The Effect of Memorization on the Retention and Learning Acquisition of Programming Practice

Hyesung Park
Georgia Gwinnett College

Kihyun Kim
Jacksonville State University

Cindy Robertson
Georgia Gwinnett College

Donghyun Kim
Kennesaw State University

To foster a skillful IT workforce, it is crucial to balance fundamental programming concepts and advanced coding skills that students can apply to complicated real-life problems. One of the biggest deterrents for students to succeed in a computing field is the struggle with learning new programming languages and how to code effectively. Based on Bloom's Taxonomy, this study develops a holistic view of the student learning process in computer programming and suggests the pedagogical insights to improve students' programming skills. This study shows the positive impact of memorization toward learning in fundamental programming class and its potential linkage to improved retention in computing programs.

INTRODUCTION

Increased demand for computing professionals in the field of computer science, information systems, information technology, and computer engineering spurs the advance of technology graduates in a varied type of industries (Dishman, 2016). According to the report from Burning Glass Technologies (2016), across five major job categories such as information technology specialists, data analysts, artists and designers, engineers, and scientists, there is an increasing demand for candidates who have coding skills. This is evidenced by the fact that there were seven million computing-related job openings in 2015. Programming jobs are one of the fastest-growing job categories, which is 50% faster than the overall job market and 12% faster than the market average. Despite ever-increasing requirements in the recent job market for applicants with strong programming skills, the attention from the younger generation toward the significance of these skills is largely deficient. For instance, in 2015, computer science accounted for only 2% of all high school AP programs. The retention rates in computing-related degree programs in most higher education institutions are not impressive and relatively high drop rates are frequently observed. Among the students who declare computing-related majors in their freshman and sophomore years, between 30 to 40% of these students either transfer schools or change their majors (Beaubouef & Mason, 2005).

It is well-known that programming skills are very challenging to improve and these skills are a significant barrier for many students in computing majors to complete their degree program. Unlike other knowledge sets and skills in other disciplines, programming is relatively new to most computing students since few students have been exposed to it before entering higher education institutions. Frequently, in an introductory class, there is a vast range of programming experience, ranging from those who have never programmed before to those that have more experience. Some students spend an extensive amount of time and make a significant effort, but still struggle to understand fundamental programming concepts and logical structures in introductory level programming classes. This struggle frustrates some students to the point that they give up at the very beginning stage of their computing program and eventually change their majors, leading to low retention rates in computing programs. The program drop rate between the freshman and sophomore years is as high as 30 to 40% in many U.S. higher education institutions (Dishman, 2016), and this becomes the norm for computing programs. So the question becomes, how can we reduce the drop rate and improve the retention rate in programming related courses? To answer this crucial question, this study investigated fundamental programming courses and applied the first three levels of Bloom's Taxonomy.

LITERATURE REVIEW

Bloom's Taxonomy

According to Bloom's Taxonomy, the learning stage can be decomposed into six levels from the lowest to highest-order of thinking (Michael & Coffman, 1956). The lowest level is remembering, which aims to see if students can retrieve relevant information from their long-term memory. Understanding, the next level, is where students can determine the meaning of instructional messages, including oral, written and graphic communication. After the understanding phase, in the applying level, students can carry out or use the procedure in a given task. Then in the next level, analyzing, students can break the material apart and detect how each part relates to each other and the overall structure. Evaluation is the next level where students can make a judgment based on criteria and standards. Finally, the highest order of the learning levels is creating. In this level, students can handle all of the complexity in the framework and can create a new form or an original product (Krathwohl, 2002).

TABLE 1
BLOOM'S TAXONOMY

Bloom's Level	Description
Remembering (lowest-order)	Students can retrieve relevant information from their long-term memory
Understanding	Students can determine the meaning of instructional messages, including oral, written and graphic communication
Applying	Students can carry out or use a procedure in a given situation
Analyzing	Students can break material into its constituent parts and detect how the parts relate to one another and an overall structure or purpose
Evaluating	Students can make a judgment based on criteria and standards
Creating (highest-order)	Students can put elements together to form a novel, coherent whole or make an original product

Originally Adapted from David Krathwohl (2002). A Revision of Bloom's Taxonomy.
Readopted from Bloom's Taxonomy, Yale Center for Teaching and Learning, https://ctl.yale.edu

Based on the fact that student retention rates in computing programs are low because students get frustrated when they have difficulty learning how to program, this study focuses on reshaping the classroom pedagogy of fundamental programming courses. Our goal is not to create an easy course, but to design a course which can effectively teach students how to start coding and improve relevant skills. It is very challenging to address all the issues referred to at the beginning of this paper. Therefore, this paper only focuses on the first three levels of Bloom's taxonomy in learning. The goal of this study is to design a course for students that provides the highest learning experience while minimizing the challenges associated with learning programming skills.

Bloom's Taxonomy model follows a sequential order of learning - it explains how learners handle a higher level of complexity and maturity of content. In order to reach the next level of learning in their programming classes, students need to master the lower level of thinking in the model. We think that the student's level of motivation is positively correlated with the student's learning process of computer programming. Motivation through memorization, which provides a positive reward to students if they can memorize the assigned program codes as homework, decreases the student's anxiety level and helps them to achieve the next level of learning of programming concepts and logical structures. This eventually leads to the next level in Bloom's Taxonomy.

Memorization

Learning how to program can be a very daunting task for students. When they take their first programming course, students tend to struggle because it is a new concept and they have no clear sense of where to start the learning process. Frequently, the frustration resulting from this struggle causes many students to give up on their programming class, and may even end up dropping out of school or changing their majors. This puts a lot of pressure on instructors in introductory programmings course to make the learning process as easy as possible and to continue to encourage and motivate their students. These instructors often have to come up with creative ways to teach these new skills to their students.

Bloom's taxonomy model states that memorization could help these students who are in the very early stage of the learning process. While Memorization is not all about learning, it is a basic but significant tool for the learning process. Memory-related strategies have been shown to relate to success in learning a second language. For example, this success was seen in a course designed for native-English speaking learners of foreign languages (Oxford & Ehrman,1995). Since learning a programming language is very much like learning a second spoken language, we believe these memory-related strategies could be used to enhance student learning in introductory programming classes. Many people think that memorization is an old-school skills-driven teaching method, but if you can use memorization as a motivational tool, then that old-school skills-driven teaching method can be turned into an accepted scholastic balanced learning tool (Orlin, 2013).

METHODOLOGY

In this study, we anticipate to answer the research questions below:

- *Does memorization help improve the levels of motivation to learn?*
- *How does the motivation level of students impact their learning achievement of computer programming?*

In this study, memorization was utilized in the following way. When students arrived in the classroom, they were given a small piece of code to memorize. After they felt like they had successfully memorized it, they had to write that section of code down on their papers. Then they handed their papers to the instructor and were given questions related to the memorized program. These questions were designed to guide the students to apply the algorithm they just memorized to another program. This memorization, recall and application process was designed to teach the students an algorithm that could be applied to many other problems, gradually giving them a deeper knowledge base. According to Bloom's lowest order of learning, remembering, we designed the materials for the students to memorize so that they had to memorize one small sample of code every class for two months.

After the memorization process, the goal is for students to be able to apply what they have learned to a new task in order to solve a problem. If this process occurs, students will have completed the first three levels of Bloom's Taxonomy - remembering, understanding, and applying. Bloom's higher learning levels were not manipulated in this study. We think to reach an advanced level of coding requires learning beyond remembering, understanding, and applying. Many studies provided that rote memorization interferes with students' creativeness and further learning progress. In fact, some researchers have found that using memory strategies on tests produced worse performance than those that used less memory strategies (Purpura, 1997). Some thought these negative results of memory strategies were due to the fact that memory strategies are often used for memorizing vocabulary and structures in the initial stages of learning a new language, but as their understanding of the language grew, learners did not need these memorization strategies as much (Oxford, 2003).

Voluntarily Engaged Students

Memorization in computing courses is fairly new to this young generation; therefore, it sometimes created high anxiety levels. Because of this, the memorization drills were not mandatory, just highly encouraged. However, most students participated in the memorization practices and positively responded to these exercises because they understood the importance of gaining a solid foundation in their programming skills for their future computing courses and their future career paths. These memorization techniques are based on the fact that if students feel success, they will be more likely to want to learn. Oxford and Shearin found that some of a student's motivation is based on their need for achievement, others are afraid of failure, and yet others are afraid of success. They found that language learning motivation will be high only if the students are expected to succeed and if the students believe that their success is worth the effort they put in (Oxford & Shearin, 1994). Thus, student motivation directs their behavior toward their goal of success in a programming class, and this motivation influences a student's persistence, effort, and energy put forth to learn something new (Ormrod, 2008). In our study, students were motivated to memorize the essential pieces of code because they learned that memorization made their future learning progress smoother and easier.

DATA ANALYSIS AND DISCUSSION

We collected data from two undergraduate fundamental programming classes at a mid-size state-funded school. Most students in the courses were freshmen and sophomores and had limited previous experience in computing and programming. Student exam score data from forty-four student subjects were collected to investigate the impact of memorization practices on student engagement and the level of achievement of basic programming concepts and advanced coding skills. Assessment of each task was composed of two student groups to investigate whether students with low memorization participation (group 1) and high memorization participation (group 2) differ in regard to the learning achievement of programming concepts through multiple choice assessment and coding skills by writing actual programs based on requested tasks. As shown in Table 2, there were 19 and 25 samples for groups 1 and 2, respectively. In addition, the students were given a survey to evaluate their perceptions of the memorization techniques.

TABLE 2
DESCRIPTIVE STATISTICS

Tasks	Groups	n	Minimum	Maximum	Mean	Std. Dev.
Programming Concept	Group 1 (low memorization participation)	19	51	94	75.95	12.380
	Group 2 (high memorization participation)	25	67	94	83.40	8.067
Programming Coding	Group 1 (low memorization participation)	19	0	100	48.32	32.385
	Group 2 (high memorization participation)	25	1	100	46.72	28.472

A one-way analysis of variance (ANOVA) was used to check the meaningful difference between the two memorization participation groups. Regarding programming concepts, there was a significant difference between group 1 and group 2 in exam scores (F = 4.495, *p* = .040), as shown in Table 3. The students that actively participated in the memorization activities (group 2) performed better than the students with low participation in the memorization activities (group 1) (μ2 = 83.40 vs. μ1 = 75.95). Regarding coding skills, however, there was no meaningful difference between the students that actively participated in the memorization activities (group 2) and those that did not (group 1) (F = 1.271, *p* = .266).

TABLE 3
ANOVA RESULTS

Tasks	Groups	Sum of Squares	*df*	Mean Square	F	Sig.
Programming Concept	Between Groups	475.674	1	475.674	4.495	.040*
	Within Groups	4444.872	42	105.830		
	Total	4920.545	43			
Programming Coding	Between Groups	1126.825	1	1126.825	1.271	.266
	Within Groups	37233.811	42	886.519		
	Total	38360.636	43			

* $p < .05$.

In summary, memorization significantly contributed to the learning acquisition of programming concepts. Students who participated in the memorization activities performed approximately 9.8 percent higher than those who did not in terms of understanding programming concepts. Those that participated in the memorization activities were more actively engaged with the course content, thereby increasing their understanding and reducing their frustration. The students indicated the same results in the student perception survey. This survey showed that the students found that memorization was helpful in understanding the programming concepts being taught. It allowed them to get a solid foundation in basic programming structures before having to apply those concepts later in the course. The students indicated that the memorization activities helped to alleviate the frustration associated with learning new programming concepts. Reducing student frustration is imperative in an introductory programming class because it is one of the leading causes for students to drop out of or fail these courses. With increased understanding and reduced frustration, students not only felt more successful, but they felt better equipped to apply their knowledge to coding problems.

While the students felt like the memorization activities helped them to solve the more complex coding problems, there was no significant relationship between memorization and improved coding skills. We believe that the reduction in frustration and increase in motivation gave the students increased confidence, but it was not enough to progress them to the next level of learning. Being able to apply critical thinking skills to solve a problem is not something that results from memorization practices.

This study showed that engaged students who were encouraged to memorize programming syntax in fundamental programming courses were better able to understand programming concepts and structures. However, the motivation through memorization was not enough to promote them to reach the next level of learning in Bloom's Taxonomy. As a result, we also looked at the scores of the students' standardized mathematics testing in high school, and found that those scores were highly correlated with students' programming coding skills ($r = 0.79$, $p = .000$). We believe that the critical thinking skills accumulated in the required mathematics courses in computing curriculums can lead students to the next level as we go through the step-by-step process in Bloom's framework. While this was not the focus of our study, it was an interesting result that warrants further research.

CONCLUSION

As computer systems become the critical technological infrastructure in core business environments, an increasing number of businesses are hiring talented technology professionals for not only their daily operation but as a source of competitive advantage. To cope with the increased industry demand for computing graduates, computing programs and professors at higher education institutions need to promote the discipline while maintaining the quality of their graduates. According to Bloom's Taxonomy, in order to foster a skillful IT workforce, it is crucial to balance conceptual and hands-on IT skills that can be applied to complicated real-life problems.

This study develops a holistic view of the student learning process in computer programming capability and investigates the pedagogical insights to improve students' learning acquisition of programming concepts. This study shows the impact of memorization toward learning in computing classrooms and its potential linkage to improved retention of students enrolled in computing programs. The insight provided by this study will allow instructors to design courses that achieve the highest learning experience across the various levels of learning phases while accommodating the challenges associated with learning programming skills.

REFERENCES

Beaubouef, T., & Mason, J. (2005). Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations. *The SIGCSE Bulletin*, 37(2), 103-106.

Burning Glass Technologies (2016). Beyond Point and Click: The Expanding Demand for Coding Skills Retrieved February 25, 2019, from https://www.burning-glass.com/wp-content/uploads/Beyond_Point_Click_final.pdf

Dishman, L. (2016). *Why Coding is Still The Most Import Job Skill of The Future,* Fast Company Retrieved February 25, 2019, from https://www.fastcompany.com/3060883/why-coding-is-the-job-skill-of-the-future-for-everyone

Krathwohl, D. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory into Practice,* 41(4), 212-218.

Michael, W., & Coffman, W. (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals.* Handbook I: Cognitive Domain.

Orlin, B. (2013). *When Memorization Gets in the Way of Learning: A teacher's quest to discourage his students from mindlessly reciting information.* Retrieved February 25, 2019, from https://www.theatlantic.com/education/archive/2013/09/when-memorization-gets-in-the-way-of-learning/279425/

Ormrod, J. (2008). *Excerpt from Educational Psychology Developing Learners.* 384-386. Retrieved February 25, 2019, from https://www.education.com/reference/article/motivation-affects-learning-behavior/

Oxford, R. (2003). Language Learning Styles and Strategies: An Overview. *GALA*, 1-25.

Oxford, R., & Ehrman, M. (1995). Adults' language learning strategies in an intensive foreign language program in the United States. *System*, 23, 359-386.

Oxford, R., & Shearin, J. (1994). Broadening the theoretical framework of language learning motivation. *Modern Language Journal,* 78, 12-28.

Purpura, J. (1997). An Analysis of the Relationships between Test Takers' Cognitive and Metacognitive Strategy Use and Second Language Test Performance. *Language Learning*, 42(2), 289-325.