IPRACTICE: A SELF-ASSESSMENT TOOL FOR STUDENTS LEARNING COMPUTER PROGRAMMING IN AN URBAN CAMPUS*

Benito Mendoza, José Reyes-Alamo, Huixin Wu, Aparicio Carranza
New York City College of Technology – CUNY
300 Jay Street, Brooklyn, NY 11201
718.260.5885, [bmendoza, jreyesalamo, hwu, acarranza]@citytech.cuny.edu

Laura Zavala
Medgar Evers College – CUNY
1650 Bedford Ave., Brooklyn, NY 11225
718-270-6128, rzgutierrez@mec.cuny.edu

ABSTRACT

It is very common that students attending urban campuses do not live in university residence halls or dorms. In our case, many of our students spend around one hour and a half each way between home and school. We have developed an App for mobile devices that provides computer programming exercises and quizzes to help students practice and self-assess their knowledge during their commuting time. The most relevant part of this App is that it can be used off-line while the students travel in the subway without an Internet connection. In this paper, we discuss the features of this App and how it has been used to help increase the passing rate, from 65% to 76%, in an introductory computer programming course.

INTRODUCTION

As math and other sciences, computer programming is an intellectually rigorous

subject, which requires a lot of practice to learn and master. One of the challenges while teaching computer programming is to provide students with several examples or exercises of the same topic or concept so that the students can generalize the concept and it becomes part of their background knowledge. Research suggests that students learn by acquiring particular schemas or patterns (e.g., pairings of verbal content with mathematical formulas) and that those schemas must be formed at a sufficiently abstract level to support accurate generalization to new problem types [1] [2]. To learn to program, students need lots of practice such that their volatile declarative knowledge becomes more robust procedural knowledge [3].

For students in urban colleges, finding time to practice and work on these types of subjects might become challenging. It is very common that students in urban campuses do not live in university residence halls or dorms. If the campus happens to have dorms, students may stay there for only one or two years and then find that they are more comfortable (personally and economically) joining friends to rent an apartment or a house. In particular, CITY TECH students spend more than one hour and a half each way between home and school; CITY TECH does not offers dorms. Although most of the students have mobile computing and there are many online tools and tutorials to practice computer programming [4] [5], most of their commuting time is in the subway, sometimes packed, with no Internet connection. Some of our students, even though registered as full-time students, have part-time jobs. All this makes their schedules tight to study and practice. Wouldn't it be nice if they could squeeze in a little extra time to practice on the train or bus?

In this paper, we discuss how iPractice, a self-assessment tool in a form of a mobile App, has helped our students to improve their programming skills and had helped us to increase the passing rate in our introductory programming course. When compared to previous semesters when this tool was not available, positive outcomes have been observed: i) the course passing rate increased about 11%; ii) the number of programming constructors in final projects increased by almost 15%.

Background

According to [6], the passing rate of introductory computer programming courses around the world is 66.77%. For the last past few years, our course titled Logic and Problem Solving (an introductory course to computer programming for an electromechanical engineering technology program) had suffered a very low passing rate of 65%. The original course, updated in the Fall of 2011, made use of AppInventor as tool/programming language. AppInventor is very appealing and accepted by the students because they use their phones heavily and the tool allows them to create mobile apps. However, we found that with this tool, somehow, students can create some nice Apps mainly using events with little or no programming. Some final projects of the course had a very nice user interface (graphics, sounds, etc.), with relative good functionality, but few or no computer programming structures (if-then statements, loops, functions, etc.). Also, students devote more time into the looks of the App than in the functionality. Since

AppInventor is cloud-based, instructors find it slow for showing different examples during the limited class time. Further, compiling and loading an App to a phone or an emulator take several minutes.

Project Goals

The main objectives of the project presented here are: i) to increase the passing rate of students and ii) to improve the quality of the final projects, which, in our opinion reflect the understanding and mastering of the core concepts taught in the aforementioned course. The first step we took towards this objective was re-designing the course.

While re-designing this course, our goal was to eliminate the above-described issues but at the same time keep the enthusiasm of students, which are very interested in mobile applications. In the Fall of 2012, we introduced a new syllabus. In a fifteen weeks semester schedule, we decided to introduce all the content of the syllabus (variables, basic data types, statements, Boolean expressions, branching, repetition, strings, lists, and functions) with Python, using [7] as textbook during the first six weeks. Then, we revise everything again with AppInventor for other 6 weeks using [8] as textbook. The remaining 3 weeks are used to work on a final project, which consist in developing a mobile App. We decided to introduce Python because it would allow instructors to show more examples during the class time, 2 hours a week. As mentioned before, AppInventor takes several minutes to compile a program and load it into the phone or emulator, this represents a problem for a short-time class, opening and uploading different examples is time consuming. Running an example in python is way faster, this allows to show several examples during the class time. We created a video library, a set of short videos to complement the lectures. But, most importantly, we created the first version of iPractice, which only operated on one of the two modes it can currently operate.

METHODOLOGY

According to [9] learning is a process, not a product, and it takes place in the mind and leads to change. Learning involves change, over time, in knowledge, beliefs, behaviors, or attitudes. More importantly, learning is not something done to students, but rather something students themselves do [10].

To develop mastery, students must acquire component skills, practice integrating them, and know when to apply what they have learned [11]. To learn to program, students need lots of practice. Students must not only develop the component skills and knowledge necessary to solve complex problems, but they must also practice combining and integrating them to develop greater fluency and automaticity [3]. Also, researchers in computer science education emphasize the importance to learn and master how to read and trace code, as a required step to become proficient writing code [12].

iPractice was thought to be used off-line in a mobile device, although there is an on-line web version too, during the commuting time in a possibly crowded subway. It was thought to help developing and reinforcing the learner's component skills and the

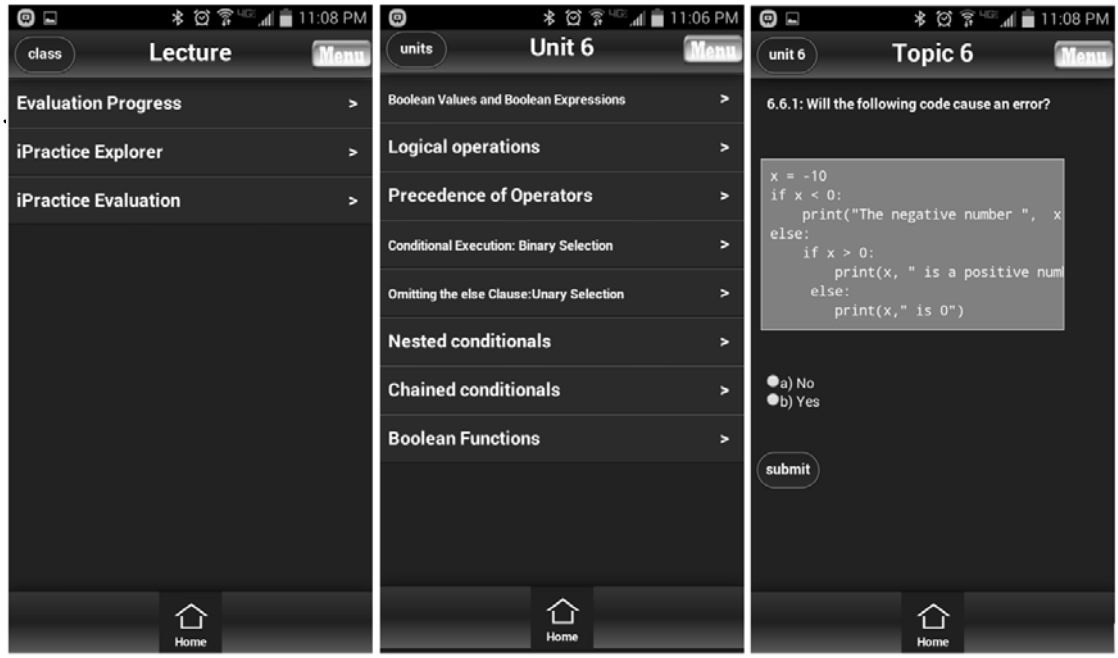

FIGURE 1. SCREENSHOTS OF THE MOBILE VERSION OF IPRACTICE OPERATING ON EXPLORER MODE. LEFT, THE MAIN OPTIONS. CENTER, ONE MENU AFTER SELECTING THE EXPLORER MODE. RIGHT, ONE QUESTION ON UNIT 6, TOPIC 6.

acquired knowledge from classroom and textbook by presenting problems to be analyzed and questions. It is not an environment where the learner can type and run code.

iPractice has two operation modes. The first one is called Explore mode, in the Fall of 2012 it was the only mode available. In this mode, the students can go over questions and problems related to every unit or chapter of the course book. Students can select different levels, unit, section, sub-section, etc., according to their needs and revise and practice related problems as many times as they want. After, submitting the answer to a given problem, iPractice provides immediate feedback, whether the answer was correct or not and why. In case the student does not understand the problem, he/she would know what part of the book should study or check again. Figure 1 shows an example of iPractice operating in this mode.

The second operation mode of iPractice is called Evaluation mode, it was introduced in the Spring of 2013. Under this mode, students complete a quiz that covers a given unit or chapter and get feedback at the end of the quiz. The result of the quiz is sent to the corresponding instructor. Instructors assign these quizzes every week to monitor the student progress. The student can answer the quiz even if there is no Internet connection, as soon as the App finds connection, it would send the results to the instructor. Figure 2 presents an example of iPractice operating on Evaluation mode.

Goal-directed practice, coupled with targeted feedback, enhances the quality of students' learning. With iPractice, students can try a given exercise as many times as they want (Explore mode). However, they have to complete a certain quiz requirements (quizzes in Evaluation mode) as part of their grade and course goals.

RESULTS

To evaluate the effectiveness of our work we collected data (passing rate and final

projects) from two semesters previous to the existence of iPractice (Fall 2011 and Spring 2012) and compared it against data collected during three semesters when a version of iPractice version was available, Fall 2012, Spring 2013, and Fall 2014. The Evaluation mode was not available during Fall 2012. We analyzed 4 sections in each semester, the ones that were taught by the professors involved in this project. A typical section has an average of 17 students, with a maximum of 20.

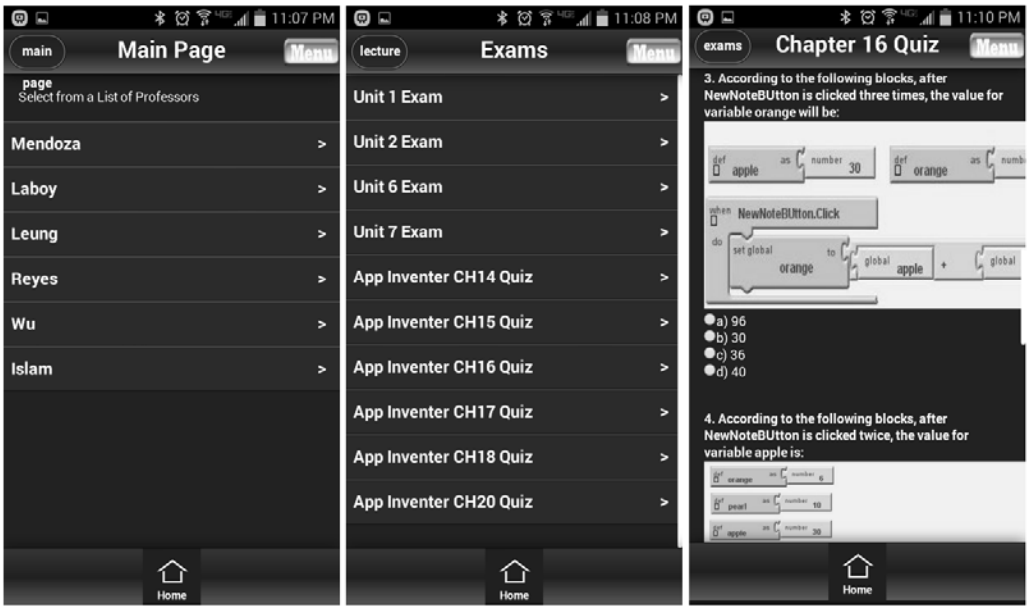

FIGURE 2. SCREENSHOT OF THE MOBILE VERSION OF IPRACTICE OPERATING ON EVALUATION MODE. LEFT, LIST OF PROFESSORS TEACHING THE COURSE. THE CENTER IS A LIST OF AVAILABLE QUIZZES. RIGHT, AN EXAMPLE OF AN EXAM.

Passing Rate

Figure 3 shows a comparison of the passing rates. The average passing rate (percentage of students with a D or higher grade) during the two semester previous to iPractice was 65%. Notice that there was almost no difference in the passing rate between former semesters and Fall 2012, when only iPractice Explore mode was available. At that moment there was no way to monitor how iPractice was used. However, the story is different when comparing the last two semesters; with evaluation mode students are required to use iPractice as weekly homework. The average passing rate of these semesters, when the two modes of iPractice were available, is 76%. This is a passing rate increment of more than 11%, when comparing the last two semesters to previous semesters. The results, indicate that the use of the Evaluation mode of iPractice has a positive impact in the passing rate, maybe because completing those quizzes is now part of the students' grade.

Figure 4 shows the grades distribution across the semesters. Although the number of students getting an A grade decreased for some semesters after Fall 2011, the rate of

A grades is about the same among Spring 2012, Fall 2012, and Spring 2013. This result could indicate that iPractice do not influence the grade of good or average students, most of the passing students get B's or C's. However, the results of Fall 2014 show something different. Most of the students got A's, and the number of D's was decreased. This might be an outlier. We will need more data from future semesters to observe this phenomena. Obviously, the number of students getting an F decreased in Spring 2013 and Fall 2014 for about 12%, compared with Fall 2011 and Spring 2012.

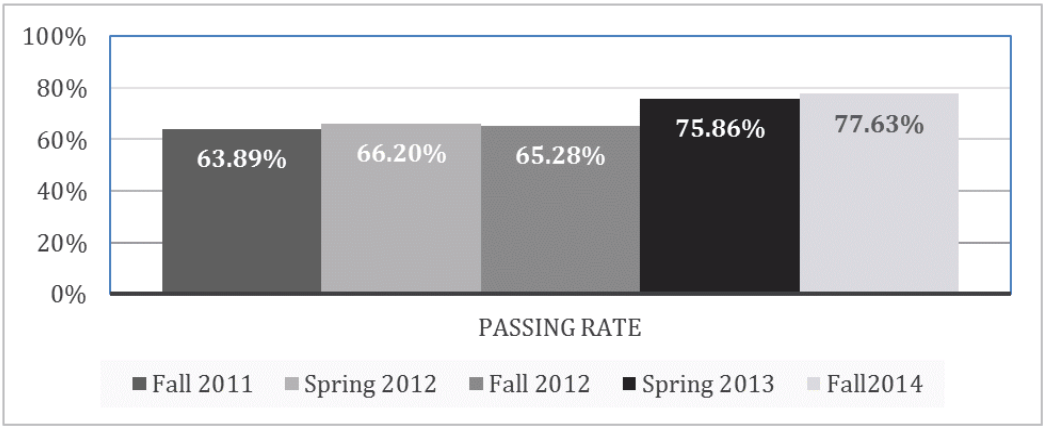

FIGURE 3. PASSING RATE ACROSS DIFFERENT SEMESTERS

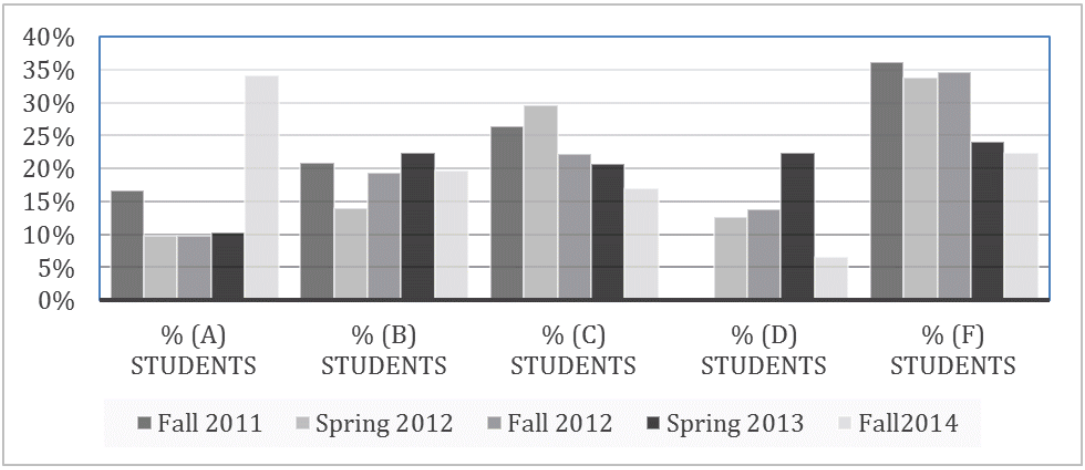

FIGURE 4. GRADE DISTRIBUTION ACROSS DIFFERENT SEMESTERS

Students' final projects

For their final project, a mobile App of their election using AppInventor, students can work individually or in pairs. To evaluate that students put in practice they learned and the impact of iPractice, we check their final projects. In addition to the App

functionality and the interface, we check that students use the computer programming concepts, patterns, and constructors taught in the course. We analyzed more than 35 final projects of each semester. As metric to evaluate the projects, we filtered out the projects with less than 3 important programming structures or patterns, among them: if, if-then, loops, functions or procedures, and lists. Thus, we counted in how many of this projects use 3 or more important programming structures.

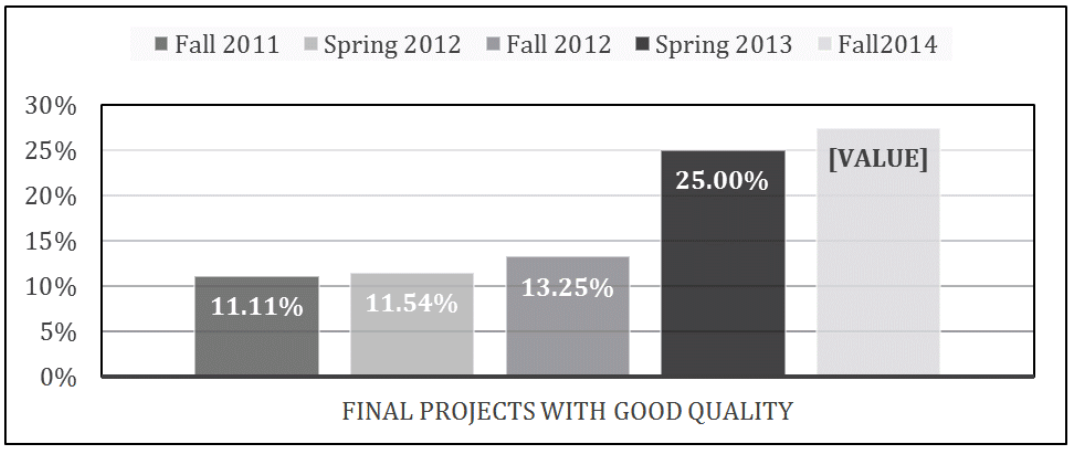

FIGURE 5. COMPARISON OF STUDENTS' FINAL PROJECT. PERCENTAGE OF PROJECTS USING THREE OR MORE CORE COMPUTER PROGRAMMING STRUCTURES (IF-THEN, LOOPS, FUNCTIONS, LISTS, ETC.)

In the semesters previous to iPractice, in average, less than 12% of the projects met the criteria. In Fall 2012, when the first iPractice was introduced, this percentage was increased to 13.25%, an increment of more than 1%. Finally, in Spring 2013 and Fall 2014, when the full version of iPractice was ready, this percentage went up higher than 26%, an average increment of almost 14%. Figure 5 shows a graphical comparison of the numbers just mentioned.

CONCLUSIONS AND FUTURE WORK

iPractice has allowed students to spend extra time practicing and be better prepared for their exams and final projects. Since its full version was introduced, the passing rate of our course has increased by 11%. A new version is being prepared to be used in another college in a C++ programming course. New problems are being developed. Creating problems and questions is the most challenging part while developing a tool like iPractice. We are working towards incorporating Automated Item Generation (AIG) into our tool. This technique makes use of templates with embedded variables to represent the structure of a problem. The templates can be instantiated to specific questions by an algorithm that replaces the variables with values. Thus, a large amount of questions can

be created from one single template.

REFERENCES

[1] S. P. Marshall. “Schemas in Problem Solving”. New York: Cambridge University Press, 2007.

[2] C. A. Weaver and W. Kintsh. “Enhancing students' comprehension of the conceptual structure of algebra word problems”. Journal of Educational Psychology, vol. 84, no. 4, pp. 419-428, 1992.

[3] F. E. Ritter, K.-C. Yeh, M. A. Cohen, P. Weyhrauch, J. W. Kim and J. N. Hobbs. “Declarative to Procedural Tutors: A Family of Cognitive Architecture-based Tutors”. In 22nd Annual Conference on Behavior Representation in Modeling and Simulation (BRiMS), Ottawa, Canada, 2013.

[4] “codecademy,” [Online]. Available: http://www.codecademy.com/. [Accessed 20 April 2015].

[5] “Computer Science Circles”. [Online]. Available: http://cscircles.cemc.uwaterloo.ca/. [Accessed 20 April 2015].

[6] C. Watson and F. Li. “Failure Rates in Introductory Programming Revisited”. In Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education, Uppsala, Sweden, 2014.

[7] A. B. Downey. “Think Python: How to Think Like a Computer Scientist”. O'Reilly Media, 2012.

[8] D. Wolber, H. Abelson, E. Spertus and L. Looney. “App Inventor: Create Your Own Android Apps”. O'Reilly, 2011.

[9] R. E. Mayer. “The promise of educational psychology”. Volume 2: Teaching, Upper Saddle River, NJ: Merril Prentice, 2002.

[10] S. A. Ambrose, M. W. Bridges, M. DiPietro, M. C., Lovett, and M. K. Norman. “How Learning Works: Seven Research-Based Principles for Smart Teaching”. San Francisco, CA: Jossey-Bass, 2010.

[11] J. Sprague, D. Stuart and D. Bodary. “The Speaker's Handbook”. Wadsworth Cengage Learning, 2012.

[12] M. Linn and M. J. Clancy. “The case for case studies of programming problems”. Communications of the ACM, vol. 35, no. 3, pp. 121-132, 1992.