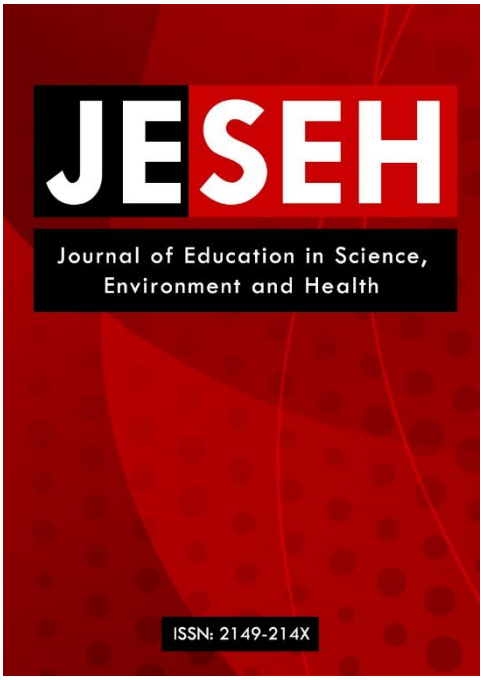


**ISSN:** 2149-214X

**Journal of Education in Science, Environment and Health**

www.jeseh.net

# The Effect of Flipped Learning Used in Text-Based Coding Education on Students' Computational Thinking Skills and Attitudes: A Mixed Methods Study

**Nergiz Agacdelen[1], Oguz Cetin[2]**
[1] Nevsehir Private Kardelen College
[2] Nigde Omer Halisdemir University

**To cite this article:**

Agacdelen, N. & Cetin, O. (2026). The effect of flipped learning used in text-based coding education on students' computational thinking skills and attitudes: A mixed methods study. *Journal of Education in Science, Environment and Health (JESEH), 12*(3), 276-294. https://doi.org/10.55549/jeseh.897

JESEH
Journal of Education in Science, Environment and Health
Volume 12, Issue 3, 2026
https://doi.org/10.55549/jeseh.897

# The Effect of Flipped Learning Used in Text-Based Coding Education on Students' Computational Thinking Skills and Attitudes: A Mixed Methods Study

**Nergiz Agacdelen, Oguz Cetin**

## Article Info

***Article History***

Published:
01 July 2026

Received:
04 February 2026

Accepted:
21 May 2026

***Keywords***
Flipped learning
Computational thinking
Coding education
Text-based programming
Attitude
Mixed methods

## Abstract

This study aims to examine the effectiveness of the flipped learning (FL) model, an innovative teaching approach in coding education. The research investigates whether implementing text-based coding education using the FL model creates a significant difference in middle school students' self-efficacy perceptions regarding computational thinking skills and their attitudes toward coding compared to traditional teaching. Designed using an explanatory sequential mixed-methods approach, the study employed a quasi-experimental design. The study group consisted of a total of 60 students enrolled in the 6th grade of a private middle school, divided into an experimental (n=30) and a control (n=30) group. During the eight-week process, the experimental group learned the Small Basic programming language according to the flipped learning (FL) model, while the control group received traditional instruction. Quantitative data were collected using the CTSPSS and CEAS scales, while qualitative data were collected through semi-structured interviews with selected students from the experimental group. The findings showed that the experimental group obtained significantly higher scores than the control group on both scales. Qualitative findings also revealed that students came to class prepared thanks to flipped learning (FL), were able to learn at their own pace, and developed positive attitudes towards coding. In conclusion, it was determined that the flipped learning (FL) model is an effective strategy in text-based coding instruction.

## Introduction

The 21st century is defined as a period that prioritizes skills such as processing, analyzing, restructuring information, and producing creative solutions rather than merely accessing information. At the center of this set of skills lies computational thinking, a systematic problem-solving approach inspired by the fundamental principles of computer science (Wing, 2006). Computational thinking encompasses mental processes such as understanding complex problems, breaking solutions into algorithmic steps, recognizing patterns, abstraction, and data analysis (Shute et al., 2017; Wing, 2006). One of the most effective tools for developing these skills is coding education. Coding not only involves teaching a programming language but also supports the development of higher-order skills such as logical reasoning, critical thinking, creativity, and problem solving (Grover & Pea, 2018).

The global trend in coding education is toward making this discipline a fundamental part of basic education. Countries such as Estonia, the United Kingdom, and Finland have integrated coding into their curricula from an early age (Bocconi et al., 2016). In Türkiye, the Ministry of National Education (MoNE) has included algorithm and coding instruction at the primary education level within the scope of the Information Technologies and Software course since 2018. Although block-based visual tools such as Scratch are generally used at the initial stage of education, transitioning to text-based programming languages (e.g., Small Basic, Python, JavaScript) is inevitable for acquiring real programming skills and developing abstract thinking capacity (Weintrop et al., 2016). However, due to factors such as the rigidity of syntax rules, the lack of immediate visual feedback, and the inadequacy of error messages, text-based coding can be demotivating and challenging for students, thereby increasing cognitive load during the learning process (Bers, 2018).

It is precisely at this point that there is a need to overcome such difficulties by restructuring the learning process. The flipped learning model is one of the approaches that has most prominently addressed this issue in recent years. This model proposes that the theoretical knowledge transmission traditionally conducted in the classroom be carried out by students in their own time and space through digital materials such as videos, animations, and

interactive simulations, while class time is allocated to active learning activities such as discussion, problem solving, project development, and teacher-guided practice (Bergmann & Sams, 2023; Bishop & Verleger, 2013). By offering students opportunities to control their own learning pace, repeat content as much as needed, and engage in deeper interaction in the classroom, this model holds great potential, especially in practice-oriented fields such as coding where individual differences are pronounced (Lo & Hew, 2017).

The literature includes numerous studies demonstrating the positive effects of flipped learning on general academic achievement, motivation, and student engagement (Zainuddin & Halili, 2016). Previous studies examining flipped learning in technology-supported environments reported improvements in student achievement, engagement, and learning motivation (Zainuddin & Halili, 2016; Zheng et al., 2020). However, most of these studies have focused on block-based coding or have measured only academic achievement quantitatively. Research that applies flipped learning in text-based coding education and simultaneously examines both cognitive (computational thinking skills) and affective (attitude, self-efficacy) dimensions in depth using mixed methods is quite limited (Lo & Hew, 2020). Although numerous studies have reported positive effects of flipped learning on student engagement and academic achievement, some research has found limited or non-significant differences between flipped learning and traditional teaching methods depending on contextual factors such as instructional design, duration of implementation, and student characteristics (Erdem, 2018; Lo & Hew, 2017). There is a particular need for comprehensive studies conducted with middle school students that shed light on the Turkish context.

Aiming to fill this gap, the present study addresses the following main research problems:
"Is there a statistically significant difference in the computational thinking skills and attitudes toward coding of middle school students who receive text-based coding education conducted according to the flipped learning approach compared to those who receive traditional instruction conducted according to the existing curriculum?" and "What are the experiences of middle school students regarding text-based coding education implemented through the flipped learning approach, their perceptions of this process, the difficulties they encounter, and the effects of this instructional model on their computational thinking skills and attitudes toward coding?"

Within the framework of this problem, the sub-problems are as follows:
1. Is there a statistically significant difference between the computational thinking skills scores of middle school students in the experimental group receiving text-based coding education based on the flipped learning approach and those in the control group receiving education based on the traditional curriculum?
2. Is there a statistically significant difference between the attitudes toward coding of middle school students in the experimental group receiving text-based coding education based on the flipped learning approach and those in the control group receiving education based on the traditional curriculum?
3. What are the views of the students in the experimental group regarding the implemented flipped learning model?

**Purpose of the Study**

In today's education systems, individuals are expected not only to access information but also to process, analyze, restructure, and use this information effectively, as well as to engage in creative problem-solving processes. In this context, computational thinking skills and coding education have become fundamental components of 21st-century learning skills (Brennan & Resnick, 2012; Wing, 2006). Coding not only involves software development but also encompasses cognitive processes such as algorithmic thinking, systematic analysis, abstraction, and modeling. Therefore, introducing students especially at the secondary education level to these skills at an early age is of great importance in enabling them to adapt to the demands of the modern era (Grover & Pea, 2013).

In recent years, there has been a noticeable increase in the use of innovative educational approaches that go beyond the limits of traditional teaching practices. One such approach is the flipped learning model, which encourages students to come to class prepared in advance while allowing class time to be devoted more to higher order cognitive processes such as practice, analysis, and production (Bergmann & Sams, 2023). The flipped learning approach places students at the center of the learning process by providing differentiated content tailored to individual learning pace and needs, and by enabling students to manage their own learning processes (Zainuddin & Halili, 2016). Especially in technology-supported learning environments, the implementation of the flipped model has been shown to positively affect students' academic achievement.

The flipped learning approach involves students engaging in knowledge-based activities outside the classroom through videos, presentations, or digital documents, while class time is focused on activities such as group work,

project development, and problem solving (Abeysekera & Dawson, 2015). This approach can enable students to experience more active and productive learning processes, particularly in practice-oriented disciplines such as coding education. Within the scope of coding education, the use of text-based programming languages (e.g., Python, JavaScript) contributes to the development of students' skills such as algorithmic thinking, debugging, and the use of conditional statements (Weintrop et al., 2016).

In this regard, the main purpose of this study is to examine the effects of implementing text-based coding education with middle school students within the framework of the flipped learning model on their computational thinking skills and attitudes toward coding. The study aims to compare the effectiveness of the flipped learning approach with that of existing instructional practices and to reveal the contribution of this model to coding education.

In particular, computational thinking skills enable students to acquire higher-order cognitive skills such as analytical thinking, data analysis, algorithm development, problem solving, and debugging (Shute et al., 2017). However, the development of these skills is possible not only through theoretical instruction but also through constructivist, student-centered, and practice-based learning approaches. In this sense, the flipped learning approach supports the development of computational thinking by providing students with opportunities to access theoretical knowledge outside the classroom and to engage in active practices under teacher guidance during class time.

In addition, attitudes toward coding encompass students' interest in coding courses, their motivation, and their perceptions of self efficacy. Developing positive attitudes toward coding increases students' active participation in the learning process and ensures the permanence of learning outcomes (Abdüsselam & Uzoğlu, 2022). Examining whether text-based coding education supported by the flipped learning model positively affects students' attitudes is important in terms of the quality of educational programs.

**Significance of Study**

With the advancement of technology, many innovations have emerged in our lives. These technological developments have brought about changes in all fields, including education. The basic skills expected of learners in education are no longer the same as before. In a constantly developing and changing world, individuals are expected to possess skills such as algorithmic thinking and problem analysis (Scot, 2018). Computational thinking is one of these skills.

Computational thinking is generally defined as a systematic problem-solving process involving abstraction, algorithmic reasoning, pattern recognition, and logical thinking skills (Wing, 2006; Grover & Pea, 2013). Computational thinking is defined as a multidimensional construct including algorithmic reasoning, problem solving, abstraction, and collaborative processes (Shute et al., 2017; Weintrop et al., 2016). Computational thinking essentially refers to a systematic approach to the problem-solving process. These skills involve identifying solution steps, organizing them systematically, decomposing complex problems into manageable parts, and evaluating possible solutions through algorithmic reasoning (Shute et al., 2017; Weintrop et al., 2016). Computational thinking skills are directly related to coding, as algorithmic thinking forms the basis of coding.

One of the most important contributions of coding education to students is its support for the development of computational thinking skills. Research has shown that learning programming also significantly enhances mathematical thinking capacity (Taylor et al., 2010). The methods and models used in coding education are of great importance. One of the models that can be used in coding education, which has become increasingly widespread in recent years, is the flipped learning model. When middle school students receive text-based coding education for the first time, they often experience difficulties in understanding the subject. This is because text-based coding education can be more challenging for students due to its more abstract and complex structure compared to visual programming tools. Considering this situation, it can be suggested that the flipped learning model may be used as an effective approach to help students grasp the fundamental logic of text-based coding, increase the permanence of learning, and transfer the knowledge they acquire. Through digital learning environments, the flipped learning model offers students opportunities for flexible learning by allowing access to instructional content independent of time and place while supporting self-paced learning processes (Abeysekera & Dawson, 2015; Lo & Hew, 2017).

The findings of this study are expected to provide practical contributions to educational practices as well as to form a theoretical basis for future research in this field. In addition, this study contributes to the literature by

examining flipped learning within a text-based programming environment and by analyzing both cognitive and affective outcomes simultaneously through a mixed-methods approach.

## Method

### Research Model

This study was designed using an explanatory sequential mixed methods design, in which quantitative and qualitative data collection and analysis techniques are used together in a sequential manner (Creswell & Creswell, 2017). In this design, quantitative data are first collected and analyzed; subsequently, qualitative data are gathered in order to explain and interpret the quantitative findings in greater depth. In the quantitative phase, a quasi experimental design specifically, a nonequivalent pretest–posttest control group model was employed to conduct causal comparisons. This model is frequently used in educational research to examine the effects of an intervention while preserving the natural structure of classroom settings (Büyüköztürk, 2002).

Table 1. Experimental design

| Groups | Pre-test | Intervention | Post-test |
|---|---|---|---|
| Experimental | CTSPSS<br>CEAS | Flipped Learning Model | CTSPSS<br>CEAS<br>Interview |
| Control | CTSPSS<br>CEAS | Existing Curriculum | CTSPSS<br>CEAS |
| CTSPSS: Computational Thinking Skills Self-Perception Scale | | | |
| CEAS: Coding Education Attitude Scale | | | |

### Study Group

The study group consists of 6th-grade students enrolled in a private middle school located in Nevşehir during the fall semester of the 2024–2025 academic year. In sample selection, convenience sampling, one of the non-probability sampling methods, was used due to ease of implementation and accessibility (Etikan et al., 2016). The study group comprises a total of 60 students selected from different classes within the same school that have similar socioeconomic and academic profiles. The assignment of the groups as experimental or control groups was carried out impartially. Accordingly, 30 students constituted the experimental group and 30 students constituted the control group.

### Data Collection Instruments

*Quantitative Data Collection Instruments*

*Computational Thinking Skills Self-Efficacy Perception Scale (CTSPSS):* This scale, used to measure students' self-efficacy beliefs regarding computational thinking skills, consists of 36 items and 5 sub-dimensions (e.g., algorithmic thinking, problem solving, abstraction). It has a 3-point Likert-type response format ("Yes = 3," "Partly = 2," "No = 1"). In the original scale development study, the Cronbach's alpha internal consistency coefficient was reported as .943. Within the scope of the present thesis study, the Cronbach's alpha (α) value was calculated by the researcher as 0.853.

*Coding Education Attitude Scale (CEAS):* This scale, developed by Karaman and Filiz (2019) to determine students' attitudes toward coding courses, consists of two dimensions general positive attitude (28 items) and general negative attitude (13 items) and a total of 41 items. It is rated on a 5 point Likert scale ranging from "Strongly Disagree = 1" to "Strongly Agree = 5." Negative attitude items are reverse-coded when calculating the total score. The reliability coefficient reported in the original study is .94. In this thesis study, the Cronbach's alpha (α) value was calculated by the researcher as 0.951.

*Qualitative Data Collection Instrument*

*Semi-Structured Interview Form:* To collect qualitative data, five open-ended questions prepared in line with expert opinions were used. These questions were directed to five volunteer students selected from the experimental group after the completion of the 8-week experimental implementation, in order to obtain in-depth data. In this way, it became possible to obtain a richer perspective on the research topic and to evaluate different viewpoints. In addition, collecting the opinions of different individuals on the same topic contributed to providing solutions to the research problem (Yıldırım & Şimşek, 2021). The interviews were conducted on a voluntary basis in the computer laboratory where the instruction was implemented.

The interview questions included in the semi-structured interview form are presented below:
1. What is text-based coding? Can you explain it?
2. What is the flipped learning model? Can you explain it?
3. Did you experience any difficulties during the implementation of the flipped learning model? If yes, what were they?
4. Do you think that teaching text-based coding through the flipped learning model was beneficial for you? Can you explain?
5. Do you think that the flipped learning model should be used in every unit of this course? Can you explain?

The interview transcripts obtained were analyzed using the thematic analysis method in accordance with qualitative data analysis procedures. To increase the reliability of the codes, the coders (the researcher and the supervising teacher) independently examined the codes, and the final version was agreed upon through joint evaluation. Thus, students' perceptions of text-based coding and the flipped learning model, the difficulties they experienced, the skills they gained, and their suggestions were systematically categorized. The findings obtained during the analysis process were presented under themes in a manner that supports the qualitative aspect of the study, and direct quotations from students were included to enhance the credibility of the findings (Miles & Huberman, 1994).

**Implementation Process**

The implementation process of the study lasted for 8 weeks. Prior to the implementation, the necessary ethics committee approval (Niğde Ömer Halisdemir University, Decision No: 2024/07-27), permissions from the Ministry of National Education, and approval from the school administration were obtained; voluntary participation consent was secured from students and their parents. Within the scope of the 6th-grade "Information Technologies and Software" curriculum, weekly lesson plans covering the topics of Algorithms and Small Basic were prepared in order to support students' development in algorithmic thinking and text-based coding.

Table 2. Weekly lesson plan schedule

| | |
|---|---|
| Week 1 | Information was provided about algorithms, the fundamentals of algorithms, and their purposes of use |
| Week 2 | Information was provided about decision and conditional structures in algorithms. Simple algorithm examples were also presented. |
| Week 3 | The Small Basic program was introduced, and its installation was explained. Information was provided on the concepts of TextWindow, BackgroundColor, ForegroundColor, Write, and WriteLine. |
| Week 4 | The Small Basic application and its components were used. Information was provided on Arithmetic Operators, Comparison Operators, and Logical Operators. |
| Week 5 | The topic was integrated with prior learning. Information was provided on TextWindow, Variables, the Math class, Power, Max, Min, SquareRoot, Round, and Remainder, and related examples were implemented. |
| Week 6 | The topic was integrated with prior learning. Information was provided on the Read and ReadNumber commands, and related examples were implemented. |
| Week 7 | Information was provided on If–Else conditional structures, the Clock object, and its methods, and related examples were implemented. |
| Week 8 | Information was provided on the For loop, While loop, and the GraphicsWindow class, and related examples were implemented. |

*Week 1:* Information was provided about algorithms, the fundamentals of algorithms, and their purposes of use. In the first week, a pre-test was administered to the students. Subsequently, the topics of what an algorithm is, the purposes of using algorithms, and how algorithms are constructed were covered. Algorithms are considered fundamental components of computational thinking and support students in developing systematic approaches to

solving problems (Wing, 2006; Cormen et al., 2009). In computer science, algorithms are the building blocks of problem solving and form the foundation of programming languages. Algorithms are used in many fields ranging from daily life to artificial intelligence, thereby promoting structured thinking (Knuth, 1997).

*Week 2:* Information was provided about decision and conditional structures in algorithms. Simple algorithm examples were also presented. In the second week, decision and conditional structures in algorithms were introduced, and simple examples were included. The flexibility and functionality of algorithms largely depend on decision and conditional structures. Through these structures, programs can branch in different directions depending on specific conditions. For example, control statements such as "if…else" are commonly used to implement such conditional branching. In addition, flowcharts can be used for visualization to facilitate a better understanding of these concepts (Cormen et al., 2009).

*Week 3:* The Small Basic program was introduced, and step-by-step instructions were provided on how to download and install the program on a computer. Subsequently, basic commands and concepts such as TextWindow, BackgroundColor, ForegroundColor, Write, and WriteLine were emphasized. In the practical activities, students performed simple text output examples by setting background and text colors of their own choice.

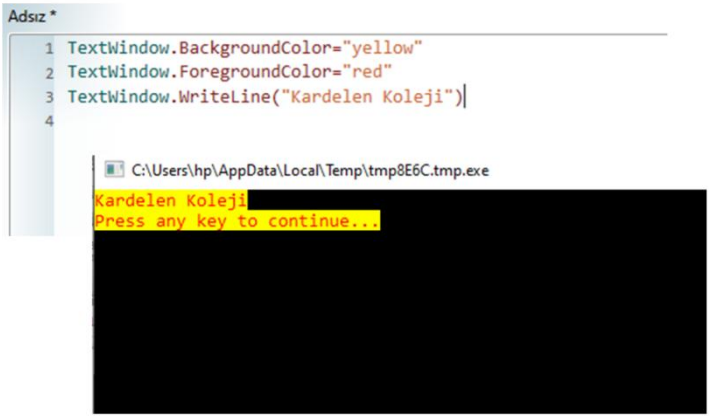


Figure 1. Study on the concepts of TextWindow, BackgroundColor, ForegroundColor, Write, and WriteLine

*Week 4:* The Small Basic application and its components were used. Information was provided about Arithmetic Operators, Comparison Operators, and Logical Operators.
The basic operators supported by Small Basic were explained. These operators are used to perform mathematical and logical operations. Arithmetic Operators include +, −, *, and /. Comparison Operators include =, <, >, <=, and >=. Logical Operators include And, Or, and Not.

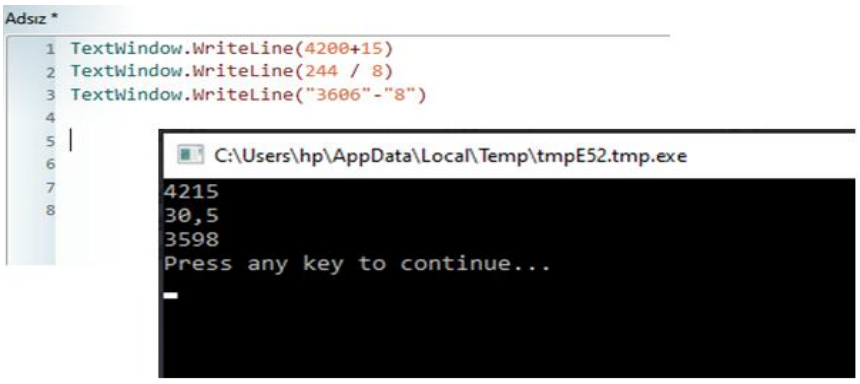


Figure 2. Operators activity

*Week 5:* The topic was integrated with prior learning. Information was provided on TextWindow, Variables, the Math class, Power, Max, Min, SquareRoot, Round, and Remainder, and related examples were implemented.
At this stage of the study, the Math class was integrated into the programming environment. Through this class, various mathematical operations were performed on defined variables, and calculation processes were carried out. In the programming domain, mathematical operations constitute one of the fundamental building blocks (Turing,

1936). The four basic arithmetic operations (addition, subtraction, multiplication, and division), which are standard in all programming languages, are indispensable components of algorithmic solutions (Arslan et al., 2022). The use of the Math class enables the implementation of more complex mathematical functions beyond these basic operations.

• Math.Power, Math.SquareRoot: Exponentiation and square root operations
• Math.Max, Math.Min: Finding the maximum and minimum values
• Math.Round, Math.Remainder: Rounding numbers and finding remainders

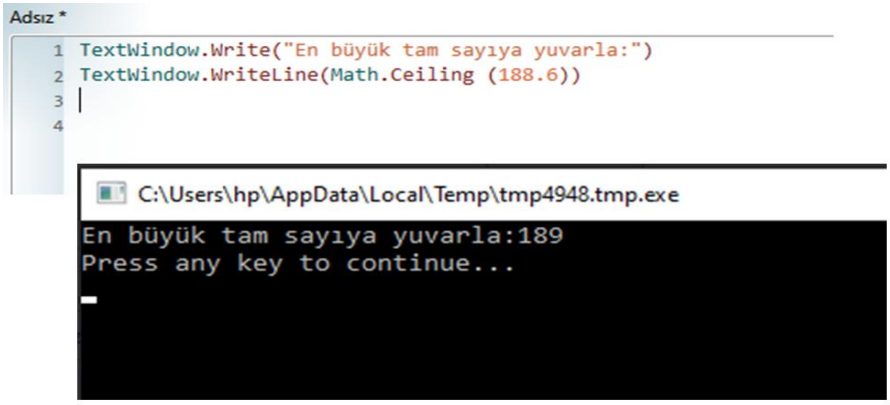


Figure 3. Math class usage activity

*Week 6:* The topic was integrated with prior learning. Information was provided on the Read and ReadNumber commands, and related examples were implemented. The topic of receiving user input through the Read and ReadNumber commands was covered. This initiates user interaction at a basic level and represents one of the fundamental steps in human–computer interaction. It is essential for a program to interact with the user.

*Commands:*
TextWindow.Read: Receiving textual input
TextWindow.ReadNumber**:** Receiving numerical input
During the lesson, students developed programs that receive input from the user, process this information, and display the output on the screen using these commands. In this way, the foundations of creating interactive applications in the programming process were established.

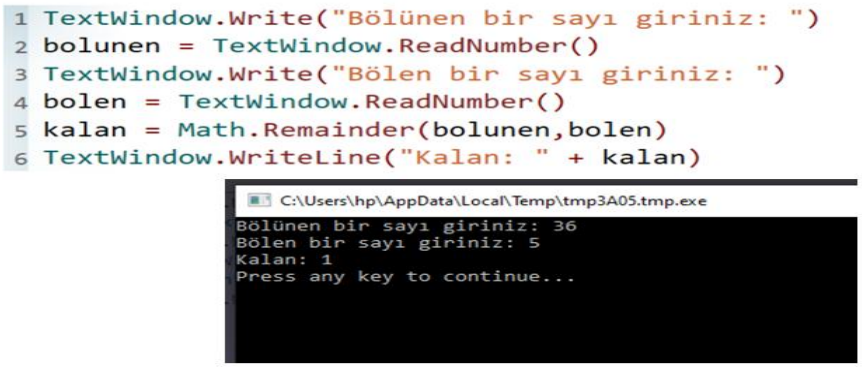


Figure 4. Read and ReadNumber commands activity

*Week 7:* Information was provided on If–Else conditional structures, the Clock object, and its methods, and related examples were implemented. In this week, advanced uses of If–Else structures were introduced together with the Clock object. Time-related programming structures contribute to students' understanding of dynamic systems and conditional decision-making processes in programming environments (Cormen et al., 2009). These programming structures provide software with dynamic flexibility and conditional decision-making capability. They allow different execution flows to be carried out depending on user inputs. Such decision mechanisms enable algorithms to produce different responses under varying conditions (Kandemir, 2018).

*Commands:* Clock.Hour, Clock.Minute, Clock. SecondTime control embedded within If–Else structures


Figure 5. If–Else commands activity

*Week 8:* Information was provided on the For loop, While loop, and the GraphicsWindow class, and related examples were implemented. For and While loops are used for repetitive operations. The GraphicsWindow class was used to demonstrate, through hands-on activities, how visual outputs can be created. These structures are employed to repeat specific operations or to continue processes until a certain condition is met (Demirkol, 2020). Loops enable repetitive tasks to be performed in a concise and efficient manner. The GraphicsWindow class is used for visualization and animation purposes.

*Loop Structures:*
For i = 1 To 10 While condition

*GraphicsWindow Commands:*
GraphicsWindow.DrawRectangle,
DrawLine, FillEllipse, SetPixel

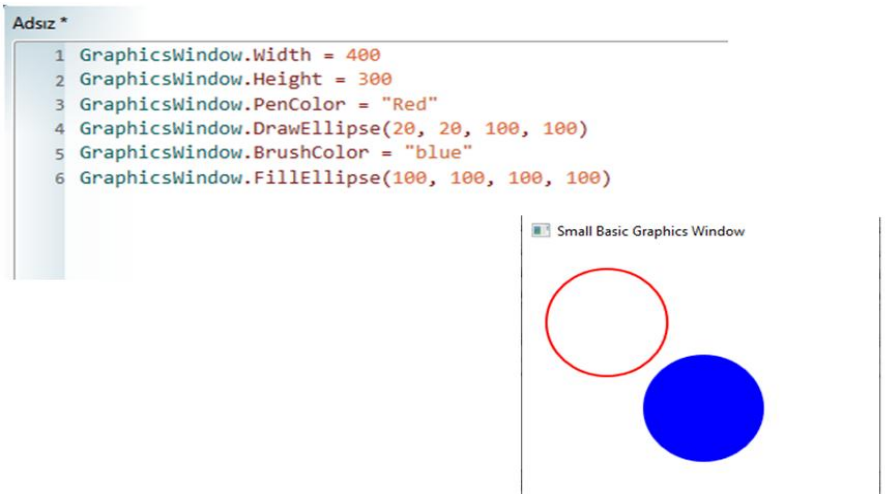


Figure 6. Visual output activity

*Implementation Process in the Experimental Group*

At the beginning of the study, the necessary institutional permissions were obtained in accordance with ethical principles, and the purpose, process, and methodology of the research were explained to the students in detail. During the eight-week experimental study, students reviewed and watched the lesson videos prepared by the teacher and uploaded to the Edpuzzle platform as part of the pre-class preparation phase. Within the scope of the study, students' use of the pre-class preparation videos was systematically monitored through the Edpuzzle platform. Through the teacher panel, the duration for which each student watched the video content was recorded and analyzed.

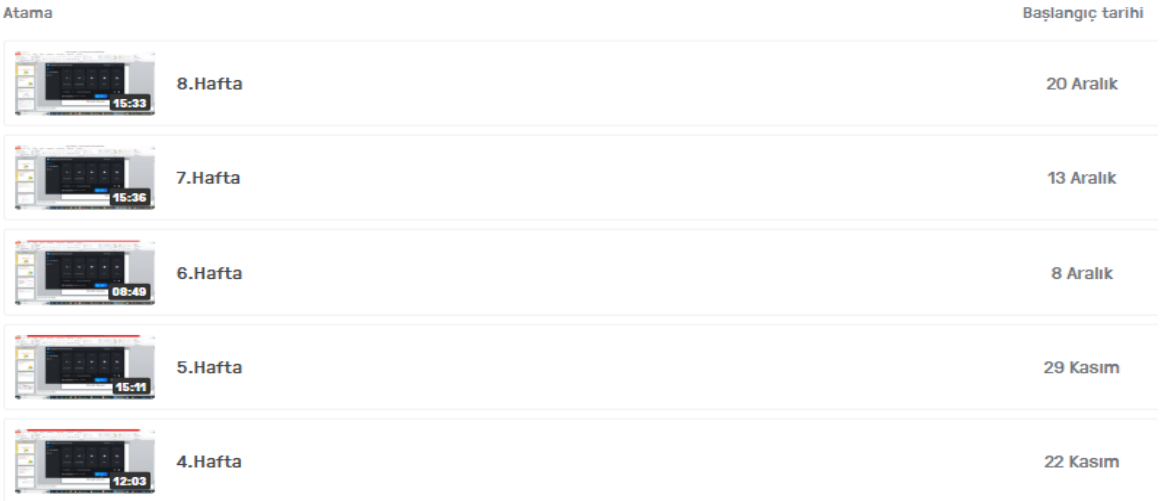


Figure 7. Date information of the lesson

## Data Analysis

*Quantitative Data Analysis*

Data analysis is defined as the systematic application of appropriate statistical methods to transform the data collected during the research process into scientifically valid results (Büyüköztürk, 2002). The implementation phase of the study was conducted in a controlled manner during the 2024–2025 academic year. Before initiating the data analysis process, the normality of the score distributions obtained from the scales was examined using the Shapiro–Wilk test, since the number of participants in the study group was below 50. In the analysis of the data, descriptive statistics such as mean, median, mode, standard deviation, skewness, and kurtosis values were used. Skewness and kurtosis values were calculated for both scales in order to determine the type of comparison to be conducted. The results of the homogeneity and normality tests conducted to determine whether the CTSPSS and CEAS scores met the assumptions of parametric tests are presented in Table 3.

Table 3. Descriptive analysis of the scales

| Scale | $\overline{X}$ | sd | Median | Skewness | Kurtosis | Shapiro-Wilk | |
|---|---|---|---|---|---|---|---|
| | | | | | | Statistic | p |
| CTSPSS | 2.01065 | .145294 | 2.02778 | -.027 | 1.015 | .958 | .039* |
| CEAS | 3.41748 | .578524 | 3.18293 | -.480 | -.528 | .870 | .000* |

*p<0.05 level

Normality analyses of the *Computational Thinking Skills Self-Efficacy Perception Scale* and the *Coding Education Attitude Scale (CEAS)* used in the study were evaluated based on skewness, kurtosis coefficients, and Shapiro–Wilk test statistics. For the computational thinking skills scale, the skewness value was found to be 0.027 and the kurtosis value was 1.015. These values fall within the ±1.5 range, indicating that the data are close to a normal distribution (Tabachnick & Fidell, 2013). However, the Shapiro–Wilk test result yielded p = .039, which is below the .05 significance level, indicating that the normality assumption was not statistically met. Nevertheless, it is known that the Shapiro–Wilk test may show excessive sensitivity to deviations from normality when the sample size is small (n < 50–100) (Ghasemi & Zahediasl, 2012). Therefore, considering the total sample of 60 participants consisting of 30 students in the experimental group and 30 students in the control group, minor deviations were not considered sufficient to preclude the use of parametric tests.

For the Coding Education Attitude Scale (CEAS), the skewness value was −0.480 and the kurtosis value was −0.528, both of which also fall within the ±1 range. However, the Shapiro–Wilk test result of p = .000 clearly indicates a violation of the normality assumption. This finding suggests that non-parametric tests may be safer when analyzing CEAS data. Nevertheless, the acceptable skewness and kurtosis values and the moderate sample size support the flexible use of parametric tests (Field, 2018). In conclusion, for both scales, it is recommended that decisions regarding the normality assumption be based on multiple criteria, including not only p values but also skewness–kurtosis values and sample size. Therefore, an independent samples t-test was used for comparative analyses in this study.

Prior to the implementation, pre-test data were collected using the relevant scales. After the completion of the intervention, post-test data were obtained using the same scales. In this study conducted with 60 students, quantitative data were transferred to the SPSS for Windows 24.0 statistical software package, and analyses were carried out using this software. After data entry was completed, reverse-coded items in the scales were recoded.

*Qualitative Data Analysis*

In qualitative data analysis, methods such as thematic analysis, content analysis, or discourse analysis are used to achieve an in-depth understanding of the data. This process focuses on interpreting participants' experiences, behaviors, and social contexts (Creswell & Creswell, 2017). In this study, thematic analysis was adopted as the qualitative data analysis technique. Thematic analysis involves dividing data into meaningful units, classifying these units according to their content, and grouping similar content to form themes (Braun & Clarke, 2006). In this context, interview transcripts were carefully examined after the transcription process and structured within the framework of themes determined in line with the research questions. Meaningful statements were selected from the data, grouped according to similar content, sub-themes were created from these groups, and direct quotations from participants that strongly represented each sub-theme were included.

To ensure the reliability of coding in the thematic analysis process, coding was repeated at different times, and the relationships among data, codes, sub-themes, and themes were continuously reviewed throughout the analysis process. This approach enabled the meaningful organization and in-depth interpretation of interview data (Yıldırım & Şimşek, 2021). The findings obtained as a result of the analysis formed the basis for data interpretation. In this study, a semi-structured interview form was used with students. Semi-structured interviews are data collection tools used by researchers in qualitative studies to obtain in-depth information (Büyüköztürk et al., 2019).

## Findings

### Quantitative Findings

In this section, quantitative comparisons conducted for both groups are presented.

Table 4. Independent samples t-test analysis of the experimental and control groups based on CTSPSS and CEAS pre-test results

| Scale | Group | n | $\overline{X}$ | sd | df | t | p |
|---|---|---|---|---|---|---|---|
| CTSPSS | Experimental | 30 | 2.02593 | .139163 | 58 | .812 | .420 |
| | Control | 30 | 1.99537 | .151985 | | | |
| CEAS | Experimental | 30 | 3.00569 | .221853 | 58 | 1.604 | .114 |
| | Control | 30 | 2.88862 | .332434 | | | |

Table 5. Independent samples t-test analysis of the experimental and control groups based on CTSPSS and CEAS post-test results

| Scale | Group | n | $\overline{X}$ | sd | df | t | p |
|---|---|---|---|---|---|---|---|
| CTSPSS | Experimental | 30 | 2.50185 | .136655 | 58 | 15.026 | .000* |
| | Control | 30 | 1.97593 | .134457 | | | |
| CEAS | Experimental | 30 | 3.93089 | .192485 | 58 | 15.276 | .000* |
| | Control | 30 | 2.90407 | .313850 | | | |

*p<0.05 level.

In the CTSPSS pre-test analysis, the mean score of the experimental group was determined as $\overline{X}$ = 2.03, while the mean score of the control group was $\overline{X}$ = 2.00; the results yielded t(58) = .812 and p = .420. Similarly, in the CEAS pre-test, the mean score of the experimental group was $\overline{X}$ = 3.01 and that of the control group was $\overline{X}$ = 2.89, with t(58) = 1.604 and p = .114. In both measurements, p values greater than .05 indicate that there was no statistically significant difference between the groups. The literature emphasizes that interpretations based solely on significance levels are insufficient and that sample size, homogeneity of variance, and effect size should be evaluated together (Cohen et al., 2018). In this study, considering that each group consisted of 30 participants, along with small t values and low mean differences, the findings support the conclusion that the groups were equivalent at baseline. Therefore, it can be stated that prior to the intervention, the experimental and control groups

were statistically similar in terms of both self-efficacy perceptions related to computational thinking and attitudes toward coding. According to the CTSPSS post-test results, the mean score of the experimental group was calculated as $\bar{X}$ = 2.50185, while the mean score of the control group was $\bar{X}$ = 1.97593. The standard deviation values were .136655 and .134457, respectively, and the t-test results yielded t(58) = 15.026 with p = .000. This result indicates a highly statistically significant difference between the groups (p < .001).

Similarly, when the CEAS post-test scores were examined, the mean score of the experimental group was found to be $\bar{X}$ = 3.93089, whereas the mean score of the control group was $\bar{X}$ = 2.90407. The corresponding standard deviations were .192485 and .313850, and the analysis yielded t(58) = 15.276 with p = .000. This finding also demonstrates a statistically significant difference between the two groups in terms of attitudes toward coding. Considering the high t values and significance levels obtained for both scales (p < .001), it can be concluded that the text-based coding instruction implemented in the study had a significant and positive effect on the experimental group students' self-efficacy perceptions related to computational thinking as well as their attitudes toward coding. Within the scope of the study, in order to determine the progress of both groups throughout the process, changes in their mean scores were also compared within each group separately.

It should be noted that the reported values represent scale mean scores rather than raw total scores, which may lead to relatively small standard deviation values. Therefore, the high t values observed in the analyses should be interpreted together with effect size statistics.

In order to better interpret the magnitude of the observed differences, effect size values were calculated. Cohen's d was used to determine the practical significance of the flipped learning intervention. The effect size for computational thinking self-efficacy was found to be d = 3.89, while the effect size for attitudes toward coding was calculated as d = 3.95. According to Cohen (1988), effect size values of 0.20 indicate a small effect, 0.50 a medium effect, and 0.80 or above a large effect. Therefore, the obtained values indicate a very large effect size.

Table 6. Results of the t-test analysis comparing the experimental group's pre-test and post-test mean scores within the group

| Group | | n | $\bar{X}$ | sd | df | t | p |
|---|---|---|---|---|---|---|---|
| Experimental | CTSPSS- pre-test | 30 | 2.0259 | .13916 | 29 | -11.707 | .000* |
| | CTSPSS- post-test | 30 | 2.5019 | .13666 | | | |
| Experimental | CEAS- pre-test | 30 | 3.0057 | .22185 | 29 | -13.065 | .000* |
| | CEAS- post-test | 30 | 3.9309 | .19248 | | | |

* p<0.05 level

The findings indicate that students in the experimental group showed a significant improvement in both their computational thinking skills and their attitudes toward coding after the text-based coding instruction implemented through the flipped learning approach. The increase observed in the students' scores on the CTSPSS scale (t(29) = −11.707, p < .001) demonstrates that the applied instructional method supported the fundamental components of computational thinking, such as problem solving, algorithmic thinking, and logical reasoning. On the other hand, based on the data obtained from the CEAS scale, it was determined that there was also a statistically significant increase in students' attitudes toward coding (t(29) = −13.065, p < .001).

Table 7. Results of the t-test analysis comparing the control group's pre-test and post-test mean scores within the group

| Group | | n | $\bar{X}$ | sd | df | t | p |
|---|---|---|---|---|---|---|---|
| Control | CTSPSS- pre-test | 30 | 1.9954 | .15198 | 29 | .590 | .560 |
| | CTSPSS- post-test | 30 | 1.9759 | .13446 | | | |
| Control | CEAS- pre-test | 30 | 2.8886 | .33243 | 29 | -.578 | .568 |
| | CEAS- post-test | 30 | 2.9041 | .31385 | | | |

According to the analysis results obtained within the scope of the findings, no significant difference was found between the pre-test and post-test scores of the students in the control group. The mean pre-test score of the control group students on the CTSPSS scale was calculated as 1.9954, while the mean post-test score was 1.9759. The results of the paired-samples t-test revealed a value of t(29) = 0.590, p = .560, indicating that the existing instructional method did not have a statistically significant effect on computational thinking skills. Similarly, the

mean pre-test score obtained from the CEAS scale was 2.8886, and the mean post-test score was 2.9041; this change was also not statistically significant ($t(29) = -0.578$, $p = .568$).

**Qualitative Findings (Thematic Analysis)**

As a result of the analysis of the data obtained from the interviews conducted with the students in the experimental group, the following themes were created based on thematic analysis.

1. Perceptions of Text-Based Coding
2. Characteristics of the Flipped Learning Model
3. Challenges Encountered
4. Contributions to the Learning Process
5. Views on Applicability

The themes that emerged as a result of the analyses were presented by supporting them with direct quotations and interpretations; thus, students' perceptions of text-based coding and the flipped learning model were revealed in detail. This process was conducted in accordance with qualitative research standards, thereby increasing the validity and reliability of the findings. At the end of this process, five main themes and sub-themes related to these themes were identified. The findings were supported by direct quotations from student opinions.

*Theme 1: Perceptions of Text-Based Coding*

*Sub-Theme 1.1: Definitional Approaches*
Most of the students defined text-based coding as the process of giving commands through writing.
"Text-based coding is a type of programming that we do by writing commands. For example, Small Basic is text-based programming." (S1)
"Text-based coding is explaining to the computer what it should do in written form." (S2)

*Sub-Theme 1.2: Difficulty and Need for Attention*
It was stated that text-based coding requires more attention and discipline compared to block-based coding. When an error is made, the program not running increased the students' need to be careful.
"Writing code with words requires a bit more attention." (S5)
"When it is written with words and symbols at every stage, if there is any mistake, the program we wrote does not work." (S3)

*Sub-Theme 1.3: Perceived Value and Emotional Reactions*
While some students found this type of programming enjoyable, others described it as difficult at first but instructive afterward.
"Everything is written with words, numbers, and symbols. Although it is difficult at first, it makes sense once you learn it." (S4)
"In this type of coding, codes are written line by line. Writing code with words requires a bit more attention. Even games can be made. After block-based coding, I felt like an engineer with this programming." (S5)

The findings obtained reveal that students are able to make sense of the concept of text-based coding not only in a definitional manner but also in functional and affective dimensions. Students defining text-based coding through written commands, symbols, and numbers; making comparisons with block-based coding and interpreting it at their own level; indicate that they have developed a basic cognitive awareness in this field. In addition, expressing their positive and negative experiences related to the process shows that students question the learning process and can perform self-evaluation. The strategies they developed in the face of difficulties encountered while writing code, increasing their attention, and performing error analyses demonstrate that text-based coding carries not only a technical but also a pedagogical value. As a result, when students' perceptions of text-based coding are examined, it can be said that this process contributes to their development both cognitively and affectively and is a learning tool that can be effectively used in educational environments.

*Theme 2: Characteristics of the Flipped Learning Model*

*Sub-Theme 2.1: Change in the Learning Process*

It was stated that with the flipped learning model, students come to class prepared, are more active in class, and can manage the process better.
"In the flipped learning model, we watch a video before the lesson and then do activities in class. Thanks to this method, instead of repeating the topic in class, the teacher does applications with us. When I learn the topic before the lesson, I become more active in class." (S1)
"In this model, we learn the topic at home and learn in more detail in class. I think this provided more permanent learning because we can communicate more with the teacher." (S2)

*Sub-Theme 2.2: Student-Centeredness and Participation*
The student-centered nature of the model and students directing the process were emphasized as positive features. "I think the most important feature is that we direct the lesson, not the teacher. After watching the video at home, I had the opportunity to ask what I wondered about in class. I used to listen passively, but in this model I participate more." (S3)

*Sub-Theme 2.3: Flexible Learning*
The fact that videos can be paused and rewatched facilitated learning. "I didn't understand the flipped model at first, but later I got used to it. Especially being able to pause and watch the videos as much as I want is very good. It also became easier to have one-on-one interaction with the teacher in class. Therefore, it was more efficient." (S4)

The research findings reveal the effects of the flipped learning model on students' learning processes in a multidimensional manner. Participants stated that thanks to this model, they came to class prepared, engaged in more activities in class, and found opportunities for one-on-one interaction with the teacher. Students' ability to use video content in accordance with their individual learning pace personalizes the learning process and enables them to focus more deeply on the content. This situation increases motivation for learning and contributes to the creation of a student-centered learning environment. In addition, more efficient use of in-class time allows students to interact more effectively with both the teacher and others.

*Theme 3: Challenges Encountered*

Within the scope of the research, the challenges students encountered in the flipped learning process were grouped under four sub-themes. These themes reflect time management, technical and content-related difficulties experienced by students in the learning process, level differences in group work, and the absence of difficulties.

*Sub-Theme 3.1: Time management and lack of preparation*
Some students stated that they could not allocate enough time to watch the videos that needed to be watched before the lesson. In particular, the intensity of other courses caused students to be unable to watch the videos on time and come to class unprepared.
"Sometimes I couldn't find time to watch the video at home. When my other courses are intense, I come to that day's lesson unprepared. This makes it difficult for me to fully understand the activities done in class." (S1)

*Sub-Theme 3.2: Technical and content-related difficulties*
They stated that technical problems such as slow internet connection negatively affected the video watching process. Especially the long loading time of long videos reduced students' motivation.
"I had difficulty watching the videos because my internet was sometimes slow. Especially in long videos, I had to wait a lot. This reduces my motivation." (S2)
"The videos sometimes progress very fast. I have to rewind the parts I don't understand, but still some topics remain incomplete. It is also difficult to ask the teacher everything in class because time is not enough." (S3)

*Sub-Theme 3.3: Level differences in group work*
Some of the students stated that differences in preparation levels among members in group work negatively affected the activities. Students who came to class after watching the video stated that when working with friends who had not watched it, disruptions occurred in the flow of activities.
"In class, sometimes there were disconnections in activities because some friends came at different levels. I have watched the video, but if others haven't, group work becomes difficult." (S4)

*Sub-Theme 3.4: Student who did not experience difficulties*
Some students stated that they did not experience any difficulties in the process, that they watched the videos before the lesson on time and actively participated in class activities.

"I did not experience any difficulties. I participated in the lesson by watching the videos on time." (S5)

These findings reveal that the applicability of the flipped learning model may not be at the same level for every student. Time management problems cause students to neglect their individual responsibilities and to benefit insufficiently from in-class activities. Technological problems, on the other hand, lead to loss of motivation and learning disruptions, especially in the process of accessing video content. In addition, differences in students' readiness levels create imbalance in interactive practices such as group work and may negatively affect the learning process. These findings show that for the flipped learning model to be applied effectively, not only instructional design but also students' individual support needs and environmental conditions should be taken into consideration. Therefore, in order to make this model sustainable and inclusive, it is of great importance to provide students with guidance on time management, technical support, and learning responsibility.

*Theme 4: Contributions to the Learning Process*

According to the research findings, the flipped learning model made positive contributions to students' learning processes from different perspectives.

*Sub-Theme 4.1: Coming Prepared*
Students stated that thanks to the videos they watched before the lesson, they came to class by learning the topic in advance. This situation contributed especially to a decrease in error rates during the application phase and to obtaining more efficiency from the lesson.
"Thanks to the videos, I understand the topic in advance and come to class prepared. This helped me make fewer mistakes while writing code." (S1)

*Sub-Theme 4.2: Activity and Self-Confidence*
Some students stated that with the flipped learning model, more time was allocated to the application part of the lesson, and this increased their self-confidence in coding. It was emphasized that text-based coding topics, which were previously perceived as difficult, were understood more easily thanks to seeing them in advance through videos and practicing them in class.
"Text-based coding normally seemed difficult to me, but with the flipped model, when I first watched the videos and then practiced in class, I understood the topic better. Now I feel more confident while writing code." (S2)
"I think the most important contribution is that we can do more activities in class. Thanks to this situation, we were able to speed up by making fewer mistakes while writing code." (S3)

*Sub-Theme 4.3: Permanence*
The ability to watch videos repeatedly helped students reinforce the topics and increase the permanence of learning.
"Since I could watch the videos several times, I understood the topic better. This increased the permanence of what I learned." (S4)

The research results reveal that the flipped learning model provides significant and multidimensional contributions in the process of text-based coding education. Students stated that thanks to this model, they were able to develop their coding skills more effectively and made fewer mistakes in in-class applications. The pre-learning process enabled students to come to class prepared; this allowed them to use in-class time more efficiently and understand complex commands more comfortably. In addition, the flipped learning approach reduced students' cognitive load and encouraged deeper learning; uncertainties experienced in the coding process were replaced by a more planned and secure learning process. The increase in self-confidence toward learning directly affected students' success in a field with low error tolerance such as text-based coding and enabled them to participate in the learning process more willingly. In this context, the flipped model stands out as an effective instructional strategy in courses that require technical knowledge such as coding education.

*Theme 5: Views on Applicability*

When students' views on the future applicability of the flipped learning model were examined, different perspectives emerged.

*Sub-Theme 5.1: Partial implementation proposal*

Some students stated that the flipped learning model does not need to be implemented in every unit and that it may be more beneficial especially in theoretically oriented topics.
"I think it can be applied in some units, but it is not necessary in all. For example, watching videos is useful in theoretical topics, but in practical units, direct in-class application may be more effective." (S1)
"I think if the topic is complex, video support would be good. But I don't think it is necessary for simple topics." (S4)

*Sub-Theme 5.2: Defense of full applicability*
Some students argued that the model should be used in all units. According to this view, having basic knowledge about the topic before the lesson makes the learning process in class more efficient.
"It can be applied in all units because having prior knowledge on every topic helps me understand better in class. It was especially very useful in topics such as coding and algorithms." (S2)
"Yes, it should be applied. Because thanks to this model, we learn both at home and in class." (S5)

*Sub-Theme 5.3: Criticism of the model in terms of time*
Some students think that implementing the flipped learning model in every unit may be inefficient in terms of time. It was stated that especially the weekly video watching duration was perceived as a waste of time by some students.
"No, it is not necessary in every unit. Also, I think watching videos every week is a waste of time." (S3)

The findings reveal that although the flipped learning model is an effective instructional strategy, applying it in the same way and in a rigid manner in all units may not always yield ideal results. While students emphasized that the model provides meaningful contributions especially in complex or application-based topics, they stated that this method may be inefficient in terms of time management in some simple or repetitive topics. This situation shows that the flipped learning approach should be planned flexibly and structured according to the characteristics of the subject. The frequency and form of the model's implementation should be adjusted by taking into account instructional objectives, content structure, and student needs. In this way, it will be possible both to benefit from the advantages offered by the model and to maintain student motivation and balance in the learning process. As a result, the flipped learning model should be implemented not with a single rigid pattern throughout the entire instructional process, but in a content-oriented, differentiated, and need-sensitive manner.

## Discussion and Conclusion

The present study examined the effects of the flipped learning approach on middle school students' computational thinking skills and attitudes toward coding. The findings indicate that the implementation of the flipped learning model contributed positively to both cognitive and affective dimensions of learning. Improvements observed in computational thinking skills and attitudes toward coding suggest that instructional environments combining technological support with active learning experiences may facilitate multiple aspects of student development.

A significant increase was observed in computational thinking scores among students in the experimental group following the implementation process. This finding is consistent with Wing's (2006) conceptualization of computational thinking as a process involving problem-solving, abstraction, algorithmic reasoning, and logical thinking. Computational thinking has increasingly been viewed not only as a technical programming skill but also as a broader cognitive competency supporting systematic approaches to solving complex problems. Similarly, Grover and Pea (2013) emphasized that computational thinking develops through active participation in programming experiences and structured problem-solving processes. Shute et al. (2017) further argued that coding activities provide opportunities for learners to strengthen higher-order thinking skills by encouraging reasoning, analysis, and iterative problem solving.

The findings also revealed improvements in students' attitudes toward coding after the instructional intervention. Positive attitudes toward coding are associated with increased participation, stronger engagement, and sustained learning experiences (Abdüsselam & Uzoğlu, 2022; Kalelioğlu, 2015). One possible explanation for this outcome may be related to the structure of the flipped learning model itself. Allowing students to access learning materials before class may reduce uncertainty and increase learners' confidence during classroom activities. Students who have prior exposure to instructional content may participate more actively and approach coding tasks with greater self-confidence.

Qualitative findings provide additional support for this interpretation. Students frequently stated that they were able to review instructional videos whenever needed and experienced increased opportunities for interaction with

the teacher during classroom activities. These experiences align with previous research suggesting that flipped learning transforms classroom sessions from content delivery settings into more interactive and student-centered environments (Bishop & Verleger, 2013; Zainuddin & Halili, 2016). Similarly, Lo and Hew (2017) reported that flipped learning environments may increase student participation and engagement by providing opportunities for more active classroom experiences.

The overall findings are also supported by findings from meta-analytic studies. Cheng et al. (2019) reported that flipped classroom practices generally have positive effects on student learning outcomes across different educational settings. Similarly, van Alten et al. (2019) found that learning outcomes improved particularly when classroom time in flipped environments was used for active learning and application activities. Zheng et al. (2020) additionally reported positive effects on both academic achievement and learning motivation. The consistency between these findings and the results of the present study strengthens the interpretation that flipped learning may represent an effective instructional approach for coding education.

Students' opinions regarding text-based coding also provide valuable insights. Participants described text-based coding as more abstract and demanding than block-based environments and indicated that greater attention and precision were required during programming activities. Similar findings have also been reported in studies emphasizing that novice learners may experience higher cognitive demands in text-based programming environments (Weintrop et al., 2016; Bers, 2018). Despite these challenges, students also reported improvements in conceptual understanding, reductions in coding errors, and increased confidence over time.

The distribution of cognitive demands across different stages of learning may help explain these findings. Rather than concentrating all learning processes within classroom sessions, students had opportunities to familiarize themselves with concepts before class and devote classroom time to guided practice and problem-solving activities. Particularly in coding education, where learners frequently encounter syntax errors and logical difficulties, teacher-supported classroom activities may contribute to deeper understanding and more meaningful learning experiences.

Although many studies report positive outcomes associated with flipped learning environments, findings in the literature are not entirely consistent. For example, Erdem (2018) did not identify significant differences between flipped and traditional instructional approaches. Such inconsistencies may stem from differences in implementation duration, participant characteristics, technological infrastructure, or instructional design. Therefore, the effectiveness of flipped learning should be interpreted within the context in which it is implemented rather than being considered universally effective.

Overall, the findings suggest that integrating text-based coding activities with a flipped learning environment contributes not only to the development of computational thinking skills but also to students' learning experiences and perceptions of coding. Considering the increasing importance of computational thinking and digital literacy skills in contemporary education, instructional environments that support both academic performance and learner engagement may provide important contributions to educational practice.

## Recommendations

In line with the research findings, it has been concluded that the flipped learning model can be used as an effective instructional approach in coding education. In this context, it is recommended for educational practitioners that the flipped learning model be systematically integrated, especially into courses that include text-based programming content. It is important for teachers to support students' pre-class preparation by developing digital content (videos, online quizzes, interactive code editors, etc.), as this allows in-class time to be devoted to practice. In this way, students will be able to manage their individual learning processes and take an active role in classroom activities. The scope of in-service training programs should be expanded to enhance teachers' digital pedagogical competencies, and guidance should be provided to teachers on implementing the flipped learning model. In particular, content integrated with the flipped learning model should be developed for information technologies courses and made accessible to all teachers.

Strategic planning can be carried out to integrate the flipped learning model with innovative instructional approaches into the existing curriculum. Especially in practice-oriented courses such as information technologies, mathematics, science, and technology, the widespread adoption of these models will support students in developing both 21st-century skills and individual learning autonomy. Students' access to an appropriate home

study environment aligned with this model directly affects its success. Families should be encouraged to take on supportive roles in this process and to monitor students' out-of-school learning activities.

For future research, it is recommended that longitudinal studies be conducted to test the applicability of the flipped learning model across different age groups and educational levels. The limitation of this study to a short-term (8-week) implementation provides limited data on long-term learning outcomes. In addition, multidimensional studies should be carried out to examine the effects of the flipped learning model not only on academic achievement but also on variables such as students' self-confidence, motivation, problem-solving strategies, and creative thinking skills. Considering the cognitive difficulties, syntactic errors, and conceptual misunderstandings that students encounter while coding, the impact of the flipped learning model in these areas should be analyzed in depth.

Finally, how the flipped learning model interacts when used together with other instructional methods (for example, project-based learning, collaborative learning) should also be investigated. Supporting coding education not only with individual learning but also with collaboration-based learning processes will enable students to develop skills in social learning environments as well. In this way, the flipped learning model can gain a structure that supports not only individual achievement but also a collective learning culture. In this context, newly developed instructional models are important in terms of strengthening the pedagogical foundations of digital transformation in education.

## Scientific Ethics Declaration

* The authors declare that the scientific ethical and legal responsibility of this article published in JESEH journal belongs to the authors

## Conflict of Interest

* The authors declare that they have no conflicts of interest

## Funding

* The authors declare that no specific funding was received from any agency in the public, commercial, or non-profit sectors for this research.

## Acknowledgements or Notes

* This article is derived from an unpublished master's thesis titled "The effect of flipped learning used in text-based coding education on students' computational thinking skills and attitudes," published by the Institute of Educational Sciences at Niğde Ömer Halisdemir University.

## References

Abdüsselam, M. S., & Uzoğlu, M. (2022). Investigation of middle school students' attitudes toward coding according to different variables. *International Journal of Turkish Education Sciences, 2022*(18), 81–92. https://doi.org/10.46778/goputeb.1028285

Abeysekera, L., & Dawson, P. (2015). Motivation and cognitive load in the flipped classroom: Definition, rationale and a call for research. *Higher Education Research & Development, 34*(1), 1–14. https://doi.org/10.1080/07294360.2014.934336

Arslan, R., Azgınoğlu, N., & Taşyürek, M. (2022). *C programming from beginner to advanced level*. Nobel Publishing.

Bergmann, J., & Sams, A. (2023). *Flip your classroom: Reach every student in every class every day* (Rev. ed.). International Society for Technology in Education.

Bers, M. U. (2018). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.

Bishop, J. L., & Verleger, M. A. (2013). The flipped classroom: A survey of the research. *ASEE Annual Conference and Exposition Proceedings*. https://doi.org/10.18260/1-2--22585

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice*. European Commission Joint Research Centre. https://doi.org/10.2791/792158

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the Annual Meeting of the American Educational Research Association* (pp. 1–25).

Büyüköztürk, Ş. (2002). Factor analysis: Basic concepts and its use in scale development. *Educational Administration: Theory and Practice, 32*(32), 470–483.

Büyüköztürk, Ş., Kılıç-Çakmak, E., Akgün, Ö., Karadeniz, Ş., & Demirel, F. (2019). *Scientific research methods* (6th ed.). Pegem Academy.

Cheng, L., Ritzhaupt, A. D., & Antonenko, P. (2019). Effects of the flipped classroom instructional strategy on students' learning outcomes: A meta-analysis. *Educational Technology Research and Development, 67*, 793–824. https://doi.org/10.1007/s11423-018-9633-7

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates.

Cohen, L., Manion, L., & Morrison, K. (2018). *Research methods in education* (8th ed.). Routledge.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT Press.

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). Sage.

Demirkol, Z. (2020). *Coding for children*. Pusula Publishing.

Erdem, E. (2018). *Investigation of different instructional strategies in programming instruction within block-based environments in terms of various variables* [Master's thesis, Başkent University].

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics, 5*(1), 1–4. https://doi.org/10.11648/j.ajtas.20160501.11

Field, A. (2018). *Discovering statistics using IBM SPSS statistics* (5th ed.). Sage.

Ghasemi, A., & Zahediasl, S. (2012). Normality tests for statistical analysis: A guide for non-statisticians. *International Journal of Endocrinology and Metabolism, 10*(2), 486–489. https://doi.org/10.5812/ijem.3505

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In *Computer science education: Perspectives on teaching and learning in school* (pp. 19–38).

Karaman, U., & Büyükalan-Filiz, S. (2019). Development of the attitude scale toward coding education. *Future Visions Journal, 3*(2), 36–47. https://doi.org/10.29345/futvis.80

Kalelioğlu, F. (2015). A new way of teaching programming skills to K–12 students: Code.org. *Computers in Human Behavior, 52*, 200–210. https://doi.org/10.1016/j.chb.2015.05.047

Kandemir, C. M. (2018). Text-based programming. In Y. Gülbahar & H. Karal (Eds.), *Programming instruction from theory to practice* (pp. 297–336). Pegem Academy.

Knuth, D. E. (1997). *The art of computer programming* (3rd ed.). Addison-Wesley.

Lo, C. K., & Hew, K. F. (2017). A critical review of flipped classroom challenges in K–12 education: Possible solutions and recommendations for future research. *Research and Practice in Technology Enhanced Learning, 12*(1), Article 4. https://doi.org/10.1186/s41039-016-0044-2

Lo, C. K., & Hew, K. F. (2020). A comparison of flipped learning with gamification, traditional learning, and online independent study: The effects on students' mathematics achievement and cognitive engagement. *Interactive Learning Environments, 28*(4), 464–481. https://doi.org/10.1080/10494820.2018.1541910

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Sage.

Scot, A. L. (2018). *21st century learning for early childhood framework*. Partnership for 21st Century Learning.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Tabachnick, B. G., & Fidell, L. S. (2013). *Using multivariate statistics* (6th ed.). Pearson.

Taylor, M., Harlow, A., & Forret, M. (2010). Using a computer programming environment and an interactive whiteboard to investigate some mathematical thinking. *Procedia – Social and Behavioral Sciences, 8*, 561–570. https://doi.org/10.1016/j.sbspro.2010.12.078

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*(1), 230–265.

van Alten, D. C. D., Phielix, C., Janssen, J., & Kester, L. (2019). Effects of flipping the classroom on learning outcomes and satisfaction: A meta-analysis. *Educational Research Review, 28*, Article 100281.

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 199–208). ACM. https://doi.org/10.1145/2771839.2771860

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. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Yıldırım, A., & Şimşek, H. (2021). *Qualitative research methods in the social sciences* (12th ed.). Seçkin Publishing.

Zainuddin, Z., & Halili, S. H. (2016). Flipped classroom research and trends from different fields of study. *International Review of Research in Open and Distributed Learning, 17*(3). https://doi.org/10.19173/irrodl.v17i3.2274

Zheng, L., Bhagat, K. K., Zhen, Y., & Zhang, X. (2020). The effectiveness of the flipped classroom on students' learning achievement and learning motivation: A meta-analysis. *Educational Technology & Society, 23*(1), 1–15.

## Author(s) Information

**Nergiz Agacdelen**
Nevsehir Private Kardelen College
Nevsehir, Türkiye
Contact e-mail: nergizagacdelen@gmail.com
ORCID iD: https://orcid.org/0009-0005-3264-3590

**Oguz Çetin**
Nigde Omer Halisdemir University, Faculty of Education, Department of Mathematics and Science Education, Faculty Member, Niğde, Türkiye
Contact e-mail: oguzcetin@ohu.edu.tr
ORCID iD: https://orcid.org/0000-0002-0986-5137