2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #39274

Evaluating Self-paced Computational Notebooks vs. Instructor-Led Online Lectures for Introductory Computer Programming

Mr. Timothy James, Purdue University, West Lafayette

Timothy James is an instructor at the University of Pittsburgh Computer Science department, as well as a Doctor of Technology student at Purdue University. Previously, Tim has spent some time in a variety of industries including Internet consulting, finance, defense contracting, aviation maintenance, telecommunications, capital markets, and sandwich artistry. Timothy hopes to continue actively engaging the community in technical training and CS education efforts.

Dr. Alejandra J. Magana, Purdue University at West Lafayette

Alejandra J. Magana, Ph.D., is the W.C. Furnas Professor in Enterprise Excellence in the Department of Computer and Information Technology with a courtesy appointment at the School of Engineering Education at Purdue University. She holds a B.E. in Informa

Evaluating Self-paced Computational Notebooks vs. Instructor-Led Online Lectures for Introductory Computer Programming

Abstract

Teaching a new programming language to computer science students is challenging, time consuming, and fraught with error. Students face many challenges while attempting to learn a new language. Allowing students the opportunity to gain confidence in their ability by quickly learning coding and applying introductory programming concepts could help them to master these concepts more quickly and defer programming environment set up to a later date. Accelerating the learning of programming may help to offset future needs, as growth in software development jobs is projected to significantly outpace growth in instructional jobs in computer science. This study implemented two versions of the same course content in self-paced and instructor-led formats. The instructor-led format included the delivery of online lectures combined with examples and practice exercises. This was considered a more traditional approach where students were taught using slides, lectures, examples, and assignments. Participants in this study were recruited for a free introductory Python course through LinkedIn and Twitter. Participants were randomly assigned either to the instructor-led or the self-paced versions of the course. It appears that based on the scores and lower attrition rates, a student-driven approach using Colab notebooks is at least approximately as effective in helping students learn the concepts.

1. Introduction

The supply of workers capable of performing effectively in software development is not keeping up with industry demand; unfortunately, the supply of instructors capable of training those future software developers is also likely to fall short of what is necessary. Growth in software development jobs is projected to significantly outpace growth in instructional jobs in computer science through 2030 [1]; current demand is already much higher than supply by hundreds of thousands [2] which will likely attract even more software engineers to the field. The educational requirements for professors in computer science are typically much higher than for software developers, while the pay remains lower. Thus, scalable methods of delivering effective computer science education will be necessary to enable fewer instructors to reach a larger audience.

However, to teach programming at scale, one must consider the challenges in teaching computer science and the learning difficulties students experience. Cognitive load is high in computer science due to the challenges inherent in the topic and material - beyond the problem itself, there are many things to keep track of, such as variables and program context [3]. The intrinsic cognitive load includes the high level of difficulty of the material being learned. Extraneous cognitive load is what is imposed on the learner by factors outside of these - such as poorly authored resources. In addition, much of the process of getting started can be cryptic and difficult to understand; environment setup alone can prevent many students from even starting to write code [4].

The goal of this research is to compare a self-paced teaching method using computational notebooks to a more traditional instructor-led teaching method using lectures, explanations, and examples. These approaches were evaluated through a comparison of aggregate performance measures of students taught through each method. Data was collected through the delivery of these teaching methods to support answers to the following questions: (1) What are differences in student *engagement* when a course is delivered via a self-paced vs. an instructor-led approach? (2) What are differences in student *performance* when a course is delivered via a self-paced vs. an instructor-led approach? And (3) What are differences in student perceptions of their learning when a course delivered via a self-paced vs. an instructor-led approach?

2. Background

Roll, Russell, and Gašević define learning at scale as "the study of the technologies, pedagogies, analyses, and theories of learning and teaching that take place with a large number of learners and a high ratio of learners to facilitators" [5]. Massive Open Online Courses (MOOCs) are a common approach to learning at scale. These programs have faced some justified skepticism. Aleven et al. cast doubt on a typical MOOC approach: showing a sequence of videos and then completing a quiz, as this separates content and the opportunity to apply that content. Interspersing applications with instruction more frequently may improve learning [6]. Attrition is high in MOOCs, overall at 90% [7]. MOOCs may have high dropout rates, but there is an argument that they provide broader access to education [8]; MOOCs are not likely to supplant traditional educational experiences, but they can address rising demand and fulfill many functions. Roll et al. suggest that workplace skill development/training, learning for

entertainment or leisure, improving the reach of education and access to educational resources, and improving knowledge of technology and science may be well addressed by MOOCs [5].

Beyond the delivery of content at scale, concepts can be another challenge in computer science - between following program logic, context, and variables - not to mention the problem being solved - cognitive load can be high [3]. Representing these concepts in ways that may be easier to understand can also be a challenge; many students learn visually but many concepts in programming do not translate to visual representation easily [9]. The literature suggests that reducing cognitive load in the learning process should be a goal for instructors, to allow students the necessary room to focus on the content being learned: "As educators, we want to simplify the learning process to provide the maximum results" [3]. Online learning environments present learners with new challenges - tools, platforms, and mechanisms that may be unfamiliar - adding to the cognitive load of learning the topic at hand.

Educational researchers have identified pedagogical approaches and learning strategies that can support students when learning programming. For example, explanations in computer programming are important to the learning experience; encouraging students who are learning to write code to document their process can be an important learning activity, and can help to reduce the cognitive load involved in programming [10]. Worked examples are a way to demonstrate solution approaches to learners. Morrison et al. suggest using "worked example-practice pairs" [3]. This is compatible with Use-Modify-Create, a scaffolded approach that empowers students to use examples, modify the supplied code, then move on to creating their own solutions to problems [11]. Specifically, scaffolding allows instructors and other subject matter experts to provide assistance to learners so that they can achieve more than they would independently; scaffolding can be leveraged to help students solve problems and can be enhanced by technology [12].

3. Limitations

A limitation of this study is in the inherent differences between these two approaches. The self-paced approach provides instruction via a computational notebook, which allows for instructive text, examples, and executable code to be presented in the same environment (Google Colab). The instructor-led approach provides instruction via slideshows and video, examples, and executable code, but all of these are provided in different environments (Google Meet for the live videos, Google Slides for the slideshows, GitHub for the examples and executable code). While both approaches utilized the same content, the same examples, the same challenge exercises, and the same evaluations, no attempt was made to make the environments more similar. This means that the techniques of worked examples and Use-Modify-Create were utilized in the content, but the computational notebook environment may allow for better application of these techniques.

4. Instructor-led and Self-paced Conditions

This study implemented two versions of the same course content, examples, and topical coverage delivered in self-paced and instructor-led formats. The instructor-led format included the delivery of online lectures combined with examples and practice exercises. This was considered

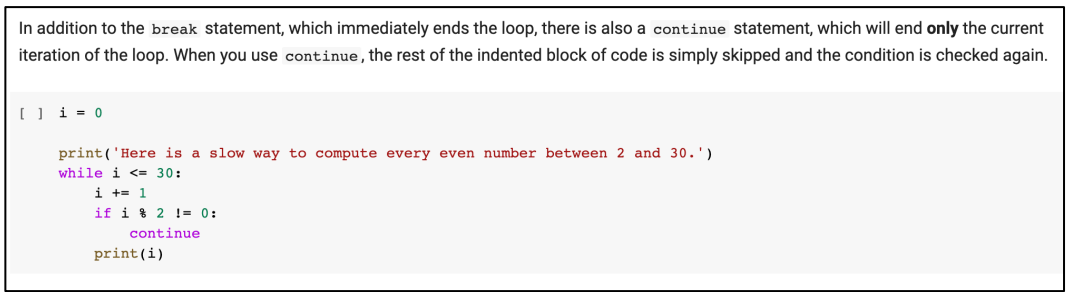
a more traditional approach where students were taught using slides, lectures, examples, and assignments.

Both courses utilized the Use-Modify-Create [11] approach, which allowed students to (1) use code or a concept provided by an instructor, (2) modify code/concepts to demonstrate changes or experiments, and (3) create new code to solve new types of problems. The Use-Modify-Create approach in the self-paced version was delivered and facilitated by using computational notebooks. Computational notebooks are web-based online interactive programming environments that allow the execution of code from a browser.

Google Colab is a hosted computational notebook service that can be used to provide description, documentation, runnable examples, and modifiable code in a single file that can be viewed through a web browser. No setup is required. Instruction and explanation can be interspersed with working, runnable, modifiable code. Colab was chosen as a tool compatible with this approach due to its ability to support pedagogy through instruction & explanations in-line with modifiable code. The interactive environment provided with computational notebooks like Colab can more directly encourage students to try new things [13].

The low requirements for setup, a structured way to convey instructional content, and ability to more closely tie instruction to code execution may make a partially automated teaching method using Colab easier for students to engage. Eliminating the requirements to install a Python interpreter, work with a text editor, write code, and run programs manually may reduce cognitive load and remove obstacles that stand in between learners and the learning objectives.

To the extent possible, the same approach and content were used in both approaches. For example, Figure 1 is section of a Colab file that covers the “continue” keyword in Python. This should closely match the delivery of content in the lecture slides and videos, as seen in Figure 2.


In addition to the `break` statement, which immediately ends the loop, there is also a `continue` statement, which will end **only** the current iteration of the loop. When you use `continue`, the rest of the indented block of code is simply skipped and the condition is checked again.

```
[ ]  i = 0

     print('Here is a slow way to compute every even number between 2 and 30.')
     while i <= 30:
         i += 1
         if i % 2 != 0:
             continue
         print(i)
```

Fig. 1. Colab content on the "continue" keyword

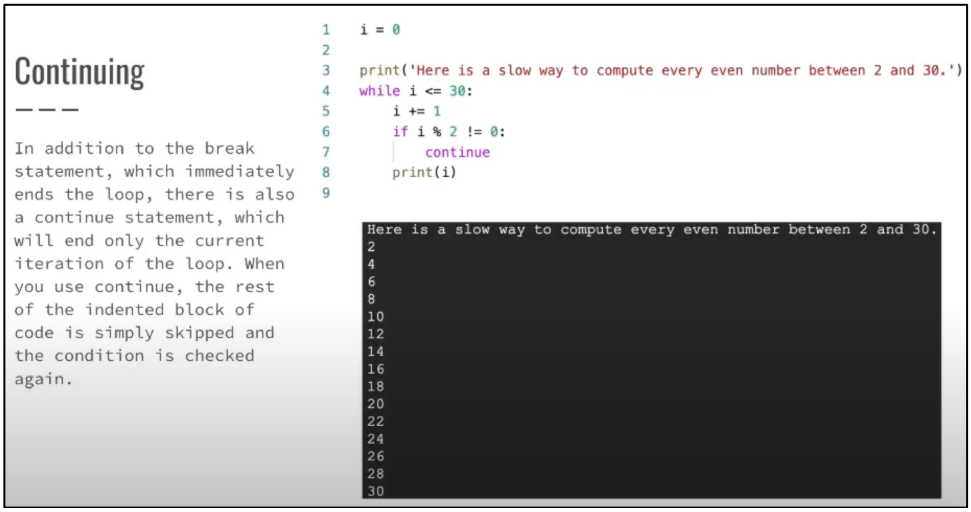


Fig. 2. Lecture slide content on the "continue" keyword

In the self-paced version of the course, an instructional Colab notebook was shared with the students two times each week (available in Appendix B). In the instructor-led version of the course, lectures were delivered two times each week via Google Meet, recorded, and shared with the students along with the slides (slides and recordings available in Appendix B). Both courses had live Q&A sessions at the end of each week.

5. Methods

This experimental study involved two versions of an introductory Python programming course using either an instructor-led approach or a learner-led approach. Each approach was delivered over a duration of four weeks. The topics for each week were as follows: week one convered introductory topics and If-Else Statements, week two introduced loops and structures, week three introduced objects and classes, and week four focused on problem solving.

5.1 Participants

The population consisted of 207 participants. The participants were recruited by posting an announcement in Linked In for registering for a free online course. According to the demographic information provided, 73.9% of the participants identified as male, 23.7% of the participants identified as female, and 2.4% of the participants identified as non-binary or did not self-identify. Regarding their nationality, 32.9% of the participants reported residing in Ghana. 28% of the participants reported residing in Nigeria, 21.7% of the participants reported residing in the U.S., 3.4% of the participants reported residing in Canada, 2.4% of the participants reported residing in Germany, and no other country was represented by 2% or more of the participants. Regarding their race, 72.9% of the participants self-identified as Black, African American, or African, 9.7% of the participants self-identified as Asian, 9.2% of the participants identified as White, 5.8% of the participants did not answer or selected "Other", no other race was represented by 2% or more of the participants. Finally, regarding age, nearly all participants (94.7%) were 35 years of age or younger. A substantial majority of all participants (81.6%) were between the ages of 21 and 35.

The participants were randomly assigned to either an instructor-led or a self-paced condition. The instrutor-led condition had a total of 104 students enrolled and the self-paced condition had a total of 103 students.

5.2 Data Collection and Data Analysis Methods

The primary data to assess students' performance were the scores from assessments that participants completed. These assessments consisted of multiple choice quizzes, programming assignments, and an examination with questions and programming problems. Beyond the scores on these assessments, simply completing the assessments is a good indicator of whether or not participants were engaged with the material; the submission of quizzes, assignments, and the exam was used to determine if students had left the course. Identifying students who have not completed the assigned evaluations can be used to measure attrition. The course also used post-course survey data to gather student confidence and perspectives on what they learned. Students were given instruction over 4 weeks. In each of the weeks, 2 multiple choice quizzes were administered, as well as a programming assignment. A final exam was administered after the completion of the 4th week of instruction.

6. Results

The results are organized into three main sections, each of them responding to the three primary research questions of the study regarding differences in student engagement, performance, and perceptions of either an instructor-led or a self-pased versions of the course.

6.1 Student Engagement

From the 104 participants who were randomly assigned to the instructor-led course, 92 students registered to participate. From the 103 participants who were randomly assigned to the self-paced course, 92 of these students registered to participate. After week 2, any participants who had not submitted any quizzes or assignments were removed from the course. This resulted in 45 students being removed from the instructor-led course (with 59 remaining) and 30 students being removed from the self-paced course (with 70 remaining). This process was the first indication of student engagement. Specifically, about 57% of the 104 students in the instructor-led course and about 73% of the 103 students in the self-paced course participated in some way by the end of week 2. This is a statistically significant result with 90% confidence. This finding suggests that more participants in the self-paced course actually participated in the first weeks of the course by submitting homework or quizzes. It is unclear if this is because it was easier to get started, or if there were other factors, but this may be a topic for future research.

As a second measure of engagement, 26 of the students in the self-paced course completed the final exam (about 37% of the 70 students that remained after Week 2); 20 of the students in the instructor-led course completed the final exam (about 34% of the 59 students that remained after Week 2).

Participation in both groups declined throughout the 4 week course - students completed fewer quizzes each week. As depicted in Figure 3, students in the self-paced course participated more through the submission of quizzes 1, 2, 3, 4, 5, and 7 than the students in the instructor-led course; about the same number of students in each course submitted quizzes 6 (23 in the instructor-led course; 24 in the self-paced course) and 8 (23 in the instructor-led course; 22 in the self-paced course).

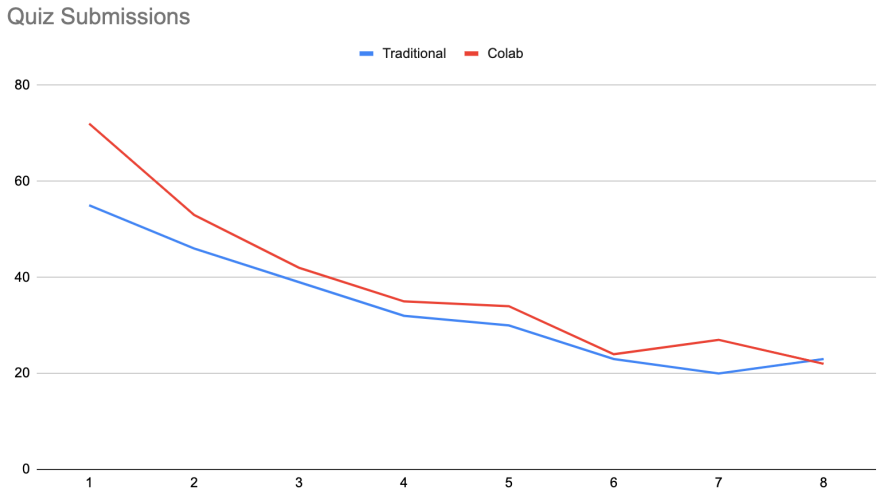


Fig. 3. Number of quiz submissions in each condition.

6.2 Student Performance

Overall, after the removal of non-participating students after Week 2, student performance was similar in the 2 courses. The average total score among the continued participants was similar in each group; 34.51% in the self-paced group and 34.76% in the instructor-led group. However, the number of passing scores was higher in each of the passing categories for the self-paced group. As specified in the course syllabi (available in Appendix B):

- Participants received a rating of "Exceptional" if their score was 95% or higher.
- Participants received a rating of "Understanding" if their score was 85% or higher.
- Participants received a rating of "Passing" if their score was 75% or higher.

As shown in Table 1, the self-paced course had a higher number of students in each of these categories.

Table 1. Student passing scores.

	Self-paced	Instructor-led
Average Score	34.51%	34.76%
Number of Students Passing	16	13
Number of Students Understanding	11	8
Number of Students Exceptional	3	2
Average Score ***without*** Student Removal after Week 2	23.45%	19.72%

Average quiz scores improved from the beginning of the course (Quiz 1) until near the end (Quiz 7). It is possible that average quiz scores improved due to lower scoring students dropping out as time progressed - i.e., students who scored lower in earlier quizzes and impacted the average did not submit the later quizzes.

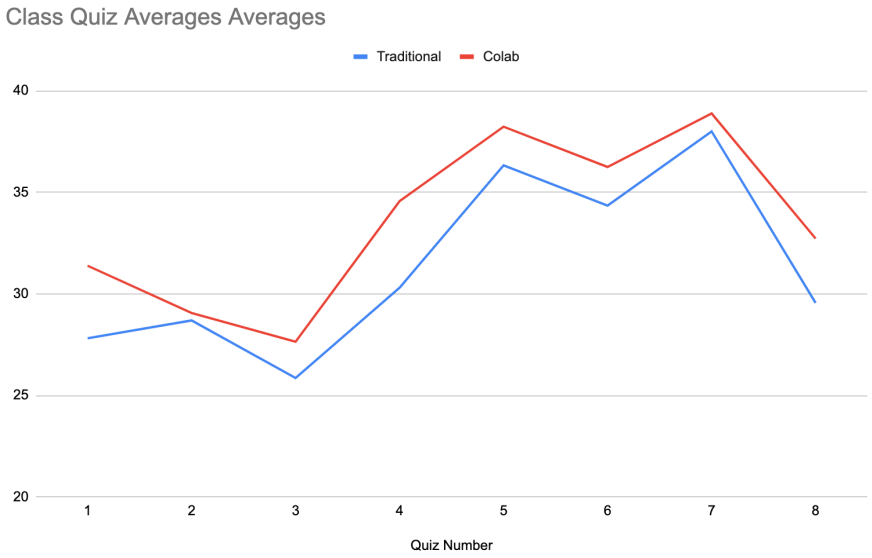


Fig. 4. Quiz scores in each condition.

While students in the self-paced course performed somewhat better than the students in the instructor-led course, and the attrition rate was somewhat more favorable in the self-paced course, the results were similar (refer to Figure 4). This finding suggests that instruction delivered in a structured format through computational notebooks could be about as effective as instruction delivered through more traditional approaches.

6.3 Student Perceptions

After the completion of the course, students were asked to complete a voluntary survey. Questions asked students to self-assess their confidence before and after the course using Likert scale of 1 (lowest confidence) to 5 (highest confidence): “How much do you agree with the following statement: Before this course, I was confident in writing Python code." and: “How much do you agree with the following statement: After taking this course, I am confident in writing Python code.” Figure 5 and Figure 6 illustrate that student confidence was similar before the course, with students in the Traditional course (n=23) rating their confidence 1.9 on average and students in the Colab course (n=22) rating their confidence 2.1 on average. After completion of the course, students in the Traditional course rated their confidence an average of 3.8, compared to 4.3 for students in the Colab course.

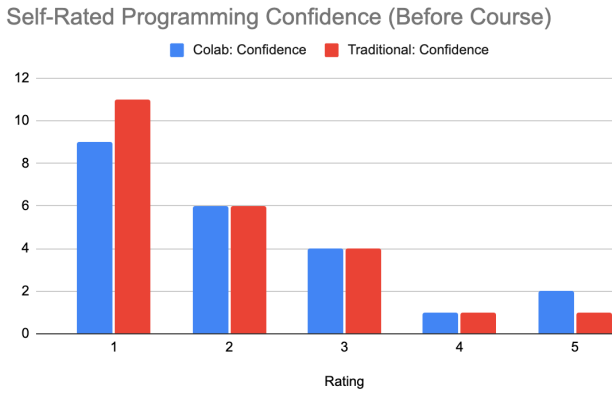


Fig. 5. Student self-rated confidence prior to the course.

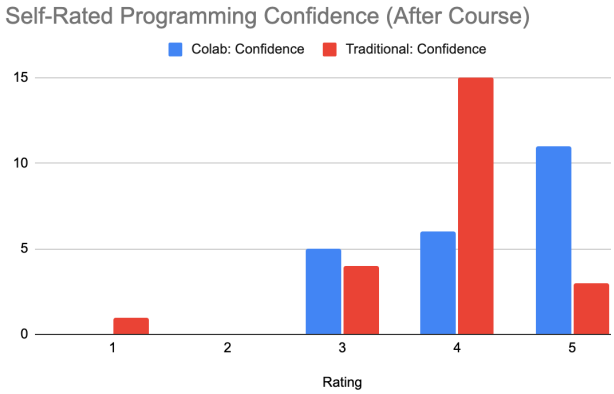


Fig. 6: Student self-rated confidence after completing the course.

One other question asked students to evaluate the time requirements on a Likert scale: How much do you agree with the following statement: "The time requirements for completing this course were reasonable." Responses are illustrated Figure 7, with students in the instructor-led course evaluating the time commitment less favorably than the students in the self-paced course (average scores of 4.3 vs 4.8, respectively).

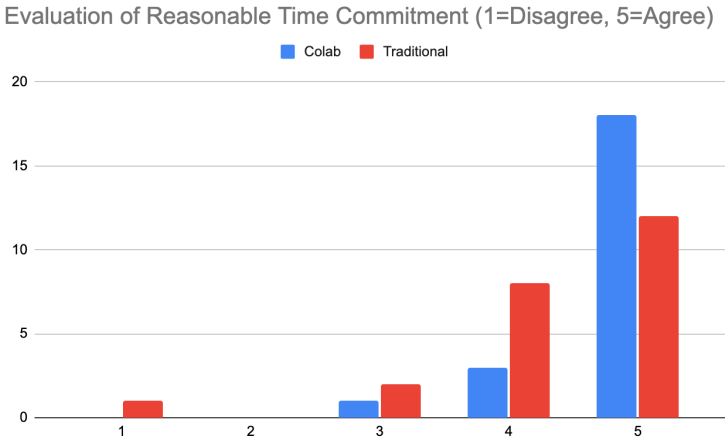


Fig. 7. Student evaluation of reasonableness of time commitment.

7. Conclusion and Future Work

Based on the quiz results, assignments, final exam, and overall scores, participants performed approximately the same in both the instructor-led group and the self-paced group. It is worth noting that producing materials for the lecture slides and Colab notebooks was time consuming, but the delivery of the lectures required significant additional time.

It appears that based on the scores and lower attrition rates, a student-driven approach using Colab notebooks is at least approximately as effective in helping students learn the concepts. In fact, given the higher passing rate (17.39%, or 16 out of 92 registered students completing with a

score of at least 75% in the self-paced group, compared to 14.13%, or 13 out of 92 registered students completing with a score of at least 75% in the instructor-led group), it is possible that the self-paced approach could be more effective in some cases. Furthermore, while attrition for both courses was high (this was expected due to the free and somewhat anonymous nature of the course), 28% of the participants in the self-paced group submitted the final exam, compared with 22% of the participants in the instructor-led group. Based on these results, it can be argued that the self-paced approach with Colab was at least as engaging as the instructor-led approach.

Based on the results observed in this study, future research questions could be evaluated. Attrition rates were high for both courses (more than 70%); reducing this attrition rate would lead to better learning outcomes. It is also worth exploring the reasons why the self-paced group fared slightly better, and if this result could be replicated with a larger group to demonstrate higher confidence in statistical significance.

References

[1] U.S. Bureau of Labor Statistics, “Software Developers, Quality Assurance Analysts, and Testers,” *https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm*, Feb. 06, 2023.

[2] T. Pham, “Analyzing The Software Engineer Shortage,” *https://www.forbes.com/sites/forbestechcouncil/2021/04/13/analyzing-the-software-engineer-shortage/* , Apr. 13, 2021.

[3] B. B. Morrison, L. E. Margulieux, and M. Guzdial, “Subgoals, context, and worked examples in learning computing problem solving,” in *ICER 2015 - Proceedings of the 2015 ACM Conference on International Computing Education Research*, Jul. 2015, pp. 267–268. doi: 10.1145/2787622.2787733.

[4] C. Wong, “Active Learning Experiences with Code Executable Blocks,” *https://medium.com/coursera-engineering/active-learning-experiences-with-code-executable-blocks-f7a4aee2c4f6*, Sep. 29, 2016.

[5] I. Roll, D. M. Russell, and D. Gašević, “Learning at Scale,” *Int J Artif Intell Educ*, vol. 28, no. 4, pp. 471–477, Sep. 2018, doi: 10.1007/s40593-018-0170-7.

[6] V. Aleven, J. Sewall, J. M. Andres, R. Sottilare, R. Long, and R. Baker, “Towards Adapting to Learners at Scale: Integrating MOOC and intelligent tutoring frameworks,” in *Proceedings of the 5th Annual ACM Conference on Learning at Scale, L at S 2018*, Jun. 2018. doi: 10.1145/3231644.3231671.

[7] H. Fake and N. Dabbagh, “Personalized Learning Within Online Workforce Learning Environments: Exploring Implementations, Obstacles, Opportunities, and Perspectives of Workforce Leaders,” *Technology, Knowledge and Learning*, vol. 25, no. 4, pp. 789–809, Dec. 2020, doi: 10.1007/s10758-020-09441-x.

[8] G. Conole, “MOOCs as disruptive technologies: strategies for enhancing the learner experience and quality of MOOCs,” *Revista de Educación a Distancia (RED)*, no. 50, Jul. 2016, doi: 10.6018/red/50/2.

[9] A. Pears *et al.*, “A Survey of Literature on the Teaching of Introductory Programming,” *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 204–223, Dec. 2007, [Online]. Available: http://ec.europa.eu/education/-

[10] C. Vieira, A. J. Magana, A. Roy, and M. L. Falk, “Student Explanations in the Context of Computational Science and Engineering Education,” *Cogn Instr*, vol. 37, no. 2, pp. 201–231, Apr. 2019, doi: 10.1080/07370008.2018.1539738.

[11] N. Lytle *et al.*, “Use, modify, create: Comparing computational thinking lesson progressions for STEM classes,” in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, Jul. 2019, pp. 395–401. doi: 10.1145/3304221.3319786.

[12] M. C. Kim and M. J. Hannafin, “Scaffolding problem solving in technology-enhanced learning environments (TELEs): Bridging research and theory with practice,” *Comput Educ*, vol. 56, no. 2, pp. 403–417, Feb. 2011, doi: 10.1016/j.compedu.2010.08.024.

[13] A. P. Lorandi Medina, G. M. Ortigoza Capetillo, G. H. Saba, M. A. H. Perez, and P. J. Garcia Ramirez, “A Simple Way to Bring Python to the Classrooms,” in *2020 IEEE International Conference on Engineering Veracruz, ICEV 2020*, Oct. 2020. doi: 10.1109/ICEV50249.2020.9289692.

Appendix A: Timeline

The study was conducted between March and May of 2022:

1. Participants were recruited on LinkedIn[1] and Twitter[2] in March of 2022.
2. The course officially started on March 21.
    a. March 21 - March 27: Introduction, If-Else Statements
    b. March 28 - April 3: Loops and Structures
    c. April 4 - April 10: Objects and Classes
    d. April 11 - April 17: Problem Solving
    e. April 18 - May 1: Exam Period

Appendix B: Course Materials

- Course syllabus for the instructor-led course:
    - https://docs.google.com/document/d/1LKOA2MsMFKeNxPKb6zgImeKkMsm9LmM3alMrwp5vfI8/edit
- Course syllabus for the self-directed course:
    - https://docs.google.com/document/d/1wRaI60qBh6jrWU9Ljdq2-fzMDqc6Rv-litQgIIS6IA0/edit
- All recorded videos for the instructor-led course are available in Google Drive:
    - https://drive.google.com/drive/folders/1jSbK3QZIB67d-2PCOJsHnNxAgky8--pv
- All lecture slides for the instructor-led course are available in Google Drive:
    - https://drive.google.com/drive/folders/1ZoMeyPJmGL3qYO-BSGbL49j5VSkTHQai?usp=share_link
- Examples from the instructor-led course:
    - https://github.com/timothyrjames/python-intro
- These Colab notebooks were used for the self-directed course:
    - https://drive.google.com/drive/folders/16v7PPfd_E0yc_aMlL-wZR2EYyArs7nBS?usp=share_link
- Assignment content was equivalent in both the Traditional and Colab courses, but the delivery mechanism was slightly different as the Traditional assignments were specified in plain text and Colab assignments were specified in a Colab notebook. The assignments can be found in Google Drive:
    - https://drive.google.com/drive/folders/1tbBqjW0X2TzHkW33frsq7cW7-ygCXCO5?usp=sharing
- All course materials are consolidated in Google Drive:
    - https://drive.google.com/drive/folders/18T3mI2jCCz5AmfItzf4DlAP5gieq07xk?usp=share_link

Appendix C: Final Score Distribution

- All scores are available (anonymized) in this spreadsheet:

---

[1] see: https://www.linkedin.com/feed/update/urn:li:activity:6901839270655488000/

[2] see: https://twitter.com/geekitarian/status/1495929603612983299

- https://docs.google.com/spreadsheets/d/1xGX0-TSYgsne9qo_FfAoW3t1tfL9s77yzAtk5ZWFGco/edit