Informatics in Education, 2025, Vol. 24, No. 1, 45–76

DOI: 10.15388/infedu.2025.01

The Effect of Cooperative Learning on Academic Performances and Computational Thinking Skills in the Computational Problem-Solving Approach

İrem Nur ÇELİK[1], Kaan BATI[2]

[1]*Hacettepe University, Graduate School of Educational Sciences, Ankara, Turkey*
[2]*Hacettepe University, Faculty of Education, Department of Mathematics and Science Education Ankara, Turkey*
e-mail: iremnceliik@gmail.com, kaanbati@gmail.com

Received: September 2023

Abstract. In this study, we aimed to investigate the impact of cooperative learning on the computational thinking skills and academic performances of middle school students in the computational problem-solving approach. We used the pretest-posttest control group design of the quasi-experimental method. In the research, computational problem-solving activities regarding 6th graders' goals of the "heat and matter" unit, were applied individually by Group 1 and cooperative learning by Group 2. These activities required students to use computational thinking skills and code using the Python programming language. The study involved 34 students from the 6th grade of a private middle school located in the capital city of Turkey. The Computational Thinking Test (CTt) and an academic achievement test were used as pre-tests and post-tests to monitor students' computational thinking skills and academic performances. Additionally, computational problem-solving activities were scored to track the progress of students' computational thinking abilities. Non-parametric Mann Whitney U and Wilcoxon T-tests were utilized to analyze the progression of pupils' computational thinking abilities and academic success, and ANCOVA was used to analyze CTt scores. Qualitative data were collected through semi-structured interviews at the end of the process to determine students' views on the computational problem-solving process. Results revealed a significant increase in students' academic achievement and computational thinking skills in both groups. A comparison of post-test scores showed no significant difference between groups. It is anticipated that the research results will make meaningful contributions to the literature concerning the progress of computational thinking skills in secondary school students.

Keywords: science education, computational thinking, cooperative learning, middle school students.

Introduction

Developing students' 21st-century skills and giving them greater importance within curricula has gained increased emphasis in recent years (Nouri *et al.*, 2020). The International Society for Technology in Education (ISTE, 2016) asserts that the role of computers and automation systems in our lives is expanding and highlights the necessity to enhance students' abilities as "*digital citizens, knowledge constructors, innovative designers, computational thinkers, creative communicators, global collaborators, and empowered learners*". The International Association for the Evaluation of Educational Achievement (IEA) evaluates the computer and information literacy levels and computational thinking skills of students as part of the International Computer and Information Literacy Study (ICILS) (Fraillon, *et al.*, 2019). Within the context of ICILS, computer and information literacy refers to a student's capacity to apply computer technology to gather, organize, and manipulate information. Computational thinking skills are the cognitive skills used for programming computers or digital devices and developing applications (Fraillon *et al.*, 2019). Wing (2008), a key figure in the debate around integrating computational thinking into education programs beyond computer science courses, believes that everyone should learn to use the methods and approaches of computer science. According to Gülbahar *et al.* (2019), individuals with computer-like information-processing abilities can display competence in problem-solving, critical thinking, and lifelong learning. In particular, there is substantial research in the context of secondary school science education investigating the teaching of computational thinking (Basu *et al.*, 2016; Grover, 2017; Ogegbo & Ramnarain, 2022). While these discussions continue, there is inadequate evidence in the literature regarding the efficacy of the collaborative approach to cultivate computational thinking skills in secondary school students.

Computational science education is defined as an interdisciplinary field that includes the disciplines of mathematics, natural sciences, computer science, and educational sciences and benefits from the power of all these fields (Yasar, *et al.*, 2000). In this context, computational physics education (Ayars, 2013; Backer, 2007; Chabay & Sherwood, 2008; Landau, et al, 2009; Landau, *et al.*, 2015), computational chemistry education (Perrin et al, 2014: Pickard *et al.*, 2014; Johnson & Engel, 2011; Miller *et al.*, 2014) and computational biology education (Fox & Ouellette, 2013; Rubinstein & Chor, 2014) have gained momentum in recent years. Landau, *et al.* (2009) suggested that computational science education should be built on problem-solving and stated that although it includes fewer theoretical science lessons compared to pure science education, it offers much more effective and meaningful learning due to the integration of science, mathematics, and computer science. Therefore, this study investigates the impact of the collaborative approach in computational problem solving on the academic achievement of middle school students in science courses, as well as their computational thinking skills development.

Theoretical Framework

Computational Thinking

Wing (2008) defines computational thinking as "*... is taking an approach to solving problems, designing systems, and understanding human behaviour that draws on concepts fundamental to computing*" (p. 3717). According to Wing, the main purpose of acquiring this skill is to provide a multidimensional thinking skill in solving problems in different areas of life. Barr *et al.* (2011) considered computational thinking as a problem-solving process and formulated it to include the following skills:

- *"Formulating problems in a way that enables us to use a computer and other tools to help solve them".*
- *"Logically organizing and analyzing data".*
- *"Representing data through abstractions, such as models and simulations".*
- *"Automating solutions through algorithmic thinking (a series of ordered steps)".*
- *"Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources".*
- *"Generalizing and transferring this problem-solving process to a wide variety of problems"* (p. 21).

Many studies in the literature define computational thinking as a problem-solving process (Grover & Pea, 2018; Weintrop, *et al.*, 2016; Yadav, *et al.*, 2014; ISTE, 2016), and it usually encompasses several sub-skills such as decomposition, abstraction, algorithm design, automation, data collection, data analysis, data representation, simulation, parallelization, and generalization (Barr & Stephenson, 2011; Conery *et al.*, 2011; Park & Jeon, 2015). The International Computer and Information Literacy Study (ICILS) 2018 assessment framework defines computational thinking (CT) as "*an individual's ability to recognise aspects of real-world problems that are amenable to computational formulation and to evaluate and develop algorithmic solutions to these problems so that the solutions can be operationalised with a computer*" (Fraillon *et al.* 2019, p. 27). According to the ICILS 2018 framework, ICT consists of two parts: conceptualizing problems and operationalizing solutions. The three dimensions that make up the conceptualizing problems part are: knowing about and understanding digital systems, formulating and analyzing problems, and collecting and representing relevant data. The two dimensions that make up the operationalization of solutions are the planning and evaluation of solutions and the development of algorithms, programs, and interfaces.

Computational Problem Solving

Computational science is an interdisciplinary field comprising the disciplines of mathematics, natural sciences, computer science, and educational sciences, leveraging the

strengths of each (Yasar *et al.*, 2000). Landau *et al.* (2015) proposed that computational science education should be founded upon problem-solving. They noted that despite involving fewer theoretical science lessons than pure science education, computational science education facilitates more efficient and significant learning by incorporating science, mathematics, and computer science. At present, it is necessary to discuss how to provide computational science education in secondary schools, as the current literature predominantly concentrates on science education in upper secondary schools (Pickard *et al.*, 2014; Miller *et al.*, 2014). At this point, computational problem-solving appears as an effective tool. Problem-based learning is a curriculum-based, student-centered approach that enables individuals to conduct research and inquiry, combine theory and practice, and use their knowledge and skills to solve existing problems (Savery, 2015). Besides computational thinking involves formulating problems appropriately to solve them with tools and methods such as computers and data analysis, finding possible solutions, and using these solutions to solve other similar problems (Barr, *et al.*, 2011; Wing, 2008). In this respect, it can be stated that computational thinking is a problem-solving process closely related to computer science, and according to Grover and Pea (2018), problem formulation is an important part of this problem-solving process. Since a computer is not required to formulate a solution in a problem-solving process, computational thinking can be taught without using a computer. For this reason, there are different computational thinking teaching approaches from preschool to high school level, either plugged-in or unplugged (Lee & Junoh, 2019). In summary, computational problem-solving does not only involve the act of computer programming. According to Dierbach (2012), two things are needed to solve a problem computationally: a solution proposal that covers all relevant aspects of the problem and an algorithm that can solve the problem using this solution proposal. Dierbach (2012) defines the computational problem-solving process as follows:

1. *Analyze problem*
    a. *Clearly understand problem*
    b. *Know what constitutes a solution*
2. *Describe data and algorithms*
    a. *Determine what time of data is needed*
    b. *Determine how data is to be structured*
    c. *Find and/or design appropriate algorithms*
3. *Implement program*
    a. *Represent data within programming language*
    b. *İmplement algorithms in programming language*
4. *Test and debug*
    a. *Test the program on a selected set of problem instances*
    b. *Correct and understand causes of any errors found "*

Cooperative Learning

Cooperative learning is a teaching strategy in which students undertake their own and their peers' learning by working in small groups (Johson & Johnson, 1999; Slavin, 1999). For an activity to be cooperative, it must have five basic elements: positive interdependence, individual responsibility, face-to-face stimulating interaction, social skills, and group processing (Johson & Johnson, 1999). From this point of view, there are different perspectives on the source of student success in cooperative learning. The motivational perspective argues that students' motivation to complete the task is the driving force and believes that other processes are driven by motivation. According to the social cohesion perspective, the effect of cooperative learning on student achievement is mediated by group spirit and cohesion, and believes that the quality of group interaction is largely determined by group cohesion. The cognitive perspective argues that interactions between students will increase student achievement for reasons related to the mental processing of information rather than motivations, and according to this perspective, the opportunity for students to learn from each other mediates the construction of knowledge (Slavin, 2015). The literature provides evidence that collaborative learning positively supports students' academic achievement (Dheeraj & Kumari, 2013; Parveen & Batool, 2012; Vaughan, 2002).

Purpose and Significance of the Study

Since computational thinking is defined as a problem-solving approach and the effect of cooperative learning on academic achievement and attitude is well known, the effect of cooperative learning on teaching programming has been examined in some studies. Tsai (2002) explored the impact of strategic learning and cooperative learning on the computer performance, attitudes, and anxiety levels of junior high school students in Taiwan. The findings indicate a significant increase in computer anxiety among students in the cooperative learning group when compared to those in the control group. This reflects that the students tended to have higher anxiety towards using and learning computers in cooperative learning. In another study, Garcia (2021) investigated the impact of collaborative learning through the use of the Jigsaw Technique (JT) on the education of university-level novice programmers in computer programming. As a result of the research, he reported that the attitudes and academic achievement of students exposed to collaborative teaching increased. Li *et al.* (2022) explored the role of socially shared regulation on computational thinking performance in cooperative learning. In the experimental group students learned under the socially shared regulation of learning (SSRL) condition. The results showed that the students in the experimental group significantly outperformed their counterparts. Zhou and Tsai (2023) explored the effects of socially shared learning regulation (SSRL) on the computational thinking, learning motivation, engagement, and academic achievement of university students in collaborative learning by teaching (CLBT). Based on the results, the scores of the experimental class with SSRL in CLBT were significantly higher than those of the control class. Furthermore,

the students in the experimental class significantly improved their computational thinking (algorithmic thinking, critical thinking, and problem-solving), intrinsic motivation, engagement, and academic achievement compared to those in the control class. As can be seen, the studies in the literature have mostly focused on college and university-level students, there is not enough evidence on the performance of younger students in cooperative learning. Furthermore, the context of the studies focuses on computer science teaching, whereas in this research we are experimenting with an approach that integrates computer science with secondary school science. Another rationale for conducting the study is that students' motivation for cooperative learning in Turkey may be different due to cultural reasons. We believe that the findings of this study are important for future studies. In this study, we aimed to investigate the impact of cooperative learning on the computational thinking skills and academic performances of middle school students in the computational problem-solving approach.

Research Questions

The research problem for this research aims to investigate whether there is a significant statistical difference in academic achievement and computational thinking skills when comparing groups who complete the computational problem-solving approach collaboratively and individually at the 6th-grade level.

RQ 1. Is there a statistically significant difference between the academic achievement test pre-test and post-test scores of the students in the groups in which the computational problem-solving approach was carried out collaboratively and individually?

RQ 2. Is there a statistically significant difference between the computational thinking test pre-test and post-test scores of the students in the groups in which the computational problem-solving approach was carried out collaboratively and individually?

RQ 3. What are the student reflections of students in collaborative and individual groups about the computational problem-solving process?

Method

In this study, we aimed to investigate the impact of collaborative and individual computational problem-solving approaches on the academic achievement and computational thinking skills of middle school students. To achieve this, we employed a pretest-posttest control group design as part of the quasi-experimental method (Fraenkel *et al.*, 2012) and this design is depicted below.

To determine whether computational problem solving improves students' academic achievement and computational thinking skills, classes from 6th grade were selected as individual and cooperative study groups. The same program and measurement tools were applied to both study groups under the same conditions. The developed program

Table 1
Research design

	Pre-test	Treatment	Post-test
Group 1 (Individual)	Computational Thinking Test (CTt) Academic Achievement Test	Computational Problem Solving (Individual)	Computational Thinking Test (CTt) Academic Achievement Test
Group 2 (Cooperative)	Computational Thinking Test (CTt) Academic Achievement Test	Cooperative Computational Problem Solving	Computational Thinking Test (CTt) Academic Achievement Test

is related to the density topic of matter and heat unit and covers the 6th-grade level acquisitions. In addition, the student standards are defined by the ISTE (2016). The program based on computational problem-solving practices was implemented in both groups under the same conditions. In Group 1, a cooperative learning approach was used, while in Group 2, students worked individually. To eliminate the threat of internal validity, the applications were conducted by a science teacher and a computer teacher in both groups. The science teacher had no prior knowledge of Python and computational problem-solving. Therefore, training was given to the teacher before the applications. This training was carried out through the activities developed for the students. The Python coding activities of the students were carried out in the computer laboratory under the supervision of the science teacher and the computer teacher. The computer teacher had prior knowledge of block coding, algorithms, and Python.

Participants

The study group for this research was determined using a convenient sampling method. The research was conducted with two 6th-grade classes at a private secondary school in the Capital city of Turkey during the 2022–2023 academic year spring semester. The current study involved a total of 34 students, comprising 10 girls and 6 boys from group 1, and 7 girls and 11 boys from Group 2, as well as the science teacher of these classes. The age range of the students in the classes selected by the purposive sampling method is 12–13 years. The ratios of male and female students in the study groups were not intervened. The necessary permissions were obtained from the Provincial Directorate of National Education and the school administration. In the current study, it was determined that students had prior knowledge about block coding and that students played block coding games within the scope of the computer course. The distribution of the students within the study group is outlined in Table 2.

The age range of the students in the study group was 12–13 years. There was no definite ratio in the number of boys and girls, and the groups were not intervened in terms of gender. The study was carried out by obtaining the necessary permissions from the Provincial Directorate of National Education and the school administration regarding the study to be carried out after the research on the determined school. Semi-structured

Table 2
Study group

	Girls	Boys	Total
Group 1 (Individual)	10	6	16
Group 2 (Cooperative)	7	11	18
Total	17	17	34

interviews were conducted with the science teacher and computer teacher face-to-face or online at a specified time.

Implementation

In the current research, a program was designed by considering the outcomes of the 2018 science curriculum (MoNE, 2018) and ISTE (2016) student standards together. The four objectives of the "*Density*" topic of the "*Matter and Heat*" unit in the 6th-grade level of the science curriculum were taken as a basis. In addition, four standards under the title of "computational thinker" in the ISTE Student Standards were used. These standards and the outcomes taken from the science curriculum are presented in Table 3. A total of eight lessons containing these objectives were developed. The first 5 lessons in the prepared program were conducted by the science teacher in the science classroom. The researchers participated in these lessons as observers and supported the science teacher to carry out the implementation as planned. The last three lessons included computational problem-solving applications and students were required to code using Python. Therefore, these lessons were conducted in the computer laboratory under the leadership of a computer teacher (see Appendix 3). The science teacher and a researcher were involved in this process and supported student work. The total implementation lasted 6 weeks including the pre-test and post-test applications. In each lesson, the tasks to be completed by the students were defined (see Table 3). Since the lessons lasted 35 minutes in the school where the implementation took place, all implementations were determined to fit this duration. In each lesson, special areas were defined for students to write their solutions related to the computational problem-solving stages along with the instructions prepared for them. Some examples of the designed activity sheets are shown in the appendix.

The Python coding activities of the students were carried out through `https://www.onlinegdb.com`. The student activity sheets consisted of steps appropriate to the computational problem-solving process and each step required the use of one or more of the computational thinking skills (see Table 3). Students carried out their computational problem-solving processes on these activity sheets. The last three activity sheets, "What is it made of?", "Which liquid where?" and "Who sinks and who swims?" were collected and scored to provide evidence about the development of students' computational

thinking skills, because, in the last three activities, students were expected to use their previously acquired computational problem-solving skills in problem situations. The problem situations were prepared following the objectives in the science curriculum and activities in the science textbook were redesigned according to the computational problem-solving process.

In Group 1, activity sheets (see Appendix 2) were provided for each student in Group 1 and the students completed these activities individually. During this time, care was taken to answer the students' questions individually as much as possible. Students did not discuss or exchange ideas in small groups. One computer was allocated to each student in the computer lab. The students in Group 1 completed their studies on the computers individually. In Group 2, students were divided into groups of 4 or 5 students at the beginning of the process. An activity sheet was provided for each group. Students progressed by discussing and making joint decisions at each stage of the process. In Group 2, the cooperative learning groups were formed by the science teacher and special attention was paid to heterogeneous groups in terms of gender and academic achievement. In Group 2 (cooperative), a computer was provided to each group of students, and after the students decided on their algorithm designs together, they coded on this computer. To monitor the development of students' computational problem-solving skills, the framework defined by Dierbach (2012) was taken as a basis. This framework consists of analyzing a problem, describing data and algorithms, implementing a program, and testing and debugging dimensions. Therefore, in this current research, to monitor students' computational problem-solving skills, we scored students' performances in three dimensions; algorithm design, flow chart, and Python coding. Algorithm design refers to students' development of a systematic for problem-solving based on the stages of analyzing the problem and describing data. Flow chart refers to creating the algorithm of this solution and visualizing (abstraction) it. Finally, python is associated with coding, implementing a program, and testing and bugging.

The elements that could threaten the internal validity of the research were carefully controlled. In both Group 1 and Group 2, the pre-test and post-test and all the applications including the planned activities were carried out by the science teacher of the school with the support of the computer teacher. The researchers were only involved in the process as observers and only provided support to the teachers when necessary. Before the implementation, the teachers were informed about the whole process, and their permissions were obtained. Since the science teacher's prior knowledge about computational thinking was very low, she was given a short training and the points to be considered while conducting the studies were explained. In Group 2 (cooperative), student study groups were formed by the science teacher before the implementation, and care was taken to ensure that the groups were heterogeneous. Students did not change their groups throughout the process. In both study groups, the same activities were carried out simultaneously by the same teachers.

Table 3

Program design and objectives

Lessons	Objectives and applications	Science Education Program Goals (MoNE, 2018) & ISTE Standards	CT Skills
Lesson 1	***Pattern identification and abstraction:*** A table containing the mass, volume, and density of some substances was presented to the students. The students tried to discover the relationship between these properties of substances. Then they were asked to define the concept of "density" in their own words. Then they tried to express this relationship mathematically.	F.6.4.2.1. Defines density. ISTE / 1.5.a. Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models, and algorithmic thinking in exploring and finding solutions.	Pattern Recognition, Abstraction
Lesson 2	***Algorithms and flow charts:*** Algorithms and their types were introduced to the students. Students' algorithm writing skills were developed with the activities "Write the algorithm of the things you do from the moment you wake up until you come to school" and "Write an algorithm to prepare your favorite sandwich".	- ISTE / 1.5.c. Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.	Pattern Recognition, Algorithm Design
Lesson 3	***Using algorithms and flow charts in problem-solving.*** Students were given a problem situation about recycling. The students were asked to prepare an algorithm to solve the problem of incorrect disposal of garbage in recycling bins and to express it with a flow chart.	- ISTE / 1.5.c. Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.	Decomposition, Pattern Recognition, Algorithm Design
Lesson 4	***Turning algorithms into pseudocodes:*** Within the scope of the student camp activity, students were given a list of rules to be followed, firstly, they were provided to write an algorithm by examining these guidelines. Then, they were trained to convert these algorithms into pseudo codes. In this activity, "*if*" condition was used for the first time.	- ISTE / 1.5.d. Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.	Algorithm Design, Coding and debugging
Lesson 5	***Introduction to Python:*** Students were introduced to the Python coding language and coding activities were carried out at a simple level. Coding activities were carried out in the computer lab of the school, students wrote Python codes on `https://www.onlinegdb.com`, and student work was monitored on this platform. In this study, especially, print, sort, if, else structures, and basic arithmetic operations were studied.	- ISTE / 1.5.d. Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.	Algorithm Design, Coding and debugging

Lesson 6	***Computational Problem Solving 1: What is it made of?*** The students tried to find out what the objects with known values were made of. The students were first asked to examine all the data in the given density table, sort out the unnecessary data, and then analyze the problem by dividing it into small pieces. Then, they prepared an algorithm to determine which material the objects were made of and wrote Python code according to this algorithm. In this activity, students were expected to use "int", "input", "print" and "if"	F.6.4.2.1. Defines density. a. Emphasize that density is a distinguishing property of matter. b. Use g/cm3 as the unit of density. F.6.4.2.2. Calculates the densities of various substances as a result of the experiments he/she designs. ISTE / 1.5.a. Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions. ISTE / 1.5.b. Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making. ISTE / 1.5.c. Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving. ISTE / 1.5.d. Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.	Decomposition, Pattern Recognition, Abstraction, Algorithm Design, Coding and debugging
Lesson 7	***Computational Problem Solving 2: Which liquid where?*** In this problem, students were asked to solve how four immiscible liquids would be ordered when placed in the same container. Students were presented with a table of liquids and their densities, and they were expected to create an algorithm for the positions of four randomly selected liquids in the container and write Python code according to this algorithm. They were especially expected to use "list" and "sort" structures to solve the problem.	F.6.4.2.3. Compares the densities of insoluble liquids in each other by experimenting. ISTE / 1.5.a. Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions. ISTE / 1.5.b. Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making. ISTE / 1.5.c. Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving. ISTE / 1.5.d. Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.	Decomposition, Pattern Recognition, Abstraction, Algorithm Design, Coding and debugging

Continued on next page

Table 3 – continued from previous page

Lessons	Objectives and applications	Science Education Program Goals (MoNE, 2018) & ISTE Standards	CT Skills
Lesson 8	***Computational Problem Solving 3: Who sinks and who swims?*** The students were given a table containing the mass and volume values of the objects named K, L, M, and N and were expected to write a code to determine which objects would float in water and which objects would sink in water. In the coding phase, students were expected to write a code that calculates the density of the object by using the mass and volume values entered and decides whether the object will sink or float by comparing it with the density of water. Students especially used "input", "if", and "print" structures.	F.6.4.2.1. Defines density. a. Emphasize that density is a distinguishing feature of matter. F.6.4.2.2. Calculates the densities of various substances as a result of the experiments he/she designs. ISTE / 1.5.a. Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions. ISTE / 1.5.b. Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making. ISTE / 1.5.c. Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving. ISTE / 1.5.d. Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.	Decomposition, Pattern Recognition, Abstraction, Algorithm Design, Coding and debugging

Table 4

Computational Thinking Test Pilot Application Descriptive Statistics

Test Statistics	CTt	Academic Achievement Test
N	120	48
Min.	7	8,00
Max.	28	20,00
Mean	14,43	14,43
Standard deviation	4,36	3,97
Mean Item Difficulty Index	0,65	0,72
Mean Item Discrimination Index	0,36	0,45
KR-20 (Alpha)	0,75	0,86
KR-21	0,69	0,78

Data collection and analyses

To examine the effect of the program developed in this study on the computational thinking skills and academic achievement of secondary school students, the computational thinking test and academic achievement test were applied as pre-test and post-test. Academic achievement test pre-test and post-test applications (paper-pencil tests) were carried out in the classroom environment and lasted 30 minutes. The pre-test and post-test applications of the computational thinking test were prepared in the form of Google Forms and applied in the computer laboratory. The applications lasted 40 minutes. In addition to these, the students' computational problem-solving activity papers were collected and scored and the development of students' computational thinking skills was monitored. For this purpose, a rubric developed within the scope of this research was used. Information about the data collection tools is detailed below.

Computational Thinking Test (CTt): The Computational Thinking Test (CTt) developed by Román-González, Pérez-González and Jiménez-Fernández (2016) consists of 28 multiple-choice items that include problem situations based on block-based coding on the code.org website. CTt was designed according to guidelines for teaching and measuring CS (Buffum *et al.*, 2015), and defines computational thinking as "... the ability to formulate and solve problems based on fundamental concepts of computation and using the inherent logic of programming languages" (Román-González, *et al.*, 2017) The test was administered to a sample of 1,251 Spanish students from 5th to 10th grades and the Cronbach's Alpha reliability of the test was found to be $\alpha = 0.793$. The average difficulty index of the test items was reported as $p = 0.59$. For the adaptation of the scale into Turkish, 120 secondary school students studying in a different private school were reached. The item statistics obtained in the pilot study are presented in Table 3. Although there was a slight decrease in item statistics after the Turkish adaptation, it was evaluated that the test is a valid and reliable tool for monitoring students' computational thinking skills.

Academic Achievement Test: Within the scope of the research, an academic achievement test was developed to monitor students' academic achievement in density. The test was designed as a multiple-choice exam consisting of 20 questions. To ensure the content validity of the test, the opinions of two field experts were obtained and the test items were revised in line with these opinions. The test was applied to an equivalent sample ($n = 48$) before the application and item statistics were checked. The descriptive analysis results of the pilot application of the academic achievement test are given in Table 4.

Computational Problem Solving (CPS) Rubric: The activity papers of CPS 1, CPS 2, and CPS 3 in which the students were involved in computational problem-solving practices were collected and scored with a rubric. Algorithm design, constructing flow charts, and Python coding phases of the activities were scored as separate sub-dimensions in cooperative and individual study groups. The rubric is given in Appendix 1.

Semi-structured Interviews: Qualitative data were collected through semi-structured interviews at the end of the process in order to determine the views of the students participating in the research (cooperative and individual) on the computational problem-solving process. Semi-structured interview questions are presented in Appendix 2.

In the study, SPSS 24 was used to analyze all the data obtained from the tests. In the analyses, it was determined that the data obtained from the academic achievement test were not suitable for normal distribution. For this reason, the Mann Whitney U Test and Wilcoxon Signed Ranks Test were used to determine whether the difference between the mean academic achievement scores in individual and collaborative groups was significant. It was determined that the data obtained from the computational thinking test were coherent with the normal distribution. In addition, since a significant difference was found between the CTt pre-test scores of the groups, the ANCOVA test was applied to determine whether there was a statistically significant difference between the post-test scores. Nonparametric tests were applied for the scores obtained from the information processing problem-solving rubric.

Internal and External Reliability, Ethics

The necessary permissions were obtained from the Hacettepe University Ethics Commission and Ankara Provincial Directorate of National Education. Before the research, students and parents were informed about the applications, and their permission to participate in the application was obtained. The measurement tools applied within the scope of the study were appropriate in terms of validity and reliability and appropriate analysis techniques were applied to the collected data.

Findings

The academic achievement test developed in this study was used to monitor the academic achievement of middle school students in computational problem-solving. Firstly, the normality of the data obtained from this test was analyzed. Table 5 shows the descriptive statistics values for the academic achievement test pre-test and post-test applications.

The descriptive statistics values of the academic achievement test showed that the skewness and kurtosis values of the test were not suitable for normal distribution. More evidence was needed to examine the suitability of the academic achievement test pre-test and post-test scores of the groups for parametric statistics. For this reason, normality

Table 5

Academic Achievement Test Pre-Test – Post-Test Applications Descriptive Statistics

		Mean	sd	Min.	Max.	Skewness	Kurtosis
Pre-test	Individual	10,60	1,28	8	13	-0,53	0,87
	Cooperative	13,86	2,98	8	18	-0,30	-0,81
Post-test	Individual	18,82	1,60	14	20	-1,90	4,45
	Cooperative	19,36	1,15	16	20	-2,25	5,37

tests of the academic achievement test results were examined. The normality test results of the test are presented in Table 6.

The normality test results of the academic achievement test showed that the post-test scores of both groups were not suitable for normal distribution. Therefore, nonparametric tests were used to examine whether there was a significant difference between the academic achievement test pre-test and post-test scores of the groups. To monitor the students' computational thinking skills, the Computational Thinking Test (CTt) was used as a pre-test and post-test. Descriptive statistics values of the pre-test and post-test applications of the Computational Thinking Test (CTt) are given in Table 7.

According to Table 7, the skewness and kurtosis values of the pre-test and post-test scores of the Computational Thinking Test were coherent with normal distribution. However, normality tests of the test results were examined to obtain more evidence about whether the score distributions were suitable for parametric statistics. The normality test results of the pretest and posttest data are presented in Table 8.

Table 6

Academic Achievement Test Normality Test Results

Group		Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistics	sd	p	Statistics	sd	p
Pre-test	Individual	0,215	17	,036	0,896	17	,057
	Cooperative	0,162	14	,200	0,931	14	,318
Post-test	Individual	0,241	17	,010	0,748	17	,000
	Cooperative	0,355	14	,000	0,638	14	,000

Table 7

Descriptive Statistics of CTt Pre-Test and Post-Test Applications

		Mean	sd	Min.	Max.	Skewness	Kurtosis
Pre-test	Individual	15,13	3,63	8	21	-0,22	-0,65
	Cooperative	18,30	4,35	10	25	-0,47	-0,97
Post-test	Individual	19,75	4,46	9	26	-0,85	0,73
	Cooperative	21,06	3,60	14	26	-0,41	-0,45

Table 8

CTt Normality Test Statistics

Group		Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistics	sd	p	Statistics	sd	p
Pre-test	Individual	0,180	16	,173	0,956	16	,587
	Cooperative	0,203	17	,060	0,918	17	,138
Post-test	Individual	0,173	16	,200	0,948	16	,454
	Cooperative	0,133	17	,200	0,945	17	,387

When the normality results of the Computational Thinking Test were analyzed, it was determined that the data obtained from the pre-test and post-test applications of the groups were coherent with the normal distribution. For this reason, parametric tests were used to examine whether there was a significant difference between the scores obtained by the groups from the pre and post-tests of the Computational Thinking Test.

Findings Regarding Research Question 1

Since the pre-test and post-test results of the Academic Achievement Test of the students in Group 2 (cooperative) were not suitable for normal distribution, the pre-test and post-test scores were analyzed by the Wilcoxon Signed Rank Test. The test results are given in Table 9.

The number of subjects (N), Mean Ranks, and Sum of Ranks for Negative Ranks, Positive Ranks, and those with the same value (Ties) between the pre-test and post-test scores of the Academic Achievement Test are given in Table 8. Table 8 shows that the post-test scores of all students were higher than the pre-test scores (positive rank N = 14). The z value (-3,31) calculated to examine whether these variables were significant was found to be significant (p = 0,001; p < 0,05). Since the pre-test and post-test results

Table 9

Cooperative Group Academic Achievement Test Wilcoxon Signed Ranks Test Results

		N	Mean of Ranks	Sum of Ranks
Post-test – Pre-test	Negative Rank	0[a]	0,00	0,00
	Positive Rank	14[b]	7,50	105,00
	Ties	0[c]		
	Total	14		

a. Post-Test < Pre-Test
b. Post-Test > Pre-Test
c. Post-Test = Pre-Test

Table 10

Individual Group Academic Achievement Test Wilcoxon Signed Ranks Test Results

		N	Mean of Ranks	Sum of Ranks
Post-test – Pre-test	Negative Rank	0[a]	0,00	0,00
	Positive Rank	17[b]	9,00	153,00
	Ties	0[c]		
	Total	17		

a. Post-Test < Pre-Test
b. Post-Test > Pre-Test
c. Post-Test = Pre-Test

Table 11
Ranks of Group 1 and Group 2 Posttest Scores

		N	Mean of Ranks	Sum of Ranks
Post-test	Individual	17	14,50	246,50
	Cooperative	14	17,82	249,50
	Total	31		

of the Academic Achievement Test of the students in Group 1 (individual) were not suitable for normal distribution, the pre-test and post-test scores were analyzed with the Wilcoxon Signed Rank Test. The test results are given in Table 10.

Table 9 shows that the post-test scores of all students were higher than the pre-test scores (positive rank N = 17). The Z value calculated to examine whether these variables were significant or not was -3,64 and the significance was found to be p = 0,000 (p < 0,05). As a result, it was determined that the increase in the posttest scores of Group 1 was significant. Mann-Whitney U test was used to examine whether there was a significant difference between the Academic Achievement Test post-test results of the students in Group 1 and Group 2. The ranks of the post-test scores of the groups are given in Table 11.

The Mann-Whitney U test analyses (U = 93,50, p = 0,265) performed to examine the difference between the post-test scores of Group 1 and Group 2 showed that there was no statistically significant difference between the two groups in terms of post-test scores (Z = -1,115, p < .05).

Findings Regarding Research Question 2

To monitor the development of computational thinking skills of the students participating in the study, both CTt was used as a pre-test and post-test and the computational problem-solving activity sheets were scored and analyzed with the rubric. To examine whether there was a statistically significant difference between the CTt pre-test and post-test scores of the students in Group 1 and Group 2, the pre-tests of the groups were first analyzed. The results of the Independent Groups t-test conducted to test whether there was a statistically significant difference between the pre-test scores of the groups are given in Table 12.

Table 12
Independent Groups t-Test Results of CTt Pre-Test Results of Groups

		N	Mean	sd	df	t	p
Pre-test	Individual	16	15,13	3,63	31	-2,26	,030
	Cooperative	17	18,29	4,35			

Table 12 reveals that there is a statistically significant difference between the pre-test scores of the groups. ANCOVA was applied by determining the pre-test scores as covariates against the possibility that the possible difference between the post-test scores of the groups in the Computational Thinking Test was due to the pre-test scores. Before ANCOVA, test scores were examined for compliance with variance analysis assumptions. The data obtained from the Computational Thinking Test are continuous variables and the test scores are suitable for normal distribution. In addition, the homogeneity of the variances of the test scores was examined by Levene's test and it was determined that the variances of the post-test scores were homogeneous (F = 0,614, p = 0,439). ANCOVA results of the groups' posttest scores of the Computational Thinking Test are given in Table 13.

When Table 13 is analyzed, it is seen that when the pre-test scores of the groups are controlled, there is no significant difference between the post-test scores. This result shows that both individual and collaborative computational problem-solving practices increased the students' computational thinking skills, but at the end of the practice, there was no statistically significant difference between the computational thinking skills of the students in both groups. To monitor the development of students' computational thinking skills, computational problem-solving activities were also scored with a rubric. The descriptive statistical values of the scores of the students in both groups are presented in Table 14.

Wilcoxon Signed Ranks Test, one of the nonparametric statistics, was used to determine whether the change in the scores of the groups in the sub-dimensions of computational problem-solving was significant. Pairwise comparisons between the scores of the students in Group 1 on the sub-dimensions of computational problem-solving are given in Table 15.

When Table 15 is analyzed, it is understood that the scores obtained by the students in Group 1 from the sub-dimensions of computational problem-solving increased significantly. Although there was no significant difference between the first and second measurements in the sub-dimensions of algorithm design, flow chart, and python coding, it is understood that the students' scores increased significantly in the third measurement compared to the previous measurements. This difference can be interpreted

Table 13
ANCOVA Results of Groups' Computational Thinking Test Posttest Scores

	Sum of Squares	df	Mean of Squares	F	p	Eta Square
Corrected model	246,37	2	123,18	13,50	,000	,474
Intercept	138,86	1	138,85	15,22	,001	,337
Pre-test	232,25	1	232,25	25,45	,000	,459
Group	5,09	1	5,09	0,55	,461	,018
Error	273,70	30	9,12			
Total	11286,00	33				
Corrected Total	520,06	32				

Table 14

Descriptive Statistics of Groups' Scores of Computational Problem-Solving Sub-dimensions

		N	Mean	sd	Min.	Max.	Percentiles		
							%25	%50	%75
Individual									
CPS 1	Algorithm Design 1	16	6,19	0,91	5	8	5,25	6,00	7,00
	Flow Chart 1	16	6,75	1,23	5	9	5,25	7,00	7,75
	Python Coding 1	16	5,19	1,42	3	8	4,00	5,00	6,00
CPS 2	Algorithm Design 2	16	6,63	1,31	4	9	6,00	7,00	7,75
	Flow Chart 2	16	6,63	1,40	4	9	5,25	7,00	8,00
	Python Coding 2	16	6,31	1,25	5	10	5,25	6,00	7,00
CPS 3	Algorithm Design 3	16	7,69	1,62	5	10	6,00	8,00	9,00
	Flow Chart 3	16	7,69	1,13	5	9	7,00	8,00	8,75
	Python Coding 3	16	7,81	1,87	4	10	7,00	8,00	9,00
Cooperative									
CPS 1	Algorithm Design 1	18	7,00	0,59	6	8	7,00	7,00	7,00
	Flow Chart 1	18	7,22	0,42	7	8	7,00	7,00	7,25
	Python Coding 1	18	5,18	0,78	4	6	4,75	5,00	6,00
CPS 2	Algorithm Design 2	18	7,77	1,43	6	10	6,75	8,00	8,50
	Flow Chart 2	18	7,16	0,78	6	8	6,75	7,00	8,00
	Python Coding 2	18	6,44	1,09	5	8	5,75	6,00	7,25
CPS 3	Algorithm Design 3	18	8,55	1,46	6	10	8,25	9,00	9,25
	Flow Chart 3	18	8,83	0,75	8	10	8,00	9,00	9,25
	Python Coding 3	18	7,83	0,78	7	9	7,00	8,00	8,25

Table 15

Wilcoxon Test Results of Problem-Solving Sub-dimensions in Individual Group

		N	Mean of Ranks	Sum of Ranks	Z	p
Algorithm Design 1 – Algorithm Design 2	Negative Rank	3	5,00	15,00	-1,706	,088
	Positive Rank	8	6,38	51,00		
	Ties	5				
	Total	16				
Algorithm Design 2 – Algorithm Design 3	Negative Rank	2	3,50	7,00	-2,553	**,011**
	Positive Rank	10	7,10	71,00		
	Ties	4				
	Total	16				
Algorithm Design 1 – Algorithm Design 3	Negative Rank	1	4,00	4,00	-3,096	**,002**
	Positive Rank	13	7,77	101,00		
	Ties	2				
	Total	16				

Continued on next page

Table 15 – continued from previous page

		N	Mean of Ranks	Sum of Ranks	Z	p
Flow Chart 1 – Flow Chart 2	Negative Rank	6	7,50	45,00	-0,484	,628
	Positive Rank	6	5,50	33,00		
	Ties	4				
	Total	16				
Flow Chart 2 – low Chart 3	Negative Rank	1	8,00	8,00	-2,471	**,013**
	Positive Rank	11	6,36	70,00		
	Ties	4				
	Total	16				
Flow Chart 1 – Flow Chart 3	Negative Rank	1	7,50	7,50	-2,319	**,020**
	Positive Rank	10	5,85	58,50		
	Ties	5				
	Total	16				
Python Coding 1 – ython Coding 2	Negative Rank	1	3,50	3,50	-3,020	**,003**
	Positive Rank	12	7,29	87,50		
	Ties	3				
	Total	16				
Python Coding 2 – Python Coding.3	Negative Rank	1	2,50	2,50	-2,888	**,004**
	Positive Rank	11	6,86	75,50		
	Ties	4				
	Total	16				
Python Coding 1 – Python Coding 3	Negative Rank	0	,00	,00	-3,555	**,000**
	Positive Rank	16	8,50	136,00		
	Ties	0				
	Total	16				

as that the students got used to the process over time and were able to manage the computational problem-solving process better. The difference between the scores obtained by the students in Group 2, in which the applications were carried out collaboratively, from the sub-dimensions of computational problem solving was similarly analyzed by the Wilcoxon Signed Rank Test, and the results of the analysis are presented in Table 16.

According to Table 16, it is understood that the scores obtained by the students in Group 2 from the sub-dimensions of computational problem-solving increased significantly. Similar to the measurement results of Group 1 (individual), it was found that the students in Group 2 (cooperative) tended to get higher scores throughout the process. However, the striking result here is that the sub-dimensional scores of the students in the cooperative groups showed a significant difference from the first measurements. This situation can be interpreted as that students support each other's learning in cooperative groups. Since the development of students' computational thinking skills was observed in both groups, it was examined whether there was a difference between these developments. The difference between the scores obtained by the students in Group 1 and Group 2 from the sub-dimensions of computational problem

Table 16
Wilcoxon Test Results of Problem-Solving Sub-dimensions in Cooperative Group

		N	***Mean of Ranks***	***Sum of Ranks***	***Z***	***p***
Algorithm Design 1 – Algorithm Design 2	Negative Rank	4	2,50	10,00	-2,070	**,038**
	Positive Rank	7	8,00	56,00		
	Ties	7				
	Total	18				
Algorithm Design 2 – Algorithm Design 3	Negative Rank	4	5,50	22,00	-2,003	**,045**
	Positive Rank	10	8,30	83,00		
	Ties	4				
	Total	18				
Algorithm Design 1 – Algorithm Design 3	Negative Rank	4	4,00	16,00	-3,071	**,002**
	Positive Rank	14	11,07	155,00		
	Ties	0				
	Total	18				
Flow Chart 1 – Flow Chart 2	Negative Rank	4	4,00	16,00	-0,378	,705
	Positive Rank	3	4,00	12,00		
	Ties	11				
	Total	18				
Flow Chart 2 – Flow Chart 3	Negative Rank	0	,00	,00	-3,461	**,001**
	Positive Rank	15	8,00	120,00		
	Ties	3				
	Total	18				
Flow Chart 1 – Flow Chart 3	Negative Rank	0	,00	,00	-3,831	**,000**
	Positive Rank	18	9,50	171,00		
	Ties	0				
	Total	18				
Python Coding 1 – Python Coding 2	Negative Rank	0	,00	,00	-3,508	**,000**
	Positive Rank	15	8,00	120,00		
	Ties	3				
	Total	18				
Python Coding 2 – Python Coding.3	Negative Rank	0	,00	,00	-3,852	**,000**
	Positive Rank	18	9,50	171,00		
	Ties	0				
	Total	18				
Python Coding 1 – Python Coding 3	Negative Rank	0	,00	,00	-3,874	**,000**
	Positive Rank	18	9,50	171,00		
	Ties	0				
	Total	18				

Table 17
Kruskal Wallis Test Results for Problem Solving Sub-dimensions in Individual and Cooperative Groups

Compare of Groups		N	Mean of Ranks	Chi-square	df	p
Algorithm Design 1	Individual	16	12,91	7,531	1	**,006**
	Cooperative	18	21,58			
	Total	34				
Algorithm Design 2	Individual	16	13,81	4,368	1	**,037**
	Cooperative	18	20,78			
	Total	34				
Algorithm Design 3	Individual	16	14,63	2,699	1	,100
	Cooperative	18	20,06			
	Total	34				
Flow Chart 1	Individual	16	15,69	1,327	1	,249
	Cooperative	18	19,11			
	Total	34				
Flow Chart 2	Individual	16	15,50	1,320	1	,251
	Cooperative	18	19,28			
	Total	34				
Flow Chart 3	Individual	16	12,44	6,605	1	**,003**
	Cooperative	18	22,00			
	Total	34				
Python code 1	Individual	16	17,31	0,012	1	,914
	Cooperative	18	17,67			
	Total	34				
Python code 2	Individual	16	16,50	0,330	1	,566
	Cooperative	18	18,39			
	Total	34				
Python code 3	Individual	16	18,50	0,326	1	,568
	Cooperative	18	16,61			
	Total	34				

solving was analyzed by the Kruskal Wallis Test and the results of the analysis are presented in Table 17.

Table 17 shows that only in the algorithm design sub-dimension, the students in the collaborative group showed higher performance, while there was no significant difference between the scores of the groups in the other sub-dimensions. In both groups (groups 1 and 2), students showed a significant increase in scores throughout the process. At the end of the process, we found that the scores achieved by the students did not differ significantly from each other. On the other hand, students working in cooperative groups were more advantageous. Especially at the beginning of the process, students in cooperative groups progressed faster by supporting each other in comprehending the computational problem-solving process.

Findings Regarding Research Question 3

During in-class observations, we observed that the students who worked cooperatively spent a little more time designing their algorithms and creating flow charts, but reached a common conclusion through discussions. Students who worked individually also created their algorithms correctly most of the time and progressed at their own pace. However, we noticed that they used less data when formulating problems. In the python coding dimension, we observed that students who were interested and excited about coding were more productive when working individually. Some motivated students working in collaborative groups requested to work individually because they wanted to express their ideas freely. This was an unexpected situation for us. The students who worked individually had more chances to try more as they worked without time constraints while writing their codes with Python. They spent discussion time trying different situations in coding. This was also true for debugging; in collaborative groups, the source of the error was discovered more quickly, but it was observed that each student wanted to try different ideas when the code gave an error, which led to a loss of time. According to the observations made during the lesson, it was observed that the interest and motivation of the students working in co-operation were higher.

The results obtained from the interviews with the students showed that the students in both groups had a positive view of the process. They stated that the anxiety they experienced about coding at the beginning of the process decreased during the process. One of the students expressed this situation as follows; "*I think it was fun, the lessons were different, I was scared when you said that we would solve problems by coding in this lesson, but I think it was easy, sometimes I try to code something myself now*" (G2/S3 – Group 2 – Student 3). In the interviews, the opinions of the students about what they had the most difficulty in the process were collected. The students mostly stated that they had difficulty in finding the source of the error (debugging) if the codes they wrote did not work. For example, one of the students expressed himself as follows; "*I had the most difficulty when the codes I wrote did not work, I asked for help from my teachers when I could not run the code, but there were also times when I solved it myself, then I felt very happy"* (G1/S4). Although we provided Python coding language training to the students throughout the process, they made mistakes while transferring correctly designed algorithms to the code. The errors were usually script-related and this problem could not be fully resolved during the process. In addition, the students stated that they had problems in using the symbols in the flow chart. According to the results obtained from the interviews, the flow chart step was the most boring part for them. They stated that coding on the computer (python coding step) was cooler. Students often expressed that they enjoyed the python coding step much more and as the activities progressed, it was observed that they preferred to use the time they allocated from algorithm design and flow chart steps in the coding part. The computer teacher stated that the students shared their questions with him during lunch breaks and even experimented on the codes they wrote in computer classes where the application was not carried out. When the students were asked to make critical reflections during the interviews, one of them responded as follows: "*we were able to do the algorithm design and flowcharts with my friends, but*

it would be better if I had the chance to do the coding myself" (I2/S13). These kinds of views were predominant and we interpreted this as students wanting to be alone on the computer.

The students' experiences also supported their interest in computers. During the interviews, a student in Group 2 stated as follows; '*the computer lesson is very fun and, in the lesson, hmmm I started to try different codes on the computer. Maybe I will do something myself in the future, maybe I will do something like a web page*' (G2/S7). In addition, the students thought that the experience they had and the skills they acquired were valuable. We observed that some students' career expectations could be shaped in this direction. The school we conducted the study was a private school and students and families were aware of this issue. This may be a factor for students' interest, but during the interviews, a student expressed himself as follows; "*yes, I think coding will be of great importance in my future life, my family supports me in this regard, I already liked science class, but it is more fun now*" (G1/S6). In our interviews, we determined that the students in both groups were interested in coding. This process was very enjoyable for them. The clearest finding that we determined by the purpose of the study was that although group work was effective, students demanded and preferred to work individually in the coding step.

Results and Discussions

In this study, we aimed to investigate the impact of cooperative learning on the computational thinking skills and academic performances of middle school students in the computational problem-solving approach. Before the study, we had predicted that the computational thinking skills and academic achievement of the students would be higher in the group in which the computational problem-solving instruction was carried out cooperatively because according to the cognitive perspective of cooperative learning, the opportunity for students to learn from each other mediates the construction of knowledge (Slavin, 2015). According to Gillers (2014), in cooperative learning, students need to coordinate group interactions and also take responsibility for other members to learn. According to cooperative learning, when students understand that individual contributions improve the group's performance, they are willing to carry out their responsibilities. However, the results of the study showed that the academic achievement and computational thinking skills of students in both individual and cooperative groups increased significantly throughout the implementation. When the academic achievement post-test scores of the groups were compared, no significant difference was found between the post-test scores. In other words, the academic achievement of the students increased at a similar level in the groups in which the collaborative and individual applications were carried out. When the development of the groups' computational thinking skills was analyzed, no statistically significant difference was found between the post-test scores when the pre-test scores were controlled. This result indicates that both individual and collaborative computational problem-solving practices increased students' computational thinking skills, but there was no statistically significant difference between the computa-

tional thinking skills of the students in both groups at the end of the practice. The results obtained from the scoring of the activity papers showed that the scores of the students in both groups in the sub-dimensions of computational problem solving increased significantly, only in the algorithm design sub-dimension the students in the collaborative group showed higher performance, and there was no significant difference between the scores of the groups in the other sub-dimensions.

Although these findings are similar to the findings of reference studies in the literature (Li *et al.*, 2022; Zhou & Tsai, 2023), there are some differences. Studies in the literature reveal that students' computational thinking skills and academic achievement increase significantly in favor of the collaborative group. The findings of this study did not reveal a statistically significant difference between cooperative and individual teaching groups. One of the reasons for the lack of the expected difference between the groups may be that the applications carried out were found to be new and interesting for both student groups. Proponents of problem-based learning assume that the problem-solving process is an approach that increases student motivation and differences in student motivation can be measured by situational interest (Rotgans & Schmidt, 2012). Situational interest is a person's immediate emotional response to certain stimuli in the learning environment (Hidi 1990; Mitchell 1993) and can be increased by external factors such as well-organized content and challenging problem situations. In particular, we believe that the coding activities carried out in the computer lab supported the situational interest of the students, which increased their motivation towards the process. We expect that our prejudices about the cooperative learning group may be justified with longer-term studies.

According to the data obtained from in-class observations and interviews, students working in collaboration completed the algorithm and flow diagram steps much faster. Students working individually also needed much more time, although they designed their algorithms correctly. On the other hand, the demand for individual work was quite dominant in the Python coding step. Students working individually had the opportunity to test their codes much more. In the cooperative groups, when the codes they wrote gave errors, each student wanted to try his/her solution for debugging, which resulted in a loss of time. According to the observations made during the lesson, we observed that the students who worked collaboratively had higher interest and motivation towards the lesson. This intuition was strengthened by the students' effort to solve the problem together with their friends and their acceptance within the group regardless of the result. Studies in the literature reveal that the problem-solving approach can be effective for students to solve the problems they encounter while coding (Scherer *et al.*, 2020). Uysal (2014) stated that problem-solving teaching methods can also effectively improve the academic performance and problem perception of students learning coding. However, although algorithmic thinking seems to be quite easy in terms of structure determining the instructions and necessary steps is a process that is quite challenging and requires patience. To reach the result, students need to continue their studies with diligence and determination. In this process, trying many times and not achieving success causes many people to give up and give up the steps (Korkmaz *et al.*, 2017). Cooperative learning can be a solution to this problem by supporting students' motivation.

In the context of programming, computational thinking is considered problem-solving (Kalelioğlu *et al.*, 2016), and problem-solving skills are often associated with non-verbal intelligence (Tsavara *et al.*, 2022). For example, Marinus *et al.*, 2018 found a positive relationship between programming ability and non-verbal intelligence. Similarly, Çiftci and Bildiren (2020) found positive effects of coding lessons on children's non-verbal cognitive abilities. In our study, in parallel with these findings, we found that cooperative learning was more effective in the steps where students' verbal interactions were supported (algorithm design and flow chart). In Python coding, that is, in the step where non-verbal intelligence was utilized, students preferred individual work more. From this point of view, it should be taken into consideration that the measurement tools we used in our study, especially the CTt, tested programming skills individually. Although we conducted a collaborative process, we collected our quantitative data individually and these findings are limited in evaluating students' collaborative computational thinking skills.

Limitations and Implications

In this study, the activities developed for the density topic at the 6th-grade level were used. Although the students included in the study were familiar with block coding, they were unfamiliar with the computational problem-solving approach and Python coding language. Before starting the problem-solving process, the students were given exercises on algorithm design, pattern recognition, transforming the algorithm into pseudocode, and Python coding language. However, we think that more time should be allocated to these activities. In addition, the computer teacher and science teacher working together made it easier for us to manage the process because the science teacher did not have enough prior knowledge. We also trained the science teacher before the implementation, but she was not competent enough to manage the coding sections. In both groups, the attitude of the students towards the process was very positive and they were highly motivated about the implementation. We attribute this to the fact that this process was quite new to the students. They also enjoyed doing science-related work in the computer lab. Most students expressed that they wanted to sit alone at the computer. In the co-operative groups, who would sit at the computer was sometimes a matter of discussion. Although students in both groups showed improvement in academic achievement and computational thinking skills, we were a little disappointed that they wanted to be alone during the coding steps. We believe that researchers should conduct deeper research on this issue. The excitement that the students experienced in the coding part of the activities and the feeling of "I succeeded" when they worked on the codes provided them with great happiness and motivation, which caused them not to pay much attention to algorithms and flow diagrams and to want to quickly switch to writing code in Python. For this reason, a certain amount of time can be allocated for each learning outcome, and as gradual progress is made, the time allocated for algorithm and flow diagram learning can be decreased and the time allocated for code writing can be increased. This suggestion may change depending on the level of the class and the development process of the class.

References

Ayars, E. (2013). *Computational Physics with Python.* California State University, Chico.

Backer, A. (2007). Computational physics education with Python. *Computing in Science & Engineering, 9*(3), 30–33.

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

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning, 11*(1), 1–35.

Buffum, P.S., Lobene, E.V., Frankosky, M.H., Boyer, K.E., Wiebe, E.N., & Lester, J.C. (2015). A practical guide to developing and validating computer science knowledge assessments with application to middle school, in: Proc. 46th ACM Tech. Symp. Comput. Sci. Educ., pp. 622–627. http://dx.doi.org/10.1145/2676723.2677295

Chabay, R., & Sherwood, B. (2008). Computational physics in the introductory calculus-based course. *American Journal of Physics, 76*(4), 307–313.

Çiftci, S., & Bildiren, A. (2020). The effect of coding courses on the cognitive abilities and problem-solving skills of preschool children. *Computer Science Education, 30*(1), 3–21. https://doi.org/10.1080/08993408.2019.1696169

Corral, J. M. R., Balcells, A. C., Estevez, A. M., Moreno, G. J., and Ramos, M. J. F. (2014). A game-based approach to the teaching of object-oriented programming languages. *Comput. Educ. 73*, 83–92. https://doi.org/10.1016/j.compedu. 2013.12.013

Dheeraj, D., & Kumari, R. (2013). Effect of co-operative learning on achievement in environmental science of school student. *International Journal of Scientific and Research Publications, 3*(2), 1–3.

Dierbach, C. (2012). *Introduction to computer science using python: A computational problem-solving focus.* Hoboken: Wiley Publishing.

Fox, J. A., & Ouellette, B. F. (2013). Education in computational biology today and tomorrow. *PLoS Comput Biol, 9*(12), e1003391.

Fraenkel, J. R., Wallen, N. E., & Hyun, H. H. (2012). *How to design and evaluate research in education* (8th ed). McGraw-Hill Humanities/Social Sciences/Languages.

Fraillon, J., Ainley, J., Schulz, W., Friedman, T., & Duckworth, D. (2019). *Preparing for life in a digital world: the IEA International Computer and Information Literacy Study 2018 International Report*. Amsterdam: IEA.

Garcia, M. B. (2021). Cooperative learning in computer programming: A quasi-experimental evaluation of Jigsaw teaching strategy with novice programmers. *Education and Information Technologies, 26*(4), 4839–4856.

Gillies, R. (2014). Cooperative learning: developments in research. *International Journal of Educational Psychology, 3*(2), 125–140. https://doi.org/10.4471/ijep.2014.08

Grover, S. & Pea, R. (2018). Computational Thinking: A competency whose time has come. In Sentance, S., Carsten, S., & Barendsen, E. (Eds), *Computer Science Education: Perspectives on teaching and learning*, Bloomsbury.

Grover, S. (2017). Assessing Algorithmic and Computational Thinking in K-12: Lessons from a Middle School Classroom. In: Rich, P., Hodges, C. (eds) *Emerging Research, Practice, and Policy on Computational Thinking*. Educational Communications and Technology: Issues and Innovations. Springer, Cham. https://doi.org/0.1007/978-3-319-52691-1_17

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. Computer science education: *Perspectives on Teaching and Learning in School, 19*, 1257–1258.

Gülbahar, Y., Kert, S. B. & Kalelioğlu, F. (2019). Bilgi işlemsel düşünme becerisine yönelik öz yeterlik algısı ölçeği: geçerlik ve güvenirlik çalışması. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 10*(1), 1–29. https://doi.org/10.16949/turkbilmat.385097

Hidi, S. (1990). Interest and Its Contribution as a Mental Resource for Learning. *Review of Educational Research, 60*(4), 549–571.

Hurt, T., Greenwald, E., Allan, S., Cannady, M. A., Krakowski, A., Brodsky, L., Collins, M. A., Montgomery, R., & Dorph, R. (2023). The computational thinking for science (CT-S) framework: Operationalizing CT-S for K–12 science education researchers and educators. *International Journal of STEM Education, 10*(1), 1. https://doi.org/10.1186/s40594-022-00391-7

International Society for Technology in Education (ISTE). (2016). *ISTE standards for students.* Retrieved from: https://www.iste.org/standards/for-students

Johnson, L. E., & Engel, T. (2011). Integrating computational chemistry into the physical chemistry curriculum. *Journal of Chemical Education, 88*(5), 569–573.

Johnson, D. W., & Johnson, R. T. (1999). Making cooperative learning work. *Theory into Practice, 38*(2), 67–73.

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing, 4*(3), 583–596.

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.

Landau, R. H., Bordeianu, C. C., & Paez, M. J. (2009). Computational Physics with Python. In The 4th International Conference on Virtual Learning ICVL.

Landau, R. H., Páez, M. J., & Bordeianu, C. C. (2015). *Computational physics: Problem solving with Python.* John Wiley & Sons.

Lee, J., & Junoh, J. (2019). Implementing unplugged coding activities in early childhood classrooms. *Early Childhood Education Journal, 47*(6), 709–716.

Li, J., Liu, J., Yuan, R., & Shadiev, R. (2022). The influence of socially shared regulation on computational thinking performance in cooperative learning. *Educational Technology & Society, 25*(1), 48–60.

Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018). Unravelling the cognition of coding in 3-to-6-year olds. In *Proceedings of the 2018 ACM conference on international computing education research* – ICER '18, august (pp. 133–141). https://doi.org/10.1145/3230977.3230984

Miller, B.T., Singh, R.P., Schalk, V., Pevzner, Y., Sun, J., Miller, C.S., *et al.* (2014). Web-Based Computational Chemistry Education with CHARMMing I: Lessons and Tutorial. *PLoS Comput Biol 10*(7): e1003719. https://doi.org/10.1371/journal.pcbi.1003719

Mitchell, M. (1993). Situational interest: Its multifaceted structure in the secondary mathematics classroom. *Journal of Educational Psychology, 85*(3), 424–436.

MoNE. (2018). *Fen Bilimleri Dersi Öğretim Program: 3–8* [Science Education Program; 3–8]. Ankara: MoNE. Retrieved from: http://mufredat.meb.gov.tr/Dosyalar/201812312311937-FEN%20B□L□MLER□%20□□RET□M%20PROGRAMI2018.pdf

Nouri, J., Zhang, L., Mannila, L., and Noren, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Educ. Inq. 11*, 1–17. https://doi.org/10.1080/20004508.2019.1627844

Ogegbo, A. A., & Ramnarain, U. (2022). A systematic review of computational thinking in science classrooms. *Studies in Science Education, 58*(2), 203–230.

Olelewe, C. J., and Agomuo, E. E. (2016). Effects of B-learning and F2F learning environments on students' achievement in QBASIC programming. *Comput. Educ. 103*, 76–86. https://doi.org/10.1016/j.compedu.2016.09.012

Özdinç, F., Kaya, G., Mumcu, F., & Yıldız, B. (2022) Integration of computational thinking into STEM activities: an example of an interdisciplinary unplugged programming activity, *Science Activities, 59*(3), 151–159, https://doi.org/10.1080/00368121.2022.2071817

Park, S., & Jeon, Y. (2015). Teachers' perception on computational thinking in science practices. *International Journal of Education and Information Technologies, 9*(1), 180–185.

Parveen, Q., & Batool, S. (2012). Effect of Cooperative Learning on Achievement of Students in General Science at Secondary Level. *International Education Studies, 5*(2), 154–158.

Perrin, B.S. Jr, Miller, B.T., Schalk, V., Woodcock, H.L., Brooks, B.R., Ichiye, T. (2014). Web-Based Computational Chemistry Education with CHARMMing III: Reduction Potentials of Electron Transfer Proteins. *PLoS Comput Biol 10*(7): e1003739. https://doi.org/10.1371/journal.pcbi.1003739

Pickard, F.C. IV, Miller, B.T., Schalk, V., Lerner, M.G., Woodcock, H.L. III, Brooks, B.R. (2014). Web-Based Computational Chemistry Education with CHARMMing II: Coarse-Grained Protein Folding. *PLoS Comput Biol 10*(7): e1003738. https://doi.org/10.1371/journal.pcbi.1003738

Román-González, M., Pérez-González, J.-C. & Jiménez-Fernández, C. (2016). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior, 30*, 114.

Román-González, M., Pérez-González, J.-C. & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test, Comput. Hum. Behav. 72, 678–691. http://dx.doi.org/10.1016/j.chb.2016.08.047

Rotgans, J. I., & Schmidt, H. G. (2012). Problem-based Learning and Student Motivation: The Role of Interest in Learning and Achievement. In G. O'Grady, E. H. J. Yew, K. P. L. Goh, & H. G. Schmidt (Eds.), *One-Day, One-Problem* (pp. 85–101). Springer Singapore. https://doi.org/10.1007/978-981-4021-75-3_5

Rubinstein, A., Chor, B. (2014). Computational thinking in life science education. *PLoS Comput Biol 10*(11): e1003897. https://doi.org/10.1371/journal.pcbi.1003897

Savery, J. R. (2015). Overview of problem-based learning: definitions and distinctions. *Interdisciplinary Journal of Problem Based Learning, 9*(1): 5–15.

Scherer, R., Siddiq, F., and Viveros, B. S. (2020). A meta-analysis of teaching and learning computer programming: effective instructional approaches and conditions. *Comput. Hum. Behav. 109*,106349. https://doi.org/10.1016/j.chb.2020.106349

Slavin, R, E. (1999) Comprehensive approaches to cooperative learning, *Theory into Practice, 38*(2), 74–79, https://doi.org/10.1080/00405849909543835

Slavin, R. E. (2015) Cooperative learning in elementary schools, *Education 3–13, 43*(1), 5–14, https://doi.org/10.1080/03004279.2015.963370

Tsai, M. J. (2002). Do male students often perform better than female students when learning computers?: A study of Taiwanese eighth graders' computer education through strategic and cooperative learning. *Journal of Educational Computing Research, 26*(1), 67–85.

Tsarava, K., Moeller, K., Román-González, M., Golle, J., Leifheit, L., Butz, M. V., & Ninaus, M. (2022). A cognitive definition of computational thinking in primary education. *Computers & Education*, *179*, 104425.

Vaughan, W. (2002) Effects of Cooperative Learning on Achievement and Attitude Among Students of Color, *The Journal of Educational Research, 95*(6), 359–364, https://doi.org/10.1080/00220670209596610

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Phil. Trans. R. Soc. 366*, 3717–3725

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 1–16. https://doi.org/10.1145/2576872

Yasar, O., Rajasethupathy, K. S., Tuzun, R. E., McCoy, R. A., & Harkin, J. (2000). A new perspective on computational science education. *Computing in Science & Engineering, 2*(5), 74–79.

Zhou, X., Tsai, CW. (2023) The Effects of Socially Shared Regulation of Learning on the Computational Thinking, Motivation, and Engagement in Collaborative Learning by Teaching. *Educ Inf Technol 28*, 8135–8152. https://doi.org/10.1007/s10639-022-11527-1

I.N. Çelik obtained her undergraduate degree in Science Teaching from the Department of Science Teaching at Hacettepe University's Faculty of Education in 2020. In 2023, she completed her Master's degree in Science Education at Hacettepe University. Her Master's thesis focused on computational thinking and coding instruction. Currently, she is employed as a science teacher in a private middle school, while pursuing her doctoral studies in Science Education at Hacettepe University.

K. Bati, Associate Professor, completed his undergraduate studies at Hacettepe University, Department of Science Teaching, in 2005. Subsequently, he was employed as a research assistant at Hacettepe University in 2009, following a period of working as a science teacher in a number of private institutions. In 2010, he was awarded a Master of Science in Science Education, and in 2014, he was awarded a Doctor of Philosophy in Science Education by Hacettepe University. Her master's thesis focused on scientific process skills and problem-solving, while her doctoral thesis addressed modelling, the nature of science and critical thinking. During the 2018–2019 academic year, he undertook postdoctoral studies at The Ohio State University (USA) as part of the TÜBİTAK 2219A project. The author, has been employed as an associate professor at Hacettepe University since 2020.

Appendix 1
Rubric for Computational Problem-Solving Activity Sheets

Skills	*Scoring*				
	Excellent (5)	***Good (4)***	***Acceptable (3)***	***Needs Improvement (2)***	***Poor (1)***
Algorithm Design	The algorithm is completely correct, all the steps have been completed correctly one by one.	The algorithm is correct, but there is a lack of expression in the steps.	The algorithm is logically correct, but there are deficiencies in the process steps.	The ordering of the algorithm steps is incorrect and/or there are deficiencies, it should be imp-roved.	The algorithm's all wrong, and the instructions are wrong.
Preparing Flow Chart	All steps in the flow diagram are completely correct and the symbols are correctly transferred to the flow diagram.	The flow diagram is correct, with no missing steps, or errors in the use of symbols.	The sequence of the flow diagram is correct, one or several steps are missing, and there are errors in the symbols.	There are errors in the steps and symbols of the flow diagram.	The flow diagram is logically faulty, does not proceed sequentially and symbols are not used.
Coding and Debugging	The Python code is written correctly and works very well.	The Python code is complete, but the steps are not working due to minor syntax errors.	Python code is written, and completed, but the steps are incorrect.	Python code was tried to be written, but there are missing and incorrect steps.	Python code is incomplete, not written

Appendix 2
Semi-structured Interview Questions

Q1. What did you do well in this process and why? What were the positive and negative aspects of the process for you?

Q2. In which area did you have more difficulties during this process?

Q3. At which stage of the process did you need more help?

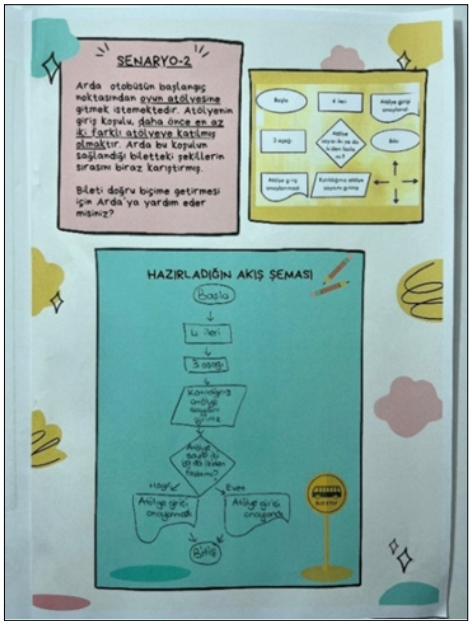
Q4. From whom did you get the most support during the process? Did you have any difficulties with this?

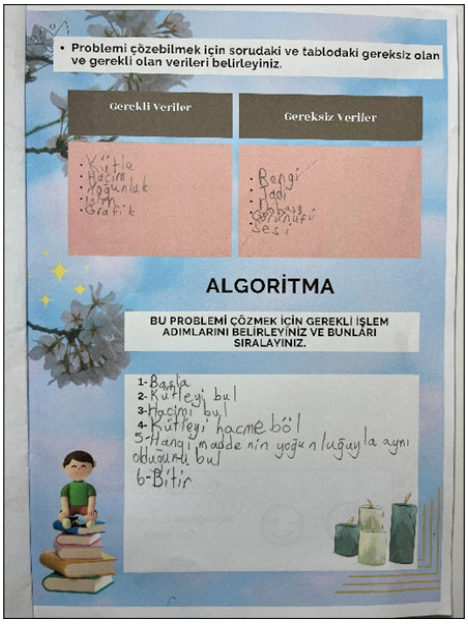
Q5. In which process do you think you should improve yourself more?

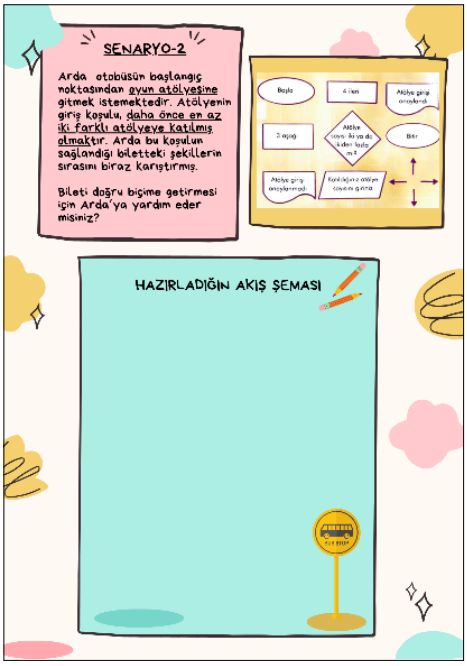
Q6. What would you do differently in your next computational problem-solving study?

Q7. Do you think you have improved yourself at the end of the process? In which area do you feel better?

Appendix 3
Examples of Activity Sheets







Appendix 4
Photos from Implication