EXPERIENCES IN PYTHON PROGRAMMING LABORATORY FOR CIVIL ENGINEERING STUDENTS WITH ONLINE COLLABORATIVE PROGRAMMING PLATFORM

Qi Cao[1], Li Hong Idris Lim[2], Vicki Dale[3], Nathalie Tasler[3]

[1]*School of Computing Science, University of Glasgow, Singapore Campus (SINGAPORE)*
[2]*School of Engineering, University of Glasgow, Singapore Campus (SINGAPORE)*
[3]*Academic & Digital Development, University of Glasgow (UNITED KINGDOM)*

Abstract

In higher education, software programming is an important fundamental skillset for students. Introductory software programming courses are offered to students majoring in not only computing science (CS), but also other schools including mechanical engineering, civil engineering, aerospace engineering, mathematics, business, and so on. Many students, particularly non-CS students, feel that software programming is not easy to learn and demonstrate good performance. Hands-on exercises in laboratories and tutorials are important to reinforce the learning of the theorical aspects of software programming. Clear guidance and immediate assistance from lecturers are highly appreciated in laboratory sessions to help students in their troubleshooting of software bugs. Teaching and learning face major challenges or even disruptions in the pandemic of COVID-19, due to the physical lock down of cities. Online learning or home-based learning has become a norm during the pandemic. As such, it introduces additional difficulty in the conduct of hands-on programming laboratory sessions, as lecturers and students are not in the same physical classroom. It may take a longer time for students to troubleshoot small software bugs in a programming code without immediate advice from lecturers. Students may become anxious or de-motivated in hands-on programming. In this paper, an online collaborative programming platform is employed to conduct laboratory sessions in the learning of Python programming subject, which is offered to Year 1 Civil Engineering students. It enables students and lecturers to work together or troubleshoot programming codes “virtually” from different physical locations. A case study and its effectiveness are explored by comparing the results before and after utilizing the collaborative programming. Interesting results are discussed on the learning process of Civil Engineering students who do not have any prior Python programming skills.

Keywords: Software programming learning, Python programming online, Collaborative learning, Collaborative programming platform, Laboratory programming exercise.

1 INTRODUCTION

In the current pandemic, almost all subjects have been converted to online learning or home-based learning through live video streaming tools, such as Zoom, Microsoft Teams, etc. It brings about new challenges to students and lecturers as online learning reduces the opportunities of class interaction and debates. Online learning lacks vital interactivity found in traditional face to face (F2F) classroom [1]. These challenges become more severe for Year 1 students who are newly enrolled into universities as they will need some time to adapt to a new environment in their university’s life. Compared to traditional F2F lectures, students tend to feel less engaged with online learning [2]. New methodologies and technologies are expected to fulfil requirements of online learning. Many learning institutions have been adopting online teaching in the current pandemic. Group allocation methods are used to enhance teaching experiences for small group discussions and interactions using online breakout rooms [[3]. Some other challenges are raised for students in online learning, such as difficulty in maintaining focused, experience of Zoom fatigue after long sessions, absence of peer support, lack of clear guidance from instructors, and so on. [4].

Software programs are widely used in various industries. As such, software programming courses are popular in Higher Education Institutions (HEI) [5]. Besides computing science (CS) students, software programming courses are also offered to non-CS students majoring in civil engineering, mechanical engineering, aerospace engineering, mathematics, data analytics, business, and social sciences, and so on. For Civil Engineering cohorts, software programming such as Python, Java, C languages, etc., may not be core subjects. However, these subjects are very useful in some computational tasks in Civil Engineering such as surveying, levelling, etc.

Common teaching methods for software programming subjects consist of contact hours including weekly lectures, laboratory sessions, after class assignments, and so on. Students learn programming theory knowledge in lectures, hands-on coding exercises in laboratory sessions and after class assignments. The laboratory sessions are important to enable students to practice hands-on software programming, with lecturers being around to provide immediate guidance and help. Software programming skills cannot be gained without the necessary practice and exercise. Students can experience developing software programming codes by themselves, followed by compiling, assembling and executing the programming codes. Students can also obtain a significant sense of achievement when they get their programming codes working, as per theory.

Software programming questions are unique and very different from those of other subjects. A tiny mistake or single program bug in the codes of software programs will stop the execution and operation of the entire software program. In traditional F2F classroom settings, students can raise questions and interact with lecturers on the spots whenever they encounter difficulties, software bugs, or programming errors. Software bugs and troubleshooting are not easy to describe or communicate over the phone or chat rooms. The best way would be to get the lecturers and students to see and work on the same computer screen so that they can go through the programming codes together. However, with subjects moving to online learning, it introduces new difficulties in laboratory sessions where students are not able to receive real time assistance from lecturers to troubleshoot software program bugs. Students may be stuck at a single lab exercise question for a long time, and not able to complete all labs exercise questions on time. It will cause anxiety to students in the study of programming. If the challenge cannot be resolved on time, students may gradually become demoralized in learning software programming subjects, thereby resulting in higher failure rates [6][7].

Many cities experienced huge difficulties causing challenges in school learnings due to the lockdown in the pandemic of COVID-19 [8]. With the wide adoption of online learning by schools across most countries nowadays, software programming courses also need to be converted, in terms of their delivery modes, into blended learning and collaborative learning. Blended learning has been explored in teaching software programming courses in HEI [9][10], with several ways being introduced, including F2F collaboration, online collaboration, online self-paced learning, etc. [11]. According to prior work in the literature, blended learning is more effective, compared with traditional learning method in programming courses [11]. Some prior works have also been reported to use collaborative tools to teach students in the learning of software programming subjects. In teaching MATLAB programming to four engineering disciplines including Civil Engineering, online help is offered through discussion forums on a Virtual Learning Environment course page, where students could use this space to help each other or contact lecturers [12]. Many tools for collaborative coding have been introduced following the “what you see is what I see” metaphor, such as Jimbo [13], Flesce [14], etc. Visual Paradigm tool is employed to conduct virtual laboratory sessions for online courses in learning software programming subjects, where students can build diagrams by using the Unified Modeling Language approach [15]. Student are asked to share their Cloud workspaces to their lecturers, such that the lecturers can see the progression of students’ work [16]. A web-based integrated development environment (IDE), named Collabode is utilized for collaborative software development, by enabling multiple software programmers to share their programming codes immediately, which allows for more than one active programming contributor to the same piece of codes [17]. A collaborative Teaching Assistant System is developed as an extension for the LMS Moodle pages in teaching introductory computer science courses [18][19]. A web-based tool, EasyCoding is reported in [20] to automatically produce activity guides and construction to help lecturers produce new activity guides. It aims to motivate students by performing laboratory exercises in learning C programming. Another tool named Botzone is reported in [21] as a competitive and interactive platform for general programming courses education. A tool, Kodr is reported as a modular customizable gamified learning platform to track student performance through coding challenges [22].

Some of these platforms have a deep learning curve requiring students to be equipped with prior programming skills before mastering the usage of collaborative tools. Large cohorts enrolling into universities may exhibit a wide spectrum of prior programming skills [23]. Some students may have zero software programming knowledge when enrolled into HEI. They may need a longer time to get used to the online learning in software programming courses. In this paper, an online collaborative programming platform named as repl.it Classroom [24] is explored to teach Python programming subject to Year 1 Civil Engineering students in the joint degree programme, which is a partnership between University of Glasgow, Singapore campus (UGS) and Singapore Institute of Technology (SIT). According to the Zoom poll at the beginning of the first lecture, 95% of these Year 1 Civil Engineering students have not learnt software programming subjects on C, Java, MATLAB, or Python languages. Almost 100% of them have zero Python programming skills in their previous study.

The research question (RQ) to be addressed is how to provide a low barrier to entry, user friendly and effective online learning method to non-experienced students for hands-on laboratory exercises of software programming subject.

A case study is conducted by utilizing this collaborative programming platform to conduct laboratory sessions in the Python programming subject for an online learning configuration. It enables students and lecturers to work together to troubleshoot programming codes "virtually" in laboratory sessions and after class. Each of them can see the same piece of code from different physical places. It allows seamless online communication and work on the same code. It can address the concerns and challenges in laboratory programming practice in online learning. The results of the laboratory coding tests are evaluated if students without any prior Python programming skills can benefit from the online collaborative programming platform by receiving timely assistance from lecturers virtually.

2 METHODOLOGY

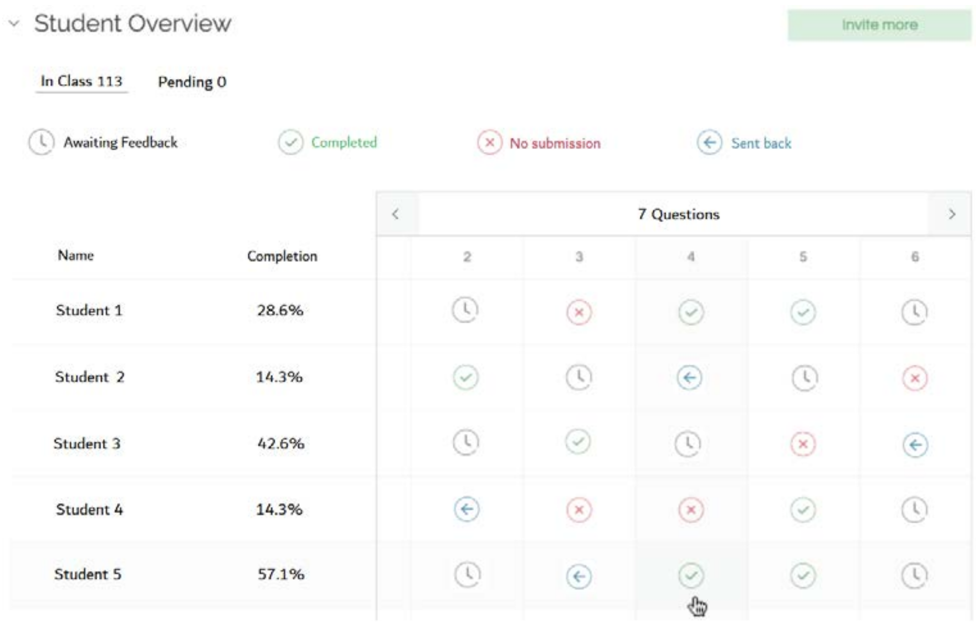
Python Programming is offered to all Year 1 students of Civil Engineering in the joint degree program. Students have the option to apply exemption for this subject, if they have learnt this subject in their previous study with minimally a "*C*" grade achieved. There are 113 students enrolled into this subject in AY2020/21.

It is a six-week subject worth five credits. The teaching mode is mainly through Zoom online due to COVID-19 pandemic. The contact hours include a two-hour lecture weekly on Zoom for theory of Python programming, and a two-hour laboratory exercises bi-weekly for hands-on practice. There are seven hands-on exercise questions in each laboratory session.

It is a 100% CA subject. Its assessment components consist of two open-book quizzes and one open-book class test. Quiz 1 is conducted in Week 3, covering learning contents of Week 1 - 2. Quiz 2 is conducted in Week 5, covering learning contents of Week 3 - 4. The class test is conducted in Week 6, covering all learning contents.

With theory learned in the weekly lectures, students are asked to work on hands-on development of Python programming codes in the bi-weekly laboratory sessions. If the students are not able to finish these laboratory exercise questions in each laboratory session, they are allowed to continue working after the labs. This provides flexibility to students who may need more time to acquire Python programming knowledge and can practice slowly at their own convenience after class.

Students are instructed to install an IDE tool named Anaconda Python on their own laptops. Anaconda IDE contains a Python programming editor, Jupyter notebook. Students can practice programming examples and laboratory questions on their own laptops where developed source codes are stored.


Student Overview

Invite more

In Class 113 Pending 0

🕓 Awaiting Feedback ✓ Completed ✗ No submission ← Sent back

		7 Questions				
Name	Completion	2	3	4	5	6
Student 1	28.6%	🕓	✗	✓	✓	🕓
Student 2	14.3%	✓	🕓	←	🕓	✗
Student 3	42.6%	🕓	✓	🕓	✗	←
Student 4	14.3%	←	✗	✗	✓	🕓
Student 5	57.1%	🕓	←	✓	✓	🕓

Figure 1. Example view for students working status in online collaborative programming platform.

Students are also instructed to enroll in a web-based online collaborative programming platform, *repl.it Classroom* for laboratory programming exercises. It consists of an online IDE that is capable of compiling and executing multiple software languages, including Python language. Students do not need to install any software tool in their own laptops. The entire development cycle from coding, testing and submissions is through this online collaborative programming platform. For each laboratory question, lecturers can pre-define a few automatic test cases on the platform.

This online collaborative programming platform provides a graphic user interface (GUI) for lecturers to view the programming exercise status of each student. It provides a very convenient snapshot for lecturers to identify the students who are encountering difficulties, as well as the exercise question where the students are working on. As such, lecturers can better prioritize and provide timely responses online to students who are stuck in software code bugs. An example GUI view indicates four different statuses of each student as shown in Fig. 1. These four statuses are explained as follows.

- *Awaiting feedback*: the student encounters difficulty in a programming question, awaiting help from lecturers.
- *Completed*: corresponding questions are done, and student has passed the pre-defined automatic test cases successfully.
- *No submission*: the student has not submitted the programming codes for the corresponding question yet.
- *Sent back*: lecturers have provided comments and feedback to the student on how to troubleshoot the codes.

To evaluate the effectiveness of the online collaborative programming platform, the laboratory programming exercises are conducted differently by *Method 1* and *Method 2* as follows.

Method 1. Perform Python programming using Jupyter notebook IDE and submit to LMS portal.

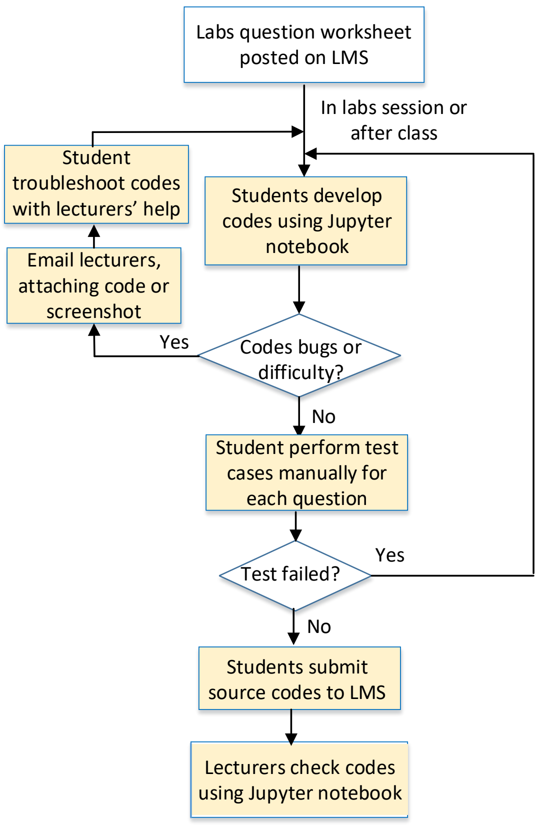
In the first laboratory session, students develop Python programming codes using Jupyter notebook IDE installed on their laptops. Laboratory exercises worksheet is handed out to students through Learning Management System (LMS) online portal. Learning outcomes are evaluated by Quiz 1, which contains three open-ended programming questions with similar format of laboratory exercise programming questions.

Method 2. Perform Python programming on the online collaborative programming platform.

In the second and third laboratory sessions, students are instructed to develop and submit their Python codes to the online collaborative programming platform. Laboratory exercises worksheet is handed out to students by both LMS online portal and the online collaborative programming platform. Learning outcomes are evaluated by Quiz 2, that contains three open-ended questions with similar format of exercise questions.

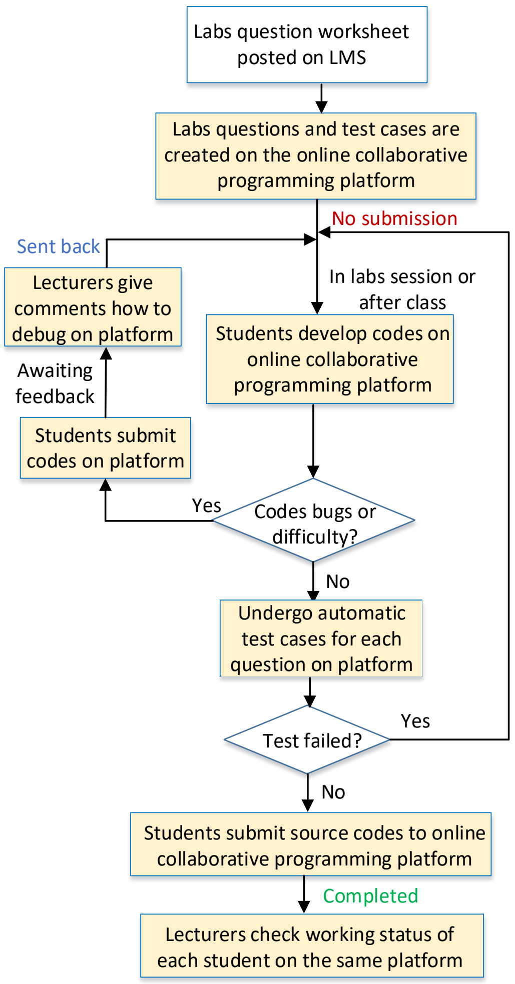
The working flowchart of *Method* 1 is shown in Fig. 2. Students develop programming codes using Jupyter notebook IDE on their own laptops. If students encounter difficulties or code bugs, they can drop email to lecturers by attaching their codes with bugs or screenshots of error messages. The lecturers can help to identify code mistakes using the local IDE. But lecturers may lose track or overlook some email, due to the large number of email communications with students. With this level of assistance, students develop programming codes and undergo test cases manually to make sure correct results being obtained. If manual test cases are failed, students need to revise the codes. Once students complete coding these laboratory questions, they submit the codes to the LMS online portal. Lecturers can then check all submissions at the LMS portal. But the LMS portal is not an IDE that is capable of compiling and executing Python codes. To check if the submitted programming codes work correctly, lecturers need to export and execute the submitted codes to the Jupyter notebook IDE one by one.

The working flowchart of *Method* 2 is shown in Fig. 3. Each laboratory question and automatic test cases are created on the online collaborative programming platform. Students develop Python codes for each question on the same platform. If students encounter code bugs or difficulties, they can submit the current codes to the online collaborative programming platform. The working status of this question becomes *Awaiting feedback*, where students are waiting for lecturers' help to troubleshoot code bugs. Lecturers will receive email notifications, and login to the same platform to check these codes. The lecturers provide debugging comments and then return to the students for further development. The working status of this question changes to *Sent back*. Students can modify the codes next. Once the codes are ready for testing, the Python codes are submitted for undergoing automatic test cases. If automatic test cases are failed, students need to revise the codes on the platform. The exercise question is marked as *Completed* only if it passes all test cases successfully. Lecturers can check the working status of each student on each question real time.



Labs question worksheet posted on LMS
In labs session or after class
Student troubleshoot codes with lecturers' help
Students develop codes using Jupyter notebook
Email lecturers, attaching code or screenshot
Yes
Codes bugs or difficulty?
No
Student perform test cases manually for each question
Yes
Test failed?
No
Students submit source codes to LMS
Lecturers check codes using Jupyter notebook

Figure 2. Work flowchart of Method 1.



Labs question worksheet posted on LMS
Labs questions and test cases are created on the online collaborative programming platform
No submission
Sent back
Lecturers give comments how to debug on platform
In labs session or after class
Students develop codes on online collaborative programming platform
Awaiting feedback
Students submit codes on platform
Yes
Codes bugs or difficulty?
No
Undergo automatic test cases for each question on platform
Yes
Test failed?
No
Students submit source codes to online collaborative programming platform
Completed
Lecturers check working status of each student on the same platform

Figure 3. Work flowchart of Method 2.

The laboratory exercise questions are not gradable that are mainly for hands-on practices to students. Students can develop programming codes for each laboratory question in the laboratory session and after class. They are asked to submit their source codes with test cases passed within 1 week.

3 CASE STUDY

A case study is conducted to evaluate the results of two methods for students learning of Python programming. Quiz 1 is conducted in Week 3 to test students' learning for lecture contents of Week 1-2 and laboratory Session 1. While Quiz 2 is conducted in Week 5 to test students' learning for lecture contents of Week 3-4 and laboratory Session 2. There are three quiz questions in both quizzes with the same CA weights. The topics are covered by questions of each quiz are shown in Table 1.

Table 1. Python programming topics covered by two quizzes.

Quiz 1 topics	*Quiz 2 topics*
• Input and output of Python programming • Various Python data types (Number; String; Boolean; List) • Python operators (Arithmetic; Assignment; Comparison) • Python Branching (if-else Statement; if-elif-else Statement)	• Flow of execution in Python programming • Various loops (For Loops; While Loops) • Python functions (Calling; Returning; Parameters and arguments)

Both open book quizzes are conducted on the LMS portal online, where students access the online quizzes remotely from home or off campus. Students can use any reference books, but not allowed to discuss with their peers or share their solutions each other. The time duration of each quiz is 45 minutes, that is selected to give students sufficient time on developing and testing their Python codes. But it may be insufficient for them to discussing with their peers or share solutions. At the ending time of each quiz, students need to submit their developed Python codes to the LMS portal to be marked by lecturers.

Table 2. Result comparisons of two quizzes.

	Mean score	***Standard deviation***
Quiz 1 for *Method 1*	69.91	15.97
Quiz 2 for *Method 2*	75.01	13.56
Difference (%)	7.3%	-15.1%

Quiz results for both *Method* 1 and *Method* 2 are evaluated and compared in the case study. The mean scores and scores' standard deviations of each quiz are shown in Table 2. It is observed that the mean score of Quiz 2 is improved by about 7.3%. The scores' standard deviation of all students is decreased by about -15.1% in Quiz 2. According to the policy of the University, Quiz and exam results are confidential information and cannot be shared or published. In order to discuss the result distributions without breaching confidentiality, the grades are merged and presented into three coarse grades: (1) Grade A- and above, (2) Grade from B- to B+, (3) Grade C+ and below. The coarse grade distributions of both quizzes are shown in Fig. 4. It is observed that more students score at Grade A- and above in Quiz 2 compared to those of Quiz 1.

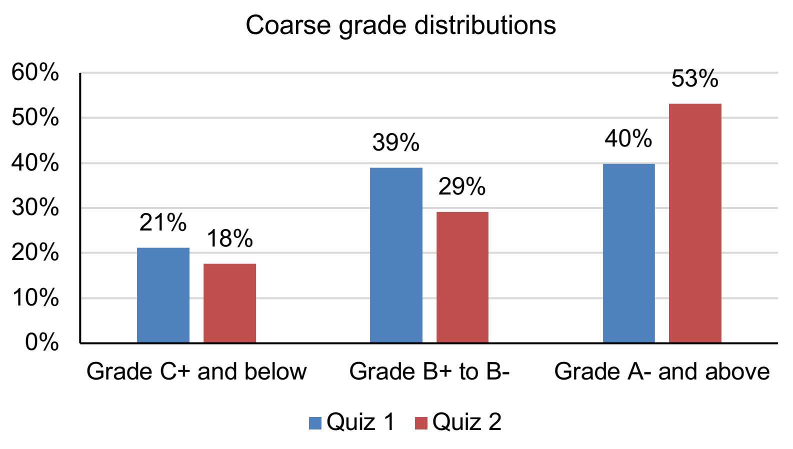


Figure 4. Coarse grade distribution comparisons of Quiz 1 and Quiz 2.

3.1 Discussions

It is observed from the results that positive impacts are brought by Method 2 with the online collaborative programming platform. It helps students in learning Python programming subject more effectively. In laboratory sessions, some students may not complete all exercise programming questions, due to various reasons. For example, some students face difficulties due to lesser prior programming skills. Others encounter software bugs in programming codes, which take time to troubleshoot one by one without swift guidance from lecturers. For those students who cannot complete all questions in labs sessions, they can keep working on the remaining questions online. Students may expect to receive continuous guidance on troubleshooting from lecturers in laboratory sessions and after class.

From the viewpoint of lecturers, the online collaborative programming platform brings across benefits like increasing of teaching efficiency. With most of courses moved online, it is not easy to help students troubleshooting software bugs on the spot physically. It enables collaborative programming that lecturers and students can work together virtually and speed up the problem-solving process in laboratory programming exercises.

However, there is a potential limitation in the case study. Two open book quizzes are conducted online where students remotely assess from home or other places off-campus. Students are asked not to discuss with peers or share solutions with each other. In reality, it is difficult to enforce as it depends on the integrity of students. It is possible that some students may sit next to each other and share answers in quizzes, which makes the evaluation of results inaccurate. Moving forward, a better way to ensure accurate quiz results is through physical face-to-face quizzes with invigilators.

4 CONCLUSIONS

Learning of software programming and laboratory programming exercises face new challenges with the trend of online learning and home-based learning. The challenges are more severe for non-CS students without prior programming skills. In this paper, a learning method by utilizing an online collaborative programming platform for laboratory hands-on programming practices is introduced. The platform is offered to Year 1 Civil Engineering students learning a Python programming subject. It provides a lower entry barrier into Python without the deep learning curve. It enables students and lecturers working together virtually on the laboratory programming exercises questions, where students and lecturers can see the real time working status and the same piece of code virtually online. With seamless communication and timely response from lecturers, students can continue their programming exercise questions in the laboratory or after class. The working flowchart utilizing the online collaborative programming platform is presented and compared with a conventional method on laboratory programming practice. Two quizzes are conducted to evaluate the learning outcomes of both methods. The evaluation results show that students make notable improvement in Quiz 2 through the laboratory practices on the online collaborative programming platform. It provides a viable option for hands-on laboratory practicing of software programming subjects, which provides a user friendly and effective learning method to non-experienced students on online programming learning.

REFERENCES

[1] A. Hron and H. Friedrich, “A review of web-based collaborative learning: Factors beyond technology,” *Journal of Computer Assisted Learning*, vol. 19, no. 1, pp. 70–79, 2003. https://doi.org/10.1046/j.0266-4909.2002.00007.x.

[2] N. Kemp and R. Grieve, “Face-to-face or face-to-screen? Undergraduates' opinions and test performance in classroom vs. online learning,” *Frontiers in psychology*, vol. 5, 2014. https://doi.org/10.3389/fpsyg.2014.01278

[3] A. Miller, M. Barr, W. Kavanagh, I. Valkov, and H. Purchase, “Breakout Group Allocation Schedules and the Social Golfer Problem with Adjacent Group Sizes,” *Symmetry*, vol. 13, no. 1, 2021. https://doi.org/10.3390/sym13010013

[4] S. Asgari, J. Trajkovic, M. Rahmani, W. Zhang, R. Lo and A. Sciortino, “An observational study of engineering online education during the COVID-19 pandemic,” *PLoS ONE*, vol. 16, no. 4, 2021. https://doi.org/10.1371/journal.pone.0250041

[5] S. Alhazmi, M. Hamilton, and C. Thevathayan, “CS for all: catering to diversity of master’s students through assignment choices,” *49th ACM Technical Symposium on Computer Science Education*, 2018.

[6] L. Margulieux, B. Morrison and A. Decker, “Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples,” *International Journal of STEM Education*, 2020. https://doi.org/10.1186/s40594-020-00222-7

[7] L. Porter, M. Guzdial, C. McDowell, and B. Simon, “Success in introductory programming: what works?,” *Communications of the ACM*, vol. 56, pp.34-36, 2016. https://doi.org/10.1145/2492007.2492020

[8] Y. Cai and Q. Cao, *When VR Serious Games Meet Special Needs Education*. Publisher: Springer Nature, 2021. https://doi.org/10.1007/978-981-33-6942-9

[9] T. Bati, H. Gelderblom, and J. Van Biljon, “A blended learning approach for teaching computer programming: design for large classes in Sub-Saharan Africa,” *Computer Science Education*, vol. 24, no. 1, pp. 71–99, 2014.

[10] C. Jacobs, G. Gorman, H. Rees, and L. Craig, “Experiences with efficient methodologies for teaching computer programming to geoscientists,” *Journal of Geoscience Education*, vol. 64, no. 3, pp. 183–98, 2016.

[11] A. Alammary, “Blended learning models for introductory programming courses: A systematic review,” *PloS one*, vol. 14, no. 9, 2019. https://doi.org/10.1371/journal.pone.0221765

[12] C. Warren, “MATLAB for Engineers: Development of an Online, Interactive, Self-study Course,” *Engineering Education*, vol. 9, no. 1, pp.86-93, 2014. https://doi.org/10.11120/ened.2014.00026.

[13] S. Ghorashi and C. Jensen, (2016). “Jimbo: a collaborative IDE with live preview,” *9th International Workshop on Cooperative and Human Aspects of Software Engineering*, 2016, https://doi.org/10.1145/2897586.2897613

[14] A. Carzaniga, D. Rosenblum, and A. Wolf, “Design and evaluation of a wide-area event notification service,” *ACM Transactions on Computer Systems*, 2001.

[15] A. Dennis, B. Wixom, and D. Tegarden, *System Analysis & Design, UML Version 2.0: An Object-Oriented Approach,* 3rd ed. Publisher: John Wiley & Sons, 2009.

[16] V. Riabov, “Teaching Online Computer-Science Courses in LMS and Cloud Environment,” *International Journal of Quality Assurance in Engineering and Technology Education*, vol. 5, pp. 12-41, 2017. https://doi.org/10.4018/IJQAETE.2016100102

[17] M. Goldman, G. Little, and R. Miller, (2011). “Real-time collaborative coding in a web IDE,” *24th annual ACM symposium on User interface software and technology*, pp. 155–164, 2011. https://doi.org/10.1145/2047196.2047215

[18] L. Echeverría and R. Cobos, “Designing the assessment of the collaborative learning process in LMS courses,” *IEEE International Conference on Computer Supported Cooperative Work in Design*, pp. 218–223, 2015.

[19] L. Echeverría, R. Cobos, L. Machuca, and I. Claros, “Using collaborative learning scenarios to teach programming to non-CS majors,” *Computer Applications in Engineering Education*, vol. 25, pp. 719–731, 2017. https://doi.org/10.100 2/cae.21832

[20] M. Almeida, L. Alves, M. Pereira, and G. Barbosa, “EasyCoding - Methodology to Support Programming Learning,” *1st International Computer Programming Education Conference*, 2020. https://doi.org/10.4230/OASIcs.ICPEC.2020.1

[21] H. Zhou, H. Zhang, Y. Zhou, X. Wang, and W. Li, “Botzone: an online multi-agent competitive platform for AI education,” *23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pp. 33-38, 2018. https://doi.org/10.1145/3197091.3197099

[22] A. Draz, S. Abdennadher, and Y. Abdelrahman, “Kodr: A Customizable Learning Platform for Computer Science Education,” *European Conference on Technology Enhanced Learning*, pp. 579–582, 2016.

[23] C. Alvarado, G. Umbelino, and M. Minnes, “The persistent effect of pre-college computing experience on college CS course grades,” *49th ACM Technical Symposium on Computer Science Education*, 2018.

[24] Replit Classroom. https://replit.com/. (Note: *Replit Classroom* has been re-branding into *Replit Teams* in 2021).