Kodr: A Customizable Learning Platform for Computer Science Education

Amr Draz[1(✉)], Slim Abdennadher[1], and Yomna Abdelrahman[2]

[1] Computer Science and Engineering Department,
German University in Cairo, New Cairo, Egypt
{amr.deraz,slim.abdennadher}@guc.edu.eg
[2] Institut fr Visualisierung und Interaktive Systeme,
University of Stuttgart, Stuttgart, Germany
Youmna.Abdelrahman@vis.uni-stuttgart.de
http://met.guc.edu.eg
http://vis.uni-stuttgart.de

Abstract. There are innovative systems designed for computer science education that teach programming concepts. However, many of them lack formal testing and comparison in a real course setting. This work intends to introduce a tool for teaching, evaluating, and assessing computer science students. Kodr is a modular gamified learning platform designed to evaluate varying problem types through gathering data about students performance. We conducted two studies in the wild with more than one thousand students to evaluate the initial design of Kodr. The first study evaluated two methods of teaching. The first method is to solve programming problems from scratch, the second, is to debug an incorrect solution of those problems. The results of the study yielded no significant difference between the two styles. The second study found significant positive correlations between Kodr's activity data and student's final course grades. Qualitative feedback gathered from students also evaluated Kodr as quite helpful.

Keywords: Computer education · Gamification · Debugging · Python · Web-based · Offline-ready

1 Introduction

It is a general conception that computer science is difficult to learn and that most of the programming courses have a very high failure rate [5]. There are several factors that can explain students' failure to acquire programming skills such as problem solving abilities, self-efficacy and an inability to form the correct mental model [8]. Therefore, recent years have witnessed a growing interest in fostering computer science education due to a large demand in labor force, as well as a call to develop computational thinking abilities in young students. This interest fostered an environment for innovation in computer science teaching pedagogy. In order to investigate various teaching methods suitable for teaching programing

K. Verbert et al. (Eds.): EC-TEL 2016, LNCS 9891, pp. 579–582, 2016.
DOI: 10.1007/978-3-319-45153-4_67

in an introduction to computer science course, we developed a teaching tool named Kodr. Kodr is a modular customizable gamified learning platform used to run and track student performance through coding challenges. Kodr combines several features from tools like Coding Bat, Python Tutor [4], Pythy [3], PILeT [1], as well as TurningCraft's CodeLab [2]. Kodr is web-based and offers offline execution of Python code and programing assignments similar to Pythy. Kodr possesses PILeT's capability of accommodating varying problems types designed to test different teaching methods. Kodr hosts a wide variety of programing problems with varying difficulty, similar to Coding Bat. Kodr also offers teachers the option to fully customize the coding problems similar to CodeLab (Fig. 1).

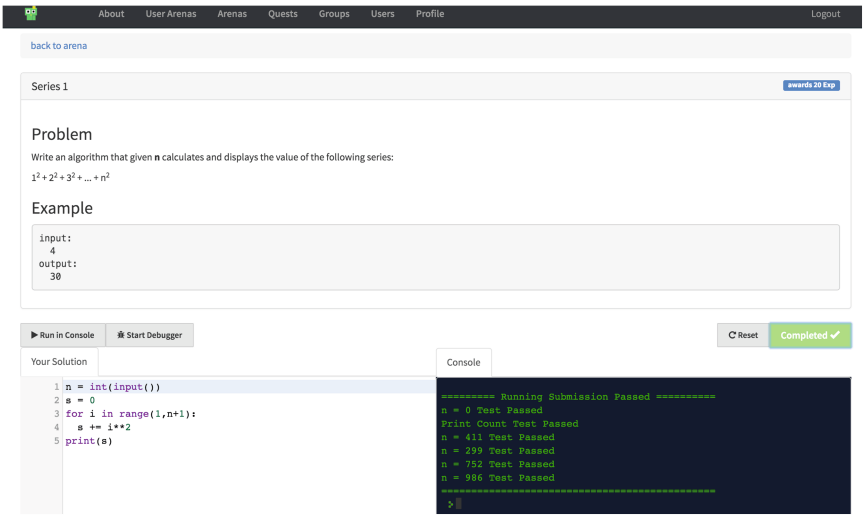

Fig. 1. Example of a Challenge in Kodr. Here you see an editor with debugging capabilities, a description section for the problem, and a console output section where output and submission result is printed.

Kodr extends on tools such as Pythy and Python Tutor by having an offline debugger. It helps novices while tracing their code. Additionally, it provides problems and assignments in a gamified context presented as challenges in arenas and quests with achievements respectively, awarding points on completion. Kodr has support for Javascript, Java, and Python programing challenges. In addition Kodr was designed with a completely modular challenge module capable of hosting programing game challenges similar to games like Help Gidget [6] as well as media manipulation challenges, accommodating a wider possible set of problems types than PILeT[1]. Kodr uses programmable tests suits similar to behavior driven tests which offers more flexibility for teachers than tools such

[1] A stand alone version of Kodr's python challenges can be viewed at pythondebugger.xyz.

as CodeLab and Coding Bat. The test suite allows submission evaluation not just through input and output but also through static analysis of the submitted code itself. The test suit also features tags that can be used to award students additional points and badges. Kodr keeps track and record data about students behavior patterns as they solve problems in order to collect evidence for evaluating student's problem solving abilities. It reports on student's progress for teachers and provides a training data set for turning the tool into an adaptive tutor that can automatically adapt itself to the student based on their perceived performance.

2 Evaluation

For Kodr's first design iteration, a study was carried out on 1078 engineering and business informatics students. 830 were male students and the rest were female students with an average age of 18. All students had no previous background in computer science/programing. The testing phase lasted for one semester spanning over 4 months.

A Pearson product-moment correlation coefficient was computed to assess the relationship between the number of challenges completed on Kodr and course grade. There was a positive correlation between the two variables: $r = 0.572$, $n = 1076$, $p < 0.01$. The positive correlation between challenge completion and final grade presents an indication that the more students engage on the Kodr system, the higher the likelihood the student would get a good grade.

250 students also agreed in a questionnaire administered post course. voting 58 % for "I had no trouble figuring out how to use Kodr". 88 % for "Kodr made it easy for me to find out how well my programs' were close to the problem solution". 80 % for "I generally found Kodr helpful in supporting my studies".

The most preferred features in Kodr for students were, being able to revisit previous lab problems (76 %), getting automatic feedback about their solution (65 %), being web-based (64 %), and being able to step through code using the debugger (63 %).

Kodr was designed to evaluate teaching methods. Accordingly, we carried out during the semester an experiment comparing the effect of whether, when faced with solving programing problems, starting from scratch (code first) as opposed to debugging a buggy solution of the same problem (debug first), would aid novice programmers in developing better understanding of programing concepts.

The study was carried out in our course delivered by two lecturers and thirteen teaching assistants. To control for the large variability, a semi random assignment was carried out across tutorial groups and lecture groups such that every teaching assistant and instructor taught both groups equally across majors. Only 449 of the 1078 students; all freshmen average 18 years of age, with minimal to no knowledge of computer science or programing, fit the criteria and opted in for completing both pre and post test. The experiment followed a between-subjects design with a pre-post test. Participants where administered the pre-test in their first lab prior to any exposure to programing concepts. The

post-test was administered after the midterm (2 month), which marks the end of the programing and algorithms section of the course. The questionnaire administered was taken from [7], as it was validated. An independent-samples t-test for difference in pre-post tests noted an almost significant difference between the control ($M = 1.81, SD = 1.701$) and experiment groups ($M = 2.11, SD = 1.75$); $t(448) = 1.86$, $p = 0.064$, which was insufficient to reject the null at <0.05 but lied in the 90th percentile, which presents some room for future research.

3 Conclusion

Kodr has been used to evaluate a teaching method and aid students in learning programing. The testing results show that the tool contributed positively in the delivery of the course content. Once the data gathered through out the semester will be analyzed, we will be capable to answer more questions about students' learning patterns and to train Kodr to become an adaptive tutor capable of modulating challenge type and difficulty.

References

1. Alshaigy, B., Kamal, S., Mitchell, F., Martin, C., Aldea, A.: Pilet: an interactive learning tool to teach python. In: Proceedings of the Workshop in Primary and Secondary Computing Education, WiPSCE 2015, pp. 76–79. ACM, New York (2015). http://doi.acm.org/10.1145/2818314.2818319
2. Barr, V., Trytten, D.: Using turing's craft codelab to support CS1 students as they learn to program. ACM Inroads **7**(2), 67–75 (2016). http://doi.acm.org/10.1145/2903724
3. Edwards, S.H., Tilden, D.S., Allevato, A.: Pythy: improving the introductory python programming experience. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE 2014, pp. 641–646. ACM, New York (2014). http://doi.acm.org/10.1145/2538862.2538977
4. Guo, P.J.: Online python tutor: embeddable web-based program visualization for CS education. In: Proceedings of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE 2013, pp. 579–584. ACM, New York (2013). http://doi.acm.org/10.1145/2445196.2445368
5. Guzdial, M., Soloway, E.: Teaching the nintendo generation to program. Commun. ACM **45**(4), 17–21 (2002). http://doi.acm.org/10.1145/505248.505261
6. Lee, M.J.: How can a social debugging game effectively teach computer programming concepts? In: Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research, ICER 2013, pp. 181–182. ACM, New York (2013). http://doi.acm.org/10.1145/2493394.2493424
7. Lee, M.J., Ko, A.J.: Comparing the effectiveness of online learning approaches on CS1 learning outcomes. In: Proceedings of the Eleventh Annual International Conference on International Computing Education Research, ICER 2015, pp. 237–246. ACM, New York (2015). http://doi.acm.org/10.1145/2787622.2787709
8. Ramalingam, V., LaBelle, D., Wiedenbeck, S.: Self-efficacy and mental models in learning to program. SIGCSE Bull. **36**(3), 171–175 (2004). http://doi.acm.org/10.1145/1026487.1008042