is not easy, and prospective teachers raise concerns regarding implementation at schools (categories d, e). This means that their attitudes regarding school students' skills and teachers' skills must be further encouraged.

With respect to RQ2: *How are teachers' computational thinking skills reflected in the development of simulations of a computational process in their field of knowledge?* We presented four simulations from four different subjects (environment studies, mathematics, logic, and history), that are suitable for all K-12 grades: elementary school, middle school, and high school. This also reflects the wide application of the pedagogical approach presented in this paper, which targeted all subject matter and all students. In other words, the preliminary analysis of the simulations shows that the concepts that the course sought to impart were achieved: computational thinking skills were applied in the context of the field of knowledge and in the context of computer simulation development. This is evident in

the simulations and also in the second category that emerged from the reflection analysis, where students assert that simulation development promotes understanding of concepts in the subject matter. We can add that this also fulfills our 4P4CT pedagogical model vision, since students actively construct their knowledge while developing an outcome in the context of their teaching field of knowledge.

Further to our literature review, we suggest that our main contribution is to provide a holistic approach to teaching computational thinking skills that enriches students' learning with respect to both content and skill. This is achieved largely by increasing the learners' intrinsic motivation, since according to the self-determination theory (Ryan & Deci, 2000), their needs for competence, autonomy, and relatedness are fulfilled. This observation is especially important for the case described in this paper—prospective teachers of all ages and all subjects—since the proposed approach has the potential to be applied to the entire education system.

Furthermore, our data were collected in the context of project development while applying active learning pedagogy, rather than in the context of a theoretical discussion about the essence of CT and its teaching, as presented, for example, in Yadav et al., (2017) or in the context of lesson plans that preservice teachers submitted and analyzed in the case described by Yang et al., (2018a, 2018b). Our approach further highlights the potential contribution of the 4P4CT framework to the preservice teachers' perception of CT in a way that addresses Yang et al.'s claim that teachers had "difficulties conceptualizing and integrating CT in conjunction with content and pedagogy, particularly during authentic vis-à-vis hypothetical lesson designs" (p. 368).

Project development pedagogy is one method used to improve preservice teachers' CT (Dong et al., 2023). Nevertheless, based on our proposed 4P4CT framework, which builds on constructivism and constructionism, we apply a pedagogy that integrates learning CT skills and disciplinary knowledge in a meaningful context for preservice teachers by developing a computational process related to teachers' discipline. Butler and Leahy (2021) and Bal et al. (2022) applied a similar approach, but while these two works applied constructivism perspectives in either a digital learning specialism program or asynchronous micro–credential courses, respectively, our approach is implemented by one course taught in a variety of study programs to prospective teachers of all ages and all subject matters, and therefore can be suitable to many study programs, by a large community of teacher educators.

The vision we impart to the preservice teachers in the course implies that:

(a) CT skills should be imparted as early as possible so that students can assimilate and use them throughout their years of schooling and beyond.
(b) CT skills should be applied across all disciplines, because they have the potential to promote learning in all areas. Furthermore, such applications will enable students to deepen their understanding of subject matter as well as foster their CT skills.
(c) CT skills should be implemented and developed in relevant contexts of problem-solving.

Although CT does not necessarily involve programming and can be manifested through other forms of problem solving (Bell, 2009; Bell et al., 2009; Taub et al., 2009), we do see a great advantage in implementing CT using computerized programming environments. Furthermore, since the implementation of computerized simulation engages teachers with technology, we propose that 4P4CT be conceived as part of teacher technological

pedagogy content knowledge (TPACK) (Gür & Karamete, 2015; Koehler & Mishra, 2009; Mouza et al., 2017). Many students did not understand what code meant and did not believe that they could develop it. Their success in developing simulations builds their confidence in the use of technology from a broader perspective. Specifically, preservice teachers are taught and trained in the course in a manner that enables them to implement 4P4CT with their school students in their discipline of expertise. This pedagogical method is in line with teacher training approaches that emphasize that teacher training programs should not focus only on theory and academic research but should also use the same pedagogical methods that preservice teachers will actually be using in the future with their students (Canaleta et al., 2014). The analysis of the selected students' outcomes, as reflected in the answers to the research questions presented at the beginning of this section, showed that the central concepts of the course were learned and reflected the success of the 4P4CT pedagogical model.

While we suggest that the foundations of CT and its practical principles of programming be taught at schools by CS teachers, we also believe that CT skills should be used when learning all disciplines. Accordingly, we envision the development of teachers' CT skills through a mandatory course for preservice teachers of all ages and disciplines and propose that the course presented in this paper can leverage the integration of CT throughout the educational system. This course is also suitable for professional development programs for in-service teachers. This extensive course description enables educators to implement similar courses.

As this study is only one step in our investigation of the 4P4CT pedagogical framework, we present only qualitative data, and neither measure students' learning nor attitudes by, for example, pre and post tests, nor used a control group. Our ongoing research of the course, examines other course assignments outcomes, including evaluating the simulations development using dedicated research tools to examine how the five CT dimensions are reflected in the simulation development. Moreover, further investigations can address how the course graduates apply what they have acquired in the course in their classes.

Data availability Data is available from the first author upon request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Ackermann, E. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*. Routledge.

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction, 31*, 100329.

Araujo, A. L. S. O., Andrade, W. L., Guerrero, D. D. S., & Melo, M. R. A. (2019). How many abilities can we measure in computational thinking?: *A study on bebras challenge. Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 545–551

Armoni, M. (2019). Computing in schools on the knowledge of CS teachers' educators. *ACM Inroads, 10*(2), 10–13.

Avargil, S., Herscovitz, O., & Dori, Y. J. (2012). Teaching thinking skills in context-based learning: Teachers' challenges and assessment knowledge. *Journal of Science Education and Technology, 21*(2), 207–225.

Azungah, T. (2018). Qualitative research: Deductive and inductive approaches to data analysis. *Qualitative Research Journal, 18*(4), 383–400.

Bal, I. A., Alvarado-Albertorio, F., Marcelle, P., & Oaks-Garcia, C. T. (2022). Pre–service teachers computational thinking (CT) and pedagogical growth in a micro–credential: A mixed methods study. *TechTrends, 66*(3), 468–482. https://doi.org/10.1007/s11528-022-00732-x

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, 38*(6), 20–23.

Bebras, *International challenge on informatics and computational thinking*. Retrieved from https://www.bebras.org/

Bell, D. T. (2009). Computer Science Unplugged. Retrieved from https://csunplugged.org/en/

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology, 13*(1), 20–29.

Bennett, J., Campbell, B., Hogarth, S., & Lubben, F. (2007). A systematic review of the effects on high school students of context-based and science-technology-society (STS) approaches to the teaching of science. Retrieved Aug, 2007.

Bennett, J., Lubben, F., & Hogarth, S. (2007b). Bringing science to life: A synthesis of the research evidence on the effects of context-based and STS approaches to science teaching. *Science Education, 91*(3), 347–370.

Bertoni, A. (2019a). A reverse engineering role-play to teach systems engineering methods. *Education Sciences, 9*(1), 30. https://doi.org/10.3390/educsci9010030

Bertoni, A. (2019b). A reverse engineering role-play to teach systems engineering methods. *Education Sciences, 9*(1), 1. https://doi.org/10.3390/educsci9010030

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education-Implications for policy and practice* (No. JRC104188). Joint Research Centre (Seville site).

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016b). *Developing computational thinking in compulsory education*. European Commission, JRC Science for Policy Report.

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., & Stupurienė, G. (2022). *Reviewing Computational Thinking in Compulsory Education* (No. JRC128347). Joint Research Centre (Seville site).

Boholano, H. (2017). Smart social networking: 21st century teaching and learning skills. *Research in Pedagogy, 7*(1), 21–29. https://doi.org/10.17810/2015.45

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver*. Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Buteau, C., Sacristán, A. I., & Muller, E. (2019). Roles and demands in constructionist teaching of computational thinking in university mathematics. *Constructivist Foundations, 14*(3), 294–309.

Butler, D., & Leahy, M. (2021). Developing preservice teachers' understanding of computational thinking: A constructionist approach. *British Journal of Educational Technology, 52*(3), 1060–1077. https://doi.org/10.1111/bjet.13090

Cabrera, L. (2019). Teacher preconceptions of computational thinking: a systematic literature review. *Journal of Technology and Teacher Education, 27*(3), 305–333.

Çakır, R., Şahin, H., Balci, H., & Vergili, M. (2021). The effect of basic robotic coding in-service training on teachers' acceptance of technology, self-development, and computational thinking skills in technology use. *Journal of Computers in Education, 8*(2), 237–265.

Canaleta, X., Vernet, D., Vicent, L., & Montero, J. A. (2014). Master in teacher training: A real implementation of active learning. *Computers in Human Behavior, 31*, 651–658.

Capraro, R. M., & Slough, S. W. (2013a). Why PBL ? Why STEM? Why Now? In R. M. Capraro, M. M. Capraro, & J. R. Morgan (Eds.), *STEM project-based learning. an integrated science, technology, engineering, and mathematics (STEM) approach.* Sense Publishers.

Capraro, R. M., & Slough, S. W. (2013b). Why PBL? Why STEM? Why now? In R. M. Capraro, M. M. Capraro, & J. R. Morgan (Eds.), *STEM project-based learning an integrated science technology engineering, and mathematics (STEM) approach.* Sense Publishers.

Chang, Y. H., & Peterson, L. (2018). Pre-service teachers' perceptions of computational thinking. *Journal of Technology and Teacher Education, 26*(3), 353–374.

CodeMonkey. *Coding for kids: Introducing programming games for the next generation*. Retrieved from https://www.codemonkey.com

CSTA (2017). Computer science standards. Retrieved from https://www.doe.k12.de.us/cms/lib/DE01922744/Centricity/Domain/176/CSTA%20Computer%20Science%20Standards%20Revised%202017.pdf

CSTA, K. (2017). *Computer science standards*. Computer Science Teachers Association.

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress*, Referenced in https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Dagienė, V., & Futschek, G. (2019). On the way to constructionist learning of computational thinking in regular school settings. *Constructivist Foundations, 14*(3), 231–233.

Denning, P. J. (2009). Beyond computational thinking. *Communications of the ACM, 52*(6), 28–30.

Denning, P. J., & Tedre, M. (2022). Computational thinking: A disciplinary perspective. *Informatics in Education, 20*(3), 361–390.

DeSchryver, M. D., & Yadav, A. (2015). Creative and computational thinking in the context of new literacies: Working with teachers to scaffold complex technology-mediated approaches to teaching and learning. *Journal of Technology and Teacher Education, 23*(3), 411–431.

Dong, W., Li, Y., Sun, L., & Liu, Y. (2023). Developing pre-service teachers' computational thinking: a systematic literature review. *International Journal of Technology and Design Education*. https://doi.org/10.1007/s10798-023-09811-3

Drew, V., & Mackie, L. (2011). Extending the constructs of active learning: implications for teachers' pedagogy and practice. *Curriculum Journal, 22*(4), 451–467.

Ebner, M., Holzinger, A., & Maurer, H. (2007). Web 2.0 technology: Future interfaces for technology enhanced learning?. In *Universal Access in Human-Computer Interaction. Applications and Services: 4th International Conference on Universal Access in Human-Computer Interaction, UAHCI 2007 Held as Part of HCI International 2007 Beijing, China, July 22–27, 2007 Proceedings, Part III 4* (pp. 559–568). Springer

Fosnot, C. T. (2013). *Constructivism: Theory, perspectives, and practice*. Teachers College Press.

Google. (2019). Google for education: Computational thinking. Retrieved from https://edu.google.com/resources/programs/exploring-computational-thinking/

Günbatar, M. S. (2019). Computational thinking within the context of professional life: Change in CT skill from the viewpoint of teachers. *Education and Information Technologies, 24*(5), 2629–2652. https://doi.org/10.1007/s10639-019-09919-x

Gür, H., & Karamete, A. (2015). A short review of TPACK for teacher education. *Educational Research and Reviews, 10*(7), 777.

Harel, I., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.

Harper, B. (2018). Technology and teacher-student interactions: A review of empirical research. *Journal of Research on Technology in Education, 50*(3), 214–225. https://doi.org/10.1080/15391523.2018.1450690

Hazzan, O., Ragonis, N., & Lapidot, T. (2020). Computational thinking. In *Guide to teaching computer science: An activity—based approach* (3rd ed., pp. 57–74). Springer.

Hestness, E., Ketelhut, D. J., McGinnis, J. R., & Plane, J. (2018). Professional knowledge building within an elementary teacher professional development experience on computational thinking in science education. *Journal of Technology and Teacher Education, 26*(3), 411–435.

Hodhod, R., Khan, S., Kurt-Peker, Y., & Ray, L. (2016). Training teachers to integrate computational thinking into K-12 teaching. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 156–157, https://doi.org/10.1145/2839509.2844675

Holbrook, J. (2014). A context-based approach to science teaching. *Journal of Baltic Science Education, 13*(2), 152–154.

Hu, C. (2011). Computational thinking: What it might mean and what we might do about it. *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*. https://doi.org/10.1145/1999747.1999811

Hughes, K., Fletcher, C. L., DeLyser, L. A., & Owen, A. (2017). Building CS teaching capacity: Comparing strategies for achieving large scale impact. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. https://doi.org/10.1145/3017680.3017685

Israel-Fishelson, R., & Hershkovitz, A. (2020). Persistence in a game-based learning environment: The case of elementary school students learning computational thinking. *Journal of Educational Computing Research, 58*(5), 891–918.

ISTE, & CSTA. (2011). Operational definition of computational thinking for K–12 education. Retrieved from https://id.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Jacob, S. R., & Warschauer, M. (2018a). Computational thinking and literacy. *Journal of Computer Science Integration*. https://doi.org/10.26716/jcsi.2018.01.1.1

Jacob, S. R., & Warschauer, M. (2018b). Computational thinking and literacy. *Journal of Computer SciEnCe Integration*. https://doi.org/10.26716/jcsi.2018.01.1.1
K-12 Computer Science Framework Steering Committee. (2016). *K-12 computer science framework*. ACM.
Kanemune, S., Shirai, S., & Tani, S. (2017). Informatics and programming education at primary and secondary schools in Japan. *Olympiads in Informatics, 11*(2017), 143–150.
Kellow, J. M. (2018). Digital technologies in the New Zealand curriculum. *Waikato Journal of Education*. https://doi.org/10.15663/wje.v23i2.626
Koehler, M., & Mishra, P. (2009). What is technological pedagogical content knowledge (TPACK)? *Contemporary Issues in Technology and Teacher Education, 9*(1), 60–70.
Kong, S. C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education, 151*, 103872.
Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior, 72*, 558–569.
Lambić, D., Đorić, B., & Ivakić, S. (2021). Investigating the effect of the use of code org. on younger elementary school students' attitudes towards programming. *Behaviour & Information Technology, 40*(16), 1784–1795.
Lamprou, A., & Repenning, A. (2018). Teaching how to teach computational thinking. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. https://doi.org/10.1145/3197091.3197120
Larrivee, B. (2008). Meeting the challenge of preparing reflective practitioners. *The New Educator, 4*(2), 87–106.
Lee, I., Grover, S., Martin, F., Pillai, S., & Malyn-Smith, J. (2020). Special issue on computational thinking from a disciplinary perspective. *Journal of Science Education and Technology, 29*(1), 1573–1839.
Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020a). Computational thinking is more about thinking than computing. *Journal for STEM Education Research, 3*(1), 1–18.
Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020b). Computational thinking is more about thinking than computing. *Journal for STEM Education Research, 3*, 1–18.
Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61.
Marangunić, N., & Granić, A. (2015). Technology acceptance model: A literature review from 1986 to 2013. *Universal Access in the Information Society, 14*(1), 81–95.
Mike, K., Ragonis, N., Rosenberg-Kima, R. B., & Hazzan, O. (2022). Computational thinking in the era of data science. *Communications of the ACM*, *65*(8), 33–35. https://doi.org/10.1145/3545109
Morad, S., Ragonis, N., & Barak, M. (2021a). The validity and reliability of a tool for measuring educational innovative thinking skills. *Journal of Teaching and Teacher Education, 97*, 103193.
Morad, S., Ragonis, N., & Barak, M. (2021b). An integrative conceptual model of innovation and innovative thinking base on synthesis of literature review. *Thinking Skills and Creativity, 40*, 100824.
Mouza, C., Yang, H., Pan, Y. C., Ozden, S. Y., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*. https://doi.org/10.14742/ajet.3521
OECD (2019). *Computer science and PISA 2021*. OECD Education and Skills Today. Retrieved from https://oecdedutoday.com/computer-science-and-pisa-2021/
Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
Park, R. (2016). *Preparing students for South Korea's creative economy: The successes and challenges of educational reform*. Asia Pacific Foundation of Canada.
Peel, A., Dabholkar, S., Anton, G., Wu, S., Wilensky, U., & Horn, M. (2020). A case study of teacher professional growth through co-design and implementation of computationally enriched biology units. In *Proceedings of international conference of the learning sciences (ICLS 2020)* (pp. 1950–1957). https://repository.isls.org/bitstream/1/6478/1/1950-1957.pdf
Piaget, J. (1973). *To understand is to invent: The future of education*. Penguin Books.
Pollock, L., Mouza, C., Guidry, K. R., & Pusecker, K. (2019). Infusing computational thinking across disciplines: Reflections & lessons learned. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 435–441).

Prins, G. T., Bulte, A. M., & Pilot, A. (2018). Designing context-based teaching materials by transforming authentic scientific modelling practices in chemistry. *International Journal of Science Education, 40*(10), 1108–1135.

Ragonis, N. (2018). Computational thinking: Constructing the perceptions of pre-service teachers from various disciplines. In S. Pozdniakov & V. Dagienė (Eds.), *Informatics in schools. Fundamentals of computer science and software engineering. ISSEP 2018. Lecture notes in computer science*, Vol. 11169 (pp. 167–179). Springer.

Rahardjanto, A. (2019a). Hybrid-PjBL: Learning outcomes, creative thinking skills, and learning motivation of preservice teacher. *International Journal of Instruction, 12*(2), 179–192.

Rahardjanto, A. (2019b). Hybrid-PjBL: Learning Outcomes, Creative Thinking Skills, and Learning Motivation of PreseReinholz, D., Slominski, T., French, T. A., Pazicni, S., Rasmussen, C., & McCoy, B. (2018). Good problems within and across disciplines. *Journal of Research in STEM Education*, *4*(1):37–53.

Reinholz, D., Slominski, T., French, T. A., Pazicni, S., Rasmussen, C., & McCoy, B. (2018). Good problems within and across disciplines. *Journal of Research in STEM Education, 4*(1), 37–53.

Resnick, M. (2012). *Let's teach kids to code*. Retrieved from https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. S., & Silverman, B. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.

Roessingh, H., & Chambers, W. (2011). Project-based learning and pedagogy in teacher preparation: Staking out the theoretical mid-ground. *International Journal of Teaching and Learning in Higher Education, 23*(1), 60–71.

Rose, D. E. (2012). Context-based learning. In N. M. Seel (Ed.), *Encyclopedia of the sciences of learning*. Springer.

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist, 55*, 68–78.

Sabitzer, B., Antonitsch, P. K., & Pasterk, S. (2014). Informatics concepts for primary education: Preparing children for computational thinking. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*. https://doi.org/10.1145/2670757.2670778

Schön, D. A. (1987). *Educating the reflective practitioner: Towards a new design for teaching and learning in the profession*. Jossey-Bass.

Seegerer, S., & Romeike, R. (2018). Computer science as a fundamental competence for teachers in other disciplines. *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*. https://doi.org/10.1145/3265757.3265787

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Stanisavljević, J. D., Pejčić, M. G., & Stanisavljević, L. Ž. (2016). The application of context-based teaching in the realization of the program content "the decline of pollinators." *Journal of Subject Didactics, 1*(1), 51–63.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*, 103798. https://doi.org/10.1016/j.compedu.2019.103798

Taub, R., Ben-Ari, M., & Armoni, M. (2009). The effect of CS unplugged on middle-school students' views of CS. *ACM SIGCSE Bulletin, 41*(3), 99–103. https://doi.org/10.1145/1595496.1562912

The Royal Society. (2012). *Shut down or restart? The way forward for computing in UK schools*. The Royal Society.

Thornberg, R., & Charmaz, K. (2014). Grounded theory and theoretical coding. In U. Flick (Ed.), *The Sage handbook of qualitative data analysis.* Sage.

Tisue, S., & Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. *International Conference on Complex Systems, 21*, 16–21.

Ung, L. L., Labadin, J., & Mohamad, F. S. (2022). Computational thinking for teachers: Development of a localised E-learning system. *Computers & Education, 177*, 104379.

Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Harvard University Press.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. M. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine, 60*, 20–23.

Wing, J. M. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In P. J. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 205–220). Springer.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014a). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 5*, 16. https://doi.org/10.1145/2576872

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014b). Computational thinking in elementary and secondary teacher education. *Transactions on Computing Education, 5*(1–5), 16. https://doi.org/10.1145/2576872

Yang, H., Mouza, C., & Pan, Y. C. (2018a). *Examining pre-service teacher knowledge trajectories of computational thinking through a redesigned educational technology course*. International Society of the Learning Sciences Inc.

Yang, H., Mouza, C., & Pan, Y. (2018b). Examining pre-service teacher knowledge trajectories of computational thinking through a redesigned educational technology course. In J. Kay & R. Luckin (Eds.), *Rethinking learning in the digital age: making the learning sciences count, 13th International Conference of the Learning Sciences (ICLS) 2018* (Vol. 1, pp. 386–375). International Society of the Learning Sciences.

Yasar, O., & Veronesi, P. (2015). Computational pedagogical content knowledge (CPACK): Integrating modeling and simulation technology into STEM teacher education. *Society for Information Technology & Teacher Education International Conference*. (pp. 3514–3521).

Yilmaz, F. G. K., Yilmaz, R., & Durak, H. Y. (2018). A review on the opinions of teachers about the development of computational thinking skills in K-12. In *Teaching computational thinking in primary education* (pp. 157–181). IGI Global

Noa Ragonis is a senior lecturer. She is the head of M.Ed. in Integrative STEM Education and the head of the Computer Science Department, at Beit Berl College Faculty of Education. Ragonis is a graduate of the Faculty of Mathematics and Computer Science at Bar-Ilan University. She received her MSc and PhD degrees at the Weizmann Institute of Science and was a postdoctoral fellow at the Faculty of Education in Technology and Science, Technion. Her research mainly focuses on cognitive aspects of teaching and learning of Computer Science and Computational Thinking, in particular in relation to Logic Programming and Object Oriented Programming, as well as integrating ICT and innovation in teaching and learning processes. She has published over 60 articles in journals, conferences and chapters in books. She has authored the book Guide to Teaching Computer Science (2011; 2014; 2020, Springer) and has authored ten Computer Science high-school textbooks and teachers' book guidelines, in relation to OOP, Logic Programming, and Computational Models. Engaged in pre-service and in-service teachers' preparation programs.

Additional details can be found on her homepage: https://www.beitberl.ac.il/english/lecturers/pages/noaragonis.aspx

Rinat Rosenberg-Kima is a faculty member at the Faculty of Education in Science and Technology in the Technion – Israel Institute of Technology. She holds a BA in Computer Science and Economics from Tel-Aviv University, an MA in Psychology from Tel-Aviv University, and a PhD in Educational Psychology and Learning Systems from Florida State University. After her PhD, Rinat moved to do her postdoc at the Lawrence Hall of Science in UC Berkeley, where she led the analytics team in a digital science curriculum project for middle school. After 10 years in the USA, Rinat moved back to Israel and researched educational social robots that facilitates small group discussions as part of a postdoc at the Faculty of Engineering at Tel-Aviv University. For this study she won Sagol School of Neuroscience's Minducate scholarship and a fund from Schmidt Futures program. In between, Rinat taught various courses in the department of Computer Sciences at the Open University, Israel and developed and programmed various educational software. Her research areas include Learning Sciences, Learning Design, Human- Robot Interaction, Computer Science Education, Learning Analytics, and Educational Data Mining.

Orit Hazzan joined the Department of Education in Technology and Science in October 2000. Her research focuses on (a) cognitive and social processes in computer science, software engineering and data science education and (b) policy and strategic planning of STEM education. Within these frameworks, her research examines the individual, the team and the organization levels, in high schools, academia and hi-tech organizations. She has published about 120 papers in professional refereed journals and conference proceedings and seven books. Her main organizational role are: in 2006–2008, Hazzan served as the Technion's Associate Dean of Undergraduate Studies; in 2007–2010 she chaired the High School Computer Science Curriculum Committee assigned by the Israeli Ministry of Education; in 2011–2015 Hazzan was the Dean of the Technion's Faculty of Education in Science and Technology, and in 2017–2019, Hazzan serves as the Technion Dean of Undergraduate Studies. Additional details can be found in her homepage: https://orithazzan.net.technion.ac.il/

Authors and Affiliations

Noa Ragonis[1] · **Rinat B. Rosenberg-Kima**[2] · **Orit Hazzan**[2]

✉ Noa Ragonis
noarag@beitberl.ac.il

Rinat B. Rosenberg-Kima
rinatros@ed.technion.ac.il

Orit Hazzan
oritha@ed.technion.ac.il

[1] Department of Computer Science, Faculty of Education, Beit Berl College, Beit Berl, Kfar Saba, Israel

[2] Faculty of Education in Science and Technology, Technion – Israel Institute of Technology, Haifa, Israel