Southwestern Business Administration Journal

Volume 19
Issue 1 *Special Covid-19 Issue*

Article 1

2021

Teaching coding in a virtual environment: Overcoming challenges

Marion S. Smith
Texas Southern University

Teaching coding in a virtual environment: Overcoming challenges

Marion Smith
Texas Southern University
marion.smith@tsu.edu

Keywords: *active learning, coding, introductory programming education, virtual classroom*

Abstract

Educational research suggests that teaching techniques are subject matter specific. Teaching techniques in introductory programming classes are centered around two approaches used by students in learning. One approach is where students develop a thorough understanding of what they are learning. This is referred to as "deep learning". Other students use a "surface approach" where they perform the tasks required from them. The persona of the instructor and the choice of instructional materials used within a class determines which approach the student will adopt. Active teaching techniques fosters "deep learning". With the need to adapt active teaching techniques to a virtual educational environment, this paper discusses how to modify these techniques to a first course in programming.

Introduction

Educational research suggests that teaching techniques are subject matter specific (Mayer 2004). Teaching techniques in introductory programming classes are centered around two approaches used by students in learning. One approach is where students develop a thorough understanding of what they are learning. This is referred to as 'deep learning". Characteristics of students engaged in deep learning include students' ability to answer open-ended questions, participate in interactive learning activities where students construct meaning and process ideas; students first understand some basic ideas and then apply these ideas in new experiences. Other students use a "surface approach" where they perform the task required from them. A student with a surface learning approach lacks depth of learning that supports the recall and knowledge application for success on assignments and assessments; the student cannot make connections between topics and does not take the time to make those connections. The table that follows summarizes some of the characteristics of students exhibit in deep and surface approaches to learning (Houghton 2004).

	Deep learning	Surface learning
Definition	Examining new facts and ideas critically, and tying them into existing cognitive structures and making numerous links between ideas.	Accepting new facts and ideas uncritically and attempting to store them as isolated, unconnected, items.
Characteristics	Looking for meaning. Focusing on the central argument or **concepts** needed to solve a problem. Interacting actively. Distinguishing between argument and evidence. Making connections between different modules. Relating new and previous knowledge. Linking course content to real life.	Relying on rote learning. Focussing on outwards signs and the **formulae** needed to solve a problem. Receiving information passively. Failing to distinguish principles from examples. Treating parts of modules and programmes as separate. Not recognising new material as building on previous work. Seeing course content simply as material to be learnt for the exam.
Encouraged by students	Being intrinsically curious about the subject. Being determined to do well and mentally engaging when doing academic work. Having the appropriate background knowledge for a sound foundation. Having time to pursue interests, through good time management. Positive experience of education leading to confidence in ability to understand and succeed.	Studying a degree for the qualification and not being interested in the subject. Not focussing on academic areas, but emphasising others (e.g. social, sport). Lacking background knowledge and understanding necessary to understand material. Not enough time / too high a workload. Cynical view of education, believing that factual recall is what is required. High anxiety.
Encouraged by teachers	Showing personal interest in the subject. Bringing out the structure of the subject. Concentrating on and ensuring plenty of time for key concepts. Confronting students' misconceptions. Engaging students in active learning. Using assessments that require thought, and requires ideas to be used together. Relating new material to what students already know and understand. Allowing students to make mistakes without penalty and rewarding effort. Being consistent and fair in assessing declared intended learning outcomes, and hence establishing trust (see Constructive Alignment).	Conveying disinterest or even a negative attitude to the material. Presenting material so that it can be perceived as a series of unrelated facts and ideas. Allowing students to be passive. Assessing for independent facts (short answer questions). Rushing to cover too much material. Emphasizing coverage at the expense of depth. Creating undue anxiety or low expectations of success by discouraging statements or excessive workload. Having a short assessment cycle.

Figure 1: Characteristics of Deep and Surface Approaches to Learning

The persona of the instructor and the choice of instructional materials used within a class determines which approach the student will adopt (Bubica, 2014) (Murray 1990). Active teaching techniques fosters “deep learning”. For programming courses, classroom and lab environments are used for active learning rather than passive learning. Active learning is an instructional approach where students are engaged in problem-solving activities, group discussion and reflection activities to foster thinking about the subject. Active learning engages students with something that develops their skills. Active learning techniques include activities where students interact with material, participate in informative assessment, contribute to peer instruction, join group activities, and is involved in case studies (Anonymous 2020). In a traditional programming class, typical active learning techniques include hands-on coding activities, pop quizzes, and pair programming (Brown 2018). The advantage of active learning is that while engaging students, the instructor can observe the students, coach, and give immediate feedback.

In contrast to active learning, passive learning typically involves one-way communication from the teacher to the student; the teacher is the focus of attention with students observing the teacher. Examples of passive learning by students include the following teaching techniques and use of media: direct instruction – lecture, watching a video, modeled instruction, reading assignments and listening to guest speakers. The disadvantages of passive instruction include the lack of consistent feedback by teachers. Without feedback, instructors do not know if the students understood the content taught and the student is left with the impression that there is only one way to answer a question or solve a problem. In the past, to teach students to code, teachers used techniques that have been typically used in a traditional classroom using passive teaching techniques.

Students and instructors face challenges in the online classroom. For example, both students and instructors face challenges with their lack of technological proficiency that lead to technical issues. Students often have problems with time management and navigating the instructional interface. Not only do instructors have problems with course design. (Akilandeswari 2020) (Anonymous 2020) (Occupational Outlook Handbook 2020), but instructors of programming classes have additional challenges to consider. These challenges must be considered in the course design while implementing active learning techniques within an online class. Accordingly, this paper describes how to modify active teaching techniques commonly used in face-to-face courses and make them usable for an online first course in programming at JHJ School of Business.

The first challenge

Students who believe it is difficult to learn to program creates a challenge that must be overcome. This belief inhibits their commitment to the course and as a result, they have already decided to limit the amount of time they devote to the course. This is a characteristic of the "surface approach" to learning. These students have the intention to perform the tasks required from them to complete the course but limit their time devoted to developing their skills. The instructor must counter this "myth" that programming is difficult to learn by communicating to students that programming is a learned skill (Guzdial 2015) and can be learned if the students study systematically. Thus, as with any learned skill, students must buy into the idea that practice and patience is required to be successful. In a traditional classroom, the physical presence of students and instructor facilitates delivering the message that programming is a learned skill. In the classroom, the instructor can react to a student instantaneously and modify the interaction and the message without disruption. However, in an online environment, where communication is not necessarily synchronous, how does the instructor overcome this myth?

In a face-to-face course, the way an instructor can counter the "myth" *that programming is difficult* is by direct communication with the students in a synchronous meeting or conversation held during the class. In contrast, a virtual instructor must arrange to meet students in the virtual classroom during scheduled "office hours" or by appointment using video conferencing. Furthermore, if a student has no access to the virtual classroom during office hours or the student cannot commit to attending a synchronous meeting because of other obligations or technology insufficiencies, the instructor has to find a way to connect to the student before the student internalizes the idea that there is "not enough time" or has "too high a workload" to commit to the course in order to gain a thorough understanding of the content of the course. The instructor must counter this limitation by holding a one-on-one or personal conversation with the student outside the posted office hours or

virtual class meetings. This can be accomplished by a phone call at an agreed upon time. During these phone call meetings, the instructor can explain how the course is designed. For introductory programming classes, students need reassurance that the course was specifically designed for students with no background in programming. Taking time to meet with students shows them that the instructor is interested in their well-being and success in the class. Furthermore, meeting with students at their convenience initiates a level of trust between the student and the instructor that the student can draw upon when needed to approach the instructor for further guidance.
Another way to counter the "myth" that programming is difficult, is to have an asynchronous dialog where the student posts his accomplishments in a blog type of environment. Using the discussion feature of Blackboard, an instructor can monitor student postings and immediately react by recognizing a student's accomplishment (Chickering and Gamson). Acknowledging a student's accomplishment is a way is to build the student's confidence. Instead of using the discussion as an "assignment", the discussion is used as a platform to announce an accomplishment. The accomplishment is a completed activity. The activity can be a graded lesson that is in the form of a game where the student earns badges or certificates or an online tutorial that awards the student a grade. Students share with the class the certificate or grade on the discussion, and the instructor congratulates them on the accomplishment and awards a grade. This combination recognition and grade confirms to the students that the instructor is interested in their accomplishments. This offsets a student's mindset that programming is difficult to learn (Arnold 2020).

An example of an earned badge by a student on an interactive tutorial on Python from the SoloLearn.com web site is shown in Figure 2. Students can post their badges and certifications on a discussion on a course management system like Blackboard. In the discussion, students share hints and experiences in completing the lesson. By sharing their reflection, students often explain that the tutorial was easy to follow and fun to complete. Often students explain how they successfully overcame an obstacle and explain what they learned from their mistakes. Activities that award the student a badge or certificate are typically assigned to reinforce textbook readings. By doing so, students are exposed to the same concepts more than once. This helps them recognize that the foundations of the programming language can be described in more than one way and demonstrated on more than one programming platform. For example, in the textbook Lambert's *Fundamentals of Python* published by Cengage, arithmetic operations are discussed under the section topic "Expressions", whereas on SoloLearn arithmetic operations are discussed under the section topic "Simple Operations".

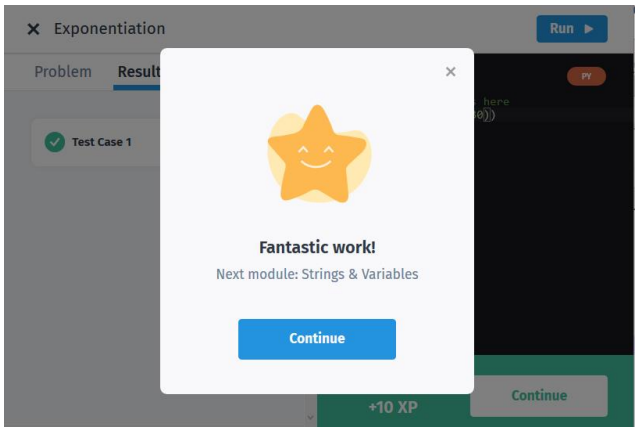

Figure 2 SoloLearn.com

The second challenge

Another challenge faced by instructors is the perception by some Management Information Systems (MIS) majors that there is little benefit from gaining programming literacy. Some MIS majors see programming as a skill for software developers, MIS majors in general have no interest

in becoming a software developer. MIS majors see themselves as future managers and see programming as a topic in one of many technology related subjects that will make them more marketable. To support this lack of interest by MIS majors, the list of duties on the Occupational Outlook Handbook by the U.S. Bureau of Labor Statistics for computer and information systems managers does not mention programming or coding, but does state,

> *Computer and information systems managers normally must have a bachelor's degree in a computer- or information science–related field. These degrees include courses in computer programming, software development, and mathematics. Management information systems (MIS) programs usually include business classes as well as computer-related ones*.

With this perspective, students often do not give a programming class the attention required to make them successful. Furthermore, many students believe that to be a successful programmer, one must have some passion for it. In many cases, students often have little or no passion for learning to code.

How does the instructor overcome this lack of interest? The instructor must make the course a positive experience for the student that in turn results in a positive attitude towards the subject of programming. Overcoming the lack of interest can be accomplished by introducing new topics and then revisiting these topics over the duration of the course. This can easily be accomplished in an online course that uses a course management system and the integration of smart content. This instructional delivery system makes it easy to introduce and repeat topics with different types of interactive assignments and assessments (Broslowski 2018) thus actively engaging students in the online course environment.

Many integrated programming textbooks such as Lambert's *Fundamentals of Python* published by Cengage, contains different types of interactive assignments. One type is referred to as annotated examples and another type is semantic code assessment problems. An annotated example is characterized by the introduction of a topic that enables students to execute sample code, on an interactive web page, and review the output. See figure 3 where the assignment operator is introduced. (Notice that the vocabulary of this text refers to the assignment statement and avoids the term operator.) On this interactive web page students can execute the code and see the result in simulated console window.

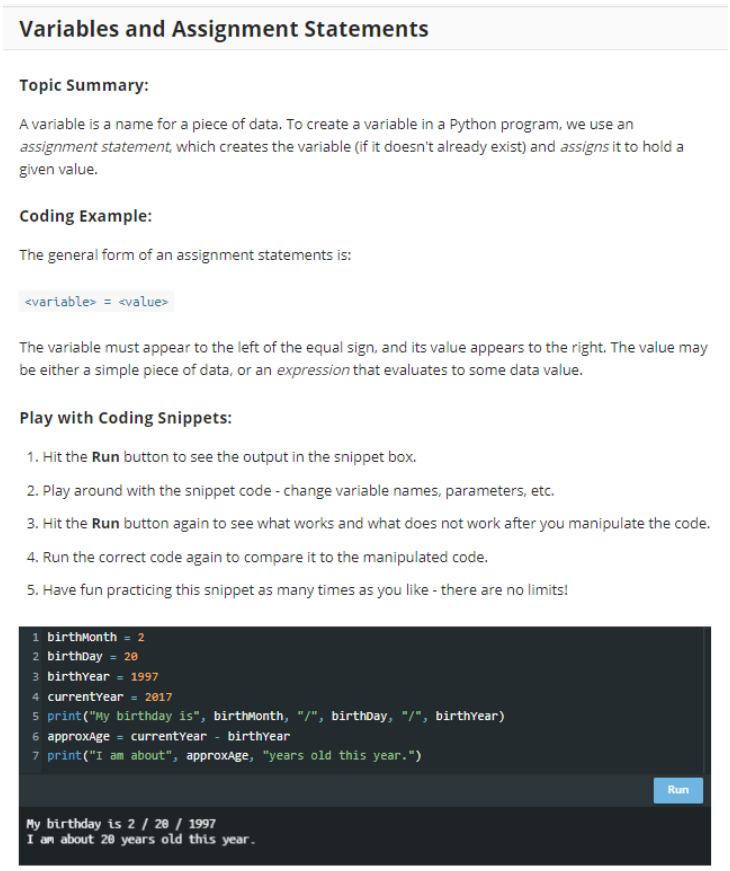

Figure 3 Coding Snippets

Semantic code assessment problems, another interactive activity, requires the student to study existing code and modify it to meet the given specifications. A sample assessment is shown in figure 4. The sample is taken from *Lambert's Fundamentals of Python* published by Cengage. The student is given several different test criteria to determine if the modified program meets the specifications.

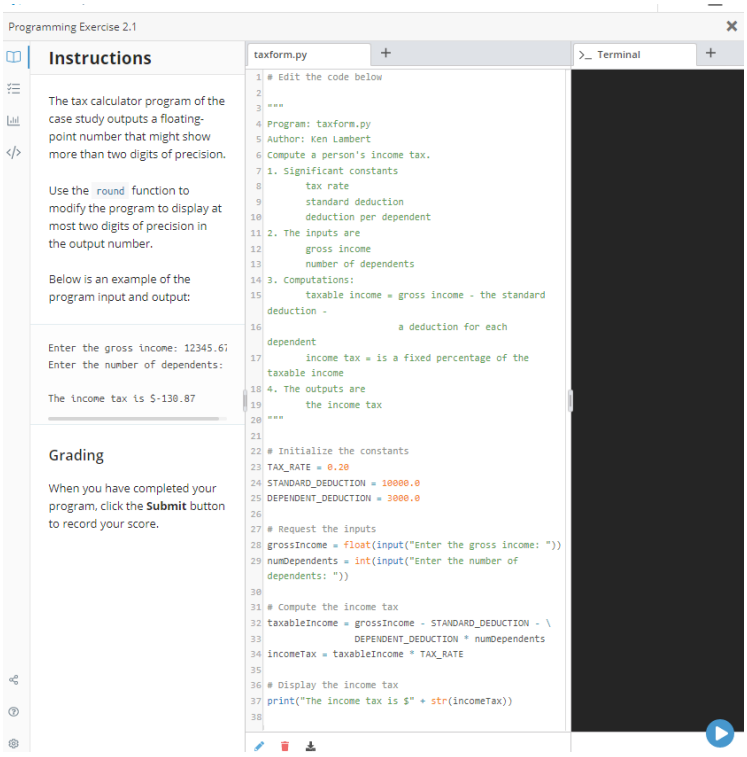

Figure 4 Semantic code assessment problems

Another example of smart content found on interactive web pages is code construction problems also known as Parson's problems.

Code construction problems are used on the SoloLearn website. As described by Brusilovsky et all (Broslowski 2018).

> "Parson's problems are code construction exercises in which students do not need to type code. This type of smart content was originally introduced by Parsons and Haden (2006) as Parson's Puzzles where a limited number of code fragments is presented in a random order. Each fragment may contain one or more lines of code. To solve the puzzle, the student must construct the described program by putting the fragments in the correct order."

As shown Figure 5, the student clicks and drags one of the possible answers to a place the right answer in "fill-in-the-blank" coding exercise.

Placing these smart activities throughout the course builds a student's interest through continued engagement with the course which in turn gives the student a positive attitude towards mastering programming concepts.

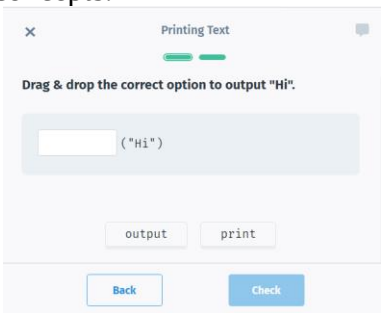

Figure 5 Parson's problem

The third challenge

In a traditional classroom, peer instruction is often used in teaching programming (Gill 2006) (Brusilovsky 2018) (Yildiz 2020). With peer instruction, students are engaged in a challenging programming question or problem and build a solution using the integrated development environment (IDE). Students discuss the solution with their peers or teach their peers the solution (Quinn 2018)j. In a classroom or computer lab where students sit together, it is easy to facilitate peer instruction. In an online environment, how does the instructor emulate peer instruction? Can an instructor required peer-to-peer interactions when so many of students believe peer-to-peer activities are a waste of time (Jaggars 2013).

One technique that emulates the peer instruction in a virtual environment is to assign students to a team. Within the team, students are required to create solutions to problems individually and then meet as a team asynchronously or synchronously to review and discuss the individual solutions. During the review, students critique each other's solutions and then adopt a consensus or final version of the solution. The final solution is submitted to the instructor as the solution to the problem and the instructor provides detailed feedback to the team members on the strengths and weaknesses of the solution.

The Fourth challenge

In a face-to-face classroom, instructors create programs in front of their students. Live coding is a technique where students learn from watching the instructor demonstrate the use of a programming tool while writing code. Some of the things a student learns from watching and listening to the instructor are the approach to the solution (top-down or bottom-up solution), use of the Integrated Development Environment (IDE) interface, use of editing shortcuts, and use of intellisense capabilities of the IDE. Students follow the instructor's lead and create the program on their computers. The instructor can pause and walk around the classroom and observe the progress of the students and give individual guidance to a student as needed. This technique is almost impossible to recreate in a virtual environment. A further difficulty in implementing live coding in the virtual classroom is the physical limitation of the student's interface to the virtual classroom. Many students use tablets, smartphones, and laptop computers to view the virtual classroom. These devices do not have sufficient screen size to allow students to view the online demonstration and use other software at the same time. One solution to this limitation is to create courses using a platform such as LinkedIn Learning. This solution enables the instructor to provide supplemental instruction materials for the course. For example, LinkedIn courses on Python provides transcripts for each instructional video. Therefore, if students need to install Python, students can watch a video and then complete the installation task on their computers while referring to the transcript after watching video.

Conclusion

Thus, with a minimum additional work, coupled with informed creativity, any teacher can increase their instructional effectiveness in online programming courses (and online courses in other areas). Accordingly, student learning can move from scratching the surface of programming knowledge to a deeper understanding of basic computer programming. With deeper understanding, students have the ability to apply that knowledge to solve a variety of problems commonly faced by managers of information systems.

REFERENCES

Akilandeswari, S. (2020, May 31). Overcoming the challenges of online teaching. Retrieved February 12, 2021, from https://www.teachingenglish.org.uk/blogs/sakilandeswari/overcoming-challenges-online-teaching

Anonymous (2020) Active learning techniques for the classroom. (2020, November 18). Retrieved February 12, 2021, from https://learninginnovation.duke.edu/resources/art-and-science-of-teaching/active-learning-techniques-classroom/

Anonymous (n.d.) Understanding learners. (n.d.). Retrieved February 12, 2021, from https://teaching.unsw.edu.au/understanding-learners

Anonymous (2020) 10 online learning challenges & how to overcome these problems. (2020, July 02). Retrieved February 12, 2021, from https://www.embibe.com/exams/online-learning-challenges-and-solutions/

Arnold, S. (2020, August 03). 10 important research findings on games in education. Retrieved February 12, 2021, from https://braveintheattempt.com/2017/07/31/10-important-research-findings-on-games-in-education/

Broslowski, P., Malmi, L., Hosseini, R., Guerra, J., Sirkiä, T., & Pollari-Malmi, K. (2018). An integrated practice system for learning programming in Python: design and evaluation. *Research and practice in technology enhanced learning*, *13*(1), 18. https://doi.org/10.1186/s41039-018-0085-9Bubica, Nikolina & Boljat, Ivica. (2014). Strategies for Teaching Programming to Meet New Challenges: State of the Art.

Brown NCC, Wilson G (2018) Ten quick tips for teaching programming. PLoS Comput Biol 14(4): e1006023. doi:10.1371/journal.pcbi.1006023

Chickering and Gamson. (n.d.). Retrieved February 12, 2021, from https://citt.ufl.edu/resources/the-learning-process/designing-the-learning-experience/chickering-and-gamson/

Computer and information Systems managers : Occupational Outlook Handbook. (2020, September 01). Retrieved February 12, 2021, from https://www.bls.gov/ooh/management/computer-and-information-systems-managers.htm#tab-2

Gill, T., & Hu, Q. (2006). Peer-to-Peer Instruction in a Programming Course. Decision Sciences Journal of Innovative Education, 4(2)

Guzdial M. Top 10 Myths About Teaching Computer Science; 2015. https://cacm.acm.org/blogs/blogcacm/189498-top-10-myths-about-teaching-computer-science/fulltext.

Houghton, W. (2004) Engineering Subject Centre Guide: Learning and Teaching Theory for Engineering Academics. Loughborough: HEA Engineering Subject Centre.

Jaggars, Shanna S. , Edgecombe, N & Stacey, Georgia W. (2013). *Creating an effective online environment*. From https://files.eric.ed.gov/fulltext/ED542153.pdf

Mayer RE. Teaching of Subject Matter. Annual Review of Psychology. 2004; 55(1):715±744. https://doi.org/10.1146/annurev.psych.55.082602.133124 PMID: 14744232

Mazur, Eric (1997). Peer instruction: A User's Manual Series in Educational Innovation Prentice Hall, Upper Saddle River, NJ.

Murray, H. G., Rushton, J. P., & Paunonen, S. V. (1990). Teacher personality traits and student instructional ratings in six types of university courses. Journal of Educational Psychology, 82(2), 250–261. https://doi.org/10.1037/0022-0663.82.2.250

Occupation Outlook Handbook on computer and Information Systems Managers U.S. Bureau of Labor Statistics Computer and Information Systems Managers : Occupational Outlook Handbook: U.S. Bureau of Labor Statistics (bls.gov) Retrieved on Feb. 7, 2021

Parker, B., & Hankins, J. (2002). AAHE's seven principles for good practice applied to an online literacy course. Journal of Computing Sciences in Colleges, 17(4). doi:10.36614/jcsc

Parsons, D., & Haden, P. (2006). Proceedings of the Australian conference on computing education - acse '06. ACE '06: Proceedings of the 8th Australasian Conference on Computing Education, 52, 157-163.

Quinn, B. (2018, April 01). The beautiful noise of peer instruction: An interview with Beth Simon. Retrieved February 12, 2021, from https://dl.acm.org/doi/10.1145/3194243

Yildiz, T. (2020, August 31). The Effect of Peer Instruction Method in Programming Education to Students' Attitudes towards Course and Programming Self-Efficacy. Retrieved January 15, 2021, from https://eric.ed.gov/?id=EJ1267971