2018 ASEE Zone IV Conference: Boulder, Colorado Mar 25

Supplemental External Assignments Incorporating Immediate Feedback for use in Entry-level Coding Courses to Promote In-Class Active Learning

Dr. John M Pavlina, Embry-Riddle Aeronautical University, Prescott

Ph.D. obtained at the University of Central Florida in Orlando under the direction of Donald Malocha. Researched Surface Acoustic Wave wireless sensors for use in NASA applications. Post Doctoral research performed at the Albert-Ludwigs-Universität Freiburg on disaster scenario cell phone location. At the university hospital affiliated with Albert-Ludwigs-Universität Freiburg research was conducted on prostate cancer ablation using HIFU and MRI. Currently working as an assistant professor at ERAU in Prescott, AZ.

Mr. Brennan Robert Gray

Supplemental Outside-of-Class Assignments Incorporating Immediate Feedback for use in an Entry-level Coding Class to Promote In-Class Active Learning

John M. Pavlina and Brennan Gray
Embry-Riddle Aeronautical University, Prescott-Campus

Abstract

Introductory computer programming classes remain difficult for incoming students with little to no experience or interaction with the background processes of a computer. Most students only interact with the graphical user interface, and therefore possess little understanding of the actual code running the programs they use daily. The first class introducing such coding concepts provides a significant challenge to many because of the various concepts that need to be learned quickly. The students must not only learn the basics of whatever programming language they are utilizing but also the fundamentals of programming a computer. It benefits the students to provide opportunities to learn coding interactively both inside and outside of prescribed class time. Difficulties arise in implementing in-class active learning strategies from the beginning of term when these first-time students must quickly internalize new vocabulary and familiarize themselves with the coding environment. When some of the basics of acquiring computer language may be performed outside of class, additional instruction time can then be allotted to active learning in the classroom. In order for these beginner students to succeed in assignments outside of class, feedback must be given immediately regarding the code being attempted. One of the main benefits of obtaining instant accuracy feedback centers on an increase in student engagement. This work examines the use of assignments with interactive learning activities. These assignments are placed on a website, such as codevolve.com, where the student can follow step-by-step instructions and then run the code to determine if the outcome is as it should be. This feedback students receive reaches far beyond simply whether or not the code will compile but actually checks the outcome of the code to ensure compliance with a desired end result. The exercises and resulting feedback provide students with an advantage during classroom instruction. The observed results of students performing the outside exercises showed increased engagement and competency during the in-class active learning assignments when compared to the control. They required less direct intervention because of common syntax errors and basic commands as these issues had been previously addressed in their out-of-class activities. An example of an exercise will be shown to demonstrate the activity and feedback system. Results from student surveys as well as a qualitative and quantitative look at student performance will be presented.

Background

Many students prefer to spend lecture time in interactive learning activities[1,2,3]. However, active learning techniques in entry-level coding classes prove challenging for students who come into

the university with no programming experience. These students generally lag behind their peers with previous experience and tend to fail the course.

Concepts such as active learning and the "inverted" or "flipped" classroom are familiar to students in the contemporary university environment[4,5]. It is no longer a question for instructors of whether these concepts work[6] but rather how best to integrate them into one's own course.

However, introductory computer programming courses remain tricky to navigate in an active learning environment. The students at Embry-Riddle who are required to take this course come from very different backgrounds, levels of experience, and degree programs. The breakdown of majors for students enrolled in the author's EGR 115 entry-level coding class in the Spring and Fall of 2017 is shown in Table 1. Because of the inconsistency of coding experience in first-year students, the instructor quickly discovers a large disparity of prior knowledge that must be addressed. The results from a first week survey in the Fall 2017 are shown in Table 2. One can see that a large portion of the students come into the course with absolutely no prior programming experience.

Table 1: Division of student by degree

Major	Spr 17	Fall 17
AE	22	43
ME	4	8
EE	0	5
CE	0	2

Table 2: Students response to level of programming experience

Programming Experience	Never	High School	High School AP	College/ University	At a job	Self Taught
# of students	28	13	4	7	1	5

The EGR 115 programming course introduces students to computing using MATLAB. The choice of MATLAB is due to the usefulness in later courses to all of the majors. It is generally understood that while computer engineers and electrical engineers will go on to utilize the programming aspects of the course more, everyone is to be taught the same. Because of the emphasis on programming, students must learn all of the basic concepts (loops, branching, and functions) and apply them to engineering problems. The author's first attempt at teaching this course to coding beginners resulted in many disgruntled and unhappy students as discovered on the end-of-year teaching evaluations. The students were assumed to have significant prior programming knowledge and therefore felt lost during the majority of the course.

In the subsequent semester, adjustments were made, the pace and material to be covered in class was altered, and the use of more in-class activities was introduced more liberally. However, such a strategy ended up as an overcorrection, since more than half of the class with some prior programming experience received no benefit.

This disparity in experience and need to balance class time between total beginner and slightly experienced students prompted the desire to create a way for novice students to obtain help outside of the class time. Of course, contemporary students have access to massively online open

courses (MOOCs) such as MIT OpenCourseWare to learn programming languages. Unfortunately, the completion percentage of students attempting a MOOC are below 40% with the average being about 15%[7], and such online offerings do not teach exactly what must be learned for the student's particular course at their own university.

With a custom assignment designed by the instructor for students of the course, rookie programmers have the possibility of absorbing introductory MATLAB terminology and syntax so that the application of larger concepts, e.g. loops, can be explored during class time.

Description of Course

The course is EGR 115: Introduction to Computing for Engineers is described as follows: "This is an introductory course in programming and computing for scientists and engineers. The course introduces students to the following aspects of software engineering: specification, requirements, design, code, and test. This course uses a problem-solving approach for developing algorithms. The following topics will be included: data types and related operations, looping, decision, input/output, functions, arrays, files, and plotting." The class extends beyond simply learning how to program but also delves into problem solving and software engineering concepts.

EGR 115 students are expected to analyze scientific and engineering problems, design algorithmic solutions to these problems, and implement the algorithms. The course is broken down into eight main topics as shown in Table 3.

Table 3: Basic outline of the course topics covered in EGR 115

Week #'s	Topics
Week 01-02	Introduction to MATLAB
Week 03	Plotting with MATLAB
Week 04	Problem Solving with Top-Down-Design
Week 05	Branching Statements
Week 06-08	Looping Statements / Vectorization
Week 09-10	Functions
Week 11	File I/O
Week 12-14	Applications /Advanced Topics

Only the first few classes are devoted to MATLAB and MATLAB syntax. Those familiar with MATLAB understand this means the instructor must cover a wealth of information including terminology and the MATLAB environment in a short time. Because of the focus on algorithm development and software engineering practices, only a few class sessions can be spent learning the basics. Therefore, out-of-class activities were created to cover some of these basic topics for total beginners.

Eleven exercises were developed and implemented for this course. The activities start with the basic operations available in MATLAB and work up to using characters and strings inside MATLAB. The list of supplemental assignment topics is as follows:

- Basic Operations
- Addressing Matrices
- Operations with Vectors
- Operations with Matrices
- Input and Output to the Command Window
- Relational Operators and Masks
- Branching statements: If-else
- For Loops
- Loops: Break and Continue
- Vectorization
- Characters and Strings

Basic Codevolve Lesson Description

The supplemental activities were designed on a website called Codevolve.com. The website was founded with a mission to make technical education scalable and accessible to as many people as possible. This platform allows educators to create custom lessons for learning any aspect of a programming language. The lessons can be set up many different ways but the most basic layout is shown in Figure 1. The activities may be run in a browser tab or integrated into a learning management system (LMS). CANVAS is the LMS used at ERAU, and it supported the tracking and grading of the Codevolve student assignments.

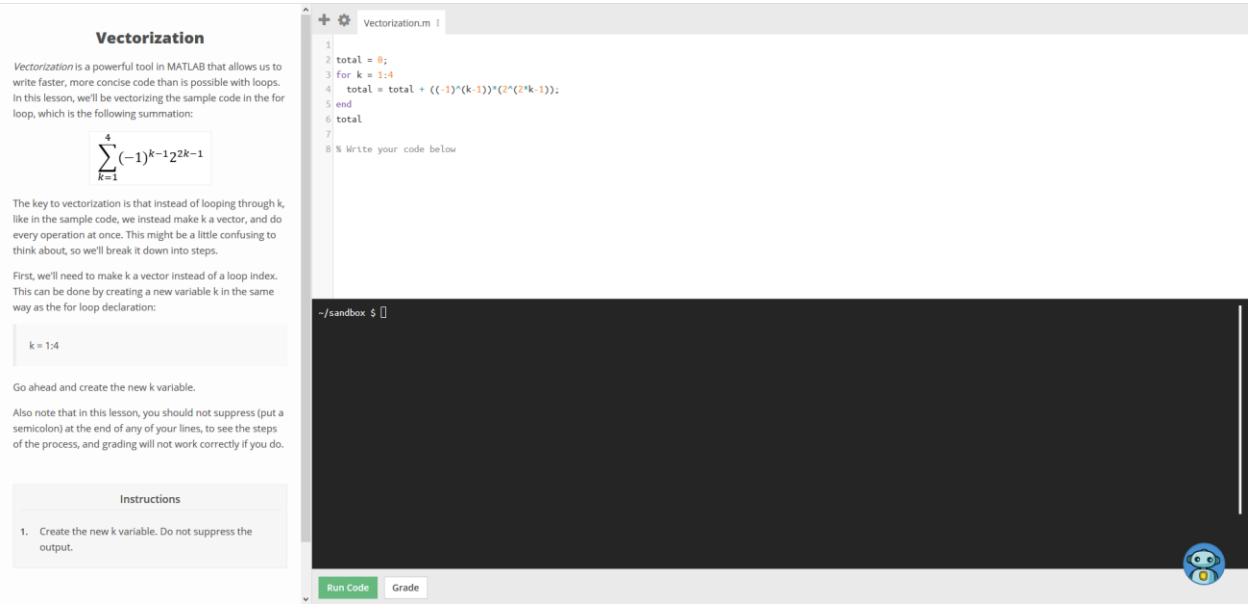

Figure 1: The layout of a generic lesson within the codevolve.com website.

Various parts of the assignments layout are shown in figures 2 – 5. Commands can be typed in an editor window, Figure 2, exactly how they would be entered into an .m file. The editor window runs in the browser, no MATLAB software needed. The results display in the terminal, Figure 3.

```
1
2 total = 0;
3 for k = 1:4
4   total = total + ((-1)^(k-1))*(2^(2*k
5 end
6 total
7
8 % Write your code below
9 k = 1:4
```

Figure 4: Code Editor Window

```
total = -102
k =

   1   2   3   4
```

Figure 3: Command Terminal Window

Vectorization

Vectorization is a powerful tool in MATLAB that allows us to write faster, more concise code than is possible with loops. In this lesson, we'll be vectorizing the sample code in the for loop, which is the following summation:

$$\sum_{k=1}^{4}(-1)^{k-1}2^{2k-1}$$

The key to vectorization is that instead of looping through k, like in the sample code, we instead make k a vector, and do every operation at once. This might be a little confusing to think about, so we'll break it down into steps.

First, we'll need to make k a vector instead of a loop index. This can be done by creating a new variable k in the same way as the for loop declaration:

```
k = 1:4
```

Go ahead and create the new k variable.

Also note that in this lesson, you should not suppress (put a semicolon) at the end of any of your lines, to see the steps of the process, and grading will not work correctly if you do.

Figure 2: Lesson instruction and general information

The left side of the interface layout shows the blocks of steps for the assignment. As shown in Figure 4, the student can read the lesson information and note the purpose of the block. The goal of the assignment is listed as an instruction, Figure 5, and tells the user specifically what needs to be done in order to complete the block and move on.

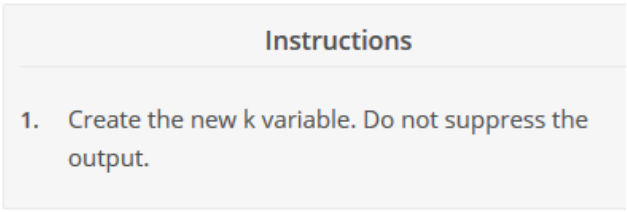

Figure 5: The instruction shown to the user that is to be completed.

If the student types an incorrect entry, a small codey bot in the bottom corner informs him or her of their error and provides a hint or suggestion as to how to fix the answer, Figure 6. If, however, the student enters the correct answer, the bot displays a 'Good work!' message. The user is then able to move to the next code block or end the activity.

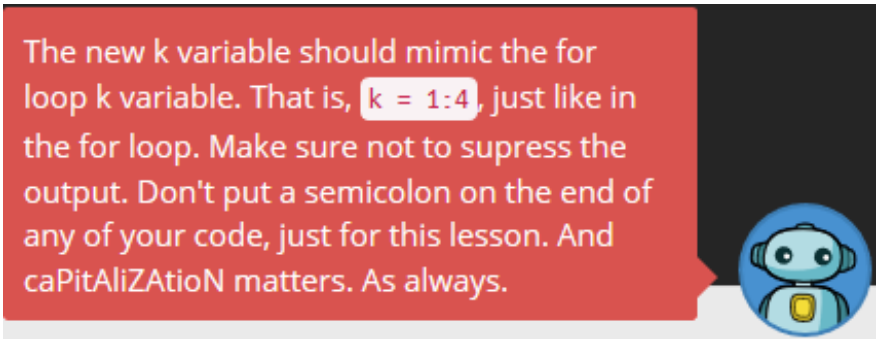

Figure 6: Example output from the helper bot when the answer is wrong. This mechanism allows for more than simply right and wrong to be displayed.

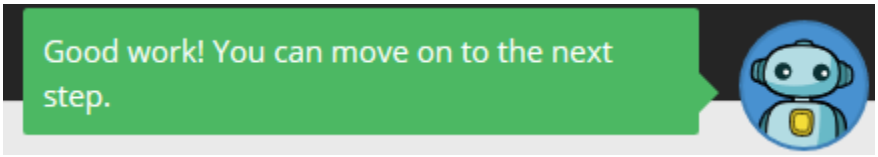

Figure 7: Example output of the helper bot when the answer is correct.

Discussion

The supplemental lessons were created in the spring semester of 2017 and offered as a beta test to a single section of 29 students. At the time, only 5 activities existed, covering multiple topics. This first group of students was offered the option of performing the outside activities. At the end of the semester, a survey was given to gauge student response to the activities.

The student survey questions are as follows. Note that the coding activities were referred to as "reading supplements" at the time.

1. I find the format of the reading supplements helpful to the way that I learn.
2. I feel that the reading supplements engage my interest.
3. I find that the reading supplements help me in understanding the readings and lectures.

4. I find that the reading supplements confuse me.
5. I would learn more if the reading supplements were more structured
6. The problems worked in the reading supplements help me in working other problems on my own.
7. I feel that I need more guidance in the reading supplements
8. I find it helpful to get feedback from the reading supplements.
9. I feel that the reading supplements should be given prior and not after the material is covered in class.
10. I think that I would learn better if a different format were used for the reading supplements (suggested below).

Students rated each question on a 5 to 1 scale with the rating ranging from: strongly agree, agree, neutral, disagree, and strongly disagree. The results of this first survey are as shown in Table 4. However, it should be stated that half of the class only performed one or two of the activities. Of those that completed all the activities, the results showed a much more positive benefit. The evidence suggests the previously-mentioned disparity in prior knowledge may have influenced the more intermediate students to only complete a few of the assignments if they felt they already knew the material.

Table 4: Results from student survey performed during the spring 2017 course.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Overall Mean	3.85	3.19	4.12	2.28	2.92	3.58	2.15	3.80	3.65	2.68
Completed All Mean	4.60	3.80	4.40	1.80	2.60	3.80	2.20	4.40	3.20	2.40

Some of the comments offered by the students regarding the supplemental activities also provide insight into why some may have performed all of assignments while others performed only one or two.

The question posed to students was the follow: “What aspects of the activity need improvement / any suggestions?”

Student responses:

- “They are a good learning tool but without a grade or any consequences most students (me) won't do too many of them.”
- “Less help, make it a tiny bit harder. …”
- “Not really, they're great as is.”
- “I like it, it allows using learned material sooner than using it on the homework. Keeping it fresher in my mind.”
- “I like the reading supplements and think it is nice to see the information presented twice, sometime in different ways.”

These responses provided enough information to suggest that the activities may be useful to some, if not all, if formatted the right way.

Over the summer, the assignments were revised and edited. Changes included shortening the activities, covering just one topic at a time, and improving the feedback and reliability of the code checking. The result of this revision was a set of eleven activities covering a large portion of the course material listed previously.

In the Fall of 2017, the supplemental activities were evaluated using two sections of the same EGR 115 course. The assignments were presented as mandatory work for Section 2 and Section 1 was taught without offering the assignments. In Section 2, the activities were assigned after the corresponding material was first covered in the lecture. The students in both sections took the same exams and quizzes and solved the same homework problems.

In both sections, the students took a pre-class survey to determine previous programming experience. Observations of in-class activities and active learning portions of the course were noted by the instructor and the instructor's TA. Even from early in the course, the instructor noticed that the section that received the extra activities (Section 2) needed less help in-class during the active learning sessions. The question remained as to whether it would affect the student scores or not. The results of the two sections test scores are shown in Table 5.

Table 5 Results from each class overall, all student are included

	Exam 1	Exam 2	Final
Section 1	71.391	72.375	76.537
Section 2	72.241	70.553	74.019
Difference in averages	-0.850	1.822	2.518

Table 6 Results from those with no prior experience in programming

	Exam 1	Exam 2	Final	Midterm Grade	Final Grade
Section 1	71.577	73.769	76.423	76.264	77.177
Section 2	69.692	68.769	69.000	70.984	69.998
Difference in averages	1.615	5.000	7.423	5.280	7.179

The overall results of both sections show a marginal difference in the average scores. For those students without prior programming experience, shown in Table 6, the first exam scores are similar with a 1.6-point difference but the other scores between the two sections are quite different. I believe the supplemental activities help beginner students, but further revisions are needed.

Since most of the out-of-class learning activities are concentrated at the beginning of the semester, it could be argued that these assignments helped those students who needed help. But, as the semester continued on and new concepts were regularly introduced (most topics later in the semester did not have corresponding out-of-class activities), the students could not keep up with the pace of the course. Though the scores are low, it should be noted that that Section 1 had the highest final average score of any 20+ student class the instructor previously taught.

Future Work

While the results of the test scores did not increase dramatically, the qualitative information obtained from students was generally positive. The students felt the activities were helpful and allowed them to keep up with the class. By attaching even the smallest of grade weight to out-of-class assignments, the students were motivated to keep up with the supplemental activities.

While improved student scores would be welcomed by student and instructor alike, ultimately the goal is to encourage the students to engage with the material and be able to participate during in-class active learning activities. Such supplemental out-of-class activities do further this goal and the immediate feedback presented to the students through Codevolve proves much more useful than assigning homework problems which take time to grade and return. During the waiting period for corrected homework, lost students will continue to fall behind and ultimately not be able to keep up with the course.

It is the author's desire to continue to use these activities, and perform further assessment of student scores and perceptions in order to reach the goal of maximum student understanding and participation in active learning classroom activities.

Bibliography

(1) Pickering, James D., and David JH Roberts. "Flipped classroom or an active lecture?." Clinical Anatomy 31.1 (2018): 118-121.
(2) Gannod, Gerald C., Janet E. Burge, and Michael T. Helmick. "Using the inverted classroom to teach software engineering." Proceedings of the 30th international conference on Software engineering. ACM, 2008.
(3) Roberts, Jonathan C., et al. "The Explanatory Visualization Framework: An active learning framework for teaching creative computing using explanatory visualizations." IEEE transactions on visualization and computer graphics 24.1 (2018): 791-801.
(4) Princy, L., et al. "Effectiveness of Classroom Interactions in Engineering Colleges." Ergonomic Design of Products and Worksystems-21st Century Perspectives of Asia. Springer, Singapore, 2018. 163-175.
(5) Arakaki, D. (2017, April), Lecture Videos to Supplement Electromagnetic Classes at Cal Poly San Luis Obispo Paper presented at 2017 Pacific Southwest Section Meeting, Tempe, Arizona. https://peer.asee.org/29222
(6) Prince, M. (2004), Does Active Learning Work? A Review of the Research. Journal of Engineering Education, 93: 223–231.
(7) http://www.katyjordan.com/MOOCproject.html