Using Codecademy Interactive Lessons as an Instructional Supplement in a Python Programming Course

Jason H. Sharp
jsharp@tarleton.edu
Marketing and Computer Information Systems
Tarleton State University
Stephenville, TX, 76402, USA

Abstract

With the recent renewed interest in programming, online learning environments like Codecademy have become quite popular, boasting some 25 million members worldwide. The purpose of this paper is to describe the author's experience using Codecademy Interactive Lessons as an instructional supplement in an introductory Python programming course. The paper provides a brief background of the literature, a description of how the author implemented the interactive lessons, a discussion of the positives and negatives, the extent to which the interactive lessons met the course skill outcomes, and conclusions about the overall experience. In sum, the Codecademy Interactive Lessons fulfilled 3 of the 6 course skill outcomes and overall, the positives outweighed the negatives.

Keywords: Codecademy, Python programming, Interactive lessons, Online learning environments

1. INTRODUCTION

"Teaching the world how to code" ~ Codecademy

Codecademy provides free, online, interactive lessons for a variety of programming topics. Founded in 2011 by Zach Sims and Ryan Bubinski (Codecademy, n.d.), Codecademy purports to have 25 million learners around the world. According to their web site, Codecademy is "an education company" and the goal is to make Codecademy "the best place for our team to learn, teach, and create the online learning experience of the future" (About, n.d., para. 1). Because, "education is broken" (About, n.d., para. 4), Codecademy considers itself a disruptive force for "building the education the world needs – the first truly net native education" (About, n.d., para. 3). As the Codecademy web site asserts, "come help us build the education the world deserves" (About, n.d., para. 4).

The purpose of this paper is to describe the author's experience using Codecademy Interactive Lessons as an instructional supplement in an introductory Python programming course. The paper provides a brief background of the literature, a description of how the author implemented the interactive lessons, a discussion of the positives and negatives, an evaluation of the extent in which the interactive lessons meet the course skills outcomes, and conclusions about the overall experience. The focus of the paper centers on the implementation, evaluation, and fulfillment of course skill outcomes.

2. BACKGROUND

While there is a growing stream of research related to online learning environments in general (e.g., Herrington, Oliver & Reeves, 2003; Huang, 2002; Johnson & Aragon, 2002; Michinov, Brunot, Le Bohec, Juhel, & Delaval, 2011; Oncu &

Cacir, 2011) and smart learning content (e.g. Brusilovsky et al., 2014), studies specifically focused on the use of Codeacademy as an instructional supplement are lacking. The following are representative studies of the use of Codecademy for teaching and learning.

Kim and Ko (2017) included Codecademy in their study of online coding tutorials. They argue that the research investigating online learning environments for programming is both "sparse" and "narrow" resulting in "little holistic guidance about how to choose effective tutorials . . ." (p. 321). Based upon a set of curriculum design dimensions the authors identified four pedagogical principles to form the basis of the study including: (1) connecting to learners' prior knowledge, (2) organizing declarative knowledge, (3) practice and feedback, and (4) encouraging meta-cognitive learning. From these four guiding principles, the authors developed nine groupings by which to analyze the data collected. The nine groupings consisted of: (1) personalization, (2) utilization, (3) contents, (4) organization, (5) context, (6) actionability, (7) feedback, (8) transfer learning, and (9) support. Within these nine groupings, the authors derived 24 pedagogical principles specifically related to coding tutorials. In all but one of the principles, the authors marked them either yes or no. Across the 23 pedagogical principles Codecademy received 20 "yes" responses that it met the criteria. The authors concluded that "most online coding tutorials are still immature and do not yet achieve many key principles in learning sciences" (p. 325).

In a study exploring the design of online learning environment for programming education Olsson and Mozelius (2016) analyzed Codeacademy and MyProgrammingLab by asking the following question, "what are the most important factors in the design of virtual learning environments for self-learning of fundamental skills and knowledge"? (p. 534). They suggested that both "seem like promising additional tools for self-learning in programming courses at the university level" (p. 94). Using a case study research method they collected data via interviews, questionnaires, and group discussions. According to the authors the student's overall experience with Codecademy was positive. The immediate feedback, in particular, was cited by a students as a major benefit. Other features of Codecademy that students liked included the structure of the lessons and the forum. In sum, the authors identified the most important design factors as follows: (1) usability and user-friendliness, (2) clear and well-formulated feedback, (3) gamification, (4) unambiguous exercises, (5) GUI design and multi-modality, and (6) curriculum alignment.

In an attempt to increase student engagement and performance in a Fundamentals of Software Development course teaching Python programming, Fotaris, Mastoras, Leinfellner, and Rosunally (2016) gamified the course using the Kahoot! Classroom Response System, which is a game-based learning and trivia platform, the classroom version of the TV game show "Who Wants To Be A Millionaire?", and Codecademy's Python programming course. To implement the gamification strategy the authors replaced the traditional one-hour topical lectures with three 20-minute micro-lectures each followed by a Kahoot! session in which students responded to questions. The typical review session comprised of question and answer time was replaced with the "Who Wants To Be A Millionaire?" game consisting of Python-related questions. Finally, Codecademy's Python programming course was used for students to practice coding. For each of these components points were awarded and a leaderboard was provided in Blackboard. To gather data the authors observed student behavior, conducted an online survey, gathered self-reported data from students, and synthesized classroom administrative data such as attendance, tardiness, use of material, completion rates, and academic performance. Results of the study indicated an overall positive response from students to the gamified approach to the course, completion rates of assignments increased slightly, and overall student academic performance increased by about 8%.

Lee and Ko (2015) included the Codecademy Python course in a larger study designed to examine whether novice programmers "produced measurable learning outcomes" after using three different online learning tools. In addition to the Codecademy Python course, the tools included were Gidget and Gidget Puzzle Designer. Each of these tools represented a different form of online learning environment which Lee and Ko identified as tutorial, game, and canvas. Using a pretest-posttest research design the authors hypothesized that there would be "no difference in learner's post-test performance among the conditions after completing their assigned learning activity" (p. 238). Overall, the study indicated that none of the online learning environments resulted in statistically significant differences in student performance. However, there were statistically significant results in student performance between Codecademy and Gidget Puzzle Designer on the posttest, indicating

that structured tutorials may improve student knowledge over non-structured environments.

Figueroa and Amoloza (2015) incorporated three online interactive platforms into a multimedia course for non-computer science majors to study the impact of these platforms on programming anxiety and perceived learning. The platforms included Blockly, Code.org, and Codecademy. In this particular study, Codecademy's JavaScript programming course was used. Students were administered the Programming Anxiety Survey, consisting of six questions, before taking the course and after taking the course. The analysis of the survey data indicated a statistically significant difference between the average scores before and after taking the course. Furthermore, data collected also indicated a positive result among students in terms of perceived learning. The authors conclude that the combination of the three online interactive platforms resulted in "a significant decrease in learning anxiety and an increase in perceived learning among students who took the course" (p. 65).

3. IMPLEMENTATION

Codecademy Teaching Resources

As a part of its educational strategy Codecademy provides several teaching resources including teacher training, class resources, and classroom tracking. Teacher training allows instructors to go through the same interactive lessons as the students free of charge. Class Resources include free lesson plans and quizzes. Classroom Tracking allows the instructor to create student accounts and to track individual performance by overview and by unit. The tracking allows for the instructor to see the percentage of each individual course completed.

Course Requirements and Outcomes

While planning to teach Python programming for the first time, the author decided to implement the interactive Python lessons provided by Codecademy. The idea of these free, online, interactive lessons was appealing to the author as an additional means to potentially engage students beyond the traditional textbook materials. He was curious to see how the student's would respond to the interactive nature of the lessons and see if the students thought they were a worthwhile activity in addition to the customary quizzes, exams, and assignments.

The course itself was offered online in an 8-week summer session via Blackboard Learn 9. A total of thirty students were enrolled in the course. The majority of students were either Computer Information Systems or Information Technology majors (25 out of 30). The course consisted of 26 men and 4 females.

Rather than offer the interactive lessons as an optional supplement for which the students could complete or not complete, the author decided to require the interactive lessons as a part of the course requirements constituting 10% of the overall course grade. This decision was made to motivate the students to complete the interactive lessons. A breakdown of the course requirements and percent of course grade is provided in Table 1. A list of the knowledge and skill outcomes is provided in Appendix A.

Course Requirements	**%**
Lab Assignments	35%
Codecademy Interactive Lessons	10%
Quizzes	10%
Exams (2)	30%
Final Exam	15%

Table 1. Breakdown of Course Requirements

Topics Covered

The textbook for the course was "Starting Out with Python Programming" (Gaddis, 2018). Because the course was taught in an 8-week summer session the author covered the first six chapters: (1) Introduction to Computers and Programming, (2) Input, Processing, and Output, (3) Decision Structures and Boolean Logic, (4) Repetition Structures, (5) Functions, and (6) Files and Exceptions. While the Codecademy Python course consists of 21 individual courses covered in 36 lessons the author selected those courses which matched the content of the textbook chapters: (1) Python Syntax, (2) Tip Calculator, (3) Strings & Console Output, (5) Conditionals & Control Flow, (7) Functions, and (14) Loops (See Appendix B). Additional courses were available to provide students an opportunity to apply the concepts from the main courses. For each textbook chapter the associated interactive lessons were provided on the course schedule (See Appendix C).

4. POSITIVES AND NEGATIVES

Positives

The author identified several positive aspects of implementing the Python interactive lessons as an instructional supplement. First, and perhaps most obvious, the interactive lessons are free. With the rising cost of traditional textbooks and the additional expense of adding publisher's interactive content (e.g., MyProgrammingLab) they provide an easily accessible, no-cost

alternative which is quite attractive to both instructors and students.

Second, the interactive lessons are self-paced and students can repeat the individual courses as many times as they wish. If the student is having difficulty with a particular topic they can spend as much time with it as needed. Additionally, students can access the content at their convenience and do not need to install special software or have lab access. With Internet access and a browser the student is good to go.

Third, because of the interactive nature of the lessons student receive immediate feedback on the code that they are writing. It is no surprise to instructors that today's students prefer hands-on activity over reading a textbook or passively listening to a lecture. The author found that the feedback provided by the interactive lessons was user-friendly and provided enough guidance to scaffold the learning experience and help to solve logical or syntactical errors.

Finally, from the author's perspective, setting up and managing a Codecademy course via the Classroom Tracking interface was quite simple and intuitive. Basically, the instructor chooses the course they want to use and then can customize its name and description to match the course syllabus. The instructor then adds the students to the course and a username and password is automatically created for each student. Students can be added, edited, and deleted at any time. An easy-to-follow "Pupil Tracker Guide" is provided by Codeacademy. The students can then login and change these items if they choose. As students complete individual courses the tracking interface displays an overview of each student's progress as well as individual performance by lesson. The author then entered the completion percentage into the gradebook in Blackboard. The performance matrix can also be downloaded as a comma-separate values (.csv) file and opened and edited in Excel.

Negatives

The implementation of the Python interactive lessons was not without its negatives. As with any interactive coding environment there is limited opportunity for creativity by the students since the "solutions" are predefined. The downside of this approach is that students only see potentially one way of solving a problem – they are not allowed to think "outside of the box". Another possible downside is that they simply employ a trial-and-error approach to problem solving until they receive the correct answer rather than enlisting critical thinking skills. As with any instructional strategy students may simply rush through the interactive lessons to get them completed rather than taking their time to learn, understand, and apply the content.

From the author's perspective, there were actually very few negatives from the standpoint of creating and managing the Python course in Codecademy. It would have been nice if the students were automatically notified that their accounts were created and what their username and password was rather than the author having to send an individual message to each student in Blackboard. The fact that the author had to manually enter the percent completed values from the tracking system to the Blackboard gradebook was also a bit time-consuming.

5. EVALUATION OF SKILL OUTCOMES

In addition to identifying the positives and negatives, the author evaluated the use of Codecademy as an instructional supplement in terms of meeting the course skill outcomes (see Appendix A).

SO1: Students will create Python programs using the Python interpreter and the IDLE IDE

Because the Python lessons are embedded within the Codecademy online, interactive environment a specific interpreter and/or IDE is not used. This skill outcome was met outside of Codecademy using the Python interpreter and IDLE IDE provided on the Python website.

SO2: Students will apply the steps in the program development process

The program development process followed was that provided by Gaddis (2017): (1) Design the program, (2) Write the code, (3) Correct syntax errors, (4) Test the program, and (5) Correct logic errors. This skill outcome is partially met using the interactive lessons. The structure of the majority of the interactive lessons is to provide students with a prompt to write a single line of code and provide immediate feedback or to provide students with partial code for which they complete. Students are not required to design and write a program from start to finish. Some may find this as a shortcoming of the interactive lessons as they provide only partial snippets of code to be completed, rather than working through the full program development process.

SO3: Students will implement variables, literals, and constants

The interactive lessons provide students the opportunity to implement variables, literals, and

constants. Students are required to declare variables, literals, and constants and assign appropriate values to them. These exercises are provided in Lesson 2 - Python Syntax, Exercises 10-13; Lesson 3 - Tip Calculator, Exercises 1-5; and Lesson 4-5 - Strings & Console Output, Exercises 1-13.

SO4: Students will select appropriate arithmetic, logical, and relational operators
The interactive lessons provide students the opportunity to select appropriate arithmetic, logical, and relational operators. These exercises are provide in Lesson 2 - Python Syntax, Exercises 10-13; Lesson 3 - Tip Calculator, Exercises 1-5; Lesson 7 - Conditionals & Control Flow, Exercises 1-10.

SO5: Students will implement sequence, selection, and repetition control structures
The interactive lessons provide students the opportunity to implement sequence, selection, and repetition structures. These exercises are provided in Lesson 2 - Python Syntax, Exercises 10-13; and Lesson 9 – Conditionals & Control Flow, Exercises 11-15; Lesson 24-25 – Loops, Exercises 1-19.

SO6: Students will analyze, design, implement, test, and debug domain-specific applications which demonstrate basic computation, input/output, control structures, operators, exception handling, and functions
The interactive lessons partially provide students the opportunity to analyze, design, implement, test, and debug domain-specific applications demonstrating basic computation, input/output, control structures, operators, and functions. Coverage of functions is provided in Lesson 11-12 – Functions, Exercises 1-19. As noted in SO2, the design of the interactive lessons lack the ability for the students to create full programs from scratch, instead providing partial code snippets for completion.

In sum, the Codecademy interactive lessons met three skill outcomes (SO3, SO4, and SO5), partially met two skill outcomes (S02, S06) and did not meet one skill outcome (S01). See Table 2 for a summary of the evaluation of the skill outcomes as either met, partially met, or not met.

Outcome	Met	Partially Met	Not Met
SO1			X
SO2		X	
SO3	X		
SO4	X		
SO5	X		
SO6		X	

Table 2. Evaluation of Skill Outcomes

6. CONCLUSION

Overall, the author's experience with the Codecademy interactive lessons for Python was positive in terms of an instructional supplement to the textbook materials. The interactive lessons provided adequate depth and breadth of the Python syntax and allowed students additional coding practice with immediate feedback in an environment conducive to their own schedule and learning speed. Additionally, the interactive lessons met or partially met five of the six course skill outcomes. Another upside being that the students were afforded this opportunity with no additional cost to the course. For those perhaps interested in implementing one of the interactive lessons the management is simple and intuitive and is not a significant addition of time commitment to the instructor. While anecdotal comments might be made from the student's perspective at this point, a potential opportunity for future research is to survey students on their attitudes toward the benefits and challenges of the interactive lessons and to correlate completion percentages with overall course grade.

7. REFERENCES

About. (n.d.) Retrieved from https://www.codecademy.com/about

Brusilovsky, P., Edwards, S., Kumar, A., Malmi, L., Benotti, L., Buck, D., . . . Wollowski, M. (2014). Increasing adoption of smart learning content for Computer Science education. *Proceedings of the Nineteenth Annual Conference on Innovation and Technology in Computer Science Education*, 1-27.

Codecademy. (n.d.). In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Codecademy

Figueroa, R. B., & Amoloza, E. M. (2015). Addressing programming anxiety among non-computer science distance learners: A UPOU case study. *International Journal for*

Educational Media and Technology, *9*(1), 56-67.

Fotaris, P., Mastoras, T., Leinfellner, R., & Rosunally, Y. (2016). Climbing up the leaderboard: An empirical study of applying gamification techniques to a computer programming class. *The Electronic Journal of e-Learning*, *14*(2), 94-110.

Gaddis, T. (2017). Starting Out with Python. Pearson, Boston.

Herrington, J., Oliver, R., & Reeves, T. C. (2003). Patterns of engagement in authentic online learning environments. *Australian Journal of Educational Technology*, *19*(1), 59-71.

Huang, H. M. (2002). Toward constructivism for adult learners in online learning environments. *British Journal of Educational Technology*, *33*(1), 27-37.

Johnson, S. D., & Aragon, S. R. (2002). An instructional strategy framework for online learning environments. *Proceedings of the Academy for Human Resource Development*, 1022-1029.

Kim, A. S., & Ko, A. J. (2017). A pedagogical analysis of online coding tutorials. *Proceedings of SIGCSE*, 321-326.

Lee, M. J., & Ko, A. J. (2015). Comparing the effectiveness of online learning approaches on CS1 learning outcomes. *Proceedings of the Eleventh International Computing Education Research Conference*, 237-246.

Michinov, N., Brunot, S., Le Bohec, O., Juhel, J., & Delaval, M. (2011). Procrastination, participation, and performance in online learning environments. *Computers & Education*, *56*(1), 243-252.

Oncu, S., & Cakir, H. (2011). Research in online learning environments: Priorities and methodologies. *Computers & Education*, *57*(1), 1098-1108.

Olsson, M., & Mozelius, P. (2016). On design of online learning environments for programming education. *Proceedings of the European Conference on e-Learning*, 533-539.

Appendix A - Knowledge and Skill Outcomes

Knowledge Outcomes:

- Students will become familiar with the Python interpreter and the IDLE Integrated Development Environment (IDE)
- Students will describe the steps in the program development process
- Students will explain the characteristics of variables, literals, and constants and their appropriate usage
- Students will distinguish between arithmetic, logical, and relational operators and their appropriate usage
- Students will identify and describe sequence, selection, and repetition control structures
- Students will describe exception handling
- Students will understand the benefits of modularization and the use of functions

Skill Outcomes:

- Students will create Python programs using the Python interpreter and the IDLE IDE
- Students will apply the steps in the program development process
- Students will implement variables, literals, and constants
- Students will select appropriate arithmetic, logical, and relational operators
- Students will implement sequence, selection, and repetition control structures
- Students will analyze, design, implement, test, and debug domain-specific applications which demonstrate basic computation, input/output, control structures, operators, exception handling, and functions

ISSN: 2473-3857
v4 n4615

Appendix B - Overview of Codecademy Python Lessons Used*

Lesson	Course	Exercises	Objectives
1	1. Python Syntax	1-9	• Become familiar with Codecademy platform • Understand why Python is used and recognize basic terminology including 'variables' and 'Boolean' • Understand and create whitespace and multi-line comments
2	1. Python Syntax	10-13	• Perform mathematical operations using python syntax • Create numbers using 'modulo' • Practice creating comments, variable and arithmetic operations
3	2. Tip Calculator	1-5	• Plenary activity synthesizing lessons 1&2: Python syntax • Create a 'tip calculator' using python syntax, variables and arithmetic operations
4	3. Strings & Console Output	1-9	• Explain what a string is and how to create one • Create variables using indexing • Implement lower(), upper() and str() string methods • Compare when dot notation should be used
5	3. Strings & Console Output	10-13	• Demonstrate how to print strings and variables including how to concatenate • Explain how to convert a non-string into a string and why you would need to • Demonstrate how to use the % operator
7	5. Conditionals & Control Flow	1-4	• Understand what control flow is • Recognize and practice using 6 comparators (==, !=, <=, >=, <, >) • Explain what a comparator is
8	5. Conditionals & Control Flow	5-10	• Recognize 3 types of Boolean operations (AND, OR, NOT) • Demonstrate how to use Boolean operations to return 'True' or 'False' values
9	5. Conditionals & Control Flow	11-15	• Recognize IF, ELSE and ELIF statements • Create simple controlled flows using IF, ELIF and ELSE statements • Practice creating control flow with conditionals and Boolean operations
11	7. Functions	1-11	• Demonstrate and understand how to define a function with and without parameters • Demonstrate and understand how to call functions • Demonstrate importing functions both specific and universal • Practice creating functions
12	7. Functions	12-19	• Demonstrate and understand what the max, min, abs and type functions do • Practice making functions
24	14. Loops	1-8	• Understand how a While/ Else loop functions • Understand how to prevent an infinite loop • Create while loops integrated with lists, inputs and mathematical operators
25	14. Loops	9-19	• Plenary: Practice making loops using the correct syntax • Understand how a For/ Else loop works • Create a For/ Else loop

*Adapted from Codecademy Python Unit Overview

ISSN: 2473-3857
v4 n4615

Appendix C – Course Schedule

Date	Tentative Schedule	Assignment Due*	Quizzes**	Codecademy Lessons***
Week 1 June 12-18	Course Introduction Chapter 1 - Introduction to Computers and Programming			
Week 2 June 19-25	Chapter 2 - Input, Processing, and Output	Lab 01*	Ch 00**	
Week 3 June 26-July 2	Chapter 3 - Decision Structures and Boolean Logic	Lab 02*	Ch 02**	Python Syntax & Tip Calculator Strings & Console Output
Week 4 July 3-9	**Exam 1 (Chapter 1-3)**	Lab 03*	Ch 03**	Conditionals & Control Flow
Week 5 July 10-16	Chapter 4 - Repetition Structures	**Exam 1****		
Week 6 July 17-23	Chapter 5 - Functions	Lab 04*	Ch 04**	Loops
Week 7 July 24-30	**Exam 2 (Chapter 4-5)**	Lab 05*	Ch 05**	Functions
Week 8 July 31-Aug 6	Chapter 6 - Files and Exceptions	**Exam 2****		
Week 9 Aug 7-13	**Final Exam**	Lab 06* **Final Exam****	Ch 06**	