Frontiers in Education Technology
Vol. 4, No.4, 2021
www.scholink.org/ojs/index.php/fet
ISSN 2576-1846 (Print) ISSN 2576-1854 (Online)

Original Paper

The Effects of Self-regulated Learning Strategies on Computer Programming Achievement in Teacher Education

Gary Cheng[1]

[1] Department of Mathematics and Information Technology, The Education University of Hong Kong, Tai Po, New Territories, Hong Kong

Received: October 24, 2021 Accepted: November 12, 2021 Online Published: November 23, 2021
doi:10.22158/fet.v4n4p37 URL: http://dx.doi.org/10.22158/fet.v4n4p37

Abstract

This study investigates the effects of student use of self-regulated learning (SRL) strategies on their computer programming achievement. Ninety-six students from undergraduate teacher training programmes offered by a Hong Kong university voluntarily participated in the study. Sixty-six of them were first-year students enrolling on an introductory Java programming course, while 30 were second-year students enrolling on an advanced Java programming course. The SRL strategies adopted by participants were measured by the Motivated Strategies for Learning Questionnaire (MSLQ) and were exemplified from the reflective writing of their electronic portfolios. Their achievement in computer programming was evaluated using continuous and end of course assessments. The findings of this study suggest that higher-order cognitive strategies (i.e. elaboration, organization, critical thinking), metacognitive control strategies (i.e. self-regulation) and resource management strategies (i.e. time and study environment management, help seeking) are likely to facilitate a prolonged achievement of computer programming for both novices and non-novices. They can provide insights into designing adequate SRL strategy training to support student learning in computer programming.

Keywords

self-regulated learning strategies, computer programming, electronic portfolio, teacher education

1. Introduction

Computer programming is generally regarded as the process of designing and executing a sequence of instructions for a computer to accomplish a specific task. It has been around in schools and universities for half a century. Papert (1980) pointed out that computer programming can equip students with the ability to decompose a problem into sub-problems, to plan and design solutions to the sub-problems, as well as to detect and debug errors while implementing the solutions. Aho (2012) noted that computer programming can engage students in the thought process of how to formulate problems in ways that can be tackled by computational steps and methods. Over the last decade, computer programming has gained renewed attention among educational researchers because it is considered a key element to implementing the computational thinking concepts that are helpful to students (Resnick et al., 2009; Lye & Koh, 2014). More importantly, it is believed that computer programming can foster the development of twenty-first century skills such as creativity, critical thinking and problem solving (Lye & Koh, 2014; Pardamean & Evelin, 2014; Scherer, 2016).

Nevertheless, computer programming remains something of a challenge for novices. Previous research has indicated that computer programming is one of the most difficult and challenging areas in computing education (Mcgettrick et al., 2005; Robin, 2019). This is evidenced by the high failure rates of introductory programming courses at university level, with nearly one-third of enrolled students failing to pass the courses (Bennedsen & Caspersen, 2007, 2019; Watson & Li, 2014). Research and analysis have been undertaken to gain understanding of potential causes behind the poor performance of novices in computer programming, with focuses on the settings of teaching and learning (Palumbo, 1990; Scherer, Siddiq, & Viveros, 2020; Watson & Li, 2014) and the characteristics of students (Saeed, Yang, & Sinnappan, 2009; Shaw, 2012; Zacharis, 2010). On the other hand, there has been evidence showing that learning strategy is the leading causal attribution to achievement outcomes on computer programming (Hawi, 2010). There has been therefore a call for more research into exploring student strategies for computer programming in higher education (Watson & Li, 2014).

Self-regulated learning (SRL) is a promising approach to facilitate student learning in computer programming. It refers to an "active, constructivist process whereby learners set goals for their learning and attempt to monitor, regulate and control their cognition, motivation and behaviour, guided and constrained by their goals and contextual features in the environment" (Pintrich, 2000, p. 453). There is a substantial body of evidence showing the importance and benefits of SRL strategies on student learning in some subject areas like language and science (Akyol, Sungur, & Tekkaya, 2010; Cheng & Chau, 2013; Sun & Wang; Zheng et al., 2020) but very limited in some other areas like computer programming. To fill this gap, this study aimed to explore the effects of student use of SRL strategies on their achievement in different types of programming assessment. It was also designed to identify whether there is difference in the use of SRL strategies between novices and non-novices. Given the growing importance of computer programming in this digital age, the findings of this study will be of great interest and relevant to researchers and practitioners in computer programming education.

2. Related research

2.1 The Importance of SRL Strategies for Academic Achievement

SRL refers to a constructivist learning process where students are actively engaged to acquire knowledge and skills through setting goals, choosing strategies, sustaining motivation, monitoring and assessing progress, making attributions about outcomes (Zimmerman, 2000). The cyclical nature of SRL can enable students to evaluate their prior efforts towards the learning goal they pursue, and to regulate their subsequent efforts to achieve a better learning outcome. Research has shown that SRL is not a fixed trait but can be constantly developed through the training and practice of relevant strategies (Bellhäuser, Lösch, Winter, & Schmitz, 2016; Teng & Zhang, 2020). Research has also shown that deployment of appropriate SRL strategies is significantly connected with knowledge acquisition and academic achievement in different subject areas like mathematics (Wang & Sperling, 2020), science (Zheng et al., 2020) and language (Sun & Wang, 2020).

According to Pintrich (1999), SRL strategies can be characterized as “strategies that students use to regulate their cognition (i.e., use of various cognitive and metacognitive strategies) as well as the use of resource management strategies that students use to control their learning” (p. 459). Three types of SRL strategies are associated with the regulation of student learning, including cognitive, metacognitive control, and resource management strategies. Students can adopt cognitive strategies to rehearse, organize, elaborate and critically think about the ideas and knowledge to be learned. They can also employ metacognitive control strategies to plan, monitor and regulate their own learning for the achievement of goals. Furthermore, they can use resource management strategies to manage and control resources directly related to their learning such as study environment, time, effort and external assistance (Pintrich, Smith, Garcia, & McKeachie, 1993).

Research has revealed that students can benefit academically from appropriate use of SRL strategies. This is especially obvious in the case of online learning where self-management strategies (e.g., time management, metacognition and effort regulation) are the key to success (Broadbent & Poon, 2015; Kizilcec, Pérez-Sanagustín, & Maldonado, 2017). The use of SRL strategies can also play a major role in differentiating high and low achieving students in different learning tasks. Performance on learning tasks for the acquisition of knowledge tended to relate more to cognitive strategies like elaboration (Greene, Yu, & Copeland, 2014), but that for the understanding of knowledge tended to rely more on metacognitive strategies like monitoring (Greene, Copeland, Deekens, & Yu, 2018). Therefore, provision of adequate support in SRL strategy deployment for academically at-risk students would possibly help them to improve their academic achievement (Winters, Greene, & Costich, 2008; Zimmerman, 2001).

2.2 Research into SRL Strategies and Computer Programming

Computer programming is widely known as a difficult and challenging skill to learn and master for novices (Qian & Lehman, 2017; Robin, 2019). It was consistently reported that approximately one-third of students failed to pass their introductory programming courses in higher education worldwide (Bennedsen & Caspersen, 2007, 2019; Watson & Li, 2014). Despite the work done by some previous studies on the possible factors influencing learning success in computer programming such as the internal characteristics of students (e.g. Cheng, 2019; Seyal, Mey, Matusin, Siau, & Rahman, 2015), more research is still needed to better understand what strategies and behaviours should be adopted by students with regard to the study of computer programming (Watson & Li, 2014).

Some research has been undertaken to examine the SRL strategies used in different learning phases and environments. For example, Pedrosa, Cravino, Morgado and Barreira (2016) analyzed 401 individual weekly reflective entries collected from 97 undergraduate students for 10 weeks to identify their use of SRL strategies in different learning phases. They found that students were more inclined to use organizing and planning strategies in early phases and then shifted to strategies about the application and transformation of theoretical knowledge into hands-on programming in later phases. Çakıroğlu and Öztürk (2017) examined how students develop their SRL strategies for computer programming in a flipped classroom with problem-based activities. Data of SRL strategies were gathered from 30 undergraduates through an observation form, interviews and online learning logs. They found that students adopted task strategies, monitoring and help seeking more frequently in the face-to-face settings than in the home sessions, suggesting that students tended to take an active role in learning programming and seek help from their teacher and peers in the classroom environment.

Some other research has concerned with the impact of SRL strategies on computer programming. For example, Bergin, Reilly and Traynor (2005) conducted a study on 35 undergraduates to examine the relationship between their SRL strategies and programming performance. They found that students who adopted more metacognitive and resource management strategies could perform better in introductory programming, but cognitive strategies like rehearsal, elaboration and organization did not seem to relate to programming performance. Cigdem (2015) carried out a similar study on 267 students from a military vocational college, but no significant correlation between their SRL strategies and programming achievement was identified. Likewise, Song, Hong and Oh (2021) explored the relationship between the SRL strategies and programming performance of 105 undergraduates and reported no significant correlation.

Despite the documented benefits of SRL strategies for academic achievement in some subject areas (e.g. Sun & Wang, 2020; Wang & Sperling, 2020; Zheng et al., 2020), there have been limited studies and inconclusive findings about the impact of SRL strategies on learning achievement in computer programming. To date little has been known about which SRL strategies are effective for students to accomplish different types of programming assessment. Further investigation is thus warranted to shed light on this research issue.

3. Purpose of the Study

This study is part of a funded project, namely "Tracking students' use of self-regulated learning strategies for computer programming in teacher education". The purpose of the study was twofold: 1) to collect and analyze student reflection on their use of SRL strategies specific to computer programming in teacher education through electronic portfolio (ePortfolio) construction; and 2) to explore the relationship between the use of SRL strategies and the achievement in computer programming. Specifically, this study was guided by the following key research question: which SRL strategies correlate positively with computer programming achievement for novices and non-novices?

This research question hypothesized that some SRL strategies would lead to better learning outcomes of students in computer programming while others would not. In this study, the hypothesis was evaluated by measuring the correlation between students' SRL strategy use and their achievement in computer programming. It was also examined by identifying the differences in the use of SRL strategies between high and low achieving students at both novice and non-novice levels.

4. Research Method

4.1 Participants

A total of 66 first year university students (28 females and 38 males) and 30 second year university students (10 females and 20 males) gave their written consent to participate in this study. The first-year students were enrolled on an introductory Java programming course, while the second-year students were enrolled on an advanced Java programming course with the prerequisite of passing the introductory Java programming course. Both courses spanned over 13 weeks with 3 hours each week and they were offered by a Hong Kong university in the academic year 2018/19. All participants studied the Bachelor of Education (Honours) programme in primary or secondary education with specialization in Mathematics or Information Technology. The ages of the first-year participants were between 17 and 22 years old (M=20.1, SD=1.86) and those of the second-year participants were between 19 and 24 years old (M=20.6, SD=1.35). At the outset of the study, all first-year participants reported that they had no prior experience in computer programming.

4.2 Measures

4.2.1 Measurement of Use of SRL Strategies

The Motivated Strategies for Learning Questionnaire (MSLQ) was used to evaluate participants' use of SRL strategies (Pintrich, Smith, Garcia, & McKeechie, 1991). The MSLQ is an 81-item, self-report instrument made up of two sections, comprising motivational orientations and use of learning strategies. The motivation section contains 31 items assessing students' goals and value beliefs, their beliefs about their skills to succeed, as well as their test anxiety in a course. The learning strategies section includes 50 items concerning students' use of cognitive, metacognitive control and resources management strategies. Since its development in early 1990s, the MSLQ has been widely adopted to evaluate students' motivation (Chen, Wang, & Lin, 2015; Sung, Hwang, & Yen, 2015; Yilmaz, 2017) and use of

SRL strategies (Alario-Hoyos, Estévez-Ayres, Pérez-Sanagustín, Delgado Kloos, & Fernández-Panadero, 2017; Broadbent, 2017; Cheng & Chau, 2013; Song et al., 2021) in a wide range of educational settings.

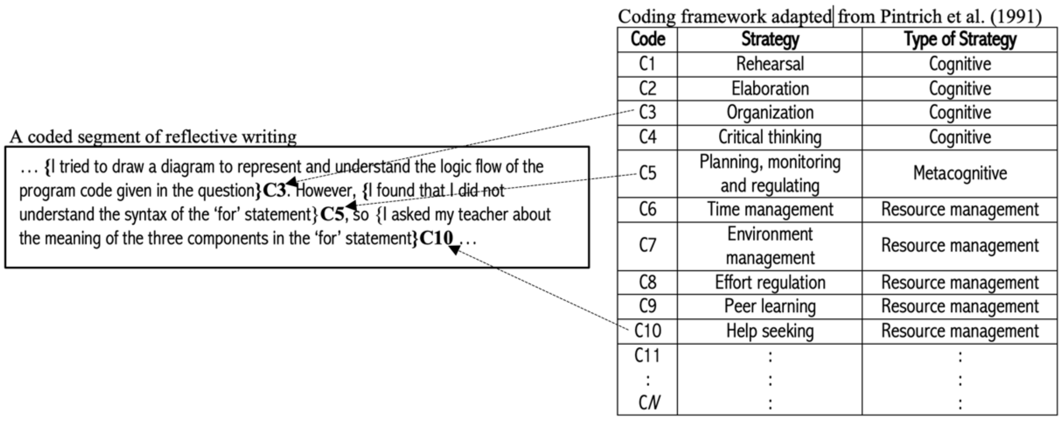
In this study, participants rated themselves on each item of the MSLQ using a seven-point Likert scale ranging from 1 (not at all true of me) to 7 (very true of me). Since this study was concerned with students' SRL strategies but not their motivational orientations, only the learning strategies section was used to examine their use of cognitive and metacognitive control strategies (rehearsal, elaboration, organization, critical thinking, and metacognitive self-regulation), as well as resource management strategies (time and study environment, effort regulation, peer learning, and help seeking). Table 1 presents details of the learning strategies section.

Table 1. Learning Strategies Section in MSLQ

Scale	Subscales	No. of Items
Cognitive Strategies	Rehearsal	4
	Elaboration	6
	Organization	4
	Critical Thinking	5
Metacognitive Control Strategies	Metacognitive Self-regulation	12
Resource Management Strategies	Time & Study Environment Management	8
	Effort Regulation	4
	Peer Learning	3
	Help Seeking	4

4.2.2 Identification of SRL Strategies Specific to Computer Programming

Student reflection on their strategies to learn computer programming was retrieved from their ePortfolio for analysis. Using open coding based on grounded theory (Charmaz, 2006), student use of SRL strategies specific to computer programming was identified. The SRL strategies for computer programming was initially based on the previous work of a general SRL strategy model (Pintrich et al., 1991). During the open coding process, a set of SRL strategies specific to computer programming and their examples can be built from student reflection. Two experienced researchers from the research team were involved in the coding process and they independently analyzed student reflection at sentence level. Discrepancies in the identification results between the two researchers were discussed to reach consensus on the method of analysis. A sample coded segment of student reflection is illustrated in Figure 1.



A coded segment of reflective writing

… {I tried to draw a diagram to represent and understand the logic flow of the program code given in the question}**C3**. However, {I found that I did not understand the syntax of the 'for' statement}**C5**, so {I asked my teacher about the meaning of the three components in the 'for' statement}**C10** …

Coding framework adapted from Pintrich et al. (1991)

Code	Strategy	Type of Strategy
C1	Rehearsal	Cognitive
C2	Elaboration	Cognitive
C3	Organization	Cognitive
C4	Critical thinking	Cognitive
C5	Planning, monitoring and regulating	Metacognitive
C6	Time management	Resource management
C7	Environment management	Resource management
C8	Effort regulation	Resource management
C9	Peer learning	Resource management
C10	Help seeking	Resource management
C11 : C*N*	: : :	: : :

Figure 1. A Sample Coded Segment of Student Reflection

4.2.3 Measurement of Computer Programming Achievement

Two types of individual assessment were applied to the introductory and advanced Java programming courses to measure student achievement in computer programming. The first was continuous assessment (CA, constituting 70% of the course grade) where students were asked to complete practical hands-on class exercises throughout the course. The second was end of course assessment (EA, constituting 30% of the course grade) where students were required to take a final written examination. Both CA and EA were designed and marked by an experienced teacher in computer programming.

4.3 Procedure

One main objective of this study was to collect and analyze student reflection on the use of SRL strategies specific to computer programming through ePortfolio practice. To develop an ePortfolio, students were asked to create a tri-weekly showcase to document three important components of SRL on their preferred online platforms (e.g. Mahara, Google Sites and Microsoft Sway). The three components included goal setting (i.e. listing specific learning goals in computer programming), artifact showcasing (i.e. selecting programming artifacts to demonstrate the progress or achievement towards the goals) and reflective writing (i.e. reflecting on the strategies adopted to achieve the goals and considering how to make improvements). The length of the reflective writing was around 200 words. Prior to the study, all students attended a technical training workshop to understand different phases of ePortfolio development. They were particularly trained and instructed to: 1) identify specific learning goals in the Goal Setting phase; 2) collect a set of learning products that demonstrate their programming ability in the Artifact Collection phase; 3) choose learning products that clearly indicate their programming achievement towards the intended goals in the Artifact Selection phase; and 4) make connections between learning goals, artifacts and strategies in the Reflective Writing phase. Figure 2 shows the development process of an ePortfolio.

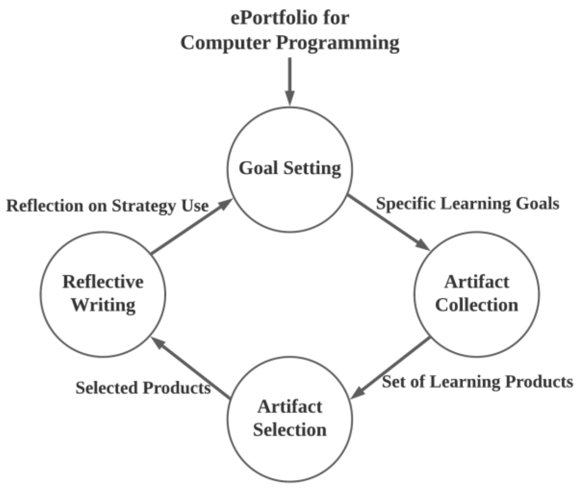


Figure 2. The Development Process of an ePortfolio

At the commencement of the study, the learning strategies section of MSLQ was administered to the participants to evaluate their use of SRL strategies. The participants then started to learn Java programming, attempt assessment tasks, and develop their ePortfolios. The course teacher would mark the assessments and the research team would analyse the results of MSLQ and annotate student reflection on ePortfolios. At the end of the course, the participants were divided into two different groups based on a median split of different types of assessment scores. Those with a score higher than the median were assigned to a high achieving group (HG), while the remaining were assigned to a low achieving group (LG). Figure 3 illustrates the procedure of this study, while Table 2 summarizes the descriptive statistics of assessment scores attained by the two achieving groups at different levels of programming courses.

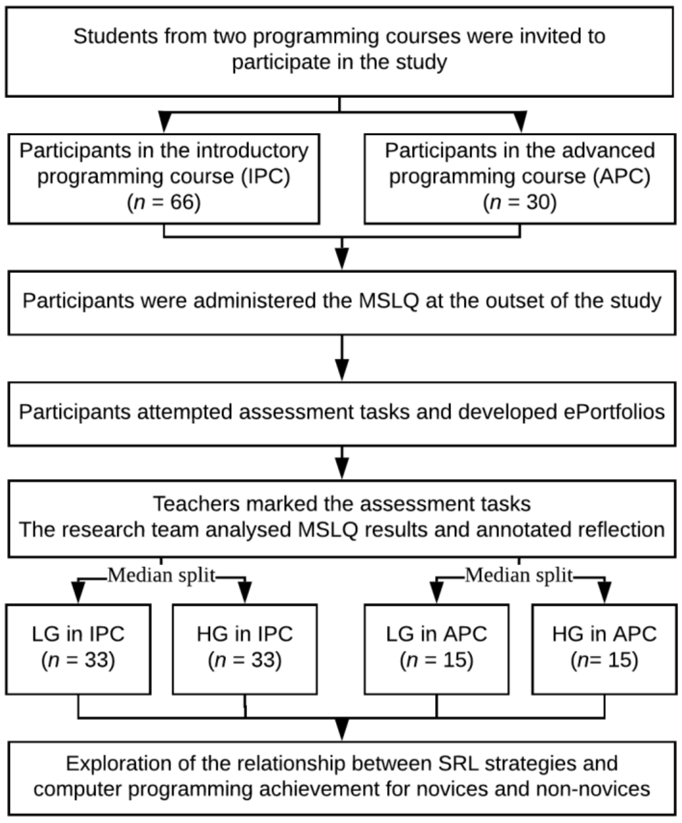

Students from two programming courses were invited to participate in the study
Participants in the introductory programming course (IPC) (n = 66)
Participants in the advanced programming course (APC) (n = 30)
Participants were administered the MSLQ at the outset of the study
Participants attempted assessment tasks and developed ePortfolios
Teachers marked the assessment tasks
The research team analysed MSLQ results and annotated reflection
Median split
Median split
LG in IPC (n = 33)
HG in IPC (n = 33)
LG in APC (n = 15)
HG in APC (n= 15)
Exploration of the relationship between SRL strategies and computer programming achievement for novices and non-novices

Figure 3. The Procedure of this Study

5. Results

5.1 Correlation between SRL Strategies Use and Assessment Scores

The internal consistency reliability among various subscales in the learning strategies section of the MSLQ was evaluated by Cronbach's alpha coefficient (α), ranging from 0.80 to 0.92 for each subscale. Since all the computed coefficients are above an acceptable level (0.70), no subscales need to be removed (Nunnally & Bernstein, 1994). The result suggests that the items in each subscale measured the same underlying construct of learning strategies.

Table 2. Descriptive Statistics of Assessment Scores Attained by HG and LG in Different Programming Courses

Level of Programming Course	Group	Median Split	No. of Participants		Assessment Score (Full Score = 100)	
			Male	*Female*	*M*	*SD*
Introductory	HG on CA	>=78.3	14	19	86.5	5.2
	LG on CA	<78.3	24	9	68.0	11.1
Advanced	HG on CA	>=83.6	10	5	89.1	2.6
	LG on CA	<83.6	10	5	73.6	13.5
Introductory	HG on EA	>=57	16	17	72.0	9.5
	LG on EA	<57	22	11	40.6	10.1
Advanced	HG on EA	>=41.8	12	3	61.3	16.2
	LG on EA	<41.8	8	7	33.3	6.4

Tables 3 and 4 present the correlation coefficients among learning strategies scales and the CA score in the introductory and advanced programming courses, respectively. As shown in the two tables, the four subscales of cognitive strategies (i.e. rehearsal, elaboration, organization, and critical thinking) were significantly positively correlated with the CA score in both the introductory and advanced programming courses (r≥0.38, p<0.01). The results indicate that novices and non-novices who reported more frequent use of cognitive strategies would attain higher CA scores in computer programming. In addition, it is found that the CA score was also significantly positively correlated with three other subscales including metacognitive self-regulation (r=0.40, p<0.05), effort regulation (r=0.47, p<0.01), and help seeking (r=0.47, p<0.01) but only in the advanced programming course. This suggests that non-novices were more likely to use metacognitive control strategies and resource management strategies to help improve their performance in CA.

The correlation coefficients among learning strategies scales and the EA score in the introductory and advanced programming courses are shown in Tables 5 and 6, respectively. Six subscales (i.e. elaboration, organization, critical thinking, metacognitive self-regulation, time and study environment, and help seeking) across three types of strategies were significantly positively correlated with the EA score in the two courses (r≥0.44, p<0.01). The results indicate that both novices and non-novices who reported more frequent use of cognitive, metacognitive control and resource management strategies would attain higher EA scores in computer programming. Moreover, the EA score was also significantly positively correlated with rehearsal (r=0.72, p<0.05), effort regulation (r=0.51, p<0.05), and peer learning (r=0.49, p<0.01) in the advanced programming course only. This suggests that non-novices were likely to adopt more diverse strategies to help improve their performance in EA.

Table 3. Correlation between SRL Strategies Use and the CA Score in Introductory Programming

	1	2	3	4	5	6	7	8	9	10
1. Rehearsal	--	0.61**	0.54**	0.43**	0.37**	0.34**	0.47**	0.58**	0.45**	0.38**
2. Elaboration		--	0.81**	0.78**	0.62**	0.57**	0.37**	0.34**	0.59**	0.59**
3. Organization			--	0.77**	0.76**	0.61**	0.35**	0.35**	0.67**	0.53**
4. Critical Thinking				--	0.75**	0.64**	0.32**	0.26**	0.68**	0.41**
5. Metacognitive Self-regulation					--	0.65**	0.35**	0.26*	0.75**	0.24
6. Time & Study Environment Management						--	0.33**	0.27*	0.63**	0.11
7. Effort Regulation							--	0.60**	0.45**	0.17
8. Peer Learning								--	0.33**	0.22
9. Help Seeking									--	0.00
10. CA Score										--

*$p<0.05$, **$p<0.01$

Table 4. Correlation between SRL Strategies Use and the CA Score in Advanced Programming

	1	2	3	4	5	6	7	8	9	10
1. Rehearsal	--	0.90**	0.90**	0.86**	0.71**	0.68**	0.68**	0.73**	0.80**	0.55**
2. Elaboration		--	0.97**	0.96**	0.86**	0.81**	0.73**	0.73**	0.91**	0.59**
3. Organization			--	0.96**	0.86**	0.82**	0.79**	0.73**	0.93**	0.58**
4. Critical Thinking				--	0.85**	0.81**	0.78**	0.67**	0.91**	0.61**
5. Metacognitive Self-regulation					--	0.95**	0.84**	0.82**	0.96**	0.40*
6. Time & Study Environment Management						--	0.77**	0.80**	0.93**	0.29
7. Effort Regulation							--	0.74**	0.86**	0.47**
8. Peer Learning								--	0.79**	0.27
9. Help Seeking									--	0.47**
10. CA Score										--

*$p<0.05$, **$p<0.01$

Table 5. Correlation between SRL Strategies Use and the EA Score in Introductory Programming

	1	2	3	4	5	6	7	8	9	10
1. Rehearsal	--	0.61**	0.54**	0.43**	0.37**	0.34**	0.47**	0.58**	0.45**	0.07
2. Elaboration		--	0.81**	0.78**	0.62**	0.57**	0.37**	0.34**	0.59**	0.50**
3. Organization			--	0.77**	0.76**	0.61**	0.35**	0.35**	0.67**	0.53**
4. Critical Thinking				--	0.75**	0.64**	0.32**	0.26*	0.68**	0.62**
5. Metacognitive Self-regulation					--	0.65**	0.35**	0.26*	0.75**	0.61**
6. Time & Study Environment Management						--	0.33**	0.27*	0.63**	0.44**
7. Effort Regulation							--	0.60**	0.45**	0.10
8. Peer Learning								--	0.33**	-0.03
9. Help Seeking									--	0.50**
10. EA Score										--

*p<0.05, **p<0.01

Table 6. Correlation between SRL Strategies Use and the EA Score in Advanced Programming

	1	2	3	4	5	6	7	8	9	10
1. Rehearsal	--	0.90**	0.90**	0.86**	0.71**	0.68**	0.68**	0.73**	0.80**	0.72**
2. Elaboration		--	0.97**	0.96**	0.86**	0.81**	0.73**	0.73**	0.91**	0.80**
3. Organization			--	0.96**	0.86**	0.82**	0.79**	0.73**	0.93**	0.77**
4. Critical Thinking				--	0.85**	0.81**	0.78**	0.67**	0.91**	0.80**
5. Metacognitive Self-regulation					--	0.95**	0.84**	0.82**	0.86**	0.74**
6. Time & Study Environment Management						--	0.77**	0.80**	0.93**	0.69**
7. Effort Regulation							--	0.74**	0.86**	0.51**
8. Peer Learning								--	0.79**	0.49**
9. Help Seeking									--	0.78**
10. EA Score										--

*p<0.05, **p<0.01

5.2 Self-reported Use of SRL Strategies

This section sought to identify the differences in the self-reported use of SRL strategies between the high and low achieving groups in the introductory and advanced programming courses. An independent samples *t*-test was used to examine if there was statistically significant difference in each subscale between the two groups classified based on different types of assessment. Tables 7 and 8 present the statistical results based on CA and EA, respectively.

Table 7 shows statistically significant differences in the use of cognitive strategies only between the high and low achieving groups classified based on CA. The high achieving groups in both programming courses reported more frequent use of cognitive strategies associated with rehearsal (Introductory level: t=2.98, p<0.01; Advanced level: t=4.35, p<0.001), elaboration (Introductory level: t=6.08, p<0.001; Advanced level: t=3.06, p<0.01), organization (Introductory level: t=4.42, p<0.001; Advanced level: t=3.00, p<0.01) and critical thinking (Introductory level: t=3.20, p<0.01; Advanced level: t=3.41, p<0.01). The four cognitive strategies tended to be effective for improving student performance in CA at both novice and non-novice levels.

On the other hand, Table 8 shows statistically significant differences in the use of not only cognitive strategies but also metacognitive control and resource management strategies between the high and low achieving groups classified based on EA. The high achieving groups in both courses reported more frequent use of the strategies associated with elaboration (Introductory level: t=4.42, p<0.001; Advanced level: t=5.09, p<0.001), organization (Introductory level: t=4.87, p<0.001; Advanced level: t=5.02, p<0.001), critical thinking (Introductory level: t=6.57, p<0.001; Advanced level: t=5.82, p<0.001), metacognitive self-regulation (Introductory level: t=5.11, p<0.001; Advanced level: t=5.56, p<0.001), time and study environment management (Introductory level: t=5.18, p<0.001; Advanced level: t=5.73, p<0.001), and help seeking (Introductory level: t=4.71, p<0.001; Advanced level: t=5.96, p<0.001). The six cognitive, metacognitive control, and resource management strategies appeared to be helpful in enhancing student performance in EA at both novice and non-novice levels.

Table 7. Means, Standard Deviations and *t*-test Values of Learning Strategies Scales for HG and LG Classified by CA

Subscales	Introductory Programming					Advanced Programming				
	LG (*n* = 33)		HG (*n* = 33)		*t*-value	LG (*n* = 15)		HG (*n* = 15)		*t*-value
	M	*SD*	*M*	*SD*		*M*	*SD*	*M*	*SD*	
Rehearsal	4.25	0.83	4.88	0.88	2.98**	3.26	0.82	4.58	0.85	4.35***
Elaboration	4.41	0.78	5.52	0.70	6.08***	3.22	1.15	4.40	0.95	3.06**
Organization	4.42	0.70	5.23	0.78	4.42***	3.20	1.15	4.36	0.95	3.00**
Critical Thinking	4.65	0.84	5.30	0.81	3.20**	3.29	1.06	4.47	0.83	3.41**
Metacognitive Self-regulation	4.63	0.80	4.82	0.77	1.00	3.70	0.95	4.18	1.04	1.32
Time & Study Environment Management	4.80	0.86	4.92	0.60	0.65	3.77	0.86	4.12	0.97	1.03
Effort Regulation	4.33	0.65	4.46	0.94	0.65	3.46	1.13	4.06	0.84	1.66
Peer Learning	4.25	0.88	4.33	1.05	0.34	3.88	0.72	4.22	0.89	1.13
Help Seeking	4.86	0.82	4.93	0.78	0.38	3.44	1.05	4.21	0.96	2.08

$^{**}p<0.01$, $^{***}p<0.001$

Table 8. Means, Standard Deviations and *t*-test Values of Learning Strategies Scales for HG and LG Classified by EA

Subscales	Introductory Programming					Advanced Programming				
	LG (*n* = 33)		HG (*n* = 33)		*t*-value	LG (*n* = 15)		HG (*n* = 15)		*t*-value
	M	*SD*	*M*	*SD*		*M*	*SD*	*M*	*SD*	
Rehearsal	4.44	0.82	4.69	0.98	1.12	3.47	0.72	4.37	1.18	2.51
Elaboration	4.52	0.82	5.41	0.80	4.42***	3.00	0.80	4.62	0.96	5.09***
Organization	4.39	0.66	5.26	0.78	4.87***	2.98	0.79	4.59	0.96	5.02***
Critical Thinking	4.42	0.69	5.53	0.68	6.57***	3.08	0.71	4.69	0.80	5.82***
Metacognitive Self-regulation	4.31	0.53	5.14	0.78	5.11***	3.22	0.56	4.65	0.83	5.56***
Time & Study Environment Management	4.46	0.71	5.26	0.53	5.18***	3.28	0.43	4.60	0.79	5.73***
Effort Regulation	4.36	0.67	4.44	0.92	0.42	3.22	0.88	4.31	0.87	3.42**
Peer Learning	4.27	0.68	4.31	1.19	0.18	3.77	0.55	4.33	0.94	2.01
Help Seeking	4.49	0.67	5.30	0.71	4.71***	3.03	0.57	4.60	0.84	5.96***

$^{**}p<0.01$, $^{***}p<0.001$

6. Discussion

6.1 Cognitive Strategies

The results of this study indicate that student use of cognitive strategies (see Tables 3 to 6) was correlated significantly with their performance in both continuous and end of course assessments. The results also indicate that students in different achieving groups adopted cognitive strategies at different levels for different types of assessment (see Tables 7 and 8). In continuous assessment, small practical exercises were designed with each focusing on a particular programming topic like conditionals, arrays or strings. Students could simply use rudimentary cognitive strategies like rehearsal or rote memorization to complete this kind of simple assessment tasks. For example, the reflective writing of high achieving students show that they often read the lecture notes multiple times in order to find the answers to the exercises. This can be seen as a way of processing the information to be learned at a surface level and can be effective for simple tasks (Aukrust, 2011). However, rehearsal strategies may not be effective for end of course assessment. In end of course assessment, students were required to attend a written examination covering a wide range of topics and abstract concepts in computer programming. Deep understanding of the programming knowledge developed by higher-order cognitive strategies seems to play a more critical role in this kind of assessment.

As can be seen from Tables 7 and 8, higher-order cognitive strategies like elaboration, organization and critical thinking can contribute to both continuous and end of course assessments. According to the reflection of high achieving students, the elaboration strategies like using analogies could help them to explain the ideas of programming by making comparison with something that are well understood in our daily life. High achieving students also reflected that the organization strategies like outlining and clustering could help them to summarize key learning topics and group similar programming concepts together. The critical thinking strategies would help students to make critical evaluation with respect to the standard of excellence (Pintrich et al., 1991). This is exemplified by the reflection of high achieving students that emphasizes on comparing the actual output of their program with that of manual processing to determine whether their program works properly. All the elaboration, organization and critical thinking strategies can potentially facilitate students to store the information to be learned in their long-term memory (Aukrust, 2011).

6.2 Metacognitive Control Strategies

As evident from Tables 3 to 6, student use of metacognitive control strategies tended to be significantly correlated with their performance in end of course assessment. A related result can be found from Tables 7 and 8 that there was statistically significant difference in metacognitive control strategies between the two achieving groups for end of course assessment only. Metacognitive control strategies are concerned with controlling and regulating one's own cognition such as planning, monitoring and regulating (Pintrich et al., 1991), and the strategies can have positive impact on the academic performance of students (Duncan & McKeachie, 2005). For instance, high achieving students demonstrated in their reflective writing that they were inclined to plan their participation in class exercises (i.e. planning), track their own attention during lectures (i.e. monitoring), and practise more on difficult programming concepts (i.e. regulating). The importance of metacognitive control strategies is particularly obvious in end of course assessment. This can be in part attributed to the fact that the scope of end of course assessment was not just limited to a specific topic or concept. To achieve a good performance in end of course assessment, students needed to concentrate well and sustain their efforts towards learning over the entire course period. The results are consistent with those of Greene et al. (2018) that metacognitive control strategies can contribute to deep understanding of concepts.

6.3 Resource Management Strategies

Resource management strategies tended to be associated more with student performance in end of course assessment than in continuous assessment (see Tables 3 to 6). Moreover, statistically significant difference in resource management strategies is found between the two achieving groups for end of course assessment only (see Table 7 and 8). High and low achievers in the two courses adopted resource management strategies for continuous assessment at nearly the same level but for end of course assessment at different levels. Time and study environment management strategies and help seeking strategies were commonly used for end of course assessment by high achieving students in both courses.

Time and study environment management strategies pertain to effectively managing time and appropriately choosing a place for study. The strategies were least relevant to continuous assessment where students were often asked to do exercises during the class, but they were particularly relevant to end of course assessment because it required students to study and review course materials outside the class. The reflection of high achieving students suggests that scheduling a regular time to study in the library every week is very helpful in studying the programming course. In addition to time and study environment management strategies, help seeking strategies were apparently related to end of course assessment. The strategies specifically refer to the ways of looking for assistance from peers, teachers or other sources, which could be useful for solving the problems arising from the wide coverage of concepts with a high level of difficulty in end of course assessment. In this study, it can be observed from student reflection that one typical source of assistance was through watching the instructional videos on computer programming available on online video sharing and social media platforms.

A worthy point to note in respect to resource management strategies is that the high achieving group in the advanced programming course appeared to use more different strategies for end of course assessment than those in the introductory programming course. For example, a significant difference in the deployment of effort regulation strategies for end of course assessment is found between the two groups in the advanced programming course but not in the introductory programming course. This suggests that the maintenance of persistence towards study in the face of difficulty plays a more significant role in learning advanced programming knowledge and skills.

Based on the results reported and discussed above, it is argued that the development of computer programming ability of novices and non-novices would require a number of SRL strategies supportive of deep understanding and sustained learning. Some of the results indicate that students' use of advanced cognitive strategies, metacognitive control strategies and resource management strategies are likely to be positive predictors of their overall programming achievement.

7. Concluding Remarks

This study aimed to investigate a research issue about the effects of Self-Regulated Learning (SRL) strategies on computer programming achievement in teacher education. The relationship between students' self-reported use of SRL strategies and their computer programming achievement were explored. The learning strategies section of the Motivated Strategies for Learning Questionnaire (MSLQ) was administered to 66 first year university students from an introductory Java programming course and 30 second year university students from an advanced Java programming course in an attempt to evaluate their use of SRL strategies, while their computer programming achievement was measured by their performance in continuous assessment (i.e. hands-on class exercises) and end of course assessment (i.e. written examination). All participating students were engaged in developing an electronic portfolio (ePortfolio) to support their reflection on the use of SRL strategies specific to computer programming for further analysis.

The results of this study indicate that all cognitive strategies were significantly positively correlated with student performance in continuous assessment, while advanced cognitive strategies, metacognitive control strategies and resource management strategies were significantly positively correlated with student performance in end of course assessment. Furthermore, the results reveal statistically significant differences in the self-reported use of the cognitive, metacognitive control and resource management strategies between high and low achievers at different programming levels. Both findings suggest that higher-order cognitive skills (i.e. elaboration, organization, critical thinking), metacognitive control strategies (i.e. self-regulation) and resource management strategies (i.e. time and study environment management, help seeking) are likely to facilitate a more prolonged achievement of computer programming.

There are two limitations in this study that could be addressed by future research. The first is the limited number of participants and the second is the self-reported data on strategy use. They may affect the generalization of the findings of this study to other contexts. As long as the limitations are recognized, this study can contribute to stimulating more discussion and advancing further research on the nature and role of SRL strategies in the pedagogy of computer programming within various contexts. Future research can also be directed at identifying strategy deployment by multiple sources of data for triangulation and exploring the impact of SRL strategy training on computer programming achievement.

Acknowledgement

This research is financially supported by the General Research Fund of the Research Grants Council of Hong Kong (No. 18601118).

References

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, *55*, 832-835. https://doi.org/10.1093/comjnl/bxs074

Akyol, G., Sungur, S., & Tekkaya, G. (2010). The Contribution of Cognitive and Metacognitive Strategy Use to Students' Science Achievement. *Educational Research and Evaluation*, *16*(1), 1-21. https://doi.org/10.1080/13803611003672348

Alario-Hoyos, C., Est évez-Ayres, I., P érez-Sanagust ín, M., Delgado Kloos, C., & Fern ández-Panadero, C. (2017). Understanding Learners' Motivation and Learning Strategies in MOOCs. The *International Review of Research in Open and Distributed Learning*, *18*(3). https://doi.org/10.19173/irrodl.v18i3.2996

Aukrust, V. G. (2011). *Learning and cognition*. Oxford: Elsevier.

Bellhäuser, H., Lösch, T., Winter, C., & Schmitz, B. (2016). Applying a web-based training to foster self-regulated learning- Effects of an intervention for large numbers of participants. *The Internet and Higher Education, 31*, 87-100. https://doi.org/10.1016/j.iheduc.2016.07.002

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, *39*(2), 32-36. https://doi.org/10.1145/1272848.1272879

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming—12 years later. *ACM Inroads*, *10*(2), 30-36. https://doi.org/10.1145/3324888

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. *ICER'05 Proceedings of the First Annual Conference on International Computing Education Research* (pp. 81-86). Seattle, New York: ACM. https://doi.org/10.1145/1089786.1089794

Broadbent, J. (2017). Comparing online and blended learner's self-regulated learning strategies and academic performance. *The Internet and Higher Education*, *33*, 24-32. https://doi.org/10.1016/j.iheduc.2017.01.004

Broadbent, J., & Poon, W. (2015). Self-regulated learning strategies & academic achievement in online higher education learning environments: A systematic review. *The Internet and Higher Education*, *27*, 1-13. https://doi.org/10.1016/j.iheduc.2015.04.007

Çakıroğlu, Ü., & Öztürk, M. (2017). Flipped Classroom with Problem Based Activities: Exploring Self-regulated Learning in a Programming Language Course. *Educational Technology & Society*, *20*(1), 337-349.

Charmaz, K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis*. Thousand Oaks, CA: Sage.

Chen, C.-H., Wang, K.-C., & Lin, Y.-H. (2015). The Comparison of Solitary and Collaborative Modes of Game-based Learning on Students' Science Learning and Motivation. *Educational Technology & Society*, *18*(2), 237-248.

Cheng, G. (2019). Exploring factors influencing the acceptance of visual programming environment among boys and girls in primary schools. *Computers in Human Behavior, 92*, 361-372. https://doi.org/10.1016/j.chb.2018.11.043

Cheng, G., & Chau, J. (2013). Exploring the relationship between students' self-regulated learning ability and their ePortfolio achievement. *The Internet and Higher Education*, *17*, 9-15. https://doi.org/10.1016/j.iheduc.2012.09.005

Cigdem, H. (2015). How does self-regulation affect computer-programming achievement in a blended context? *Contemporary Educational Technology*, *6*(1), 19-37. https://doi.org/10.30935/cedtech/6137

Duncan, T. G., & McKeachie, W. J. (2005). The making of the motivated strategies for learning questionnaire. *Educational Psychologist*, *40*(2), 117-128. https://doi.org/10.1207/s15326985ep4002_6

Greene, J. A., Copeland, D. Z., Deekens, V. M., & Yu, S. B. (2018). Beyond knowledge: Examining digital literacy's role in the acquisition of understanding in science. *Computers & Education*, *117*, 141-159. https://doi.org/10.1016/j.compedu.2017.10.003

Greene, J. A., Yu, S. B., & Copeland, D. Z. (2014). Measuring critical components of digital literacy and their relationship with learning. *Computers & Education*, *76*, 55-69. https://doi.org/10.1016/j.compedu.2014.03.008

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education*, *54*(4), 1127-1146. https://doi.org/10.1016/j.compedu.2009.10.020

Kizilcec, R. F., Pérez-Sanagustín, M., & Maldonado, J. J. (2017). Self-regulated learning strategies predict learner behavior and goal attainment in massive open online courses. *Computers & Education*, *104*, 18-33. https://doi.org/10.1016/j.compedu.2016.10.001

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. https://doi.org/10.1016/j.chb.2014.09.012

Mcgettrick, A., Boyle, R., Ibbet, R., Lloyd, J., Lovegrove, G., & Mander, K. (2005). Grand challenges in computing education - A summary. *The Computing Journal*, *48*(1), 42-48. https://doi.org/10.1093/comjnl/bxh064

Nunnally, J. C., & Bernstein, I. H. (1994). *Psychometric Theory* (3rd ed.). New York, NY: McGraw-Hill.

Palumbo, D. B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research*, *60*(1), 65-89. https://doi.org/10.3102/00346543060001065

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Pardamean, B., & Evelin (2014). Enhancement of creativity through logo programming. *American Journal of Applied Sciences*, *18*(4), 528-533. https://doi.org/10.3844/ajassp.2014.528.533

Pedrosa, D., Cravino, J., Morgado, L., & Barreira, C. (2016). Self-regulated learning in computer programming: strategies students adopted during an assignment. In *Proceedings of the International Conference on Immersive Learning* (pp. 87-101), Santa Barbara, CA, USA. https://doi.org/10.1007/978-3-319-41769-1_7

Pintrich P. R. (1999). The role of motivation in promoting and sustaining self-regulated learning. *International Journal of Educational Research*, *31*, 459-470. https://doi.org/10.1016/S0883-0355(99)00015-4

Pintrich, P. R. (2000). The role of goal orientation in self-regulated learning. In M. Boekaerts, P. R. Pintrich, & M. Zeidner (Eds.), *Handbook of self-regulation* (pp. 451-502). San Diego, CA: Academic Press. https://doi.org/10.1016/B978-012109890-2/50043-3

Pintrich, P. R., Smith, D. A. F., Garcia, T., & McKeechie, W. J. (1991). *A manual for the use of the motivated strategies for learning questionnaire (MSLQ)*. Ann Arbor, MI: National Center for Research to Improve Postsecondary Teaching and Learning, University of Michigan.

Pintrich, P. R., Smith, D. A. F., Garcia, T., & McKeechie, W. J. (1993). Reliability and predictive validity of the Motivated Strategies for Learning Questionnaire (MSLQ). *Educational and Psychological Measurement*, *53*, 801-813. https://doi.org/10.1177/0013164493053003024

Qian, Y., & Lehman, J. (2017). Students' misconception and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education*, *18*(1), 1-24. https://doi.org/10.1145/3077618

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Robins, A. V. (2019) Novice programmers and introductory programming. In S. A. Fincher, & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 327-376). Cambridge, UK: Cambridge University Press. https://doi.org/10.1017/9781108654555.013

Saeed, N., Yang, Y., & Sinnappan, S. (2009). Emerging web technologies in higher education: A case of incorporating blogs, podcasts and social bookmarks in a web programming course based on students' learning styles and technology preferences. *Educational Technology & Society*, *12*(4), 98-109.

Scherer, R. (2016). Learning from the past—The need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology*, *7*(1390). https://doi.org/10.3389/fpsyg.2016.01390

Scherer, R., Siddiq, F., & Viveros, B. S. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior*, *109*, 106349. https://doi.org/10.1016/j.chb.2020.106349

Seyal, A. H., Mey, Y. S., Matusin, M. H., Siau, N. H., & Rahman, A. A. (2015). Understanding Students Learning Style and Their Performance in Computer Programming Course: Evidence from Bruneian Technical Institution of Higher Learning. *International Journal of Computer Theory and Engineering*, *7*(3), 241-247. https://doi.org/10.7763/IJCTE.2015.V7.964

Shaw, R.-S. (2012). A study of the relationships among learning styles, participation types, and performance in programming language learning supported by online forums. *Computers & Education*, *58*(1), 111-120. https://doi.org/10.1016/j.compedu.2011.08.013

Song, D., Hong, H., & Oh, E. Y. (2021). Applying computational analysis of novice learners' computer programming patterns to reveal self-regulated learning, computational thinking, and learning performance. *Computers in Human Behavior*, *120*, 106746. https://doi.org/10.1016/j.chb.2021.106746

Sun, T., & Wang, C. (2020). College students' writing self-efficacy and writing self-regulated learning strategies in learning English as a foreign language. *System*, *90*, 102221. https://doi.org/10.1016/j.system.2020.102221

Sung, H.-Y., Hwang, G.-J., & Yen, Y.-F. (2015). Development of a contextual decision-making game for improving students' learning performance in a health education course. *Computers & Education*, *82*, 179-190. https://doi.org/10.1016/j.compedu.2014.11.012

Teng, L. S., & Zhang, L. J. (2020). Empowering learners in the second/foreign language classroom: Can self-regulated learning strategies-based writing instruction make a difference? *Journal of Second Language Writing*, *48*, 1-16. https://doi.org/10.1016/j.jslw.2019.100701

Wang, Y., & Sperling, R. A. (2020). Characteristics of Effective Self-Regulated Learning Interventions in Mathematics Classrooms: A Systematic Review. *Frontiers in Education*, *5*, 58. https://doi.org/10.3389/feduc.2020.00058

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. *Proceedings of the 2014 conference on innovation & technology in computer science education* (pp. 39-44). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/2591708.2591749

Winters, F. I., Greene, J. A., & Costich, C. M. (2008). Self-Regulation of Learning within Computer-Based Learning Environments: A Critical Analysis. *Educational Psychology Review*, *20*(4), 429-444. https://doi.org/10.1007/s10648-008-9080-9

Yilmaz, R. (2017). Exploring the role of e-learning readiness on student satisfaction and motivation in flipped classroom. *Computers in Human Behavior*, *70*, 251-260. https://doi.org/10.1016/j.chb.2016.12.085

Zacharis, N. Z. (2010). The effect of learning style on preference for web-based courses and learning outcomes. *British Journal of Educational Technology*, *42*(5), 790-800. https://doi.org/10.1111/j.1467-8535.2010.01104.x

Zheng, J., Xing, W., Huang, X., Li, S., Chen, G., & Xie, C. (2020). The role of self-regulated learning on science and design knowledge gains in engineering projects. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2020.1761837

Zimmerman, B. J. (2000). Attainment of self-regulation: A social cognitive perspective. In M. Boekaerts, P. Pintrich, & M. Zeidner (Eds.), *Self-regulation: Theory, research, and applications* (pp. 13-39). San Diego, CA: Academic Press.

Zimmerman, B. J. (2001). Theories of self-regulated learning and academic achievement: An overview and analysis. In B. Zimmerman & D. Schunk (Eds.), *Self-regulated learning and academic achievement: Theoretical perspectives* (2nd ed., pp. 1-37). Mahwah, NJ: Erlbaum.