Article

Not peer-reviewed version

Effect of Block-based Python Programming Environment on Programming Learning

Yongcheon Kim , Jamee Kim , Wonkyu Lee *

Posted Date: 27 July 2023

doi: 10.20944/preprints202307.1849.v1

Keywords: block-based Python Programming; programming environment; programming learning

Article

Effect of Block-Based Python Programming Environment on Programming Learning

YongCheon Kim [1], JaMee Kim [2] and WonGyu Lee [3]*

[1] Department of Computer Science Education, Graduate School, Korea University, Seoul 02841, Korea; yongcheon.kim@inc.korea.ac.kr
[2] Major of Computer Science Education, Graduate School of Education, Korea University, Seoul 02841, Korea; celine@korea.ac.kr
[3] Department of Computer Science and Engineering, Graduate School, Korea University, 145 Anam-ro Seungbuk-gu, Seoul, South Korea
* Correspondence: lee@inc.korea.ac.kr; Tel.: +82-2-3290-2391.

Abstract: Advancements in computing technology have resulted in significant changes in education, healthcare, and manufacturing fields. Thus, personnel training in computer-related fields is directly related to national competitiveness. Therefore, the importance of programming education has been emphasized worldwide. Programming education has been conducted since the 1980s, however beginners often find programming tedious and difficult because of the cognitive burden of using text commands. Therefore, block-based programming environments, such as Scratch and Code.org, and beginner-oriented programming environments, such as Blockly and Pencil Code, have been developed. However, they have limitations when transitioning from block to text-based programming. In this study, we conducted one semester of classes for 128 middle school, high school, and university students to determine whether an environment that allows using a text-based programming language in a block-based programming environment aids beginners' understanding of programming instructions, command usage confidence, and programming usefulness. The results confirm that the usability of a block-based environment positively influences programming perception. This study is significant because it verifies the necessity and effectiveness of a block-based environment that employs a text-based programming language in programming education for beginners.

Keywords: block-based Python programming; programming environment; programming learning

1. Introduction

Beginning with the United Kingdom in 2013, the educational curricula of various countries, such as India, Korea, Japan, and Finland, have been revised to include software and artificial intelligence (AI) content [1, 2]. Given the direct correlation between software and AI technology development, personnel training, and national competitiveness, countries worldwide are emphasizing the importance of programming education. The status of software and AI in each country can be found on the global AI index provided by Tortoise Media [3].

The emphasis on the importance of programming education is not a recent phenomenon. As computers became common in schools in the 1980s, they were used for teaching text-based programming languages, such as Basic or Logo [4]. The requirement to learn intricate concepts and syntaxes in text-based programming has posed barriers to entry for beginners [5]. Complex instructions of programming languages, such as variables, iterations, arrays, and functions, primarily cause beginners to fail in programming [6].

Extensive research has been conducted on diverse learning methods and programming environments with the aim of effectively teaching programming in educational settings [7]. Error feedback is another factor that presents challenges for beginners in programming [8]. Therefore, Scratch, which allows learning through trial and error, has been used in all stages of beginner programming, regardless of the school grade level [9]. University students who have used Scratch have been successful in programming and expressed satisfaction with their work [10].

However, because block-based programming environments use different syntaxes than text-based programming languages commonly used in the industry, they are unsuitable for university students or those preparing for employment in programming-related fields [4, 11]. Researchers have developed hybrid programming environments, such as Blockly [12] and Pencil Code [13], that convert blocks into text commands when combined; however, they do not reduce the burden of learning text commands.

In order to lower the barriers for beginners to enter the field of programming, both the advantages of a block-based environment and the sense of accomplishment imparted by text-based programming must be provided. This is because the learning efficacy positively affects learners' motivation in the programming learning [14, 15].

This study focused on beginners with prior exposure to a text-based programming language in a block-based programming environment. The objective of the study was to investigate the impact of beginners' understanding of programming instructions and their confidence in effectively utilizing those instructions on their overall positive perceptions of programming. Additionally, the study aimed to determine how positive perceptions of programming at each grade level are affected by the usability-related factors of a block-based environment that supports learning text-based programming. It is suggested that the directions of other classes must be considered based on the grade level.

2. Related Work

2.1. Factor Analysis

When developing a programming environment and evaluate its usability, the validity of the evaluation tools must be ensured. In other words, it is necessary to verify that the tools developed to evaluate usability contain questions that are suitable for evaluation. Validity refers to whether the measurement target can be measured appropriately. Additionally, to evaluate two or more pieces of content, it is important to confirm whether the corresponding constructs can be used to evaluate what they are meant to evaluate. To confirm the validity of a construct, factor analysis can be used, whose purpose is to reveal the covariance structure of the data variables. It is used to create a single construct when one variable changes with another. The process for confirming the validity of the usability evaluation tool and performing factor analysis to create the constructs is as follows:

1) Create questions, perform a usability analysis, and obtain scores for each evaluation question.
2) Calculate a matrix of the correlation coefficients between questions.
3) Extract non-rotated factors.
4) Rotate the factors.
5) Interpret and assign names based on the content of questions with high factor loadings related to rotated factors.

Factor rotation is used to obtain a structure wherein the variables of each factor can be clearly interpreted. The goal is to consider factors that are not accurately explained through factor loading, which indicates the degree to which each variable reflects a single factor, and convert them into a simple structure. By rotating to a simple structure, each variable receives a high load from only one factor and relatively low loads from others, simplifying the factor structure. Thereafter, it can be interpreted as a factor structure that is not explained by the initial factor load. The following factor equation can be used to derive factors F_1, F_2, … , F_k which include the weight coefficients a_1, a_2, … , a_k of multiple variables in the data:

$$Z_j = a_{j1}F_1 + a_{j2}F_2 + \cdots + a_{jk}F_k + U_j$$

Z_j = Standard score of the j_{th} variable

a_{jk} = Weight (coefficient) for factor $k(F_k)$ of the j_{th} variable

U_j = Unique variance of the j_{th} variable

Factor analysis was used to extract constructs for measuring their effectiveness and satisfaction toward education and to ensure their validity. Specifically, 48 evaluation items were developed to determine the relationship between motivation and achievement based on students' degree of participation in an e-learning environment. Furthermore, six constructs (psychological motivation, peer collaboration, cognitive problem solving, interaction with instructors, community support, and learning management) were extracted to conduct the study [16]. To determine why beginner programmers have low coding skill levels, factor analysis was conducted on the answers to programming problems submitted by 614 university students through a web-based learning system, and four skill-level constructs were extracted: code style, syntactic, logical error-related, and syntax debugging [17]. Factor analysis is a method used to extract underlying constructs from a set of observed variables, categorizing confirmed constructs based on shared variance and extracting content commonly explained by multiple evaluation questions.

Thus, factor analysis ensures the validity of the developed tools and can be utilized to extract factors that commonly explain the questions within an examination tool.

2.2. Regression analysis

Regression analysis is a statistical technique that enables the prediction of values for dependent variables based on the values of independent variables. It accomplishes this by establishing linear equations that represent the relationships between the independent and dependent variables. In other words, it examines the extent to which dependent variables change based on the changes in independent variables, and estimates the predictive power of independent variables with regard to dependent variables. Simple regression analysis assumes linear relationships between independent and dependent variables and is expressed as follows:

$$Y' = B_0 + B_1X_i + \varepsilon_i$$

B_0 : When X_i = 0, the expected value of Y_i (regression constant, intercept)

B_1 : Population's regression coefficient (slope of regression line)

ε_i : Error that is not explained by X_i

In regression analysis, the least squares method is utilized to estimate the intercept (β_0) and regression (β_1) coefficients. These coefficients minimize differences between the actual values of the dependent variable (Y) and the predicted values (Y') obtained using the independent variables. The goal is to find the regression equation that minimizes the overall difference between the observed and predicted values of the dependent variable.

The total change in the Y value is classified into two parts: those that can and cannot be explained by the regression equation.

$$\begin{array}{ccccc} \sum(Y_i-\bar{Y})^2 & = & \sum(Y'_i-\bar{Y})^2 & + & \sum(Y_i-Y')^2 \\ SS_T & = & SS_R & + & SS_E \end{array}$$

SS_T : Total deviation sum of squares of Y

SS_R : Change that can be explained by the regression equation (regression deviation sum of squares)

SS_E : Change that cannot be explained by the regression equation (residual sum of squares)

An analysis of the variance table for simple regression can be explained as follows:

Table 1. Block-based programming environment analysis results.

	Sum of Squres(SS)	df	Mean Square(MS)	F	Z_j
Regression	SS_R	1	$SS_R/1$	MS_R/MS_E	SS_R/SS_T
Residual	SS_E	n-2	$SS_R/n-2$		
Total	SS_T	n-1			

One criterion for judging the suitability of a regression equation is the determination coefficient(R^2). It indicates the explanatory power of an independent variable for a dependent variable and refers to the ratio of the variance, obtained using the regression equation, to the total variance of the dependent variable. The closer the value of the determination coefficient is to 1, the greater the explanatory power of the independent variable.

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_E}{SS_T}$$

Simple regression analysis is used for analyzing the predictive power of an independent variable with regard to a dependent variable, whereas multiple regression analysis is a statistical method used to determine the variable that affects the dependent variable among several independent variables. The linear equation for the multiple regression model is as follows:

$$Y_i = B_0 + B_1 X_{1i} + B_2 X_{2i} + \cdots + \ + B_k X_{ki} + \varepsilon_i$$

where β_k is the unstandardized regression coefficient, i.e., the partial slope of the regression equation. It indicates the change in the Y value when the value of a certain independent variable X_k is increased by 1 while those of the other independent variables are fixed.

There are several methods for selecting the variables that must be included in the regression model to find the optimal regression equation, such as enter, forward selection, backward elimination, and stepwise selection [18]. Stepwise selection determines the optimal regression equation through an appropriate combination of adding and removing independent variables. As the variables are added individually, the significance of the variables already included in the model is reviewed, and those that are insignificant are excluded. This method is helpful for extracting significant variables.

Multiple regression analysis has been used to determine the factors that affect students' attitudes toward computer programming, which are a sense of achievement during the programming process, self-efficacy with regard to programming, and recognition learning [19]. Additionally, studies have been conducted to predict students' levels of academic achievement in the early stages [20]. In some cases, path diagrams have been used to visualize the extent to which two or more explanatory variables affect a dependent variable [21]. This is because path diagrams of the effects of three independent variables (X, U_1, and U_2) on a dependent variable (Y) can help users understand the relationships between them and interpret their significance

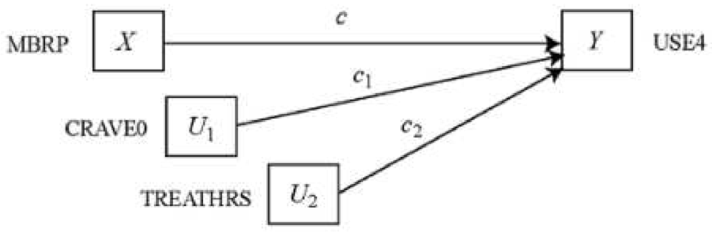
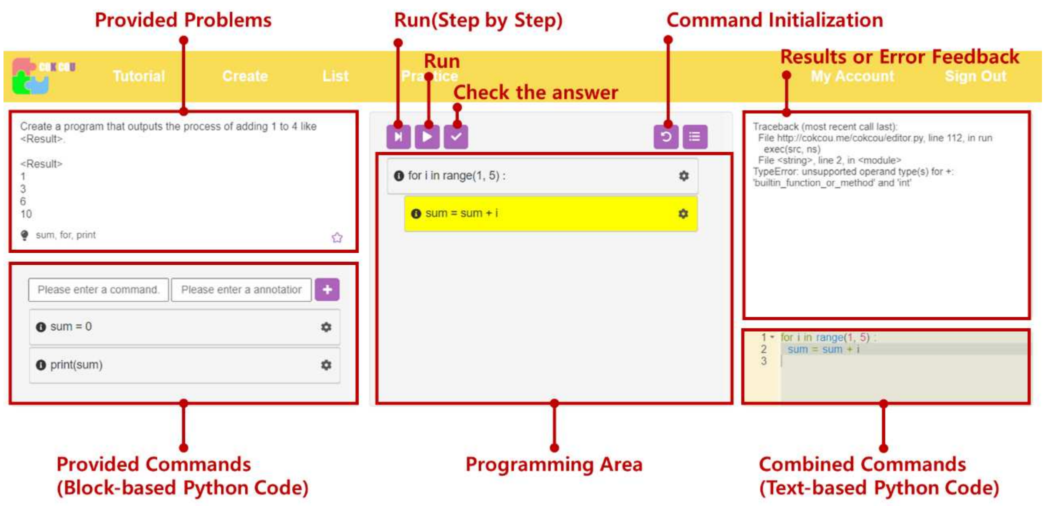


Figure 1. A path diagram of multiple linear regression.

3. Programming Environments

The programming environment used to collect the data consisted of the following components:



Create a program that outputs the process of adding 1 to 4 like <Result>.

```
<Result>
1
3
6
10
```

sum, for, print

```
sum = 0
print(sum)
```

```
for i in range(1, 5) :
    sum = sum + i
```

```
Traceback (most recent call last):
  File http://cokcou.me/cokcou/editor.py, line 112, in run
    exec(src, ns)
  File <string>, line 2, in <module>
TypeError: unsupported operand type(s) for +:
'builtin_function_or_method' and 'int'
```

```
for i in range(1, 5) :
    sum = sum + i

```

Figure 2. Programming Activity User Interface(UI).

First, "Provided Problems" area. Learners were provided with some problems and asked to develop programs based on their content. For example, if the problem was "Print the process of adding numbers 1~4," then "1, 3, 6, 10" must be printed.

Second, "Provided Commands" area. In some cases, all commands required to solve the problem were provided, and in others, too many or too few commands were provided. If too few commands were provided, the learner was required to use the "+" button to add commands directly. For the provided commands, a single-line Python program command constituted a single block. The learner could move commands by dragging and dropping them using the mouse instead of entering the text.

The third component was the "Programming" area, wherein the learners dragged and dropped commands from the provided command area to complete the program. Unnecessary commands could either be deleted or modified by clicking the Modify button (⚙). The Run (▶) button could be pressed to execute the combined commands and see the results. The Run One Step (⏯) button could be used to run a command one line at a time, obtain the results, and view the locations of commands where errors have occurred during the debugging process, which is useful. Clicking the Run or Run One Step buttons showed commands causing errors in yellow color. To initialize with the first command given, the Command Initialization(↻) button must be clicked.

The fourth component was an area for viewing "the execution results and error feedback". If there were no errors, the execution results of the combined commands were output. In case of errors, the programming environment outputs the locations of the commands where errors occurred and the causes of these errors.

Finally, the fifth component was an area to enter text-based Python commands. When block-based Python commands are combined in the programming area, the content of the "Combined command area" changes.

4. Methods

4.1. Participants

To determine the effects of the proposed programming environment on beginners' programming learning motivation, one semester of Python classes was conducted with 22 middle school, 34 high school, and 72 university students who had never used Python.

4.2. Programming Course

Additionally, to determine the effects of the proposed block-based programming environment on programming learning, we conducted classes, collected data, performed factor analysis, and then regression analysis, in that order.

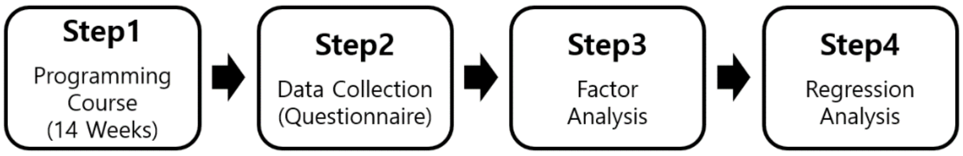

Step1
Programming Course (14 Weeks)
Step2
Data Collection (Questionnaire)
Step3
Factor Analysis
Step4
Regression Analysis

Figure 3. Procedure.

4.2.1. Procedure

The classes were held for 2 hours per week for 14 weeks, and the class content included output (print), input, operations (the four arithmetic operations, logical operations, and comparison operations), conditions (IF, ELIF, ELSE), iteration (for, while), lists, Python's built-in functions, user-defined functions, and individual projects.

4.2.2. Data Collection

Table 2 presents the content of the survey regarding the "usability of programming environment" and "perceptions of programming" answered by the 128 participants. The survey used a 5-point Likert scale, where 5 = "strongly agree" and 1 = "strongly disagree." The survey was conducted after the classes were finished.

Table 2. Questionnaire.

Category	No.	Item
Usability Of Programming Environment	A01	I understand commands
	A02	Commands are easy to use
	A03	I am confident in using commands
	A04	I have the knowledge and techniques required for using commands
	A05	I can obtain the desired results
	A06	It helps to understand "print( )
	A07	It helps to understand "input( )
	A08	It helps to understand" quadratic/comparative/logical operations
	A09	It helps to understand "if, elif, else
	A10	It helps to understand "for, while
	A11	It helps to understand "list
	A12	It helps to understand "function
	A13	It helps to understand "algorithm
	A14	The environment helps with my programming activities
	A15	I want to spend more time using the provided environment
	A16	I want to use the provided environment in the future
Perceptions Of Programming	B01	Programming helps create a better world
	B02	Programming is worth studying
	B03	Programming will be useful even after I graduate school
	B04	Programming is relevant to the environment, technology, and society
	B05	The programming class hours at school should be increased
	B06	Programmers think and make decisions rationally
	B07	I want to know more about programming

4.2.3. Factor Analysis

To determine the factors of the programming environment that affected beginners' "perceptions of programming" in this study, factor analysis was conducted through the following steps: data suitability assessment, factor extraction, and factor rotation. SPSS (version 26.0; IBM Corp., Armonk, NY, USA) for Windows was used for factor analysis.

First, to verify the suitability of the factor analysis model, the Kaiser–Meyer–Olkin (KMO) test and Bartlett's test of sphericity were performed on the survey content "degree of help with programming, " as presented in Table 3. The KMO value was 0.924, and Bartlett's test of sphericity significance probability was 0.000. Thus, the factor analysis model was suitable.

Table 3. Kaiser-Meyer-Olkin and Bartlett's Test of Sphericity.

Kaiser-Meyer-Olkin (KMO) Measure of Sampling Adequacy		0.924
Bartlett's Test of Sphericity	Approx. Chi-Square	3475.178
	df	253
	Sig.	.000

Second, the factors were extracted. Principal component analysis and exploratory factor analysis were used in this study. Table 4 presents the total explained variance. There were 23 components before extraction; however, after extraction and rotation, there were 4 components with eigenvalues greater than 1. Apparently, the four extracted factors accounted for 81.06% of the total variance and could be considered to comprise high explanatory power.

Table 4. Total Variance Explained.

	Initial Eigenvalues			Rotation Sums of Squared Loadings		
Component	Total	% of Variance	Cum. %	Total	% of Variance	Cum. %
1	13.727	59.684	59.684	6.496	28.243	28.243
2	2.223	9.664	69.348	5.772	25.095	53.338
3	1.684	7.324	76.671	3.546	15.418	68.756
4	1.010	4.389	81.060	2.830	12.305	81.060

Third, a factor rotation was performed. This study used an orthogonal rotation method based on varimax, which uses Kaiser normalization. Table 5 shows the rotated-component matrix. The following Table 5 presents the results of reducing factors using the rotated-component matrix.

Table 5. Rotated Factor Matrix.

Category	Sub-category	No.	Item	Factor			
Usability Of Programming Environment	Understanding of Programming Instructions	A11	It helps to understand "list"	0.847	0.263	0.138	0.285
		A08	It helps to understand" quadratic/comparative/logical operations"	0.845	0.203	0.258	0.314
		A09	It helps to understand "if, elif, else"	0.835	0.291	0.213	0.285
		A13	It helps to understand "algorithm"	0.818	0.227	0.189	0.276
		A10	It helps to understand "for, while"	0.792	0.338	0.093	0.306
		A07	It helps to understand "input( )"	0.731	0.290	0.237	0.356
		A12	It helps to understand "function"	0.702	0.465	0.109	0.243
		A06	It helps to understand "print( )"	0.627	0.438	0.248	0.352
	Usage Confidence	A03	I am confident in using commands.	0.352	0.822	0.002	0.236
		A05	I can obtain the desired results.	0.363	0.705	0.239	0.239
		A04	I have the knowledge and techniques required for using commands.	0.513	0.675	0.068	0.293
		A01	I understand commands.	0.467	0.626	0.208	0.335
		A02	Commands are easy to use.	0.232	0.594	0.531	0.204
	Usefulness	A16	I want to use the provided environment in the future.	0.190	0.112	0.920	0.153

		A15	I want to spend more time using the provided environment.	0.193	0.089	0.908	0.199
		A14	The environment helps with my programming activities.	0.552	0.237	0.564	0.301
Perception Of Programming	Positive Perceptions of Programming	B02	Programming is worth studying.	0.281	0.198	0.113	0.862
		B01	Programming helps create a better world.	0.211	0.238	0.119	0.846
		B04	Programming is relevant to the environment, technology, and society.	0.179	0.138	0.136	0.841
		B03	Programming will be useful even after I graduate school.	0.270	0.179	0.085	0.814
		B07	I want to know more about programming.	0.275	0.191	0.141	0.790
		B05	The programming class hours at school should be increased.	0.282	0.193	0.179	0.782
		B06	Programmers think and make decisions rationally.	0.312	0.153	0.215	0.679

The model settings were validated through dimensionality reduction. The following model was used in this study:

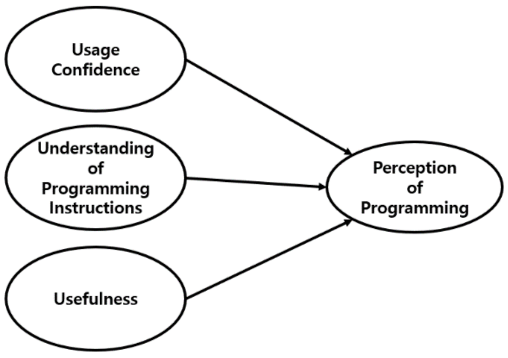

Usage Confidence
Understanding of Programming Instructions
Usefulness
Perception of Programming

Figure 4. Procedure.

The following Table 6 shows the averages and standard deviations for each grade level, according to this proposed model.

Table 6. Averages and Standard Deviations for Each Grade Level.

Factor	M(SD)			
	Middle School	High School	University	Total
Understanding of Programming Instructions	4.04(0.58)	4.35(0.74)	4.63(0.68)	4.46(0.71)
Usage Confidence	3.80(0.64)	4.28(0.75)	4.53(0.67)	4.34(0.73)
Usefulness	4.18(0.69)	4.34(0.70)	4.16(0.96)	4.21(0.86)
Positive Perceptions of Programming	4.58(0.43)	4.70(0.48)	4.72(0.57)	4.69(0.53)

4.2.4. Regression Analysis

A multiple regression analysis was performed to determine the effects of the "usability of the programming environment" on "perceptions of programming" at each grade level. Stepwise selection was used to input the independent variables into the analysis, and SPSS for Windows (version 26.0) was used for regression analysis.

5. Results

The results of analyzing the effect of the "usability of the programming environment" factor on "perceptions of programming" are as follows. The "usability of the programming environment" factor includes "usage confidence, understanding of programming instructions, and usefulness." The analysis of variance results showed that the usability factor had a statistically significant effect on programming perceptions (positive). The results of the detailed analysis of the factors that affected the programming perceptions are as follows:

Table 7. The analysis of variance summary of the correlation.

ANOVA					
Model	**Sum of Squares(SS)**	**df**	**Mean Square(MS)**	**F**	**p**
Regression	16.144	2	8.072	53.133	.000
Residual	18.990	125	0.152		
Total	35.134	127			

The analysis results showed that "understanding of programming instructions" had a statistically significant effect on "perceptions of programming (positive)" with a Beta value of 0.462 and a significance level of 0.5. The "usage confidence" factor was also statistically significant with a Beta value of 0.248 (p <.05). Therefore, it can be considered that using the programming environment and increasing the confidence made the programming perceptions more positive.

Table 8. The regression coefficient of the correlation.

Model	B	Std. Error	Beta	t	Sig
(Constant)	2.393	0.225		10.612	.000
Understanding of Programming Instructions	0.342	0.082	0.462	4.170	.000
Usage Confidence	0.178	0.080	0.248	2.233	.027

The B-value was used to predict the programming perceptions based on the usability of the programming environment. The following regression equation was derived using the B (unstandardized coefficient) value: of the three factors, the perceptions of programming value can be predicted based on "understanding of programming instructions" and "usage confidence."

(a) All Students
R = 2.393 + 0.342×(Understanding of Programming Instructions) + 0.178×(Usage Confidence)

(b) Middle School Student
R = 0.426×(Usefulness) + 2.804

(c) High School Student
R = 0.280×(Usefulness) + 0.251×(Understanding of Programming Instructions) + 2.395

(d) University Student
R = 0.375×(Understanding of Programming Instructions) + 0.277×(Usage Confidence) + 1.724

For example, out of all students (a), the positive perceptions toward programming of those who answered 5 for "understanding of programming instructions" and 3 for "usage confidence" was calculated as R = 2.393 + (0.342×5) + (0.178)×3, and their perception toward programming (positive) was 4.637.

The explanatory power of the "usability of the programming environment" factor with respect to "perceptions of programming" was 45.9%. Of this, "understanding of programming instructions" was 43.8% and "usage confidence" was 2.1%.

6. Discussions

This study analyzed the effect of the usability of a block-based environment on the positive perceptions of programming to support text-based programming learning. The analysis results showed that usability affects the positive perceptions of programming. Factors that influenced perceptions of programming varied according to school grade level. The results of this analysis are discussed below.

First, we focused on the usability aspect when conducting programming classes. Specifically, when the programming instruction understanding increased by one point, positivity toward programming increased by 0.342 points. This is consistent with the results of studies that have found that the cognitive burden caused by syntactical requirements decreases programming confidence during text programming classes [22, 23]. Therefore, to familiarize beginners with programming, a suitable programming environment that helps them satisfy the syntactic requirements without a cognitive burden is required, similar to that used in this study. The overall averages for "understanding of programming instructions" and "usage confidence" in this study environment were 4.46 and 4.34, respectively, as listed in Table 6, which confirmed that it helped learners use text commands without a cognitive burden. These results contradict the results of studies that have reported that it is difficult to transition programming languages when switching from block- to text-based programming. Additionally, they contradict the CSTA 2016 report, which states that teaching programming concepts using block-based programming has limitations [24-27]. Given that university students obtained a higher "understanding of programming instructions" (4.63) and "usage confidence"(4.53) than those of middle and high school students. In addition to the results of the studies [10] that have reported that success was achieved by using Scratch in university programming classes, it can be confirmed that the proposed environment helped university students learn text-based programming.

Second, the orientation of other classes according to school grade level must be considered. The results of analyzing factors that affect programming perceptions (positive) showed that "usefulness" was a factor for middle school students, "usefulness" and "understanding of programming instructions" were factors for high school students, and "understanding of programming instructions" and "usage confidence" were factors for university students. Therefore, there is a need to create a comfortable environment for middle and high school students to learn programming. This is consistent with the results of studies that have reported that block-based programming environments cultivate learners' interest and motivate them to continue programming because these environments have no errors [22, 28-30]. The lower the school grade level, the more sensitive the learners are to the programming environment. Based on this finding, perceptions of programming (positive) can be increased among middle school students.

For high school and university students, the "understanding of programming instructions" variable influenced "perceptions of programming (positive)". However, block-based programming environments, which are easy to use and do not cause problems such as errors, may be tedious for high school and university students and could lower their willingness to learn [31]. Therefore, interest in programming based on "understanding of programming instructions" must be increased as this factor is required to obtain appropriate programming results. Unlike middle and high school students, the "usage confidence" factor affected "perceptions of programming (positive)" among university students. Confidence in programming ability can strengthen positive perceptions among students. This is consistent with the results of studies that have reported that beginners lose confidence and interest when programming education is conducted solely in a text-based environment. However, they use commands confidently and are motivated to continue programming when text-based programming is performed in a block-based programming environment [22-23]. However, university students do programming either because it is a requirement of their course or to gain employment; therefore, text-based programming will help them in a practical manner [32]. Hence, block-based programming environments should offer practical assistance with text-based programming.

The results of this study showed that high school and university students should participate in phased programming classes using block-based programming environments instead of classes that employ only a text-based programming environment [9]. This assertion is supported by the results of studies [33] showing that students who learn programming through block-based programming tools achieve significantly better academic results than those who learn solely through conventional text-based programming tools because block-based visual programming tools provide students with an interesting experience.

7. Conclusions

Beginning with Scratch, introduced in 2006, block-based programming environments, such as Etoys and Code.org, have been widely available for beginners. These environments help introduce beginners to programming because they eliminate the possibility of errors, which is one of the main reasons beginners experience difficulties when using conventional text-based programming environments[22, 34, 35]. Studies have shown that using text-based programming languages after block-based programming environments can promote algorithmic thinking; however, other studies have found that switching to text-based programming environments results in problems [36]. Subsequently, hybrid environments such as Blockly and Pencil Code were introduced to help students transition from block-based programming environments to text-based ones; however, the students felt that the programming they were performing was not real [37].

This study analyzed whether the usability of a block-based text programming environment affects students' positivity toward programming. This environment used in this study allowed students to conduct Python programming in a block-based programming environment instead of conventional block-based, text-based, or hybrid programming environments [38], such as Pencil Code, which convert two types of commands. In other words, this study focused on programming language learning that can be used after graduating from college, and eliminating the errors that occur when beginners input commands, such as typing difficulties and syntax errors.

In future studies, environments that provide feedback and help in debugging to resolve errors must be investigated, which is one of the reasons beginners fail during the programming process. It is necessary to develop an environment that supports beginners in the process of programming to resolve problems independently and confidently by employing the programming instructions used in this study.

Author Contributions: Conceptualization, Y.-C.K. and W.-G.L.; methodology, J.-M.K. and Y.-C.K.; software, Y.-C.K.; validation, W.-G.L. and J.-M.K.; writing—original draft preparation, Y.-C.K.; writing—review and editing, Y.-C.K. and J.-M.K.; supervision, J.-M.K. and W.-G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT), grand number No. 2021R1A2C2013735.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

Wong, G. K.; Cheung, H. Y.; Ching, E. C.; Huen, J. M. School perceptions of coding education in K-12: A large scale quantitative study to inform innovative practices. *2015 IEEE international conference on teaching, Assessment, and learning for engineering* **2015**, 5-10.

Stephens, M. Embedding algorithmic thinking more clearly in the mathematics curriculum. *ICME 24 School mathematics curriculum reforms: challenges, changes and opportunities* **2018**.

Intelligence, T. The Global AI Index. Available online: https://www.tortoisemedia.com/intelligence/global-ai/ (accessed on 14 June 2023).

Moors, L.; Sheehan, R. Aiding the transition from novice to traditional programming environments. *Proceedings of the 2017 Conference on Interaction Design and Children* **2017**, 509-514.

Topalli, D.; Cagiltay, N. E. Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education* **2018**, *120*, 64-74.

Gomes, A.; Mendes, A. J. Learning to Program - Difficulties and Solutions. *International Conference on Engineering Education* 2007, 283-287.
Wing, J. M. Computational thinking. *Communications of the ACM* **2006**, *49*, 33-35.
McCall, D.; Kölling, M. A new look at novice programmer errors. *ACM Transactions on Computing Education* **2019**, *19*, 1-30.
Xinogalos, S.; Satratzemi, M.; Malliarakis, C. Microworlds, games, animations, mobile apps, puzzle editors and more: What is important for an introductory programming environment?. *Education and Information Technologies* **2017**, *22*, 145-176.
Cárdenas-Cobo, J.; Puris, A.; Novoa-Hernández, P.; Parra-Jiménez, Á.; Moreno-León, J.; Benavides, D. Using scratch to improve learning programming in college students: A positive experience from a non-weird country. *Electronics* 2021, *10*, 1180.
Emerson, A.; Rodríguez, F. J.; Mott, B.; Smith, A.; Min, W.; Boyer, K. E.; Lester, J. Predicting Early and Often: Predictive Student Modeling for Block-Based Programming Environments. *International Educational Data Mining Society* **2019**.
Seraj, M.; Katterfeldt, E. S.; Bub, K.; Autexier, S.; Drechsler, R. Scratch and Google Blockly: How girls' programming skills and attitudes are influenced. *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* **2019**, 1-10.
Deng, W.; Pi, Z.; Lei, W.; Zhou, Q.; Zhang, W. Pencil Code improves learners' computational thinking and computer learning attitude. *Computer Applications in Engineering Education* **2020**, *28*, 90-104.
Tavares, P. C.; Henriques, P. R.; Gomes, E. F. A Computer Platform to Increase Motivation in Programming Students-PEP. *CSEDU* **2017**, *1*, 284-291.
Yong, S. T.; Tiong, K. M. A Blended Learning Approach: Motivation and Difficulties in Learning Programming. *International Journal of Information and Communication Technology Education* **2022**, *18*, 1-16.
Lee, J.; Song, H. D.; Hong, A. J. Exploring factors, and indicators for measuring students' sustainable engagement in e-learning. Sustainability **2019**, *11*, 985.
Zhang, Y.; Paquette, L.; Pinto, J. D.; Fan, A. X. Utilizing programming traces to explore and model the dimensions of novices' code-writing skill. *Computer Applications in Engineering Education* **2023**.
Paisanwarakiat, R., Na-udom, A., & Rungrattanaubol, J. (2022, April). Combining Logistic Regression Analysis with Data Mining Techniques to Predict Diabetes. *Proceedings of the 18th International Conference on Computing and Information Technology* **2022**, 88-98
Gürer, M. D.; Cetin, I.; Top, E. Factors affecting students' attitudes toward computer programming. *Informatics in Education* **2019**.
Alboaneen, D.; Almelihi, M.; Alsubaie, R.; Alghamdi, R.; Alshehri, L.; Alharthi, R. Development of a web-based prediction system for students' academic performance. *Data* **2022**, *7*, 21.
Hayes, A. F.; Rockwood, N. J. Regression-based statistical mediation and moderation analysis in clinical research: Observations, recommendations, and implementation. *Behaviour research and therapy* **2017**, *98*, 39-57.
Hu, Y.; Chen, C. H.; Su, C. Y. Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis. *Journal of Educational Computing Research* **2021**, *58*, 1467-1493.
T. Jenkins. On the difficulty of learning to program. *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* **2002**, 53-58.
Noone, M.; Mooney, A. Visual and textual programming languages: a systematic review of the literature. *Journal of Computers in Education* 2018, *52*, 149-174.
Puente, E. L. Effect of the use of block-based languages in programming learning. *2022 International Symposium on Computers in Education* **2022**, 1-6.
Xu, Z.; Ritzhaupt, A. D.; Tian, F.; Umapathy, K. Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study. *Computer Science Education* **2019**, *29*, 177-204.
K-12 Computer Science Framework Steering Committee. *K-12 computer science framework*. ACM. 2016.
Papadakis, S. Evaluating the efficiency of two programming environments in shaping novices' attitudes, perceptions, beliefs and knowledge in programming: a comparison between Scratch and App Inventor. *International Journal of Teaching and Case Studies* **2019**, *10*, 31-52.
Weintrop, D.; Wilensky, U. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education* **2017**, *18*, 1-25.
Erol, O.; Çırak, N. S. The effect of a programming tool scratch on the problem-solving skills of middle school students. *Education and Information Technologies* **2022**, *27*, 4065-4086.
Cheung, J. C.; Ngai, G.; Chan, S. C.; Lau, W. W. Filling the gap in programming instruction: a text-enhanced graphical programming environment for junior high students. *ACM SIGCSE Bulletin* 2009, *41*, 276-280.
Mihci, C.; Ozdener Donmez, N. Teaching GUI-Programming Concepts to Prospective K12 ICT Teachers: MIT App Inventor as an Alternative to Text-Based Languages. *International Journal of Research in Education and Science* **2017**, 3, 543-559.

Xu, W. W.; Su, C. Y.; Hu, Y.; Chen, C. H. Exploring the effectiveness and moderators of augmented reality on science learning: A meta-analysis. *Journal of Science Education and Technology* **2022**, *31*, 621-637.

Pérez-Marín, D.; Hijón-Neira, R.; Bacelo, A.; Pizarro, C. Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?. *Computers in Human Behavior* **2020**, *105*, 105849.

Weintrop, D.; Wilensky, U. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education* **2017**, *18*, 1-25.

Blanchard, J.; Gardner-McCune, C.; Anthony, L. Bridging Educational Programming and Production Languages. *Paper for the" Every Child a Coder" workshop, ACM SIGCHI Conference on Interaction Design and Children (IDC 2015), Boston, MA* **2015**.

Powers, K.; Ecott, S.; Hirshfield, L. M. Through the looking glass: teaching CS0 with Alice. *Proceedings of the 38th SIGCSE technical symposium on Computer science education* **2007**, 213-217.

Alrubaye, H.; Ludi, S.; Mkaouer, M. W. Comparison of block-based and hybrid-based environments in transferring programming skills to text-based environments. *arXiv preprint arXiv:1906.03060* **2019**.