Three Propositions on the Education in Programming Languages

Euijin Kim
Elmer R. Smith College of Business and Technology
Morehead State University
Morehead, KY 40351

Abstract

In the age of Big Data and artificial intelligence, programming languages are important for college students because the knowledge of programming can help them understand the modern technologies and make better decisions about the technologies. Nonetheless, programming language courses are not popular among college students. To attract more students, several suggestions have been made such as computational thinking (Wing, 2006) or application of the concept with easy programming approaches(Lye & Koh, 2014). These approaches, however, overlook the nature of programming (e.g., programming is about understanding computers). In this paper, a straight approach with three propositions on the education in programming languages is suggested. It is also proposed that a student's awareness of the propositions is related to the student's successful completion of computer programs (i.e., coding).

Keywords: programming, programming languages, Bloom's revised taxonomy

1. Introduction

Modern information technologies have improved human life with convenient tools, but they also posed a threat to us. For instance, advanced artificial intelligence and robots have taken our jobs (Brynjolfsson & McAffe, 2014; Ford, 2015). This fact has scared many people because it could affect our lives. To deal with this threat, it is important to understand computers and how they work because the knowledge can help us understand the threatening technologies and thus deal with them more reasonably. In this paper, it is argued that studying programming languages is a good way to understand computers and their inner workings. The skills and knowledge of programming could also help students find better jobs in modern times. Programming skills are important for data scientists (Saltz & Stanton, 2018), andsome business schools recognized the importance of programming knowledge for their students (Weinberg, 2014)

Nonetheless, programming language courses are not popular among college students. Students seem to perceive that programming is difficult and is only for computer science students. To attract students, computer science educators proposed a conceptual framework such as computational thinking (Wing, 2006) and application of the framework with easy programming approach (Lye & Koh, 2014). These tactics proved to be useful in attracting students but could be an obstacle in the way of learning and mastering programming because programming is more than conceptual thinking. Programming is limited with technical constraints and different from conceptual thinking processes in a human brain.

In the following sections, the three propositions on the education in programming languages are explained.

2. Three Propositions on the Education of Programming Languages

Programming is about writing a series of statements (instructions). With high-level programming languages (e.g., structured, goal-oriented, or object-oriented), English-like statements with some symbols similar to those used in mathematics are written in a text file (i.e., source code file). Next, the source code file should be translated into a binary code file to be processed in a computer. Different types of programming languages implement the process differently. For instance, the compiler-based programming languages (e.g., C, C++, Swift, etc.) translate (or compile) a source code file at the development stage, which outputs an executable binary code file. The binary code file is distributed to users who run it on their computers (in this process, the computer should be the same type as the one the binary code was generated in). In an interpreter-based programming language (e.g., Python, Perl, JavaScript, etc.), the translation process happens at the user's end. In other words, the source code file is developed by the developer then delivered to a user who uses an interpreter to translate (i.e., interpret or parse) the source code file. A hybrid programming language (e.g., Java, C#, Kotlin, etc.) includes both translation processes. For instance, a Java developer creates a source code file and compiles it to generate an intermediary file (i.e., a bytecode file). The bytecode file is distributed to a user who uses a Java interpreter (i.e., Java Virtual Machine or Java Runtime Environment) to interpret it.Beyond these technical processes, there are more technical constraints of programming.

Among others, the following three propositions are so important that students should comprehend them to learn the main topics of programming and be able to complete working computer programs:

Proposition 1: Programming is about understanding how computers work. Programming is not just thinking about a series of instructions logically as is done in a human brain. When a series of numbers is to be arranged in ascending order, for instance, we can use our brain and write the numbers on paper to sort the numbers. When we program this algorithm in computer programming, we use variables, arrays, index numbers, and other programming constraints. Unless these constraints and how they work in a computer are understood, it is difficult to complete a program with the sorting algorithm.

Proposition 2: Programming is about writing text statements. In practice, easy programming (e.g., visual programming) or other convenient methods are frequently touted as a way of teaching students programming languages. To attract students, it may be necessary to use these methods because students usually don't like to type in many lines of code, but eventually they will have to write text statements to complete working programs. Even if easy-to-use tools are used (e.g., using mouse to drag a button icon and position it in a window),text statements should be written to make the button work as intended. Without a clear understanding of the text processing, students cannot complete computer programs.

Proposition 3: Programming is not mathematics. In programming, many terms and symbols are borrowed from mathematics. This adoption seems to have made students assume that the terms and symbols are used the same way as used in mathematics. For instance, the equals sign (=) is used in mathematics as an equality operator. So, in mathematics, a statement x = 5 is read as "x equals 5." The process in our brain tells us x becomes 5. In programming, this does not work the same way. For instance, in C++, a statement `x = 5` means that the variable x (which should have been declared as in `int x;`) is assigned an integer value 5. There is no becomingness in programming unlike in mathematics (as in a human brain). The term variable in programming is defined as "a named storage location in the computer's memory for holding a piece of information (Gaddis, 2019, p. 16)."On the other hand, in mathematics it is defined as "the letters that stand for numbers (Gustafson & Frisk, 2002, p. 2)." Unless these different uses in programming are understood, students are not likely to complete computer programs successfully.

Only when students are aware of these three propositions, they are likely to complete computer programs (i.e., coding) successfully. In other words, the degree of students' awareness of these propositions are likely to be related to their successful development of computer programs. The revised Bloom's Taxonomy(Anderson et al., 2001) also supports the relationships as shown in Figure 1.

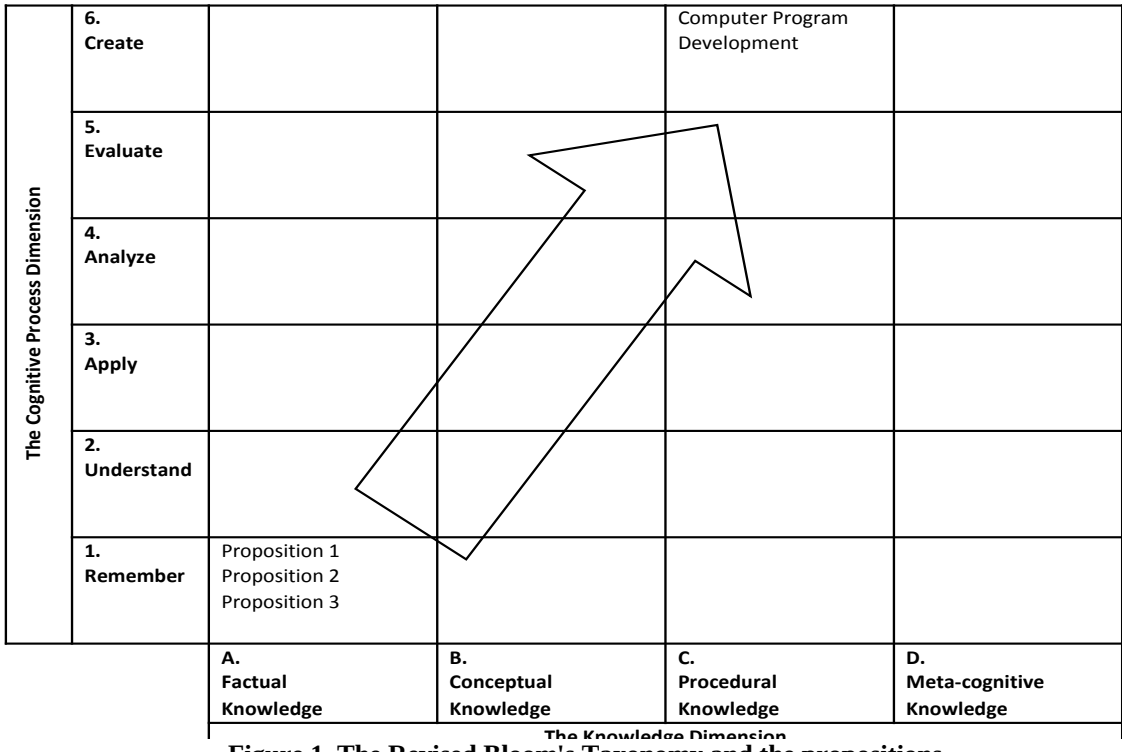

Figure 1. The Revised Bloom's Taxonomy and the propositions.

The revised Bloom's Taxonomy uses two dimensions (cognitive process and knowledge), the levels of which are assumed to be moving incrementally to the next level. This implies that a lower level (e.g., Remember in the cognitive process dimension) should be completed before moving to the next level (e.g., Understand). In the same vein, students should learn factual knowledge before moving to the next level (i.e., conceptual knowledge). Based on this logic, the following hypotheses are proposed (refer to Figure 2):

Hypothesis 1: The degree of a student's understanding on a computer's inner workings is positively related to the degree of a student's completion of a computer program.

Hypothesis 2: The degree of a student's awareness about the necessity of text statements in a computer program is positively related to the degree of a student's completion of a computer program.

Hypothesis 3: The degree of a student's awareness about the differences between programming terms and mathematical terms is positively related to the degree of a student's completion of a computer program.

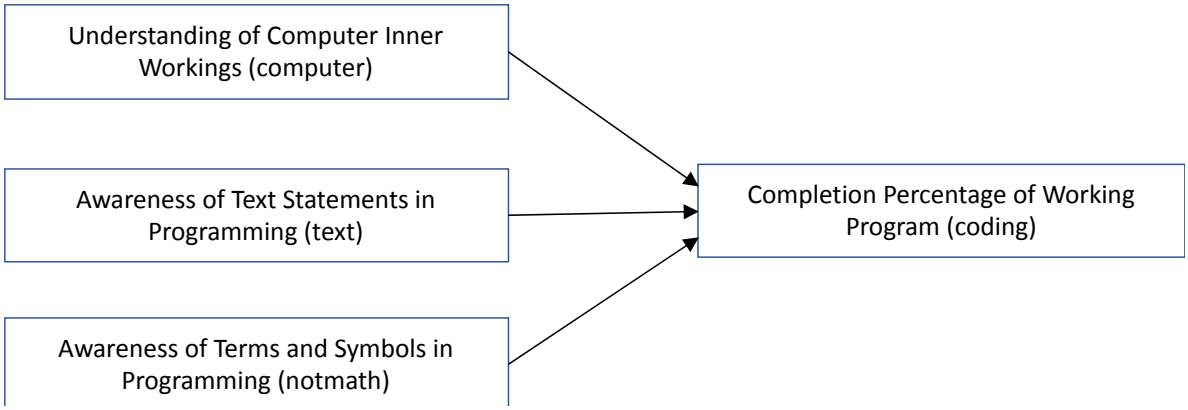

Figure 2. Research model with a dependent variable and three independent variables.

3. Research Method

Exam questions were used to measure the three propositions. Each category (i.e., a student's understanding of how computers work, a student's understanding of text data processing, and a student's understanding of the differences between programming and mathematics) included five questions, the scores of which were summed to generate a variable. This process produced three independent variables (i.e., computer, text, and notmath). The score of a student's final hands-on exam was used to measure his/her completion rate of computer program development (coding).

The questions for the independent variables were included in regular exams before which the propositions were emphasized in lectures. For online courses, video lectures were developed by the author and posted on an online course management system (e.g., Blackboard©) where students could watch them to learn the propositions and other topics of programming.

To test the hypotheses, multiple regression was used because the research model included a dependent variable (coding) with three independent variables (computer, text, and notmath).

4. Results

Eighty-six students completed the questions in a series of exams in an introductory programming language course with C++ (three sections between 2016 and 2018 were used). Seventy-three students were male and thirteen were female. Most students were college students (93%), seven percent were high school students, and four percent were traditional students.

The results of the multiple regression analysis indicated that the regression model was statistically significant, $F(3, 82) = 245.2$, $p < 0.05$. The three independent variables accounted for 89.97% of the variance of the dependent variable ($R^2 = 0.8997$, adjusted $R^2 = 0.896$). The p-values of the independent variables were also statistically significant ($p < 0.05$), indicating that the three independent variables (computer, text, notmath) were the predictors of the dependent variable (coding). Appendix lists the output of the statistical analysis.

The results implied that the three propositions were all related to students' ability to complete coding. In other words, (1) those who understood how computers work completed the coding exam more successfully,

(2) those who were aware of text typing in programming completed the coding exam more successfully, and (3) those who recognized the differences between programming terms and mathematical ones completed the coding exam more successfully.

5. Limitations

The participants of the research were the students who took introductory programming language courses, and they did not constitute a random sample. In addition, the sample size was not large enough to guarantee a robust statistical analysis. With these caveats, the results should be interpreted accordingly.

6. Conclusion

This paper suggests three propositions that should be emphasized when programming languages are taught. The three positions are (1) programming is about understanding how computers work, (2) programming is about writing text statements, and (3) programming is not mathematics. The statistical analysis provided provisional evidence that students who were aware of these propositions were likely to complete programs (coding) more successfully. The result, however, should be cautiously interpreted, and more robust studies are called for.

References

Anderson, L. W., Krathwohl Peter W Airasian, D. R., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., & Wittrock, M. C. (2001). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Blooms Taxonomy of Educational Objectives (Abridged). New York: Addison Wesley Longman, Inc.

Brynjolfsson, E., & McAfee, A. (2014). The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies. New York: W. W. Norton & Comany, Inc.

Ford, M. (2015). Rise of Robots: Technology and the Threat of a Jobless Future. New York: Basic Books.

Gaddis, T. (2019). Starting out with C++: From Control Structures through Objects, Brief Version (9th ed.). New York: Pearson.

Gustafson, R. D., & Frisk, P. D. (2002). Algebra for College Students (16th ed.). Pacific Grove, CA: Brooks/Cole.

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.

Saltz, J. S., & Stanton, J. M. (2018). An Introduction to Data Science. Thousand Oaks, CA: Sage.

Weinberg, C. (2014). B-schools finally acknowledge: Companies want MBAs who can code. Bloomberg Businessweek, 1.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

Appendix. Output of the Multiple Regression Analysis

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.94852835
R^2	0.89970603
Adjusted R^2	0.89603674
Standard Error	7.34345434
Observations	86

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	3	39668.01359	13222.6712	245.198834	7.8105E-41
Residual	82	4421.95838	53.9263217		
Total	85	44089.97197			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
Intercept	28.976504	2.032696482	14.2552045	6.6041E-24	24.9328231	33.0201848
computer	5.31149832	0.972527616	5.46153984	4.9337E-07	3.37683109	7.24616556
text	3.93280299	1.094152438	3.59438307	0.00055372	1.75618524	6.10942074
notmath	5.58776852	0.986281356	5.66549138	2.11E-07	3.62574071	7.54979632