

A Comparative Case Study to Experiences of High School Students Using Text-Based versus Hybrid-Based Environments in Programming Education¹

Alper Ünal² Fatma Burcu Topu³

To cite this article: Ünal, A., & Topu, F. B. (2022). A comparative case study to experiences of high school students using text-based versus hybrid-based environments in programming education. *e-Kafkas Journal of Educational Research*, 9, 492-512. doi:10.30900/kafkasegt.1053820

Research article

Received:05.01.2022

Accepted:07.07.2022


Abstract

This study aimed to comparatively determine the experiences of high school students in programming language education via text-based or hybrid-based programming environments. A comparative case study was conducted in this study. The participants consisted of a total of 19 high school students with no previous experience in any programming language, nine of them in the text-based programming group and ten of them in the hybrid-based programming group. The qualitative data were obtained with a semi-structured interview at the end of the 10-week programming education process and analyzed by content analysis. The findings were presented in dimensions of difficulties and conveniences in a programming language course, anxiety about the programming process, course outcomes, and their preferences for future programming courses. In each dimension, even if common codes were obtained for both groups in some themes, the effects of these codes on students differed in each group. According to the findings, in the programming process, students faced some difficulties and conveniences in terms of mental effort. While “trying to figure out where they made a mistake” created a difficulty, “using comprehensible visual elements in the hybrid-based environment” as a convenience had the highest frequency among the codes. Some situations caused learning anxiety in students such as worry about failing, while others did not. The students achieved positive and negative course outcomes. “Understanding the logic of coding and acquiring programming skills” which was one of the positive outcomes had the highest frequency. In addition, students' preferences regarding whether or not to attend the future programming courses changed for various reasons. “Unwilling to programming language education” was one of these findings. Considering the scarcity of programming education studies via text-based and hybrid-based programming environments, the results and implications of this study are to strengthen future research by providing rich data.

Keywords: Text-based programming, hybrid-based programming, programming language education, high school students

¹ This study was derived from a part of the first author's Master's Thesis and organized by under the supervision second author. Retrieved from Council of Higher Education database in Turkey (Thesis No. 551146).

²  Author, ICT Teacher, alpeerunal@hotmail.com, Ministry of National Education, Antalya, Turkey

³  Corresponding Author, Assist. Prof. Dr., burcutopu@hotmail.com, fburcu.topu@atauni.edu.tr, Atatürk University, Kazım Karabekir Education Faculty, Erzurum, Turkey

Programlama Eğitiminde Metin-Tabanlı ve Hibrit-Tabanlı Ortamları Kullanan Lise Öğrencilerinin Deneyimlerine İlişkin Karşılaştırmalı Bir Durum Çalışması¹

Alper Ünal² Fatma Burcu Topu³

Atıf: Ünal, A., ve Topu, F. B. (2022). A comparative case study to experiences of high school students using text-based versus hybrid-based environments in programming education. *e- Kafkas Eğitim Araştırmaları Dergisi*, 9, 492-512. doi:10.30900/kafkasegt.1053820

Araştırma Makalesi

Geliş Tarihi:05.01.2022


Kabul Tarihi: 07.07.2022


Öz

Bu çalışma, metin-tabanlı veya hibrit-tabanlı programlama ortamları ile yapılan programlama dili eğitimine katılan lise öğrencilerinin deneyimlerini karşılaştırmalı olarak belirlemeyi amaçlamaktadır. Karşılaştırmalı durum çalışmasının temel alındığı bu çalışmaya, herhangi bir programlama dili deneyimi olmayan, dokuzu metin-tabanlı programlama grubunda ve onu hibrit-tabanlı programlama grubunda toplam 19 lise öğrencisi katılmıştır. 10 haftalık programlama eğitimi sonunda yarı yapılandırılmış görüşme ile elde edilen nitel verilere içerik analizi uygulanmıştır. Bulgular, programlama dili dersindeki zorluklar ve kolaylıklar, programlama sürecine ilişkin kaygı, ders çıktıları ve öğrencilerin gelecekteki programlama derslerine yönelik tercihleri boyutlarında sunulmuştur. Her boyutta bazı temalarda her iki grup için ortak kodlar elde edilse de, bu kodların öğrenciler üzerindeki etkileri gruplara göre farklılık göstermiştir. Elde edilen sonuçlara göre, programlama sürecinde öğrenciler zihinsel çaba açısından bazı zorluklar ve kolaylıklarla karşılaşmışlardır. Kodlar arasında “nerede hata yapıldığını bulmaya çalışmak” zorluk olarak, “hibrit tabanlı ortamda anlaşılır görsel öğeleri kullanmak” kolaylık olarak en yüksek sıklığa sahiptir. Başarısızlık endişesi gibi bazı durumlar öğrencilerde öğrenme kaygısına neden olurken, bazıları da kaygıya neden olmamıştır. Öğrenciler olumlu ve olumsuz ders çıktıları elde etmişlerdir. Olumlu sonuçlardan biri olan “kodlama mantığını anlama ve programlama becerisi edinme” en yüksek sıklığa sahiptir. Ayrıca, öğrencilerin gelecekteki programlama kurslarına katılıp katılmama tercihleri çeşitli nedenlerle değişmiştir. Bu bulgulardan biri de “programlama dili eğitimine karşı isteksizlik”dir. Metin-tabanlı ve hibrit-tabanlı programlama ortamları ile yapılan programlama eğitimi çalışmalarının azlığı göz önüne alındığında, bu çalışmanın sonuçları ve çıkarımları, zengin veriler sağlayarak gelecekteki araştırmaları güçlendirecektir.

Anahtar Sözcükler: Metin-tabanlı programlama, hibrit-tabanlı programlama, programlama dili eğitimi, lise öğrencileri

¹ Bu çalışma, birinci yazarın yüksek lisans tezinin bir bölümünden derlenmiş ve ikinci yazar danışmanlığında düzenlenmiştir. Türkiye Yükseköğretim Kurulu veri tabanından alınmıştır (Tez No. 551146).

²  Yazar, BTR Öğretmeni, alpeerunal@hotmail.com, Milli Eğitim Bakanlığı, Antalya, Türkiye

³  Sorumlu Yazar, Dr. Öğretim Üyesi, burcutopu@hotmail.com, fburcu.topu@atauni.edu.tr, Atatürk Üniversitesi, Kazım Karabekir Eğitim Fakültesi, Erzurum, Türkiye

Introduction

Nowadays, in many areas regarding information technology, the trend is shifting towards producing and using programmable technological devices. It is a critical issue to raise the manpower to provide rational solutions for these systems all over the world. For this reason, it is necessary to provide learning environments to equip students with algorithmic thinking, programming skills, and developing software by using a programming language (Hsu & Hwang, 2021; Jancheski, 2017; Shin, Park, & Bae, 2013). Accordingly, many countries include information technology and computer science courses in the curriculum to acquire students these skills at an early age (Demirer & Sak, 2016; Grover & Pea, 2013; Ministry of National Education [MoNE], 2017, 2018; Vaidyanathan, 2013).

The various programming languages such as C, Java, Python, etc. are preferred in programming education. Each programming language has its syntax. Although the selection of programming language differs according to the purpose of use, object-oriented Python is one of the most commonly used programming languages (Github, 2019). Free download Python has a simpler syntax than other common programming languages. It can be used easily in any environment regardless of the platform as an open source without needing a compiler. These features make Python a user-friendly and powerful programming language (Adi & Kitagawa, 2019; Lutz, 2013; Sanner, 1999). Thus, in line with the recommendation of the MoNE as well (Gülbahar & Kalelioğlu, 2018), this study is based on the process of teaching Python programming language to high school students without any programming language experience.

The programming languages include a wide range of subjects and various concepts expressed in English. Beginner programmers need to learn the syntax and logic of the programming language, as well as to design algorithms (Gomez, Moresi, & Benotti, 2019; Hsu & Hwang, 2021; Tuomi, Multisilta, Saarikoski, & Suominen, 2018). However, many beginner programmers, due to their inexperience in programming, try to memorize the general rules of these languages. This is a remarkable factor that hinders the permanency of their programming success (Gomes & Mendes, 2007; Salleh, Shukur, & Judi, 2018). Computer programming is a complex mental process. It causes the students more mental effort to effectively use their working memory (Asai, Phuong, Harada, & Shimakawa, 2019; Kelleher & Pausch, 2005). It affects the learning process negatively (Mavilidi & Zhong, 2019; Moreno, 2010; Sweller, 2010). In other words, it is likely to prevent the programming education continues efficiently (Garner, 2002; Stachel et al., 2013; Yukselturk & Altiok, 2017). To reduce the mental effort, it is suggested that comprehensive content is divided the pieces, and complex tasks are presented from easy to difficult week by week. Thus, knowledge retention increases in long-term memory (Çakiroğlu et al., 2018; Mavilidi & Zhong, 2019). In addition, mentioned factors above may cause the learning process to continue with negative feelings such as boredom, low interest, high anxiety, lack of self-confidence, reluctance toward learning a programming language, and even interrupting it (Chang, 2005; Gomes & Mendes, 2007; Hsu & Hwang, 2021; Owolabi, Olanipekun, & Iwerima, 2014; Tsai, 2019).

It is suggested to teach programming languages having simple syntax to students encountering a text-based programming education for the first time. In addition, it can be preferred easy-to-use block-based programming environments to facilitate algorithmic thinking. Thus, an opportunity arises to reduce the negative emotions that discourage beginner programmers from acquiring programming skills (Asai et al., 2018; Çakiroğlu, Çevik, Köşeli, & Karaman, 2021; Mumcu, Mumcu, & Çakiroğlu, 2021; Tsai, 2019; Topalli & Cagiltay, 2018; Yukselturk & Altiok, 2017). A block-based programming environment consists of code blocks with various colours and features. In order to create an algorithm, puzzle pieces-like block structures are easily combined by drag and drop (Gomez et al., 2019; López, Otero, & García-Cervigón, 2021). Thanks to block-based environments, such as Scratch, Code.org, and Alice, students make fewer syntax errors and have low cognitive challenges (Çakiroğlu et al., 2021; Mumcu et al., 2021; Rahaman, Mahfuj, Haque, Shekdar, & Islam, 2020; Tsai, 2019), and they also have a more positive attitude towards learning programming (Seraj, Katterfeldt, Bub, Autexier, & Drechsler, 2019; Yukselturk & Altiok, 2017). Hsu and Hwang (2021) have revealed that students engaging with block-based programming tasks have low programming anxiety and a more enjoyable learning experience. Although, students cannot learn any programming language via mentioned block-

based environments (Topalli & Cagiltay, 2018; Mumcu et al., 2021), this is possible via Google Blockly as a hybrid-based programming environment (Fraser, 2015).

The hybrid-based programming environments have features of both text-based and block-based programming environments (Weintrop & Wilensky 2018). In this way, hybrid-based programming makes it possible to see the programming logic and syntax at the same time. One of these environments is Blockly which transforms blocks into code syntax of various programming languages (e.g. Python, Javascript), and shows visual and textual codes simultaneously (Jung, Nguyen, & Lee, 2021; Rahaman et al., 2020; Weintrop & Wilensky 2017). It is also possible to add new functions to the code blocks (Adi & Kitagawa, 2019; Bak et al., 2020; Fraser, 2015; Jung et al., 2021; Valsamakis, Savidis, Agapakis, & Katsarakis, 2020). Blockly is impossible for students to make syntax errors during the code writing by dragging and dropping the blocks. This enables students to focus on programming logic (López et al., 2021). These features of Blockly reduce the complexity of programming and facilitate beginner programmers to learn the programming languages (Chen et al., 2021; Rahaman et al., 2020; Sano & Kagawa, 2019; Winterer et al., 2020). According to Adi and Kitagawa (2019), Blockly is a convenient environment for novice programmers to start learning Python programming language. On the other hand, in hybrid-based programming, students also need to comprehend the logic of coding, while they combine blocks correctly. This may cause more mental effort and also make it difficult for the learning process (Debue & Van De Leemput, 2014; Ionescu, 2021; Sweller, 2010). Considering these contradictory statements in the literature, it is important to investigate students' experiences in the learning process of a programming language by using Blockly as a hybrid-based programming environment.

Previous studies are mostly in the engineering field, and on the usability of a technological system developed using Blockly (Adi & Kitagawa, 2019; Bak, Chang, & Choi, 2020; Chen, Chen, Yu, & Lee, 2021; Ionescu, 2021; Jung et al., 2021; Rahaman et al., 2020; Rodríguez-Gil et al., 2019; Weintrop, Shepherd, Francis, & Franklin, 2017; Winterer, Salomon, Köberle, Ramler, & Schittengruber, 2020). There are also available studies using Blockly in programming education at different grade levels. These studies are examined programming skills as well as the various variables such as intention and attitude (Seraj et al., 2019; Yiğit, 2016), pedagogical interactions (López et al., 2021), problem-solving (Bubnó & Takács, 2017), object-oriented programming concepts (Su & Hsu, 2017), interaction with various programming interfaces (Weintrop & Wilensky, 2018, 2019), collaborative visual programming (Valsamakis et al., 2020).

On the other hand, there are studies in the literature comparing block-based and hybrid-based programming environments. Seraj et al. (2019) compared Scratch versus Blockly, Weintrop and Wilensky (2019) compared Snap! versus Pencil. cc. Only a few studies are comparing text-based versus hybrid-based programming environments. Weintrop and Wilensky (2018) compared block-based, text-based, and hybrid-based programming environments. As to Yiğit (2016), parallel to our study, he compared text-based and hybrid-based programming environments. However, quantitative research methods were used in these studies and statistical results were presented.

As for our study, we compared the programming experiences with the qualitative research perspective of the high school students using a text-based programming environment (Python editor) in one group and using a hybrid-based programming environment (Blockly) in the other group. This study has a 10-week (40 hours) implementation process as a long period and presents detailed results and implications to contribute to the literature concerning the use of text-based and hybrid-based programming environments in the Python programming language learning process. In addition, our study could make it possible to identify factors influencing students such as mental effort, anxiety, self-confidence, and motivation for learning a programming language. Thus, it will guide future studies on programming language education in high schools by ensuring rich data. This study is also important in terms of improving the computer science course curriculum in high schools by MoNE. Consequently, this study differs from previous studies in these aspects. Accordingly, the research question of the study is following.

What are the experiences of students participating in programming language education via text-based or hybrid-based programming environments?

Method

Research Design

This study is based on a comparative case study, one of the qualitative research designs (Yin, 2003). This research design comparatively examines to understand the similarities and differences between cases (Baxter & Jack, 2008; Stake, 2006). Accordingly, in the current study, the experiences of high school students in one group using Python editor as a text-based programming environment and in the other group using Blockly as a hybrid-based programming environment during the 10-week Python programming language learning process were analyzed in detail. The results were presented comparatively and holistically according to the groups. Thus, it was aimed to obtain detailed and rich data on how different situations affect students' experiences, reveal similar and contrasting results, and suggest implications (Barlett & Vavrus, 2017; Goodrick, 2014).

Participants

Participants consisted of 19 high school preparatory-grade students in total, 9 (7 girls, 2 boys) in the text-based programming group and 10 (7 girls, 3 boys) in the hybrid-based programming group. The students in both groups were determined with the purposive sampling method. These students did not participate in any programming language training before and performed all of the 10-week Python programming language tasks of our study. Accordingly, this study was conducted with volunteers among these students to participate in the interview to obtain in-depth information about their programming experiences.

Data Collection Instrument

A semi-structured interview form with ten questions was developed by researchers to reveal the experiences of students in two groups using different programming platforms in detail. To increase the reliability of the instrument, two information technology teachers, and instructional technologies experts, female, and male students checked the intelligibility of questions in the interview form. Interview questions were asked to two female and two male students who participated in the pilot study in both groups. To ensure the validity and reliability of the study, before the interviews, the researchers stated that the students would freely express their opinions and would not receive any score because of their positive or negative opinions. They also emphasized the importance of their opinions to make better this programming education process. The face-to-face interviews were conducted with volunteer students in both groups at the end of the 10-week implementation process. Accordingly, it was expected to explain the students' positive and negative thoughts about the Python programming language learning process, and the reasons for challenging or facilitating in the programming tasks, they performed with the used programming environment. In addition, it was asked to express their programming achievements and intention to continue programming education in the future. The interviews were voice recorded with permission from students to prevent the loss of data.

Process

A 10-week Python programming language education was conducted with one group using the Python editor in text-based programming and with the other group using Blockly environment in hybrid-based programming in the computer science course the researchers. Prior to the implementation, the label and text of blocks in Blockly were fixed according to the syntactic structure of sections in Python programming language such as loops, functions, variables, etc.

Weekly task-based activities were developed according to Foundations of Programming unit objectives and subjects (Variables, Conditional Statements, Decision Structures, Loops Structures, Functions, and Lists) in high school computer science course (MoNE, 2018). These activities had the same content in both groups, while the course was carried out based on different programming approaches in each group. The programming codes in these activities were created with Python syntax for the text-based programming group, whereas it was built with the blocks for the hybrid-based programming group and was also possible to display Python syntax on the Blockly interface. Screenshots of a programming activity that had different programming interfaces (Python editor in text-based programming group and Blockly environment in hybrid-based programming groups) are shown in Figure 1 below.

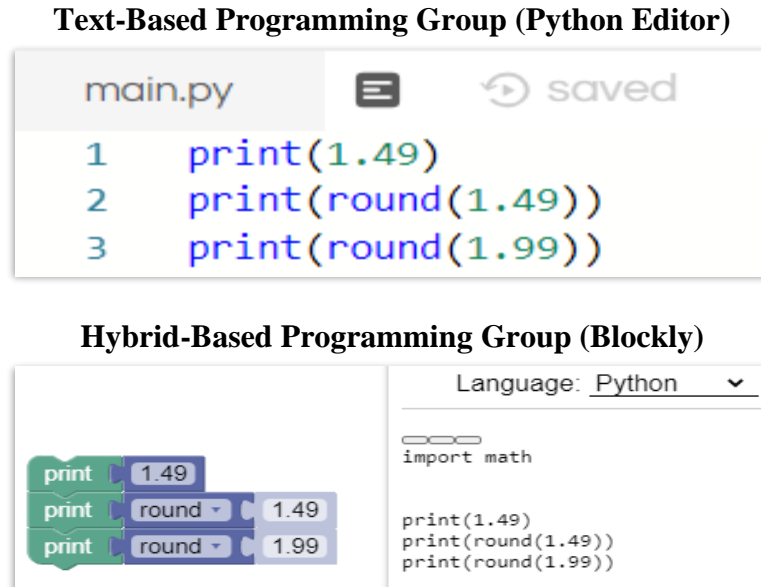


Figure 1. Interfaces of Programming Environments Used in Text-Based and Hybrid-Based Programming Groups

Before the implementation of this study, researchers obtained the Ethics Committee Approval from the university that their work, and from the MoNE in Turkey. Researchers also got students' permission in both groups by promising to keep their names confidential.

At the beginning of the implementation, during one week (four lessons), the computer science teacher, the first researcher, introduced the programming environments to each group and explained the programming education process. He also checked the technical infrastructure of the information technology classrooms for the reliability of the study. 10-week implementation was 40 hours in total for each group (weekly 4 lessons = 2 days * 2 hours). At the beginning of the first lesson of each week, the teacher presented knowledge to students about the Python programming language subject of that week. After that, as seen activity photos in Figure 2, students separately performed weekly programming tasks in the text-based programming group and hybrid-based programming group.

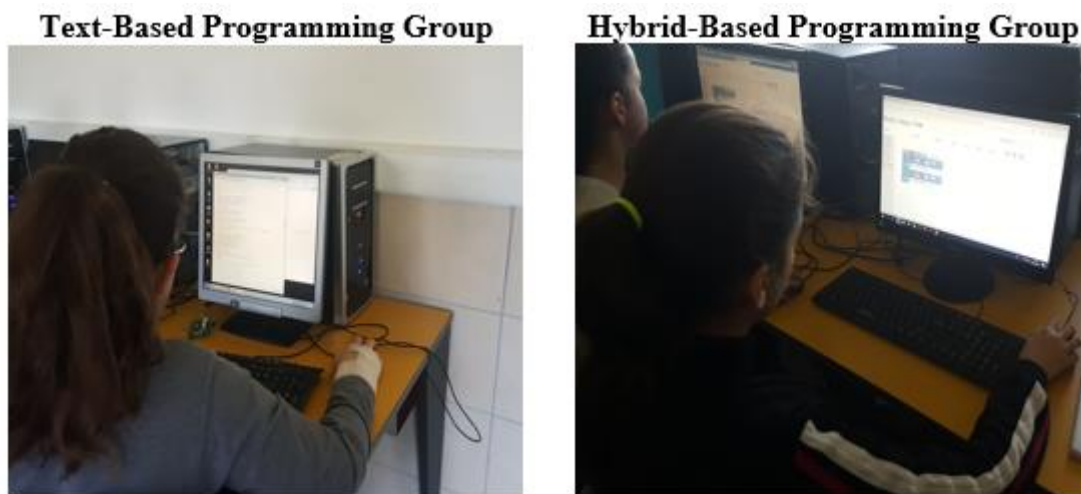


Figure 2. Activity Photos of a Programming Task in Text-Based and Hybrid-Based Programming Groups

Following the students' completion of the task, the teacher explained the correct textual Python codes step by step to both groups. He also demonstrated the appropriate order of the blocks of the task to the Blockly group. Thus, students who could not get the correct programming output were also able to complete the task in both groups.

To prevent the loss of the data and to increase the validity and reliability of the study, it was taken photos and video recordings throughout the process. After the whole implementation was conducted as face-to-face interviews with volunteer students in each group (9 in the text-based programming group and 10 in the hybrid-based programming group) by voice recording.

Data Analysis

Content analysis was conducted to reveal the students' experiences in the Python programming language education process in detail and to comparatively present these findings (Merriam & Tisdell, 2015). Accordingly, first of all, the interview voice recordings collected from students in both groups were transcribed into writing. Data were analyzed using Nvivo 12. The codes that were revealed for each group were combined into themes. According to themes, the frequencies of codes in each group were comparatively presented in tables and figures. These codes were also supported by the quotations of students' statements (SX) in each group (TG for text-based programming group, HG for hybrid-based programming group). To ensure the trustworthiness of the data analysis, the codes, frequencies, themes, and quotations were checked by an instructional technology expert by comparing with photos and videos taken throughout the process, and voice recordings obtained from the interviews.

Findings

According to the findings, even if common codes were obtained for both groups in some themes, the effects of these codes on students differed in each group. The findings were presented in the following titles: "difficulties and conveniences in a programming language course, anxiety about programming process, course outcomes, and students' preferences for future programming courses". The codes in the tables and figures were shown in different colors according to the groups (HG: green, TG: blue, Both Groups: orange), and were supported by sample quotations from the statements of the students in each group (HG/TG, SX, Female/Male).

Difficulties and Conveniences in Programming Language Course

Students in HG and TG faced some difficulties and conveniences in terms of mental effort in the programming language course. The findings are presented in Table 1.

Table 1.
Difficulties and Conveniences in Programming Language Course

Difficulties		
Themes	Codes	f
<i>Having extra mental effort to understand the programming subjects</i>	• Loops Structures	7
	• Lists	5
	• Functions	3
	• Decision Structures	2
	• Flowcharts/Algorithms	2
<i>Finding the source of error during programming</i>	• Trying to understand where one made a mistake/getting ambitious	10
	• Trying over and over	6
	• Feeling negative emotions (angry, bored) when writing erroneous codes	5
	• Giving up	4
	• Asking the classmates for help when one cannot finish the task in time/cannot do it	1
<i>Having low programming course motivation</i>	• Thinking the course is boring/unnecessary	6
	• Considering the course as trivial	2
	• Making enough efforts to pass the course exams	2
	• Lack of interest in the programming course	1
<i>Using programming environment</i>	• Getting used to a new programming environment	3

Table 1. (Continued)

	<ul style="list-style-type: none"> ● Asking the teacher for help while using the programming environment and performing tasks 	3
	<ul style="list-style-type: none"> ● Coding in Python editor 	2
	<ul style="list-style-type: none"> ● Using the English programming language 	1
<i>Using computer</i>	<ul style="list-style-type: none"> ● Dislike using computers 	1
	<ul style="list-style-type: none"> ● Inability to use computers well 	1
Conveniences		
Themes	Codes	f
<i>Using programming environment</i>	<ul style="list-style-type: none"> ● Using explanatory/comprehensible visual elements in Blockly 	4
	<ul style="list-style-type: none"> ● Coding in Python editor 	2
	<ul style="list-style-type: none"> ● Using ready-made drag and drop code blocks in Blockly 	1
	<ul style="list-style-type: none"> ● Using regulatable code blocks in Blockly 	1
<i>Understanding more easily as increasing the programming experience</i>	<ul style="list-style-type: none"> ● Revising by doing example activities 	3
	<ul style="list-style-type: none"> ● Getting used to the programming language 	2
	<ul style="list-style-type: none"> ● Enjoying coding 	1

● HG ● TG ● Both Groups

As seen in Table 1, various themes and codes emerged related to facing difficulties and conveniences in terms of mental effort in the programming language course. According to these findings, sample quotations from students' statements are as follows.

Having extra mental effort to understand the programming subjects

Some of the students in both groups had difficulty understanding loop structures, lists, and functions especially because they were complicated and related to mathematics. Besides, some students also stated that they had similar difficulties in decision structures, flow charts, and algorithms. This caused the extra mental effort to understand the programming subjects. A student's statement: "... I had difficulty in lists and loops structures. For example, when I made a mistake, the loop continued endlessly. Then it was a bit annoying (HG, S10, Male)." Another student's statement: "I didn't know how and where to use "if-else" [decision structures] very well. I felt confused. I didn't have difficulty in writing but I didn't understand the logic for it (TG, S19, Female)."

Finding the source of error during programming

The majority of the students in both groups said that they got ambitious in trying to understand where they had made a mistake and never gave up trying again and again. Yet, some of the students felt angry about making a mistake while coding and gave up coding. In addition, some of them in both groups asked their friends for help when they could not finish tasks. Therefore they had difficulty finding the source of the error during programming. A student's statement: "I made great efforts to write the code, but when I could not catch up with my friends, I asked for their help (TG, S11, Female)." The other students' statements are as follows.

I examined them one by one to see where I made a mistake. I corrected it and tried it again when I found the mistake. I asked the teacher when I did not understand. I got ambitious to do the right it. For this reason, I had difficulty in coding (TG, S14, Female).

When there was an error, I opened a new tab and wrote the program again. But I immediately gave up when there was an error in the program again. I felt frightened and my hand began to tremble (HG, S8, Female).

Having low programming course motivation

Some students in both groups said that they did not make any efforts or did not have any difficulties because they thought the course was boring and unnecessary, because they considered the course trivial and because they did not have interest in the course. This was the reason for low course

motivation. However, there were also students in both groups who made enough effort to pass the course exams. A student's statement: "I didn't make any effort in this course and I had difficulty because I thought it was unnecessary (TG, S12, Female)." Another student's statement: "I made efforts in the exam to prove to what extent I had learned as I thought that it could be beneficial to me in the future (HG, S9, Male)."

Using computers

Some of the students in the TG especially stated that their lack of interest and experience in using computers was a disadvantage in this process. A student's statement: "... I did not like using computers very much. So I had some difficulties when we were writing codes. The course content was complex (TG, S17, Male)."

Using the programming environment

Some of the students in both groups stated that it was difficult to get used to a new programming environment. So they needed teacher guidance to use the programming environment and perform tasks. In addition, students in HG had difficulty in writing codes because of coding language which is based on English. A student's statement: "I had difficulty in getting used to such an environment [Python editor] because we faced it for the first time (TG, S15, Male)." Another student's statement: "When I could not write codes at the first trial, I worried. I could do it at the second trial with help from the teacher (HG, S6, Female)." Another student's statement: "I had difficulty at first. Having it in a foreign language seemed complex to me. But then, I began to learn and get used to it as I encountered the words (HG, S7, Female)."

On the other hand, some of the HG students said that the availability of comprehensible visual elements in the Blockly environment, the regulatability of ready code blocks, and using the drag and drop method provided them with convenience. However, a minority of the TG students stated that transferring the codes on the paper into the computer- Python editor- facilitated them to understand the codes. Some students' statements are as follows. These different views of the students in each group stemmed from the interface features of the programming environments used in text-based and hybrid-based programming approaches.

I didn't have difficulty using Blockly...because we could delete and add blocks using drag and drop. That's why it was more comfortable. The figures were more explanatory, they were separated so we would not forget the codes right away (HG, S1, Female).

I had great difficulty when it was on paper. But it was easier to write codes in a programming environment... at least I learned. I saw what happened [while my codes were working]. So, I didn't have any difficulties using Python editor (TG, S19, Female).

Understanding more easily as increasing the programming experience

Some students in both groups said that they overcame the difficulties as they became more familiar with the programming language and performed example activities. In addition, a student in the TG expressed not having any difficulty because of enjoying coding. A student's statement: "...I began to like programming, coding and so on as I did activities. So, I did not have difficulty in coding (TG, S14, Female)." Another student's statement is as follows. These findings revealed that no matter which programming approach was taken as the basis, the increase in the programming experience of the students made it easier to understand the programming language.

I had difficulty at first. But then, I became more and more familiar with the subject as I encountered more activities. When I got used to it, I understood the underlying logic of coding. And I had no longer difficulty noting down what the teacher did (HG, S7, Female).

Anxiety about Programming Process

It determined that some situations in the programming process caused learning anxiety in students, while the others did not. The findings are presented in Table 2.

Table 2.
Reasons for whether or not the Learning Anxiety in the Programming Process

Themes	Codes			
	Not Anxious	f	Anxious	f
<i>Making the programming error</i>	• Think everybody can make mistakes	9	• Fear of losing popularity in the teacher's eyes	4
	• Not being hesitated to ask the teacher for help	5	• Fear of confusing the codes	2
	• Not being hesitated to make mistakes to learn	4	• Worry about the success of classmates	1
			• Worry about being mocked by classmate	1
<i>Being interested in a programming course</i>	• Like using computer	3	• Dislike using computer	1
	• Considering the course as unnecessary/ underestimate	2		
<i>Being knowledge about programming course</i>	• Attending an algorithm course previously	3	• Attending a programming course for the first time	1
<i>Self-confidence</i>	• Belief in to able to succeed	1	• Worry about failing	5

• HG • TG • Both Groups

As seen in Table 2, in the programming process, various themes and codes emerged related to situations that caused and not caused learning anxiety in students. According to these findings, sample quotations from students' statements are as follows.

Making the programming error

Some of the students in both groups worried about confusing the programming codes and losing popularity in the teacher's eyes. In addition, a few students in the HG were afraid of classmates' reactions. A student's statement: "I feared to confuse the place and order of the codes. I had panic and felt anxious because I thought about which one to do, this or that (HG, S9, Male)." Another student's statement: "I felt a little bit anxious about the change of the teacher's perspective if I could not do it (TG, S17, Male)." Another student's statement: "I was ashamed of my friends about not being able to do coding. Because some of them liked using computers whereas I did not know much (HG, S6, Female)."

However, most of the students in both groups thought that they were not professionals in programming, they did not hesitate to make a coding error and ask the teacher for help. A student's statement: "I made a coding error. But I did not hesitate to ask for the teacher's help (TG, S14, Female). Another student's statement is as follows. These findings supported that making programming errors caused anxiety in some students without considering in which group the students were.

I never hesitated to make a coding error. I was not a professional, this was my first year [in programming]. I was at the stage of learning. There were lots of classmates who were trying to learn, just like me (HG, S8, Female).

Being interested in a programming course

Some students in both groups liked using computers and did not feel any anxiety about the programming course, a student in the TG thought the opposite because of disliking using a computer. In addition, a few students in the HG and TG did not feel anxiety because of the belief of no need for such a course. A student's statement: "I have been good with computers since I was 6-7 years old and some games I played, some contained codes required at least a few English words. So, I didn't feel anxiety in this course (HG, S9, Male)." Another student's statement: "... I didn't feel anxiety. It was nice to do something related to computers. We were relaxed in IT class and this course (TG, S14, Female)." Another student's statement: "I was not interested in this course so I did not listen to lessons

(TG, S15, Male).” According to these findings, whether some students were interested in the lesson or not, regardless of which group they were in, affected their programming anxiety.

Being knowledgeable about programming course

Some students in both groups did not feel anxiety about the programming course because of attending the block-based algorithm course in secondary school. However, students in the HG stated to felt learning anxiety because of attending such a course for the first time. A student's statement: “At first, I felt a little bit anxious. Because I was going to learn a language I had never seen before, Python. I was afraid of not being able to do it (HG, S1, Female).” Another student's statement: “I didn't feel much anxiety because we had also seen algorithm [Scratch] at secondary school, but I didn't know it would be like this [Python programming language] (TG, S16, Female).” Accordingly, participation in algorithm training in previous years affected some students' anxiety about the programming language course.

Self-confidence

Although a student in the HG believed to be able to achieve the learning objectives of the programming course, some students in both groups were worried about failing. Accordingly, self-confidence affected the programming anxiety. A student's statement: “I felt anxiety at first. Because I thought it was a difficult course and I would not be successful (TG, S13, Female).” Another student's statement is as follows.

I did not feel anxiety because I thought of coding as a puzzle. First, I divided it into pieces and I thought about where to start a piece of it. I did it immediately because combining puzzle pieces was easy for me (HG, S8, Female).

Course Outcomes

Students in HG and TG achieved positive and negative course outcomes at the end of the implementation process. The findings were presented in Figure 3. According to codes related to course outcomes in Figure 3, sample quotations from students' statements are as follows.

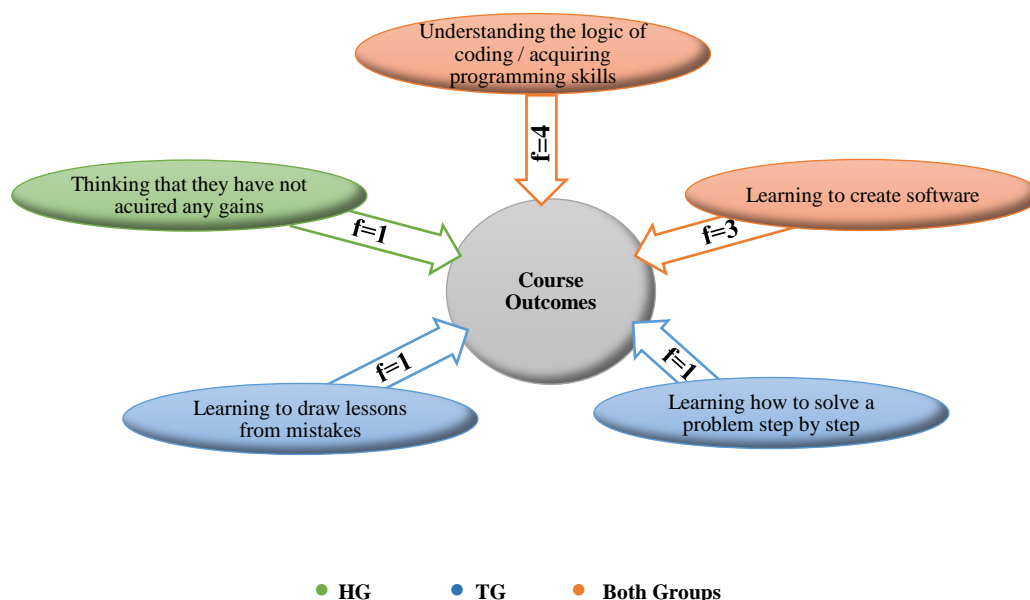


Figure 3. Findings Related to Course Outcomes

Understanding the logic of coding/acquiring programming skills and learning to create software were identified as the common positive outcomes for both groups. A student's statement: “...Now, I learned the logic of how to write programming codes (HG, S5, Male).” Another student's

statement: “I learned lots of things related to programming languages in this course. I think that there are lots of things I can do. For example, I can make a program (PG, S19, Female).”

One of the TG students learned to draw a lesson from his mistakes and another student learned how to solve a problem step by step. A student's statement: “I learned what and how to do it step by step when I encountered a problem (TG, S11, Female).” Another student's statement: “I think I learned to draw a lesson from my mistakes in computer coding course (TG, S15, Male).” Yet, a student in the HG stated that she had not acquired any gains in the course. This student's statement “I think I did not learn anything. The course did not contribute to me (HG, S4, Female).” Consequently, some of the students in both groups who attended this course achieved positive outcomes while very few had negative outcomes.

Students’ Preferences for Future Programming Courses

The students’ preferences in the HG and TG whether to attend future programming courses or not are presented in Figure 4. According to codes related to attending future programming courses in Figure 4, sample quotations from students' statements are as follows.

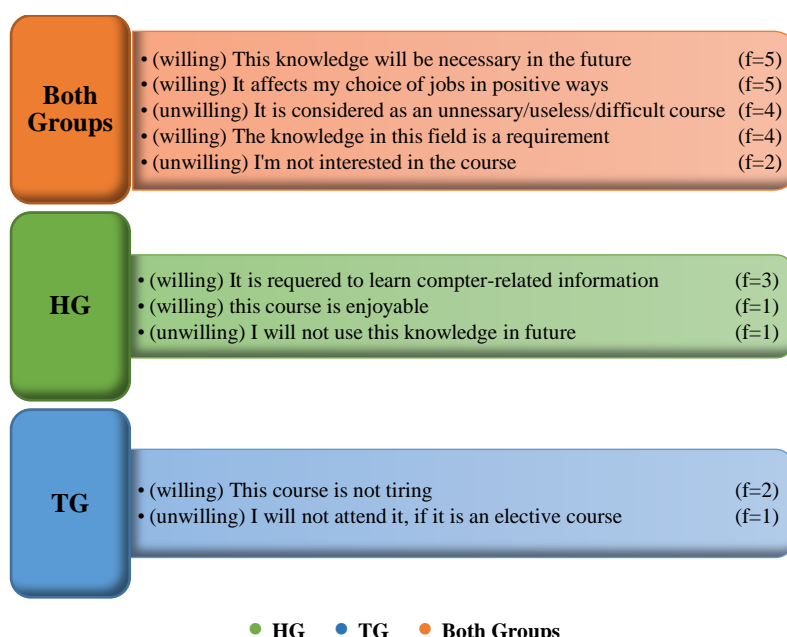


Figure 4. Students’ Preferences in HG and TG for Future Programming Courses

Many students in both groups believed that the programming knowledge was necessary for the future, it would affect their choice of jobs in positive ways, and the knowledge in this field is a requirement. A student's statement: “I think the knowledge about programming and coding will be useful in choosing a department at university or in our future job. For this reason, it is a necessary course (HG, S9, Male).” Another student's statement: “It is necessary to learn these things [programming language]. If we choose IT at university, or a job in computers, it is necessary for us to get prepared for the future (TG, S15, Male).”

A student in HG thought this course was enjoyable and a student in TG thought this course was not tiring. For this reason, they wanted to attend this course in the future. A student's statement: “I would still choose the course if it were elective. Because we had fun and we learned new things about computers (HG, S7, Male).”

However, a few students in both groups did not want to attend the future programming course because they did not have an interest in the course, they could not make sense of the need for learning a programming language, and they considered it unnecessary, and useless and difficult course. In addition, they preferred to have elective courses instead of compulsory courses. A student's statement: “...I did not like this course because I had difficulty in it. I do not think it is a necessary course (TG,

S16, Female).” Another student's statement: “I think this course is unnecessary. It should have been taught to students who liked, were interested in, and curious about it (HG, S4, Female).” Another student's statement: “I would not participate in it if it was an elective course. Because we have just started high school. In my opinion, there are more important courses than this course in high school (TG, S15, Male).” Accordingly, the majority of students in both groups were eager to attend future programming courses for various reasons whereas a few of them were not.

Discussion

In both groups, most of the students made an extra mental effort to understand programming subjects especially Loops Structures, Lists, and Functions. Çakiroğlu et al. (2021) revealed that students could not complete the nested repeat block task in mobile code activities. It can be said that this compelling situation is independent of the coding environment, and is caused by the nature of the programming subject content. Many students had also difficulty finding the source of the error during programming, and they tried over and over to correct the errors. Weintrop and Wilensky (2019) found similar results in a study that measured students' success and error levels to our study. Although this situation challenges students mentally, it is not considered a negative finding as it is a natural part of the programming process. However, some of the students, who had negative feelings such as getting angry and bored when they wrote erroneous code, got ambitious whereas some of them gave up. This situation encountered in both groups proved that the students' reactions were not related to text-based or hybrid-based programming approaches. Li (2016) reported that users acted such reactions during the programming process. The various scaffold models can be used to reduce mental effort according to the reactions observed in students during programming tasks (Salleh et al., 2018).

The majority of the students in both groups stated that they did not fear making programming errors, and were able to learn the programming skill by making mistakes. According to Asai et al. (2019), the long-period implementation in programming education may help students become familiar with programming concepts and overcome programming anxiety. In our 10-week programming training, the programming tasks performed by the students in both groups from simple to complex may have provided this result. As in our study, we recommend that the future studies spread over a long period as well.

On the other hand, It was determined that some students in both groups feared making the programming error. One of the reasons for this is the fear of losing popularity in the teacher's eyes. In addition, a few students, participating in hybrid-based programming, worried about classmates' reactions when they made a programming error. Rogerson and Scott (2010) emphasized that such external factors increased learning anxiety. This finding was in contrast with Hsu and Hwang (2021)'s study which determined that students engaging with block-based programming tasks had low programming anxiety. The finding of "classmate pressure", which emerged in our study and was not encountered in previous studies, is an important keyword that should be considered to reduce learning anxiety in the programming process. In future studies based on hybrid-based programming, it is recommended to take measures to prevent this pressure.

Some students in both groups worried about failing to learn the programming language and were not confident about it. This caused learning anxiety in students. For instance, a student in a hybrid-based programming group was worried about attending such training for the first time. According to Bosch and D'Mello (2017), the reason for such feeling was a lack of knowledge about the course. Regardless of the students who were in which group, those who attended an algorithm course previously did not feel any anxiety about the programming course. It may be able to increase the self-confidence of students by using a method to reduce negative self-assessment of them during the programming process (Gorson & O'Rourke, 2020).

In BG, most of the students thought that the interface features of a hybrid-based programming environment made programming easier. In addition, a student participating in hybrid-based programming did not have such anxiety and believed to be able to succeed in learning a programming language. Weintrop and Wilensky (2019) found out that students use Pencil. cc, a hybrid-based programming environment like Blockly, had self-confidence in terms of programming skills. Moreover, the studies in literature explained that students, who engaged with programming tasks in

such environments, had a more enjoyable learning experience and they felt that the programming process was easier (Adi et al., 2019; Hsu & Hwang, 2021; López et al., 2021; Rahaman et al., 2020). As a matter of fact, hybrid-based programming environments such as Blockly has the potential to reduce complexity in programming education, as it enables to see both the real programming code structure and blocks at the same time (Bak et al., 2020; Rodríguez-Gil et al., 2019; Sano & Kagawa, 2019). Therefore, it can be said that hybrid-based programming environments are suitable for novice programmers to start learning a programming language (Adi et al., 2019; Rahaman et al., 2020). On the contrary, Çakiroğlu et al. (2018) determined that the buttons, colors, and drag/drop features of the block-based programming environment increased the mental effort. Trying to catch the logic of coding while combining the blocks correctly requires focusing on multiple parameters. This may make difficult learning (Ionescu, 2021). These contrasting findings may have resulted from based on different programming approaches in the studies.

In our study, some of the students had difficulty in text-based programming and needed help the classmates while they were performing the programming tasks. Asai et al. (2019), and Topalli and Cagiltay (2018) confirmed that text-based programming required more mental effort. However, some others stated that using a text-based programming environment made writing code easier and thus, they enjoyed coding. Weintrop et al. (2017) also determined that a text-based programming environment improved students' programming abilities similar to professional programmers compared to a block-based one. This finding offers a different perspective for future studies.

In both groups, a small number of students had difficulty getting used to a new programming environment. Similar statements were also quoted in the literature (Moons & De Backer, 2013; Shih 2017). This factor affecting the process in negative ways just at the beginning of teaching programming can be explained as an innovation effect. Moreover, these students also needed teacher guidance while using the programming environment. This guidance is seen as an important factor to facilitate the programming language learning process, as it can reduce students' mental efforts and negative beliefs (Asai et al., 2019; Gorson & O'Rourke, 2020). Yet, during the 10 weeks, these students stated that they got used to the programming environment as the weeks go by and comprehended the programming language more easily as performing the various programming tasks each week. This is evidence that getting experience makes a less mental effort (Gomez et al., 2019; Mavilidi & Zhong, 2019). Accordingly, it can be said that novice programmers have an opportunity to improve their programming abilities thanks to these activities spreading over a long period.

A few students who did not like to use computers had learning difficulties and anxiety in the text-based programming process. According to Weintrop and Wilensky (2019), students whether or not liked the computer science course affected their attitudes toward participating in text-based programming activities. On the other hand, the majority of students in both groups did not feel anxiety about learning programming as they liked to use the computer. Owolabi et al. (2014) stated that computer proficiency reduced computer anxiety, and this might indirectly affect learning anxiety. In this case, it seems that this finding in our study was independent of the use of programming environments based on different programming approaches. Therefore, this is an interesting result that can direct new studies.

Although the significance of this course was explained to participants at the beginning of the implementation, few students in both groups thought that the lessons were boring, unnecessary, and unimportant. These students with no learning anxiety did not consider taking the course again, as they were not interested in this course. As Bubnó and Takács (2017) emphasized, eliminating this negative perception had a great role in achieving success in programming education. In addition, these students with low motivation had difficulties in the programming education process. Salleh et al. (2018) also stated that low motivation caused to increase the mental effort. According to the codes revealed in our study, the reason for this was the efforts of these students to pass this course instead of acquiring programming skills. Such a result for high school students was not available in previous studies, because the majority of these studies were conducted in engineering fields, not in high schools. For this reason, this interesting result in our study needs to be proven by new research.

In both groups, understanding the logic of coding, acquiring the programming skills, and learning to create software were achieved in the learning outcomes. These outcomes are among the target achievements determined by MoNE (2017). Thus, it can be said that this study has a positive effect on many students regardless of text-based and hybrid-based programming approaches. This result also shows that the instructor has well managed this implementation process, and has assigned the appropriate programming tasks to students considering their experiences. In addition, learning how to solve a problem step by step and understanding the programming errors were among the learning outcomes of students participating in text-based programming. These findings are evidence that learning a programming language improves problem-solving skills, and increases the desire to reach a solution for students (Lye & Koh, 2014). However, a student participating in hybrid-based programming claimed not to acquire any gain in this course. This result could stem from disregarding the course. Shih (2017) also determined that some university students performing the hybrid-based programming tasks exhibited low programming learning behavior.

In our study, a student participating in hybrid-based programming wanted to attend the future programming course again because of the enjoyable lessons. According to Seraj et al. (2019), students who performed hybrid-based programming tasks using the Blockly environment stated that this was a more interesting environment to improve their programming experience in the future. Considering these positive results, it is seen that hybrid-based programming which can be used for both textual and visual coding is a favorable approach for teaching any programming language. In both groups, the majority of the students emphasized the importance of knowing the computer science field and the necessity of programming education. In the study of Weintrop and Wilensky (2017), students participating in both block-based and text-based programming expressed similar statements. In their study, especially the students performing the text-based programming tasks expressed that such environments were important for professionalization in programming. On the other hand, even if the most of participants in our study were female, they expressed that programming education was important and necessary. However, Hsu and Hwang (2021) stated that female students had a low interest in programming. Although there is no analysis regarding gender in the focus of our study, this finding will strengthen further studies.

Conclusion and Implications

This study offers a detailed perspective in terms of comparing the experiences of a group participating in text-based programming and the other group participating in hybrid-based programming. Therefore, the results obtained from the interviews in this study provide a rich data source in terms of giving ideas to instructional designers, instructors, and researchers regarding programming language education. In this study, during the 10-week-long programming language training, the majority of students in both groups realized the necessity of learning a programming language. This implementation process is an indicator that this awareness takes a long period to become a culture. For this reason, it is recommended that MoNE include courses at each grade level that will improve programming skills. On the other hand, some of the high school students in this study may have participated in algorithm training by using the block-based programming environment while in secondary school. The possibility of this situation affecting the results of the hybrid-based group is a limitation of this study.

According to Bubnó and Takács (2017), students need to realize that they should not be afraid of computer programming, but rather, it is a process of mathematical problem solving through a machine. Therefore, it is necessary to offer various ways to them which can help to understand the programming languages. It is recommended to answer the needs of inexperienced/novice programmers that programming experts, instructional designers, and instructors, jointly develop the programming environments and conduct the programming language activities. At this point, hybrid-based programming environments can be used to increase students' beliefs about how to be able to succeed, motivation, and self-confidence to learn programming languages by having an enjoyable learning experience. In particular, as learners facing a programming language for the first time may feel anxiety, a hybrid-based programming approach that offers a variety of ease-of-use can be based to minimize this anxiety. In addition, students' interest can be increased in learning a programming language by performing more programming tasks and activities highlighting the necessity of this

course. Even if, these implementations, such as in our study, do not eliminate all programming difficulties, they can allow students to focus on logic and structures in programming rather than worrying about the basics of programming languages (Kelleher & Pausch, 2005).

Consequently, this study offers some practical implications for instructors and researchers.

It is recommended that programming language training spreads over long-period implementation in order to improve the students' programming experience and thus, overcome programming anxiety and increase their self-confidence.

In this study, individual programming activities were carried out. In future studies based on text-based or hybrid-based programming, assigning students collaborative programming tasks maybe prevent the classmate pressure. Thus, it can be compared with the results of this current study by investigating whether group activities cause positive changes in students' programming experiences.

As hybrid-based programming environments such as Blockly enable us to see both the real programming code structure and blocks at the same time it is likely to reduce complexity in programming education. Therefore, novice programmers may use to start learning a programming language.

In this study, Python editor was used in text-based programming and Blockly environment in hybrid-based programming for the Python programming language training. To verify the results of our study, it is recommended that more studies be conducted on programming language education just included in the high school curriculum.

Lisans Bilgileri

e-Kafkas Eğitim Araştırmaları Dergisi'nde yayınlanan eserler Creative Commons Atıf-Gayri Ticari 4.0 Uluslararası Lisansı ile lisanslanmıştır.

Copyrights

The works published in the e-Kafkas Journal of Educational Research are licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Ethical Approval

In this study, the authors declare that they have complied with the rules specified in the “Scientific Research and Publication Ethics Directive of Council of Higher Education” and have taken none of the actions specified in the “Contrary Actions to Scientific Research and Publication Ethics”. Furthermore all authors certify to have contributed to the study, and have participated sufficiently in this study to take public responsibility for the content, including participation in the concept, design, analysis, writing, or revision of the paper. They also declare that they have no conflict of interest, and the responsibility for any ethical violation belongs to the authors of this paper.

Ethics Committee Consent Information

Ethics Committee for the Implementation School: Ministry of National Education

Approval Date: May 15, 2018

Approval Number: 98057890-605.01-E.9496638

Ethics Committee of the University: Atatürk University Graduate School of Educational Sciences

Approval Date: May 23, 2018

Approval Number: 88179374-300-E.1800159222

Authors Contributions

A.Ü. carried out the implementation process, methodology, recourses, data collection, data analysis, validation, literature review, and original master's thesis writing. F.B.T. as a master's thesis supervisor managed the whole implementation process, methodology, recourses, data collection, data analysis, validation, literature review, and checked out the original master's thesis considering the IMRAD. A.Ü. wrote the first draft of the manuscript. F.B.T. as a corresponding author conducted the all further draft processes of manuscript.

References

- Adi, P. D. P., & Kitagawa, A. (2019, November). A review of the Blockly programming on M5Stack board and MQTT based for programming education. In *2019 IEEE 11th International Conference on Engineering Education (ICEED)* (pp. 102-107). IEEE. doi:10.1109/ICEED47294.2019.8994922
- Asai, S., Phuong, D. T. D., Harada, F., & Shimakawa, H. (2019). Predicting cognitive load in acquisition of programming abilities. *International Journal of Electrical & Computer Engineering*, *9*(4), 2088-8708. doi:10.11591/ijece.v9i4.pp3262-3271
- Bak, N., Chang, B. M., & Choi, K. (2020). Smart Block: A visual block language and its programming environment for IoT. *Journal of Computer Languages*, *60*, 100999. doi:10.1016/j.col.2020.100999
- Bartlett, L., & Vavrus, F. (2017). Comparative case studies: An innovative approach. *Nordic Journal of Comparative and International Education (NJCIE)*, *1*(1), 5-17. doi:10.7577/njcie.1929
- Baxter, P., & Jack, S. (2008). Qualitative case study methodology: Study design and implementation for novice researchers. *The Qualitative Report*, *13*(4), 544-559. Retrieved August, 10, 2021. <http://www.nova.edu/ssss/QR/QR13-4/baxter.pdf>
- Bosch, N., & D'Mello, S. (2017). The affective experience of novice computer programmers. *International Journal of Artificial Intelligence in Education*, *27*(1), 181-206. doi:10.1007/s40593-015-0069-5
- Bubnó, K., & Takács, V. L. (2017, September). *The mathability of word problems as initial computer programming exercises*. In *2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (pp. 39-44). IEEE. doi:10.1109/coginfocom.2017.8268213
- Chang, S. E. (2005). Computer anxiety and perception of task complexity in learning programming-related skills. *Computers in Human Behavior*, *21*(5), 713-728. doi:10.1016/j.chb.2004.02.021
- Chen, T. L., Chen, Y. R., Yu, M. S., & Lee, J. K. (2021). NNBlocks: A Blockly framework for AI computing. *The Journal of Supercomputing*, 1-31. doi:10.1007/s11227-021-03631-9
- Çakıroğlu, Ü., Çevik, İ., Köşeli, E., & Karaman, M. (2021). Understanding Students' Abstractions in Block-Based Programming Environments: A Performance based Evaluation. *Thinking Skills and Creativity*, *41*(2021), 100888. doi:10.1016/j.tsc.2021.100888
- Çakıroğlu, Ü., Suiçmez, S. S., Kurtoğlu, Y. B., Sari, A., Yildiz, S., & Öztürk, M. (2018). Exploring perceived cognitive load in learning programming via Scratch. *Research in Learning Technology*, *26*. 1-20. doi: 10.25304/rlt.v26.1888
- Debie, N., & Van De Leemput, C. (2014). What does germane load mean? An empirical contribution to the cognitive load theory. *Frontiers in Psychology*, *5*, 1099. doi:10.3389/fpsyg.2014.01099
- Demirer, V., & Sak, N. (2016). Programming education and new approaches around the world and in Turkey. *Journal of Theory and Practice in Education*, *12*(3), 521-546. Retrieved December, 16, 2019. <https://dergipark.org.tr/en/download/article-file/262355>
- Fraser, N. (2015). Ten things we've learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (pp. 49-50). IEEE. doi:10.1109/blocks.2015.7369000
- Garner, S. (2002). Reducing the cognitive load on novice programmers. In *Proceedings of ED-MEDIA World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 578-583). AACE. Retrieved December, 06, 2019. <https://files.eric.ed.gov/fulltext/ED477013.pdf>
- GitHub (2019). Retrieved November, 15, 2019. <https://octoverse.github.com/>
- Gomes, A., & Mendes, A. J. (2007, June). Learning to program-difficulties and solutions. In *International Conference on Engineering Education (ICEE)*. Retrieved November, 20, 2018. <http://icee2007.dei.uc.pt/proceedings/papers/411.pdf>
- Gomez, M. J., Moresi, M., & Benotti, L. (2019). Text-based programming in elementary school: A comparative study of programming abilities in children with and without block-based experience. In *Proceedings of the Conference on Innovation and Technology in Computer Science Education* (pp. 402-408). ACM. doi:10.1145/3304221.3319734
- Goodrick, D. (2014). Comparative case studies. Methodological briefs-impact evaluation no. 9 (No. innpub754). Retrieved August, 10, 2021. https://www.unicef-irc.org/publications/pdf/brief_9_comparativecasestudies_eng.pdf

- Gorson, J., & O'Rourke, E. (2020, August). Why do CS1 students think they're bad at programming? Investigating self-efficacy and self-assessments at three universities. In *Proceedings of the ACM Conference on International Computing Education Research* (pp. 170-181). ACM. doi:10.1145/3372782.3406273
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43. doi:10.3102/0013189X12463051
- Gülbahar, Y., & Kalelioğlu, F. (2018). Bilişim teknolojileri ve bilgisayar bilimi: Öğretim programı güncelleme süreci. *Millî Eğitim Dergisi*, 47(217), 5-23. Retrieved July, 27, 2021. <https://dergipark.org.tr/tr/download/article-file/539818>
- Hsu, T. C., & Hwang, G. J. (2021). Interaction of visual interface and academic levels with young students' anxiety, playfulness, and enjoyment in programming for robot control. *Universal Access in the Information Society*, 1-13. doi:10.1007/s10209-021-00821-3
- Ionescu, T. B. (2021). Leveraging graphical user interface automation for generic robot programming. *Robotics*, 10(1), 3. doi:10.3390/robotics10010003
- Jancheski, M. (2017). Improving teaching and learning computer programming in schools through educational software. *Olympiads in Informatics*, 11, 55-75. doi:10.15388/oi.2017.05
- Jung, K., Nguyen, V. T., & Lee, J. (2021). BlocklyXR: An interactive extended reality toolkit for digital storytelling. *Applied Sciences*, 11(3), 1073. doi:10.3390/app11031073
- Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2), 83-137. doi:10.21236/ada457911
- Li, X. (2016). Application of cognitive load theory in programming teaching. *Journal of Higher Education Theory and Practice*, 16(6), 57-65. Retrieved December, 01, 2017. http://www.m.www.na-businesspress.com/JHETP/LiX_Web16_6_.pdf
- López, J. M. S., Otero, R. B., & García-Cervigón, S. D. L. (2021). Introducing robotics and block programming in elementary education. *Revista Iberoamericana de Educación a Distancia (RIED)*, 24(1), 95-113. doi:10.5944/ried.24.1.27649
- Lutz, M. (2013). *Learning Python: Powerful object-oriented programming*. 5th ed. O'Reilly Media, Inc.
- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51-61. doi:10.1016/j.chb.2014.09.012
- Mavilidi, M. F., & Zhong, L. (2019). Exploring the development and research focus of cognitive load theory, as described by its founders: Interviewing John Sweller, Fred Paas, and Jeroen van Merriënboer. *Educational Psychology Review*, 1-10. doi:10.1007/s10648-019-09463-7
- Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation*. John Wiley & Sons.
- MoNE (2017). *Bilgisayar bilimi dersi öğretim programı*. Retrieved November, 30, 2017. <http://mufredat.meb.gov.tr/ProgramDetay.aspx?PID=374>
- MoNE (2018). *Bilgisayar bilimi dersi (kur 1-2) öğretim programı*. Retrieved January, 10, 2018. <http://mufredat.meb.gov.tr/ProgramDetay.aspx?PID=335>
- Moons, J., & De Backer, C. (2013). The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education*, 60(1), 368-384. doi: 10.1016/j.compedu.2012.08.009
- Moreno, R. (2010). Cognitive load theory: More food for thought. *Instructional Science*, 38(2), 135-141. doi:10.1007/s11251-009-9122-9
- Mumcu, H. Y., Mumcu, S., & Çakıroğlu, Ü. (2020). Use of Arithmetic Operation Skills in Block Based Programming Environments: A Comparative Case Study. *Journal of Computer and Education Research*, 8(16), 404-427. doi:10.18009/jcer.705822
- Owolabi, J., Olanipekun, P., & Iwerima, J. (2014). Mathematics ability and anxiety, computer and programming anxieties, age and gender as determinants of achievement in basic programming. *GSTF Journal on Computing (JoC)*, 3(4), 109-114. doi:10.7603/s40601-013-0047-4
- Rahaman, M. M., Mahfuj, E., Haque, M. M., Shekdar, R. S., & Islam, K. Z. (2020). Educational robot for learning programming through Blockly based mobile application. *Journal of*

- Technological Science & Engineering (JTSE)*, 1(2), 21-25. Retrieved August, 09. 2021. <https://rsepress.com/index.php/jtse/article/view/15/45>
- Rodríguez-Gil, L., García-Zubia, J., Orduña, P., Villar-Martinez, A., & López-De-Ipiña, D. (2019). New approach for conversational agent definition by non-programmers: A visual domain-specific language. *IEEE Access*, 7, 5262-5276. doi:10.1109/access.2018.2883500
- Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research*, 9, 147-171. Retrieved December, 01, 2017. <https://www.learntechlib.org/p/111361/>.
- Salleh, S. M., Shukur, Z., & Judi, H. M. (2018). Scaffolding model for efficient programming learning based on cognitive load theory. *International Journal of Pure and Applied Mathematics*, 118(7 Special Issue), 77-82. Retrieved February, 21, 2019. <https://acadpubl.eu/jsi/2018-118-7-9/articles/7/10.pdf>
- Sanner, M. F. (1999). Python: A programming language for software integration and development. *J Mol Graph Model*, 17(1), 57-61. Retrieved November, 18, 2019. https://www.academia.edu/1831560/Python_a_programming_language_for_software_integration_and_development?from=cover_page
- Sano, Y., & Kagawa, K. (2019). Design of a programming environment for non-procedural programming languages using Blockly. *The International Journal of E-Learning and Educational Technologies in the Digital Media (IJEETDM)*, 5(3), 93-101. doi:10.17781/p002614
- Seraj, M., Katterfeldt, E. S., Bub, K., Autexier, S., & Drechsler, R. (2019, November). Scratch and Google Blockly: How girls' programming skills and attitudes are influenced. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (pp. 1-10). ACM. doi:10.1145/3364510.3364515
- Shih, W. C. (2017, June). Mining learners' behavioral sequential patterns in a Blockly visual programming educational game. In *2017 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)* (pp. 1-2). IEEE. doi:10.1109/ICIMSA.2017.7985594
- Shin, S., Park, P., & Bae, Y. (2013). The effects of an information-technology gifted program on friendship using scratch programming language and clutter. *International Journal of Computer and Communication Engineering*, 2(3), 246. Retrieved November, 18, 2019. <http://www.ijcce.org/papers/181-J028.pdf>
- Stachel, J., Marghitu, D., Brahim, T. B., Sims, R., Reynolds, L., & Czelusniak, V. (2013). Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool. *Journal of Integrated Design and Process Science*, 17(1), 37-54. doi:10.3233/jid-2013-0004
- Stake, R. E. (2006). *Multiple case study analysis*. New York, NY: The Guilford Press
- Su, J. M., & Hsu, F. Y. (2017, July). *Building a visualized learning tool to facilitate the concept learning of object-oriented programming*. In *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)* (pp. 516-520). IEEE. doi:10.1109/IIAI-AAI.2017.180
- Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, 22(2), 123-138. doi:10.1007/s10648-010-9128-5
- Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education*, 120, 64-74. doi:10.1016/j.compedu.2018.01.011
- Tsai, C. Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior*. doi:10.1016/j.chb.2018.11.038
- Tuomi, P., Multisilta, J., Saarikoski, P., & Suominen, J. (2018). Coding skills as a success factor for a society. *Education and Information Technologies*, 23(1), 419-434. doi:10.1007/s10639-017-9611-4
- Vaidyanathan, S. (2013, December). *Opinion: We need coding in schools, but where are the teachers?* Retrieved December, 01, 2017. <https://www.edsurge.com/n/2013-12-09-opinion-we-need-coding-in-schools-but-where-are-the-teachers>

- Valsamakis, Y., Savidis, A., Agapakis, E., & Katsarakis, A. (2020, August). Collaborative Visual Programming Workspace for Blockly. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 1-6). IEEE. doi:10.1109/VL/HCC50065.2020.9127253
- Weintrop, D., Shepherd, D. C., Francis, P., & Franklin, D. (2017, October). *Blockly goes to work: Block-based programming for industrial robots*. In *2017 IEEE Blocks and Beyond Workshop (B&B)* (pp. 29-36). IEEE. doi:10.1109/BLOCKS.2017.8120406
- Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 29-36. doi:10.1145/3089799
- Weintrop, D., & Wilensky, U. (2018). How block-based, text-based, and hybrid block/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction*, *17*, 83-92. doi:10.1016/j.ijcci.2018.04.005
- Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education*, *142*(103646), 1-17. doi:10.1016/j.compedu.2019.103646
- Winterer, M., Salomon, C., Köberle, J., Ramler, R., & Schittengruber, M. (2020, September). An Expert Review on the Applicability of Blockly for Industrial Robot Programming. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 1231-1234). IEEE. doi:10.1109/ETFA46521.2020.9212036
- Yiğit, M.F. (2016). *Investigating the effect of instruction through visual programming environment on students' learning computer programming and attitudes toward programming* (Master's Thesis). Retrieved April, 15, 2019 from Council of Higher Education database in Turkey. (Thesis No. 442990). <https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp>
- Yin, R. K. (2003). *Case study research: Design and methods* (3rd ed.). Thousand Oaks, CA: Sage.
- Yukselturk, E., & Altioğ, S. (2017). An investigation of the effects of programming with Scratch on the preservice IT teachers' self-efficacy perceptions and attitudes towards computer programming. *British Journal of Educational Technology*, *48*(3), 789-801. doi:10.1111/bjet.12453