Teaching Programming Online: An Approach with Attributional Factors and Self-Efficacy

Xiaoyun He

Auburn University at Montgomery, xhe@aum.edu

Follow this and additional works at: https://aisel.aisnet.org/sais2022

Recommended Citation

https://aisel.aisnet.org/sais2022/33

This material is brought to you by the Southern (SAIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in SAIS 2022 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
TEACHING PROGRAMMING ONLINE:
AN APPROACH WITH ATTRIBUTIONAL FACTORS AND SELF-EFFICACY

Xiaoyun He
Auburn University at Montgomery
xhe@aum.edu

ABSTRACT
Despite the growth in rates of students enrolled in online courses, the data shows that online courses face the challenge of a higher attrition rate than their face-to-face counterparts. While numerous studies explored the approaches to teaching programming, few have focused on the fully online delivery mode. In this paper, we describe and discuss our approach in designing and delivering of an introductory programming course online. Drawing on attribution and self-efficacy theories, we provide a framework in which each of the main elements in our approach contributes to the attributional factors such as behaviors instigation that includes motivational behavior and learning strategy, effort, and persistence. These factors influence performance indirectly through self-efficacy.

Keywords
Introductory Programming, Pedagogical, Online Delivery, Attributional Factors, Self-efficacy.

INTRODUCTION
According to the US Bureau of Labor Statistics (2020), employment of computer and information technology careers is projected to grow 12 percent from 2018 to 2028, much faster than the average for all occupations. The demand for computing and programming skills is expected to continue to increase for the foreseeable future. Technical aspects of software development including programming are considered some of the foundational material that we will expect our students in Information Systems to master by the time of graduation (Topi, 2019).

Yet, learning to program can be an incredibly difficult task, to the point where the phrases “high failure rate” and “programming course” are almost synonymous (Bennedsen & Caspersen, 2007). In order to improve the situation, extensive research efforts have been devoted to develop and apply various teaching tools and methods (Vihavainen, Airaksinen, & Watson, 2014). However, pass rates were not found to have significantly differed over time (Watson & Li, 2014). It appears that introducing students to programming is still one of computing education’s grand challenges.

In recent years, rates of students enrolled in at least one online course have consistently grown from 31.1 percent in 2016 to 33.1 percent in 2017 and 34.7 percent in 2018 (National Center for Education Statistics, 2019). The recent Covid-19 pandemic forced many universities moving classes online. Despite the shift to online learning in higher education, online courses face an ongoing chronic dilemma - a higher attrition rate than their traditional face-to-face counterparts (Simplicio, 2019; Lee, Pate, & Cozart, 2015).

To date, a large number of studies have examined various teaching programming related techniques and practices, but few have focused on programming courses in fully online delivery format. In this paper, we share our approach in designing and teaching an introductory Java programming course online at a regional state university. Drawing on attribution and self-efficacy theories, we provide a framework in which each of the elements in our approach contributes to the attributional factors such as behaviors instigation that includes motivational behavior and learning strategy, effort, and persistence. In addition, we recognize that our students have diverse backgrounds while largely share some similar characteristics. Research shows that teaching geared to common learner characteristics can be more effective than teaching focused on individual differences (Willingham & Daniel, 2012). We discuss pedagogic considerations in our approach, which combines research-based practices with adaptations that
recognize the common learner characteristics. Students give high marks to the course’s weekly objectives, short videos with explanations and examples, and hands-on activities.

In particular, the introductory programming course is offered at a public comprehensive university in the South of the United States. Undergraduate students majoring in information systems (IS) are required to complete an introductory programming course with the option to choose either Java or C++. Students must pass this course with a letter grade of C or higher. It is deemed to be one of the core knowledge areas that the students should possess to thrive in the information age. Our objective for this course is to introduce students to programming as a problem-solving process so that they can understand the fundamental components in coding and write programs for basic problems.

**RELATED LITERATURE AND THEORETICAL BACKGROUND**

Over the past few decades, tremendous amount of research effort has been put into developing effective approaches to teaching programming. The proposed approaches include moving from a traditional lecture and lab-based approach to using flipped classroom, pair programming, game-based learning, extreme apprenticeship, etc. (Vihavainen, Airaksinen, & Watson, 2014). Yet, a recent systematic review on the curriculum, pedagogy, and languages for teaching introductory programming (Luxton-Reilly et al., 2018) concludes that “Despite many years of study and debate, some significant issues in teaching programming remain unresolved and probably unresolvable.”

For students especially for novice programmers, learning programming is still perceived as difficult. The learning challenges facing students include the perceived ability related to problem solving, motivation and engagement, difficulties in learning the syntax of programming languages, mathematical ability, etc. (Medeiros et al., 2018). Prior research has cited subject difficulty, unrealistic expectation, lack of study, lack of practice, and lack of time as the causes attributed to the failure in programming courses (Hawi, 2010; Luxton-Reilly, 2016).

Meanwhile, the advances in computer technologies has made it possible to deliver courses online via audio and/or video (live or prerecorded), including both synchronous (i.e., simultaneous) and asynchronous (i.e., not simultaneous) instruction. Chenoweth, Corral, & Scott (2016) found that an asynchronous tutorial can achieve the same learning outcomes as a traditional lecture format by using automated feedback for convergence (using PPT slides with button choices). A study conducted by He and Yen (2014) suggests that the participants in the face-to-face delivery method group were more likely to feel satisfied with the delivery method than the students using the other two delivery methods (i.e., satellite broadcasting and live video streaming). Despite the growth in the number of students enrolled in online courses, online programming teaching and learning remains a relatively undeveloped area in the literature.

Among many factors attributable to online course dropout, the lack of persistence is one area of focus (Hart, 2012; Simplicio, 2019). Successful completion of online learning largely depends on maintaining active engagement in the course activities and requires strong self-regulatory skills and increased autonomy (Lee, Pate, & Cozart, 2015). Student characteristics also plays a role in online course attrition (Cochran et al., 2014).

In achievement-related academic contexts, success may be attributed to high ability and/or effort, while failure is perceived as due to low ability and/or lack of effort (Weiner, 1972). Drawing on attribution theory (Heider, 1958), prior studies have been rather convincingly demonstrated that causal attributions influence the likelihood of students undertaking achievement activities, the intensity of work at these activities, and the degree of persistence in the face of failure. These behaviors manifestly influence the degree of learning in higher education (Weiner, 1972; Weiner, 2010). For example, when people attribute their successes to unstable causes (luck or effort) and their failures to stable causes (ability or task difficulty), the probability of persistence is low. Attributions, in combination with motivation, has been found to predict almost 50% of the variance in students’ university Grade Point Average (GPA) scores (Wilson, 2002). Research also indicates that attributing success to effort may reflect that students’ confidence in their abilities is suffering, and that attributing failures to external causes, such as task difficulty, could be detrimental to performance and learning (Siegle et al., 2009).

Although instruction is regarded as the major environmental factor affecting scholastic success, other factors can become more important when teaching does not produce the desired results. Attributional retraining, rooted in attribution theory, is one alternative intended for enhancing students' motivation and achievement striving by changing how students think about their successes and failures, resulting in the increase in performance and persistence (Perry et al., 1993). Prior research suggests that the students who received attributional retraining were less likely to fail (e.g., Stewart et al., 2011; Wilson & Linville, 1985).

While engaged in routine teaching activities, instructors may actually function as informal trainers by encouraging controllable attributions such as effort and strategy in place of immutable causes such as academic ability or intelligence (Perry et al., 1993; Stewart et al., 2011). When faced with the attributional statements (e.g., “I'm not smart enough to pass”, “The material is too
difficult”, “I was just lucky to do well on that test”), the instructor has an ideal opportunity to encourage the student to think differently about the event, by suggesting a more suitable explanation (e.g., “You do have the ability; otherwise you would not be here”, “This may be a difficult course, but you can master the material if you study harder.” “Luck has less to do with your success than your approach (strategy) to the course”). During these informal exchanges the instructor takes on the typical role of the trainer in more formal programs, by monitoring and correcting students' attributional think. Such attributional retraining is considered as an adjunct to, and possible aspect of, effective teaching (Perry et al., 1993).

Attributional factors such as the amount of effort expended and judgments of task difficulty influence performance indirectly through self-efficacy (Schunk, 1991). Self-efficacy is the belief we have in our own abilities, specifically our ability to meet the challenges ahead of us and complete a task successfully (Bandura, 1997). Studies of self-efficacy suggest that it is more likely to have high student satisfaction and perceived learning rates if students come to an online course with high confidence in their capabilities of getting a good grade, dealing with difficult topics, completing online activities, managing course schedule, etc. (Alqurashi, 2019). Self-efficacy can be improved in various ways. The most influential way to create a strong sense of self-efficacy is performance accomplishment, which is based on learners’ prior successful experiences. Providing timely, authentic and constructive feedback with encouragement can also lead to higher self-efficacy (Margolis and McCabe, 2006).

AN APPROACH WITH ATTRIBUTIONAL FACTORS AND SELF-EFFICACY

In this section, we first present a framework in which each of the main elements in our approach contributes to the attributional factors such as behaviors instigation (e.g., motivational behavior and learning strategy), effort, and persistence. These factors influence performance indirectly through self-efficacy. In addition, we describe the main elements and discuss pedagogic considerations of our approach in designing and delivering the introductory Java programming online.

Figure 1. A Framework with Attributional Factors and Self-Efficacy
Realistic and Consistent Expectations

It has been suggested that the difficulties faced by novices in learning programming may be a consequence of unrealistic expectations rather than intrinsic subject complexity (e.g., Luxton-Reilly, 2016). Faculty efforts to articulate and clarify expectations do result in a change in student attitudes regarding the acceptability of certain behaviors (Aasheim, et al. 2012). In order to enhancing the learning process and reducing attrition rates, instructors should engage students early and often (Angelino, Williams, & Natvig, 2007). By setting realistic and clear learning expectations right from the beginning, we are able to foster an online learning environment in which students are less anxious about their grades and recognize the benefit of engaging in active learning for their academic success. It helps our students develop competence and confidence as budding programmers.

Specific Weekly Objectives and Tasks

Prior studies have shown that setting objectives and goals can help present-biased students to mitigate their self-control problem. In addition, task-based objectives had large and robust positive effects on the level of task completion, and task-based goals also increased course performance (Clark et al., 2020). We created weekly modules in the Blackboard, including weekly learning objectives that are aligned with the weekly activities and assessment. The students can work on the assignments any time prior to the end of a particular week, so it provides some degree of flexibility, especially in the situation where many of our students work either part-time or full-time. We supported them consistently with communications and actions throughout the term, which we will elaborate more in the subsequent subsections.

Effective Lectures: Short Videos with Examples

With the perceived difficulty and the amount of new material for programming courses, the lack of face-to-face instructions and interactions with the instructor when delivered online could lead to anxiety among students. The anxiety has a negative impact on student performance (Slavin, 2003), as well as student engagement in any given activity (Tharayil, et al., 2018). Effectively employing demonstration videos in programming courses can be beneficial in increasing students’ confidence of performing well (Sharp, 2016). Bergmann and Sams (2012) point out that while some instructors may use generic lecture videos to flip their classroom, it is better for instructors to create their own.

We created short videos by using Kaltura Capture. The videos captured the desktop of the instructor showing students the coding statements being crafted one by one in the NetBeans IDE with the instructor narrating to explain the underlying core concepts as well as demonstrating how they work in the examples of concrete programs. The videos range from approximately 9 minutes to 18 minutes in length. The videos were then uploaded into the Blackboard for students to view at their convenience.

The short length of the videos can be a positive contributing factor. Academic studies in effective teaching have long held the belief that short attention span is one of the characteristics of modern students. McKeachie & Svinicki (2013) has maintained that attention typically increases from the beginning of the lecture to 10-15 minutes into the lecture and decreases after that point. For example, the well-known TED talks are required to follow 18 min rule, which is based on the notion that 18 min is long enough to have a “serious” presentation but short enough to hold a person’s attention.

Moreover, showing students concrete and practical coding examples in the videos appears to be effective. As noted in Zhang et al. (2020), showing introductory students well-written code as an exemplar can have many benefits. For instance, if the instructor emphasizes the modularity of the code, students may be able to recognize the syntax for defining the scope of each module. The style of the code can make a good first impression that can be emulated. The proper use of methods from the main program of a carefully chosen example can make the overall purpose of the program readable, even to novice programmers.

We received a number of positive comments from our students soon after we had started creating and placing them in the Blackboard. The helpfulness of short videos was confirmed by the students’ responses in course evaluations.

Timely and Constructive Feedback

Prior research has suggested that providing timely and constructive feedback with encouragement can lead to higher self-efficacy (Margolis and McCabe, 2006). As noted by Angelo (1993), early and regular feedback not only helps learners efficiently direct their attention and energies and helps them avoid major errors and dead ends, it also can serve as a motivating form of interaction between teacher and learner, and among learners. Further, when students learn to internalize the feedback,
they can begin to give themselves corrective feedback. Subsequently they become independent, which strengthens their confidence in their ability (Angelo, 1993).

Once the students have submitted their programming assignments, we provide constructive feedback on their submissions no later than a couple of days. We also try to find out what students are doing with the feedback that we have provided. Do they read and use the comments we provide? In other words, we want to make sure the feedback is being utilized and contributes to their learning. Based on our experience, we should not assume that students understand, sometime we, as the instructors, have to ask them.

**Frequent Communication and Help Sessions**

Sharma et al. (2020) has noted that, even though it might be obvious, frequent communication has particular applicability for undergraduate students enrolled in an online programming course. Frequent communication can keep everyone in the loop. We kept students informed about what was expected for the coming weeks and important dates via the announcements in the Blackboard and emails. We also made sure that the students truly understand what’s going on. We let the students know that we were with them and that we were all in this together, which helps alleviate the stress some of the students might be experiencing.

We offered both online office hours and optional one-on-one help sessions through either Blackboard Collaborate Ultra or Zoom. In particular, the students that utilized the help sessions considered them very helpful, as reflected by the following comments: “I do not think that I can figure out what is wrong with my program without your video help session.”; “Without the one-on-one help sessions, I would have given up doing the coding assignments.”; “I was losing confidence in my ability to identify and correct the errors in my code. Your help session really helped me to see the big picture, and I learned more about debugging a program.”.

Offering extra help to students can have greater impact on students' learning than otherwise, especially in online programming courses. Prior studies suggest that students can lose interest if they have trouble running a program and experience a lack of opportunities to get help (Sharma et al., 2020). When students are learning to program, they may easily get frustrated especially when their program is not working as expected. The sheer frustration can be overwhelming. When we help them to manage some of the frustrations of learning to program, we send out a signal to our students that we care about them and their success in our course. We have also found that when students know we care about them, they seem to be more determined to do well in our class and be persistent (Zhang et al., 2020).

**EFFECTIVENESS**

To assess the effectiveness of our approach, we looked at both student performance on final exam and semester-end course evaluations. Final exam includes both multiple-choice and short answer questions. The short answer questions require writing the code snippets to meet the requirements specified in the question. We compared student performance in four semesters, two of them were taught online, and another two were face-to-face. Note that, for each semester, the questions given in the exam are different in wording but used to test the same programming concept or technique. Overall, the students in the online classes performed as well as those in the face-to-face version.

Based on the semester-end course evaluations given by the students in all those four semesters, the online classes received as high marks as the face-to-face ones. Some of the students in the online classes provided positive comments about the usefulness of the short videos as well as the flexibility of doing the work throughout the week, while some appreciated the instructor’s extra help via the videos.

**CONCLUSION**

Programming is generally regarded as difficult for students to learn and challenging for instructors to teach. A fully online programming course can make it even more challenging than when it is taught in-person. In this paper, we describe and discuss our approach in designing and delivering of an introductory programming course online. Drawing on attribution and self-efficacy theories, we provide a framework in which each of the main elements in our approach contributes to the attributional factors such as behaviors instigation that includes motivational behavior and learning strategy, effort, and persistence. These factors influence performance indirectly through self-efficacy. The preliminary results suggest that student performance and perception of programming are similar to or even better than those in face-to-face delivery mode. We hope that our approach shared in this paper will be helpful to other instructors when teaching programming online.
REFERENCES


