International Journal of Innovation, Creativity and Change. www.ijicc.net
Volume 9, Issue 4, 2019

Quantifying the Effects of Programming Education on Students' Knowledge Representation and Perception in Computational Thinking

Youngseok Lee[a], Jungwon Cho[b*], [a]Professor, KNU College of Liberal Arts and Sciences, Kangnam University, 40 Gangnam-ro Giheung-gu, Yongin-si, Gyeonggi-do, 16979, South Korea, [b]Professor, Department of Computer Education, Jeju National University, 102 Jejudaehakno, Jeju-si, Jeju-do, 63243, South Korea, Email: [a]yslee38@kangnam.ac.kr, [b*]jwcho@jejunu.ac.kr

Students should be able to express their knowledge in a manner that computers can understand. We aim to improve students' computational thinking (CT) and to express knowledge representation through programming education. We have developed a tool to measure students' CT. Tests were conducted using pattern analysis, conditional comparison, abstraction, automation, and algorithm design. Through Python programming education, we taught students' the knowledge expression needed to solve various problems, and then conducted a post-test. We analysed the correlations between the academic performance of students and their computer-related knowledge and expression skills. Even though students were unfamiliar with computer programming terminology and concepts, programming and computing education was administered based on problems that could be solved using elementary mathematics. There was no significant difference between the results of the initial students' assessments and the results after the lecture. However, the correlation between the students' assessments and actual academic performance was high. These studies provide a pilot model of how tools can be used to express and measure students' knowledge of CT. Based on these results, students can learn a variety of techniques to express their knowledge and continue to improve upon such.

Key words: *Knowledge Representation, Computational Thinking, Student's Perception, Correlation Analysis, Programming Knowledge.*

Introduction

To prepare for a future digital society, students must learn how to communicate and collaborate with machines and computers. To do this, students need to improve their computational thinking (CT) through programming education and learn how to express their ideas in a language that computers can understand (Lee and Cho, 2019). Among the various programming languages understood by computers, Python is most suited for beginners and is used widely in a variety of fields, such as big data and machine learning (Basogain et al., 2018; Atmatzidou and Demetriadis, 2016).

In order to express knowledge, one must be able to procedurally unravel their thoughts in detail. For example, when you draw a simple square, you are illustrating the shape of a picture that exists in your mind at a particular point in time. However, a computer or machine needs to be given specific directions and distances, for example: turn left after going straight. Complex programming techniques are not necessary for knowledge expression – only simple, logical thinking is required (Chen et al., 2017; Burbaitė et al., 2018; González, 2015). Therefore, knowledge representation only requires the concept of CT, as CT provides the method and process for solving problems through computing (Adeyinka and Bashorun, 2012; Durak, 2018).

Programming education is essential to improve knowledge representation through CT (Czerkawski and Lyman, 2015; Durak et al., 2019). Programming identifies a problem, finds a way to solve it, and then gradually repeats and combines this process to complete the work (Gao et al., 2016; Broks, 2016). This means that programming naturally improves information processing and utilization, procedural thinking, logical thinking, reasoning, and problem solving (Grover and Pea, 2013; Grover and Pea, 2018; Lee and Cho, 2017; Lee and Cho, 2018).

There are various ways to represent and measure knowledge. In computer science education, a study was conducted on tools that define the concept and form of programming knowledge and thought expression (Gürsoy, 2016; Duran, 2019). In addition to the validity of the questionnaire, major analyses were performed and correlations between concepts were analysed. This method for understanding patterns and presenting a combination of appropriate conditional statements, loops, and functions is similar to that used in this study. However, there is a lack of presentation and review of detailed concepts on CT and programming elements.

Studies have been conducted to specifically analyse the process used for applying programming skills and knowledge (Bergersen and Gustafsson, 2011). In addition to reliability analysis, we have presented and verified a research model that influences

programming knowledge. This model is necessary to understand the knowledge structure of professional developers, but it is also important to study and analyse the knowledge required to use computing for problem solving.

During the programming education of college students, a study was conducted on the relationship between a student's expected grade and their actual academic grade (Lee and Cho, 2019; Smith IV, 2019; Krpan et al., 2015). The study looked specifically at the correlation between programming knowledge, confidence, and success. Although this study did not address expressing one's own thoughts, the assessment of academic performance using one's own experience with success is useful as a comparative study.

Programming education based on CT requires an assessment of the prior knowledge levels of students to allow for the preparation and provision of appropriate educational content. There are many ways to assess student knowledge, such as through written tests.

In this paper, we presented different problems and proposed various ways to solve these problems. We also made suggestions on learning and measuring CT knowledge expression. These processes were then analysed in correlation with the students' academic performance.

Materials and Methods

Questionnaire on Knowledge Representation in CT

Various studies are being conducted to measure CT, programming knowledge, and knowledge expression. Based on these studies and our previous research (Lee and Cho, 2019; Grover and Pea, 2013; Lee and Cho, 2018), 20 items for measuring CT knowledge expression based were produced and used as tools to assess students' knowledge at the beginning of the semester. When producing the test paper, the factors to measure were divided into five stages: pattern analysis, condition comparison, abstraction, automation, and algorithm design.

- Pattern analysis:

♣ When John drinks three cups of milk, John's brother drinks one cup of milk. When Sara drinks two glasses of milk, Sara's brother drinks one glass of milk. When John and Sara drink six glasses of milk, how many cups of milk will John's brother and Sara's brother drink?

♣ Adam has six chickens. Each day the six chickens lay nine eggs. One day Adam's father bought two chickens and raised eight chickens. How many eggs will each of the eight chickens produce per day?

♣ What is the total number of five-character strings that can be created using five letters: three A's and two B's?

- Conditional comparison

♣ When the relations between Kiyoung, Subin, Minji, and Taeyong are as follows, what is the correct representation of each person's height with an inequality sign?
{Subin > Kiyoung, Taeyong > Minji, Minji < Subin, Minji < Kiyoung}

♣ There are two tunnels in the amusement park. When symbols line up through the black tunnel, they line up in reverse order. Likewise, when symbols line up through the white tunnel, only the first and last symbol change. Rules and question examples are shown in Figures 1 and 2.

Figure 1. Rules of conditional comparison

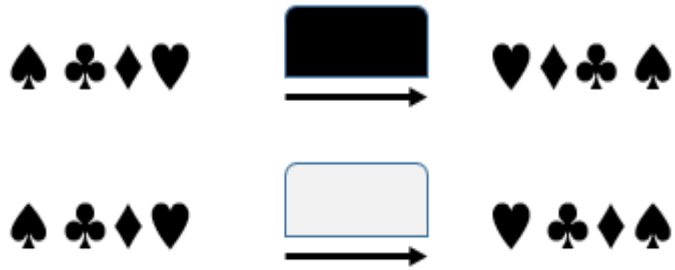


Figure 2. Example question of conditional comparison

- Abstraction

♣ Ants line up through a forest with pits. The pits are moderately deep, so some ants fall into them. If an ant does not fall into the pits and passes through the forest, the ant that was previously at the back of the line is now first. Rule and question examples are shown in Figures 3 and 4.

Figure 3. Rules of abstraction

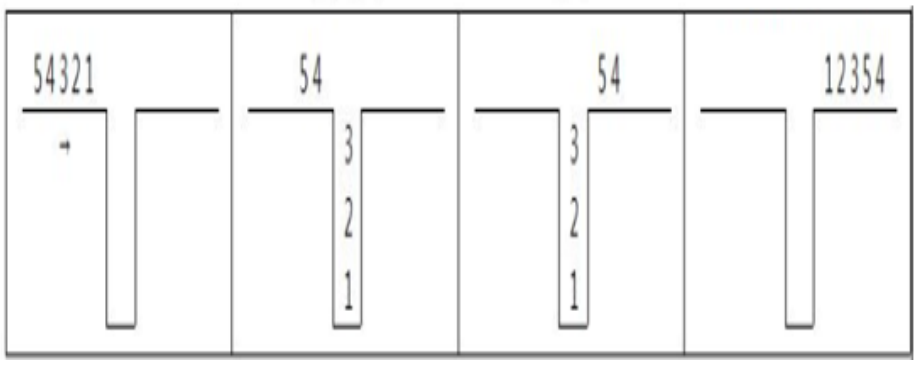
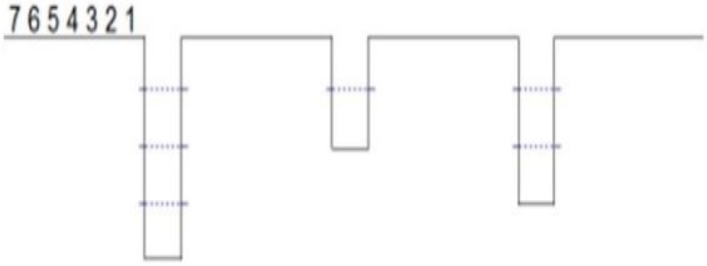


Figure 4. Example question of abstraction



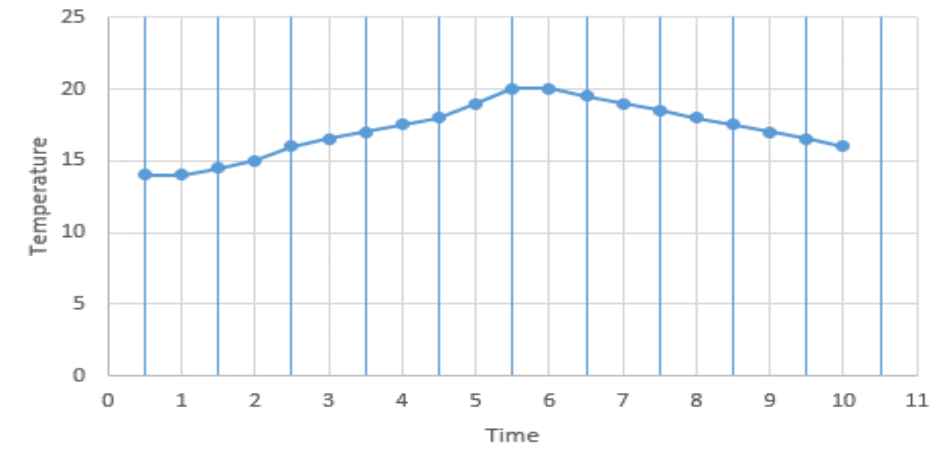
- Automation

♣ I have a car driven by a program. As shown in the following example, the car should be operated with the following program to reach the destination: go forward, turn left, turn right, go back.

♣ The following program was written to represent the temperature measured at the weather station. Which graph correctly shows the results of the program? {Repeat 20 times, 30 minutes rest, temperature display}

Example graph is shown in Figure 5.

Figure 5. Example graph of automation

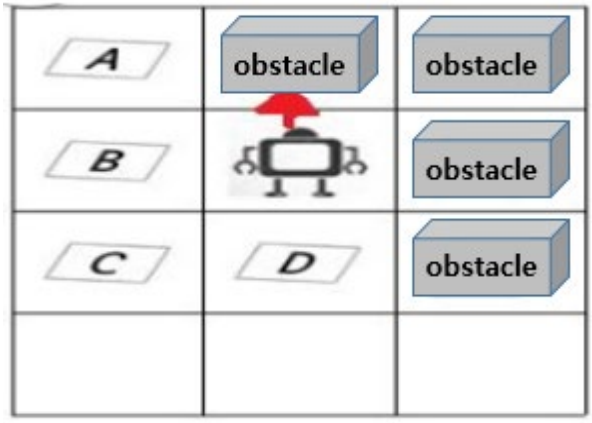


- Algorithm design

♣ There is a robot that can be programmed. The robot can move one square up, down, left, or right. The robot also has a sensor that detects obstacles. Where does the robot arrive when obstacles are in position and the program is executed in the following order? Example is shown in Figure 6.

A. Move one space to the right if there is an obstacle on the left
B. Move one space down if there is an obstacle to the top
C. Move one space to the left if there is an obstacle on the right
D. Move one space up if there is an obstacle at the bottom

Figure 6. Example of algorithm design



Research Methodology

Sample / Participants / Group

We surveyed 159 university students in Korea using questionnaires on CT and students' academic grades. The CT questionnaire that used a self-reporting method. A total of six classrooms were assessed, with 89.8% of 177 respondents completing the survey. A frequency analysis was performed to confirm the general characteristics of the study subjects. The descriptive statistics are presented in Table 1.

Table 1: Distribution of the research group according to class, gender, and year grade

		Classroom	N	%			N	%
Class	Non-science & Engineering	2	27	17.0	Gender	Female	104	65.4
		4	22	13.8		Male	55	34.6
		6	28	17.6	Year grade	1st	146	91.8
		11	27	17.0		2nd	4	2.5
	Science & engineering	12	31	19.5		3rd	6	3.8
		14	24	15.1		4th	3	1.9
Total			159	100				

The demographic distribution of the sample is as follows: of the 159 students surveyed, 55 or 34.6% were males and 104 or 65.4% were females. In terms of distribution by class, 91.8% or 146 students were in the first grade (or freshmen). In computer science education, the majority of students were in the first grade due to the nature of liberal arts. The science and engineering division was divided into two classes of 12 and 14. Diagnostic results were obtained showing that classes 4, 6, and 11 were composed of students from Global Studies, Music Studies, Korean Language, and Content Studies.

Instrument and Procedures

A CT test was developed for university students using their attitudes and satisfaction levels obtained from surveys. The CT questionnaire consisted of 20 questions. The CT test awarded one point for every correct answer and 0 points if an answer was incorrect. The Cronbach α coefficient was .928 for estimating the internal consistency and reliability of students' responses in the test.

The questionnaire was developed over the following steps: previous research was analysed, the questionnaire was created, validated, and administered online, and the results were analysed. The questionnaire was developed by modifying and supplementing questionnaires from previous research.

Python-based lectures were taught in the order of variables, data types, loops, conditional statements, and functions, and all lectures provided a variety of problematic situations as well as the techniques to solve them. We explained how to draw a turtle and used variables and loops to learn how to paint squares and connect them to make patterns.

The pre-test was conducted before the start of the course, and the post-test was conducted at the end of the course, using similar problematic scenarios. We then analysed the correlation between the students' academic grades and the results of their CT tests.

Results and Discussion

A CT test was administered at the beginning of the semester and a questionnaire relating to student satisfaction was conducted at the end of the semester. The questionnaire was analysed and processed using the SPSS Statistics 18.0 program, and the significance level was verified at $p<.05$. The results of factor analysis for the validity of questions are shown in Tables 2 and 3. KMO and Bartlett's test scores were high at .928 and the significance probability was .000, which ensured the validity of the question.

Table 2: KMO and Bartlett's Test

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.759
Bartlett's Test of Sphericity	Approx. Chi-Square	526.213
	df	190
	Sig.	.000

Table 3: An exploratory factor analysis of the computational thinking test

Item	Component				
	1	2	3	4	5
No01	-.054	.423	.486	-.248	.451
No02	.385	.529	-.050	-.151	.177
No03	.421	.157	-.303	.256	-.229
No04	.174	-.095	.716	.147	-.320
No05	.543	-.092	-.462	.137	.103
No06	.440	.261	-.091	.468	.227
No07	.227	.093	.372	.528	-.072
No08	.580	.166	.217	.215	-.148
No09	.496	.034	-.005	-.338	-.143
No10	.490	.094	-.043	-.340	.148

No11	.604	.176	-.100	.258	.028
No12	.563	.148	-.075	-.187	-.303
No13	.515	-.152	-.036	.051	-.115
No14	.384	-.436	-.029	.017	.413
No15	.392	-.190	.062	.112	.460
No16	.607	-.055	.078	-.238	-.273
No17	.528	-.558	.194	-.111	.014
No18	.447	-.102	-.002	-.194	-.201
No19	.347	-.261	.151	.043	.293
No20	.528	.228	.112	-.249	.140

Extraction Method : Principal Component Analysis.
Rotation Method : Varimax with Kaiser Normalization.

a. Rotation converged in 7 iterations.

To analyse the students' knowledge based on CT ability, the results of the post-tests were compared with the final academic performance grades, as shown in Table 4.

Table 4: Correlation between computational thinking test and actual academic performance.

		CTSUM1	CTSUM2	Score	Grade
CTSUM1	Pearson Correlation	1			
	Sig.(2-tailed)				
	N	159			
CTSUM2	Pearson Correlation	.732**	1		
	Sig.(2-tailed)	.000			
	N	158	158		
Score	Pearson Correlation	.485**	.504**	1	
	Sig.(2-tailed)	.000	.000		
	N	159	158	159	
Grade	Pearson Correlation	.488**	.550**	.923**	1
	Sig.(2-tailed)	.000	.000	.000	
	N	159	158	159	159
**. Correlation is significant at the 0.01 level (2-tailed).					

The correlation between the CTSUM1(pre-test) scores and Score was 0.485. The correlation with Grade was .488, and the correlation with the CTSUM2(post-test) scores was .732. CTSUM2 showed a very high correlation with Score, at .504, and Grade, at .550. According

to these results, the correlation between the students' CT test scores and the students' actual academic scores is highly related. At the beginning of the semester, after learning knowledge expression, both the pre-test and the post-test were found to be related to actual academic performance. Therefore, there was a correlation between students expressing knowledge and ideas through programming languages and solving various problems in their own way.

Conclusion

The development of artificial intelligence (AI) and robot technology is expected to radically change during our lifetime. Today's students will find it very natural to compete or collaborate with robots in the near future. To prepare for this, we must understand how and why the software that drives robots is made. To understand this, students must express their knowledge and organize their thoughts through programming education.

In this paper, we designed programming education in a problem-oriented form using Python, which is often provided to students by universities as an educational programming language. The students solved the problems by learning and utilizing solutions for the presented situations. In order to measure students' knowledge and CT ability, measurement tools were created and tested before and after administering the computer programming education. The correlation with the actual academic performance of the programming education was then analysed. There was a strong correlation between CT ability and students' academic performance. This finding is relevant for measuring students' knowledge expression techniques, knowledge creation, and CT by using actual students' programming experiences. Additionally, significant results were obtained from the pre-test, post-test, and academic achievement of students' knowledge expression.

Therefore, if we construct a problem-solving learning form suitable for programming education, analyse various cases, and share the results, we can proceed with desirable programming education and improve students' knowledge expression techniques. Although AI and robots will not replace humans right now, we will provide students with research on expressions of knowledge to foster their talents and actively utilize and cooperate with the principles of AI and robots. More practical research is needed on how to teach this effectively.

REFERENCES

Adeyinka, T. & Bashorun, M. (2012). Attitude of Undergraduate Students Towards Computer-Based Test (CBT). International Journal of Information and Communication Technology Education, 8:33-45. DOI:10.4018/jicte.2012040103.

Atmatzidou, S. & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. Robotics and Autonomous Systems, 75: 661-670.

Basogain, X., Olabe, M. Á., Olabe, J. C. & Rico, M. J. (2018). Computational Thinking in pre-university Blended Learning classrooms. Computers in Human Behavior, 80: 412-419.

Bergersen, G. R. & Gustafsson, J.-E. (2011). Programming skill, knowledge, and working memory among professional software developers from an investment theory perspective. Journal of individual Differences.

Broks, A. (2016). Systems theory of systems thinking: General and particular within modern science and technology education. Journal of Baltic Science Education, 15(4):408-410.

Burbaitė, R., Drąsutė, V. & Štuikys, V. (2018). Integration of computational thinking skills in STEM-driven computer science education. In 2018 IEEE Global Engineering Education Conference (EDUCON).1824-1832. IEEE.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X. & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. Computers & Education, 109: 162-175.

Czerkawski, B. C. & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. TechTrends, 59(2): 57-65.

Durak, H. Y. (2018). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. Technology, Knowledge and Learning:1-17.

Durak, H. Y., Yilmaz, F. G. K. & Yilmaz, R. (2019). Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. Contemporary Educational Technology, 10(2):173-197.

Duran, R. S. (2019). Towards a Common Instrument for Measuring Prior Programming Knowledge. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education. ACM. p. 443-449.

Gao, S., Guo, Y., Chen, J. & Li, L. (2016). Factors affecting the performance of knowledge collaboration in virtual team based on capital appreciation. Information Technology and Management, 17(2):119-131.

González, M. R. (2015). Computational thinking test: Design guidelines and content validation. In Proceedings of EDULEARN15 Conference: 2436-2444.

Grover, S. & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1):38-43.

Grover, S. & Pea, R. (2018). Computational Thinking: A competency whose time has come. Computer Science Education: Perspectives on teaching and learning in school. London: Bloomsbury Academic:19-37.

Gürsoy, D. (2016). Assessing novice programmers' performance in programming exams via computer-based test. Master's Thesis. University of Twente.

Krpan, D., Mladenović, S. & Rosić, M. (2015). Undergraduate programming courses, students' perception and success. Procedia-Social and Behavioral Sciences, 174: 3868-3872.

Lee, Y. & Cho, J. (2017). The influence of Python programming education for raising computational thinking. Int J u-and e-Serv Sci Technol, 10(8):59-72.

Lee, Y. & Cho, J. (2018). Factor Analysis of Computational Thinking for Software Education Based on Problem-Solving Learning. International Journal of Pure and Applied Mathematics, 120(6):4953-4967.

Lee, Y. & Cho, J. (2019). Knowledge representation for computational thinking using knowledge discovery computing. Information Technology and Management, 1-14. DOI: 10.1007/s10799-019-00299-9.

Smith IV, D. H. (2019). Quantifying the Effects of Prior Knowledge in Entry-Level Programming Courses. In: Proceedings of the ACM Conference on Global Computing Education. ACM, 2019. p. 30-36.