Utilising Blended Learning for Large Classes to Deliver an Introductory Programming Course

Buddhika Karunarathne and Vishaka Nanayakkara
University of Moratuwa, Sri Lanka

buddhika@cse.mrt.ac.lk
vishaka@cse.mrt.ac.lk

Abstract: Higher education institutions are constantly pushed to increase the student intake and produce industry-ready graduates. The pressure generally arises from the governing bodies, originating from socio-political factors. It is particularly evident in the field of Information Communication Technology (ICT) where there is a growing need for quality graduates. Undergraduate programmes in the field of ICT generally offer introductory computer programming courses that are mandatory at early stages of the degree programme. Consequently, the number of students registering for such courses can increase drastically. The students are coming from diversified backgrounds. While some students may have prior experience in computer programming, a larger majority may not have any prior knowledge or experience. Increased class size meets the constraint of the limited availability of physical classroom space and would require the class to be separated into smaller groups for face-to-face activities. Large classes are also challenged with the increased man-power requirement in terms of lecturers, instructors, and teaching assistants. Maintaining student engagement and interactivity becomes increasingly difficult in a large class. Grading of assessments becomes time-consuming and needs and requires increased manpower. Based on the observations and learning from an ongoing introductory programming course, this study evaluates the procedure for overcoming the challenges faced by the teaching staff, especially with a class size exceeding 1000. To cover the theoretical components, recorded video lectures are provided. The class is divided into groups consisting of approximately 100 students and live, online, interactive group discussions are conducted. The live, online smaller group discussions allow students to clarify doubts regarding the content. This also allows the distribution of the limited teaching and assistant staff. Students who need further assistance are encouraged to meet the teaching assistants physically. Thus, the workload is distributed effectively, and it helps ensure the dedicated support and attention is provided to the students. Auto graded programming assignments are effectively employed for formative and summative assessments, which reduces the grading workload significantly. The findings of this study provide insights on course design to effectively deliver the content and conduct assessments overcoming the challenges presented by increased class size.

Keywords: Blended learning, Large class, Programming course, Auto-grading

1. Introduction

The field of ICT is continuously growing and the demand for skilful individuals in various sub-domains has also been in the rise. Computer programming, software engineering, web development, quality assurance engineering, cloud computing, user interface/user experience (UI/UX) design, and information security are some of the highly sought after subcategories in the field of ICT. Around the globe, a large number of students sign up and get enrolled for study programmes that aim to transform them as individuals with knowledge, skills and attitudes expected by the respective industries.

In general, any study programme in the field of ICT would expect the students to have at least a fundamental knowledge in computer programming. This is generally done by allowing the students to take an elementary programming module at the early stages of the programme, as early as in the very first semester.

At present, engineering schools are also pressing towards requiring their students to be familiar with fundamentals of computer programming. As all branches in engineering now use computers and computer-based methodologies undergraduate students that belong to all engineering disciplines are taught programming in most universities (Peteranetz et al., 2018). Even though for may engineering disciplines may not directly involve computer programming at their core work, programming knowledge is identified as a considerable advantage for such students to successfully continue with their studies. Computer programming and scripting knowledge can be vastly helpful in conducting data analysis, developing models and simulations, and to automate tasks in useful ways which otherwise can be exceedingly time-consuming. Therefore, it should be noted that for any of the non-ICT engineering disciplines, solid knowledge in programming fundamentals with hands-on experience will be vastly useful. Liu, 2014 stated that undergraduate programmes in computer science may include several programming-related courses of varying difficulty level. Other engineering disciplines may include at least one introductory programming course in the undergraduate curriculum (Wang et al., 2017).

Effectively delivering a module in programming fundamentals can be challenging in many ways. Most students may have no prior experience in any kind of computer programming. Some students may not even have a good understanding in basic mathematical concepts. The students will be from diverse backgrounds, skill levels and they will be placed in a class to learn programming from scratch. Learning programming is not just about learning theory. Hands-on exercises and practice play a vital part in teaching and learning programming. Therefore, it is not just a case of delivering lectures and requiring the students to do assessments and examinations. It may require a considerable amount of individual attention towards the students so that they can catch up with the flow and experience an efficient learning journey.

The task becomes more challenging with increased number of students and limited resources in terms of teaching staff and infrastructure. With the increasing student intake, the teaching staff is burdened with delivering programming fundamental modules for large classes often without appropriate number of manpower to cater to such scale. Especially when the class size exceeds 1000, the teaching staff find it difficult to manage the time and resources in expectation to deliver the course content effectively.

This study identifies the situation, analyses, and attempts to provide recommendations based on a real case-study on an existing programming fundamentals course in an engineering school. The findings would be largely helpful to the teaching staff as well as the administrators to take necessary steps and adopt accordingly to maintain required standards in the module delivery.

2. Literature Review

2.1 Introductory Programming Courses

Kanika et al., 2020 identified the main reasons for teaching introductory programming course in the early stage of an undergraduate engineering program. The study emphasized that exposure to terminology and fundamental programming concepts, enhancing computation thinking and problem-solving abilities, teaching to design, implement, test, and debug a program and teaching good practices in programming were of importance for the students. Such course is typically taught using a particular programming language. In a typical introductory course, students are provided with details on data types, variable declaration, operators and expressions, input and output statements, conditional and iterative statements, arrays and strings, pointers, user-defined data types, functions and recursion, and file handling (Yuan et al., 2015). C++, Java, and Python are identified as some of the popular programming languages used in most universities (Kunkle & Allen, 2016).

The history of teaching introductory and advanced modules in computer programming dates back to mid-20th century. The programming languages have largely evolved since then and hence the teaching methodologies have also evolved. Koulouri et al., 2014 identified the complexity of teaching a programming course and the challenges faced by the educators. The teachers take up the task of teaching students how to analyse a problem logically and translate the identified solution into a computer program.

Kanika et al., 2020 went on to identify approaches used in teaching programming and clustered into visual programming, game-based learning, pair and collaborative learning, robot programming and assessment systems. The study further reviewed the efficiency of the approaches by comparing multiple studies that had been conducted. The study went on to recommend that suitably designed tools help in motivating students to learn programming. Especially the interactive programming tools have shown to help students in getting familiar with programming in a short period of time.

Prensky, 2007 identified key factors for maintaining interest levels of the modern students. The study stated that the students preferred constant interaction with technology, inductive reasoning, frequent and quick interactions with content, and good visual literacy skills were preferred by students. It went on to say that these are not easy to offer through traditional instructional methods.

Bogdanovych, A. and Trescak (2016) showed that by including game-like visual examples, the students' motivation to learn programming significantly improved. The study further implied that, even though the student started with a visual framework to learn programming, they successfully went on to learn programming language like Java, which required extensive coding.

2.2 Teaching and Learning Methods for Introductory Programming Modules

Blended learning as defined in Graham (2006) as learning systems "that combine face-to-face instruction with computer-mediated instruction" has received increasing attention in recent times. Garrison and Kanuka (2004)

identified blended learning as “the thoughtful integration of classroom face-to-face learning experiences with online learning experiences”. The method is widely adopted in higher education and has been referred to as the “new traditional model” (Ross and Gage, 2006) and the “new normal” in course delivery (Norberg et al., 2011).

Blended learning has gained significant popularity in recent years, as it offers a flexible and engaging learning environment that combines the benefits of both traditional classroom instruction and online learning (Kizilcec & Kambhampaty, 2020). The hybrid-delivery model described in one of the sources allows participants to access graduate degrees while remaining in their current positions, providing them with opportunities for professional advancement (Cable et al., 2014).

lended learning offers a variety of benefits for teaching programming fundamentals. It facilitates flexible learning, access to rich resources, and collaboration among learners and educators. With the potential to transform education, this system provides a comprehensive and interactive approach that improves comprehension, retention, and overall learning outcomes (Monika et al., 2023). Blended learning also allows for a personalized learning experience, as students can progress at their own pace and receive individualized feedback and support (Cable et al., 2014). Furthermore, the use of multimedia elements such as text, images, audio, video, and interactive content in blended learning can enhance the teaching and learning process and make programming fundamentals more engaging and accessible to students (Monika et al., 2023).

3. Problem Statement

The challenges faced by educators in efficiently delivering introductory programming course in undergraduate engineering study programmes is identified. Multiple studies have investigated how these challenges can be overcome by employing modern teaching and learning methodologies. However, it is a complicated problem for which the solution may have to take different combinations of actions to provide a sophisticated and customized overall solution.

4. Methodology

This study documents the methods adopted in delivering an introductory programming module for an engineering undergraduate programme in a state university in Sri Lanka.

4.1 The Overview of the Module

The introductory programming module which used Python as the programming language for teaching, targeted at training the students to design and develop algorithms to solve simple computational problems. Using the high-level programming language Python, it further trains the students to develop simple control applications using embedded software platforms.

The most recent offering of the module, which will be of focus in this study, comprised of more than 1000 registered students who were in their first semester of the engineering degree programme. Most of these students were yet to select their specific engineering discipline and they were required to take this module at the first semester of the programme.

The students were divided into 12 groups with each group comprising of 80-90 students and one full-time teaching assistant.

4.2 Delivery of Lectures

The panel of lectures comprised of five senior academic staff. The content of the module was distributed among the panel, and they delivered their respective parts in recorded lecture videos. The respective lecture notes and other supporting material including lecture slides, labs sheets, Jupyter Notebooks (Project Jupyter), and online quizzes were made available on the learning management system, Moodle (Moodle.org).

4.3 Live Discussions

Weekly group live online discussions were conducted by the respective lecturer. The teaching assistant in-charge of the group also participated the session. The students were expected to refer to the material, including the recorded lecture for that week, prior to joining the live discussion. The students were encouraged to come prepared with questions to be discussed during the session. The lecturer would actively take examples from the lesson and explain the content. All these sessions were conducted on the video conferencing platform, Zoom (Zoom.us).

4.4 One-on-one Support

Those students who needed further assistance were given the opportunity to meet a teaching assistant during a specified time. Each week teaching assistants would allocate few hours for one-on-one meetings with students depending on the requirement. This helped the weaker students to catch-up with the flow during the semester.

4.5 Teaching Assistant Support

Twelve full-time teaching assistants were deployed for this module. Each teaching assistant had the responsibility of managing a group of students. They were the students primary contact regarding anything related to the module. Apart from providing help with the content of the module, they also kept track of the student participation and engagement. In the case of any abnormality, they would report during the weekly meeting of the teaching staff.

4.6 Continuous Assessments (CA)

The continuous assessments comprised of lab exercises and online quizzes on Moodle. In these activities students would apply the programming knowledge into hands-on work to solve programming problems of varying difficulty levels. CodeRunner (Lobb and Harlow, 2016) which is an open-source question-type plugin for Moodle was vastly useful in conducting the continuous assessments. CodeRunner made it possible to provide auto-graded programming questions which assessed the students answer based on pre-defined test cases. A significant portion of the continuous assessments were conducted using CodeRunner type questions, which reduced the grading load by a great margin.

4.7 Final Assessment

The final assessment was conducted in a face-to-face proctored mode. Multiple choice questions (MCQ) were mainly used. Since there were over 1000 students to take the examination, this was a procedure with high physical space and human resource demands. Due to the unavailability of such a large number of computing workstations, the examination had to be given on printed paper. A large number of supervisors, invigilators, and other staff personnel had to be deployed for the examination.

4.8 Grading Management

The final examination had the highest grading load with over 1000 papers to mark. Even though only MCQs were given, since the examination was paper-based, it took a considerable amount of time to complete with the participation of the lecturers and the teaching assistants.

4.9 Teaching Manpower Management

This module had one of the largest dedicated teaching staff involved. Apart from the lecturers and full-time teaching assistants, few part-time teaching assistants were also employed. The part-time teaching assistants mainly comprised of final year students in the same undergraduate programme. All the teaching staff regularly met in a weekly meeting to discuss the weekly updates and planning for the next stages of the module. In addition to that, sub-group meetings were also arranged where necessary to discuss specific events such as continuous assessment components.

4.10 Building Question Pools

The questions pools for the continuous assessments were mainly contributed by the teaching assistants. The lecturers would join the meetings to provide guidance and review the questions. By developing a large pool of questions, it was possible to randomize the questions presented to the students during the continuous assessments.

5. Analysis and Discussion

5.1 Student Composition

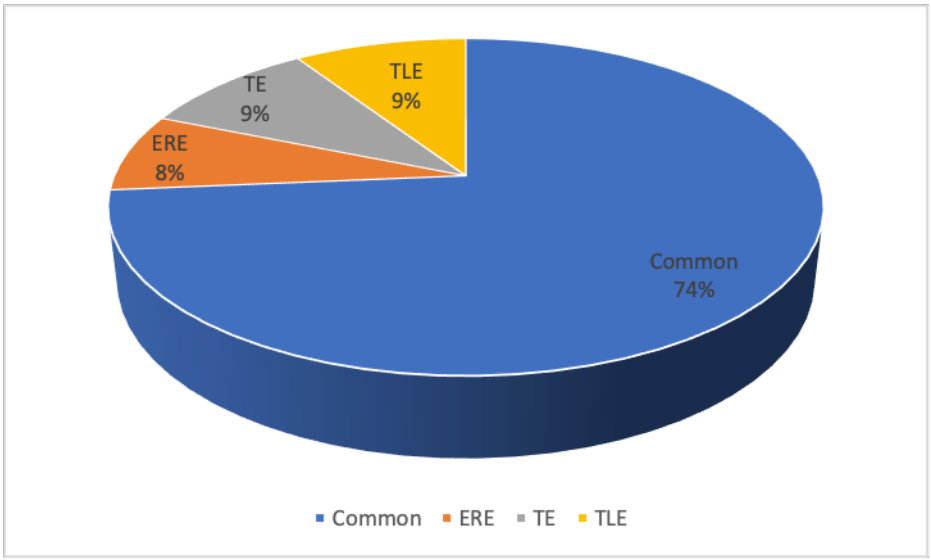

Figure 1: Enrolled students' distribution among engineering disciplines

Majority of the students (74%) were in a common group, who were yet to decide on their engineering discipline. For these students, the grade to be obtained from the introductory programming module was also countable towards the competitive score in selecting a discipline. On the other hand, student who had already included in specified disciplines, Earth Resources (ERE), Textile Engineering (TE) and Transport and Logistics (TLE) were taking the module as per the mandatory requirement.

5.2 Performance in Continuous Assessments

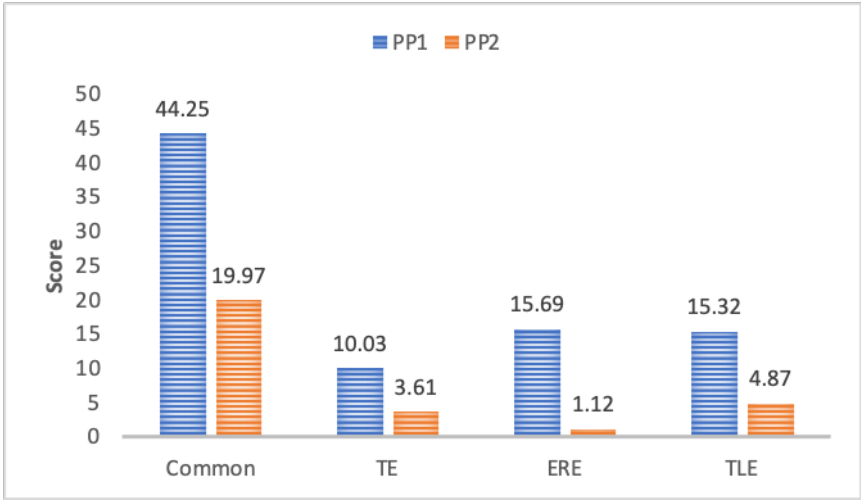

Figure 2: Average scores obtained in CA components.

Figure 2 presents the average scores (out of 100) for the two major continuous assessment components, Programming Problems 1 and 2. Bothe PP1 and PP2 presented students with programming exercises based on CodeRunner using Python. The students physically attended the lab and attempted the questions on the computer. This was conducted group-wise on different dates, as the number of workstations available in the computer lab was not sufficient for the whole class to be accommodated at the same time. Therefore, a large pool of questions was developed for this purpose as introduced in Section 4.10.

Observing Figure 2, a significant difference between the scores of the common group and the other groups can be identified. All groups found PP2 harder than PP1. Students in the common group performed significantly better in both PP1 and PP2. The need to obtain a competitive grade for the module can be considered as one

of the main reasons for the difference. The students in TE, ERE and TLE groups did not have to push for higher grades as their engineering discipline selection had already been done.

However, this should be discussed as a matter of developing appropriate attitude towards learning programming. Students tend to take the module merely as a requirement, but do not see the importance of learning programming and successful completion. Awareness should be made on the advantage of taking the course and learning programming.

5.3 Interactive Auto-Graded Assignments

One of the key aspects of the process was pushing for automatically graded programming assignments. To do a meaningful assessment in a programming course, hands-on programming components should be included. Grading of programming assignments is generally time consuming. For large classes, grading assignments individually and providing feedback becomes increasingly challenging. This is where the importance of auto-graded programming assignments is highlighted.

Figure 3 illustrates an example of using CodeRunner on Moodle for an interactive auto-graded programming assignment in Python. The student is provided with details on the expected behaviour of the program, and they can use the available space to write their code. The "Check" button can be used to test whether the program works as expected. The instructor can set-up several test cases while preparing the question. Test cases can either be specified as 'hidden' or 'visible'. The program must pass all test cases to be given full marks. It can be set-up to penalise the students based on the number of times they use the "Check" button which may lead to partial marks being given to the answer at the end. If there are errors in the code, or if some test cases fail, the CodeRunner interface will display alert messages accordingly. The student can consider such information and further refine their answer.

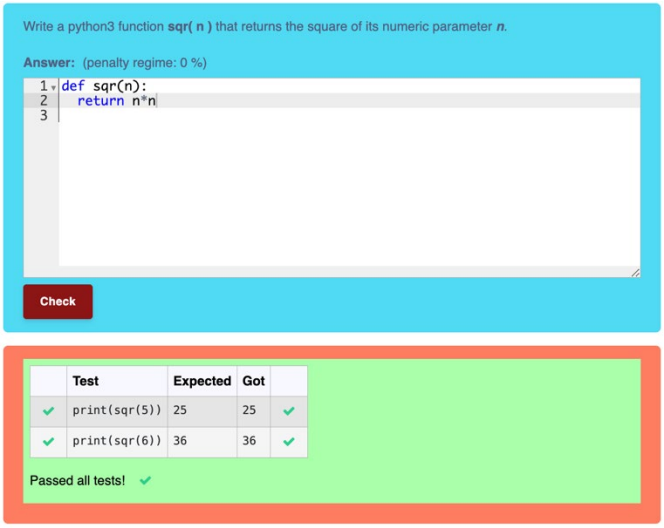

Figure 3: Example of an interactive auto-graded programming question using CodeRunner on Moodle

5.4 Utilizing Blended Learning

The delivery of the module included both face-to-face interactions as well as computer-mediated activities in the balance. Live interactive discussion sessions and one-on-one sessions with the teaching assistants maintained the level of face-to-face interactions. Recorded video lectures, text content, tutorials, lab sheets and other resources made available via Moodle encouraged self-study and helped prepare in advanced for the face-to-face sessions. Interactive, auto-graded programming assignments with feedback provided the means of conducting effective assessments for an introductory programming module.

6. Conclusion

The current study identified the applicability of blended learning for the delivery of an introductory programming course for an undergraduate degree programme. The delivery of the module faced multiple challenges due to the large number of student enrolments. Maintaining the level of face-to-face interactions became increasingly difficult and thus the teaching staff had to look for alternate and innovative ways to deliver the module efficiently. Adversities connected with limited availability of manpower as well as

infrastructure was countered by the introduction of computer-mediated interactions. The time constraints associated with grading assessments were balanced by the introduction of auto-graded assignments with the use of tools available in the learning management system. The findings of this study will be useful for institutions and academic staff in designing and delivering similar modules.

7. Limitations

The notable limitation of the approach is the conventional methods used for the final assessment. The papers had to be manually marked, in the expense of a considerable amount of time and human resources, even though it only contained MCQs. The best possible alternative is to move to computer-based examination, which can be facilitated by a quiz on Moodle. For MCQs, the quiz can be auto graded, taking out the grading burden completely. The only reason which prevented that was the unavailability of sufficient number of computer workstations for all students to attempt the examination at the same time.

A further suggestion would be to conduct the examination group-wise. Assuming that there is a lab facility with 200 usable computers, it can be suggested to prepare five distinct versions of the examination paper and let groups of 200 take the examination at different times. This would indeed require a considerable attention in preparing the questions so that fairness is maintained.

8. Future Work

The procedures described in this study should be subject to change and continuous evolvement. Some of the methods described here are purely experimental and it will take repetitive usage to determine the effectiveness. Continuously improving tools and technologies support educators to experiment with novel methods and combinations. Tools like CodeRunner thus far have been effectively deployed and the educators should look forward for better and enhanced tools and methods which will provide a better teaching and learning experience. Furthermore, online interactive coding tools like Trinket (Trinket) can be utilized for improved interactivity in programming courses.

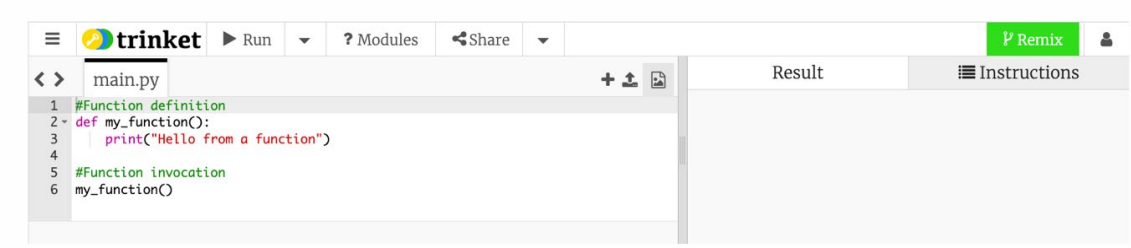

Figure 4: Online interactive coding tool, Trinket

Tools like Trinket can be seamlessly integrated to the learning management system, Moodle, to provide an uninterrupted and complete learning experience to the students.

References

Bogdanovych, A. and Trescak, T. (2016) Teaching Programming Fundamentals to Modern University Students. In *Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016) - Volume 2*, pages 308-317 ISBN: 978-989-758-179-3.

Cable, C., Knab, M., Tham, K., Navedo, D., and Armstrong, E. (2014) Why are you here? Needs analysis of an interprofessional health-education graduate degree program. *Adv Med Educ Pract*. 2014;5:83-88 https://doi.org/10.2147/AMEP.S60211.

Ericson, B. J., Rogers, K., Parker, M., Morrison, B., and Guzdial, M. (2016) Identifying Design Principles for CS Teacher Ebooks through Design-Based Research. In Proceedings of the 2016 ACM Conference on International Computing Education Research (Melbourne, VIC, Australia) (ICER '16). Association for Computing Machinery, New York, NY, USA, 191–200. https://doi.org/10.1145/2960310.2960335

Garrison, D. R., & Kanuka, H. (2004). Blended learning: Uncovering its transformative potential in higher education. *Internet and Higher Education, 7*, 95–105. https://doi.org/10.1016/j.iheduc.2004.02.001.

Graham, C. R. (2006). Blended learning systems: Definition, current trends and future directions. In C. J. Bonk & C. R. Graham (Eds.), *The handbook of blended learning: Global perspectives, local designs* (pp. 3–21). San Francisco: Pfeiffer.

Kanika, K., Chakraverty, S., and Chakraborty, P. (2020). Tools and Techniques for Teaching Computer Programming: A Review. Journal of Educational Technology Systems. 49. 170-198. 10.1177/0047239520926971.

Kizilcec, R. F., Kambhampaty, A. (2020) Identifying course characteristics associated with sociodemographic variation in enrollments across 159 online courses from 20 institutions. PLoS ONE 15(10): e0239766. https://doi.org/10.1371/journal.pone.0239766.

Koulouri, T., Lauria, S., & Macredie, R. D. (2014). Teaching introductory programming: A quantitative evaluation of different approaches. ACM Transactions on Computing Education, 14(4), Article 26.

Kunkle, W. M., & Allen, R. B. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. ACM Transactions on Computing Education, 16(1), Article 3.

Liu, X. (2014). Reform and practice in teaching data structures. In Zheng, D. (Ed.), Proceedings of the International Conference on Education Management and Management Science (pp. 2013–2016). CRC Press.

Lobb, R., and Harlow, J. (2016). Coderunner: A tool for assessing computer programming skills. ACM Inroads, 7(1), 47-51.

Monika, Bala, J. and Sunita. (2023) Scope and Challenges of Multimedia in Education Sector. IJFMR Volume 5, Issue 3, May-June 2023. DOI 10.36948/ijfmr.2023.v05i03.3868.

Moodle.org. https://moodle.org/. Accessed 26 May 2024.

Norberg, A., Dziuban, C. D., & Moskal, P. D. (2011). A time-based blended learning model. *On the Horizon*, *19*(3), 207–216. https://doi.org/10.1108/10748121111163913.

Peteranetz, M. S., Flanigan, A. E., Shell, D. F., & Soh, L. (2018). Helping engineering students learn in introductory computer science (CS1) using computational creativity exercises (CCEs). IEEE Transactions on Education, 61(3), 195–203.

Prensky, M. (2007). Paragon House. *Digital Game-Based Learning*.

Project Jupyter. "https://jupyter.org/" . Accessed 26 May 2024.

Ross, B., & Gage, K. (2006). Global perspectives on blended learning: Insight from WebCT and our customers in higher education. In C. J. Bonk, & C. R. Graham (Eds.), *Handbook of blended learning: Global perspectives, local designs*, (pp. 155–168). San Francisco: Pfeiffer.

Trinket. https://trinket.io/. Accessed 26 May 2024.

Wang, Y., Hill, K. J., & Foley, E. C. (2017). Computer programming with Python for industrial and systems engineers: Perspectives from an instructor and students. Computer Applications in Engineering Education, 25(5), 800–811.

Yuan, H.-Y., Dai, J.-G., & Peng, J. (2015). C language programming course reform and practice of teaching. In Proceedings of the Conference on Education and Teaching in Colleges and Universities (pp. 92–95). Atlantis Press.

Zoom.us. One platform to connect. "https://zoom.us/". Accessed 26 May 2024.