A Low Stakes Introduction to Computer Programming

Lori Carter
Mathematical, Information, Computer Science
Point Loma Nazarene University
San Diego, CA, USA
lcarter@pointloma.edu

***Abstract*— This Innovative Practice Category Work in Progress paper introduces a one-unit "absolute beginner" computer programming course. The goals of increasing retention and mitigating disparity in experience levels for majors in Computer Science, and increasing the interest of and accessibility for non-majors in computing were similar to those of other redesigns of introductory computing courses. The big difference was the low-stakes format. The course was designed for potential CS majors with no experience, K-12 Education majors, and other majors who want to see what computer programming is and how it might fit into their future. As a one-unit course offered during the school year, many students can take it at no additional cost, and without much risk of it adversely affecting their GPA or other courses. As a class that is not a prerequisite for any other, the class can be taught with wide interdisciplinary focus and limited technical language.**

Keywords—computer programming, introduction, beginner, K-12 educators, non-majors, low-stakes, retention

I. INTRODUCTION

In the Spring semester of 2018, Point Loma Nazarene University (PLNU) launched a one-unit course entitled "Computer Programming for the Absolute Beginner." In an effort to make the course accessible to a diverse audience, it had a prerequisite of only basic algebra, was a half-semester course, and used a text written specifically for the course. The textbook and lectures, while technically accurate, strove to use as few new terms as possible, so students could learn about programming without being overwhelmed with a new vocabulary at the same time. The course was designed with three populations in mind: students from other majors who wanted to see how programming might fit with their interests, students who thought they might want to major in computer science but had no previous experience, and education majors who wanted to be prepared to introduce computer science to their K-12 students. We were hoping that the new course would meet the needs of these three groups better than our traditional introductory course, and, as an added benefit, would alleviate some of the over-crowding in those three-semester-unit CS1 sections.

A. Students from other majors

Computer programming is becoming a skill that is required for jobs in other disciplines. In a recent Forbes article, the author suggested eight jobs that would be easier to land if the applicant had some experience in computer programming [1]. These jobs included technical writer, instructional designer, user experience designer, and product manager. Burning glass, a jobs research company, reported that people in art and design, the natural sciences, and data analysis could benefit from coding experience as well [2]. At PLNU, business professors are encouraging students who want to work in tech companies to gain some experience, and journalism professors understand that forensic journalism requires computing skills. Studies (as well as our experience) have shown that students from these populations who take the traditional CS1 course often drop the course, citing that the unexpectedly high work load for the non-required course keeps them from putting in the time they need for their major courses [3].

B. Potential Computer Science Majors with no Experience

While the number of computer science majors has risen dramatically in the last few years, retention is still an issue [4]. A comprehensive study performed by Peterson et.al, at a large university revealed that over 17% of the students that started CS1 during the session dropped prior to completing the class [3]. Of those students who were interviewed regarding why they dropped, almost one half cited lack of prior experience. They were intimidated by the students with experience who seemed to be catching on more quickly. Several stated that if they had just had a little knowledge, it would have helped a lot. At PLNU we repeatedly saw that although the majority of students in our CS1 courses did not have prior experience, those who were struggling cited their lack of experience as a reason for their inability to catch on quickly. This reasoning may be based in fact. Flores reports that students are more uncomfortable with learning new skills and concepts if they were not introduced to them prior to their pre-teen years [5]. While all students are introduced to subjects like mathematics, history, and science in elementary school, few are introduced to text-based computer programming.

This lack of experience with computer programming leads to another reason revealed by Peterson to explain why

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

students drop CS1. They are simply surprised at the amount of time and work that it takes. We have observed the same in our classes. Walker, a member of the ACM Committee on Retention points out that "Incoming students may have encountered fictionalized media representations of computing that provide a distorted or inaccurate perspective on the discipline and the people in it [6]." Several of the students who are enrolled in our current Absolute Beginner course are students who think they want to change their major to computer science. This low stakes course will at least give them an idea of what to expect before they commit to the change.

C. K-12 Education majors

In 2016, then President Obama announced his "Computer Science for All" initiative, declaring computer science to be a "new basic skill necessary for economic opportunity and social mobility [7]." Organizations such as Hour of Code [8], CodeHS [9], and Girls Who Code [10] have developed curriculum to introduce students to coding as early as kindergarten. More and more primary and secondary schools are experimenting with these curriculums. However, we were continually hearing through informal conversations with current K-12 teachers that they were either uncomfortable with teaching the lessons (often block-based programming exercises) or were afraid they would not be able to answer questions posed when students wanted information on programming beyond the simple lessons provided. Our education department was excited about the idea of introducing coding to their teachers **before** they interviewed for jobs and entered the classroom.

D. Positive affect on CS professor workloads

In many universities, there are not enough faculty available to staff the CS1 courses when they are populated not only by hopeful CS majors, but many non-majors as well. CRA reports that in 2015, the non-majors out-numbered the CS majors in the CS1 courses designed for majors [4]! In the Peterson report, the majority of the CS1 drop-outs that were interviewed were non-majors. They cited the course being too hard, too time-consuming, and taking them away from their major courses. At PLNU, prior to this new course, the rapidly expanding CS1 course for majors was the only option. Our hope is that this new course will relieve some of the pressure from that course for those who just want a "taste" of CS.

II. Related Work and where this course fits

Introductory computing classes are constantly being redesigned to better meet the needs of computer science majors and non-majors. Research, industry needs, and enrollment fluctuations often drive these changes. For majors and non-majors alike, classes are changed to be more engaging. Guzdial and Forte designed a compelling media computation course for non-majors at Georgia Tech that included Python programming and topics such as algorithmic thinking and efficiency [11]. Others have adapted this curriculum to meet more specific needs at their universities [12]. Cortina developed a course for non-majors at Carnegie Mellon which covered a variety of computing topics, but left out traditional programming [13]. Introductory computing courses are being designed to be more interesting to both majors and non-majors by combining computer science and "X." Successful examples including CS and Biology at Harvey Mudd [14] and the University of Illinois, Chicago [15] and CS and Literature at Wheaton College in Massachusetts [16]. Another successful approach has been to provide shorter, specialized courses for select students, such as the two unit introductory courses for future engineers in either MATLAB or Python that can be completed in 5-6 weeks at Cornell university [17].

Other universities have worked at revising their introductory courses for CS majors to aid in retention, address the disparity that exists in the experience level of incoming freshmen and to make programming more accessible by introducing basic programming or tough concepts with block-based languages such as Scratch [18] and Alice [19] and then transitioning to text-based languages. Pair programming [20] and peer tutoring [21] have also been shown successful at aiding retention in traditional CS1 courses. Summer "bridge" courses such as the one offered by Purdue University for incoming freshmen with no prior experience in computing [22] have been used to mitigate the difference in experience levels for CS1 students. Such summer courses generally incur an additional fee.

Like the courses just cited, the Absolute Beginner course at PLNU attempts to address the interest, retention, accessibility, and experience-disparity issues for potential CS majors and for non-majors using similar techniques. The big difference with this new course is the streamlined, low-stakes format. The secondary difference is in the intentional inclusion of education majors.

We have attempted to make the programming content compelling by using interdisciplinary examples without going deeply into secondary subjects (Biology, Literature, Media) that, while interesting, do take class time to develop and can feel overwhelming themselves [23]. We also incorporate drawing exercises that many students find to be motivating.

We do begin the course with a block-based language because that is what education majors were more likely to use in their K-12 classrooms and because they are accessible to a wide audience. We move quickly to Python, however, as our main purpose was to introduce computer programming. We do not cover topics such as algorithmic complexity because, while interesting, they do not meet the immediate needs of our audience or fit into our 1-unit constraint. Additionally, we limit the variety of technical terms and programming constructs both because of our limited time and to keep the content as simple as possible while still being informative. While we are not using formal pair programming or peer tutoring, we include lab time in the course and encourage students to work together. To ensure adequate in-class help, peers are used as lab assistants.

We expect a very narrow range of experience level as the course is clearly labelled for "Absolute Beginners."

Furthermore, it is taken in addition to the CS1 course and doesn't count towards the degree so there is no incentive for students with prior experience to take it. We hope that our one-unit course will aid in retention for the traditional CS1 course as the potential majors will now have some experience.

The main purpose of the new course is not to prepare the student for a specific next class or to teach the algorithmic thinking that is desired for all at technical universities, but rather to provide an experience that gives a knowledge base in computer programming, building confidence in the student's ability to choose the next step in computing, whatever that may be. Because it is not a prerequisite for any course, we are free to choose content based on the audience and the experience we want them to have. We are not forced to have them prepared with a certain vocabulary or set of skills.

All of this is offered as a one-unit course, open to anyone and, being offered during the school year, often easily added to the student's schedule at no additional cost. At PLNU, as is the case at many universities, the tuition is a flat fee for 12-17 semester units. Many students have one extra unit available, but they might not have two.

III. Course Design

The Computer Programming for Absolute Beginners course was designed with the audiences described in the introduction in mind. It meets for seven weeks, two days a week. Each class consists of a brief lecture followed by an assignment that is generally a combination of questions on the reading for that week and a short programming assignment. The programming assignments are interdisciplinary in nature, designed to show students how widely computing can be applied. During the first few weeks, most students completed the assignment during class time with the professor and a lab assistant available to help with any issue. As students became more confident and programming assignments more complex, whatever the student didn't finish was completed outside of class. The second class of the week also included a short written quiz designed to encourage the students to keep current.

A. The Course Content

Because one of the target audiences for the course is future K-12 educators, the first-day introduction to coding uses block-based languages (such as Scratch and Blockly) which many of the teachers will one day use when introducing coding to their students. Since we want to give education majors a broader view of programming, and are also using this class to give students an idea of what they might expect in a CS1 class, we immediately move on to Python.

The Python language was chosen because of its relatively simple structure. The lessons provided acquaint the students with syntax, input and output, variables, assignment statements, if statements, loops, algorithm development, testing, and debugging. The turtle drawing module is used to motivate the use of loops for repetition, and students are introduced to sample programs which have applications to the sciences, social sciences, business, and education. One example of an interdisciplinary programming project calculates Body Mass Index and makes recommendations based on the results. Another can be used to screen candidates for a state food assistance program.

For this first offering, all programs have been completed using the web-based tool Trinket [24]. Using Trinket provides a Python interpreter and editor to the students, allowing them to use their own computers without having to download or install software. Furthermore, we do not need to take the time to introduce a complex IDE. We only want to introduce programming.

B. The Textbook

One complaint that we have heard repeatedly in our traditional CS1 course is that it not only requires learning a new set of unfamiliar skills, but a new vocabulary as well. This can be overwhelming. The textbook used for this course was written with the goal of using as little technical language as possible. For example, while we talk about Python *instructions*, we don't use the relative synonyms *commands* or *statements*. When we talk about creating the algorithm, we don't actually use the word algorithm, but rather "set of steps." When we introduce "if" statements, we don't confuse the issue with the variety of forms an if statement could take (if-else, if-elif, switch), we stick to using one version to get the concept across. We do the same thing with loops. While the students may not be creating the most efficient code, they are getting the concepts in the most concise way possible. Each chapter is short, with many examples and illustrations. An attempt is made to include examples that appeal to a diverse group of students. This means that they are not highly mathematical, and include topics that would be of interest across genders.

IV. Impact on Enrolled Students

A. Student Enrollment

The first offering of the course began with an enrollment of 21 students. This is a large number at a small, liberal arts college for a class that is purely elective. Two of the enrolled students dropped almost immediately. The course started during the second half of the semester when many students are overwhelmed with the load they already have. Of the 19 students (12 female, 7 male) who attended for at least two weeks, two students dropped the course. One student was a chemistry major who thought the class was proceeding too slowly. The second was a business major who dropped because she thought, even with the name given, that it would be a data analytics course. This last case reinforces one of the reasons for the class. Students can find out what computer programming is without committing to a 3-4 unit course.

Of the students who remained after the drop date, the break-down of the enrollment is shown in Table 1. Almost all of the education majors signed up for the class at the encouragement of their advisors. The psychology major enrolled because she has been offered a fellowship in psychology that also involves computational techniques. She wanted to prepare herself. The business major knew that programming would help him in his future, but had tried

TABLE 1 CLASS PARTICIPANT BREAKDOWN

Student major	number
Business	1
Christian Ministry	1
CS – failed to pass CS1 (intend to try again) or CS new major with no previous experience	4
Education	7
Psychology	1
Sociology	1
Undeclared	2

unsuccessfully to teach himself. The CS majors were trying to gain confidence and experience. The other students were just curious.

B. The Impact

At the time of this writing, the seven-week class was finishing week six. The students were progressing faster than expected, and several chapters of the book had been re-written in real time to add additional material. An anonymous survey was administered to determine how the students felt about the class. Of the 17 remaining students, 16 took the survey. Many of the questions on the survey asked students to respond yes or no regarding whether a goal was met. It could be an objective for the course in general, or a goal that a student had in taking the class. These answers are recorded in Table 2. As you can see, most responses were very positive.

The students were also given an opportunity to explain their answers. All students felt that they had achieved their goal in taking the class. The goals stated by the students can be broken down into 4 categories. The numbers in parenthesis indicate the number of students expressing that goal.

- A better understanding of programming (8)
- A grasp of the basics to assist future K-12 students (3)
- Learning enough to move forward with personal study (3)
- Wanted to know if they could handle a programming class (4)

The numbers add to 18 (rather than the 16 survey takers) because a couple of students had multiple goals. Because there were seven education majors in the class, we would imagine that more of them were interested in learning the material not just for their own understanding, but so that they could pass it on but did not explicitly state that.

The two students who believed that the amount of technical language used could have been less also indicated that they found the book difficult to read at times. The other students were very positive about the level of difficulty with the class and the book. Some of the comments were:

"The class was fun!"

"Easy-to-understand concepts and language"

"The pictures and the step by step approach were incredibly helpful"

"This class breaks down difficult information into easy-to-understand language and concepts."

"This was a good way to be introduced to computing without the complexity"

"This was a good workload for 1 unit"

TABLE 2 POST-CLASS SURVEY RESULTS

Goal to be achieved	yes	no
Good feel for what computer programming is	16	0
Achieve your goal for taking the course	16	0
Not too much technical language in the class	14	2
I understand how programming can be used in my field	16	0
This class helped me make a judgement on taking future CS classes	15	0

That some students found the book difficult even with the copious examples and limited technical language is not surprising. Psychologist Martin Kutscher explains that many current students are not used to reading in any depth. In using hypertext (the mode of the internet) students are used to immediately finding the tidbit of information that they are looking for rather than following a line of reasoning. "Indeed, the average web page holds the reader for 18 seconds [25]."

Another question on the survey asked about the pace of the course. Four of the 16 students indicated that the pace was too fast and 12 said that it was just right, although several said that the course went too slow in the beginning.

V. CONCLUSIONS AND FUTURE WORK

This paper describes the design and first offering of a streamlined, low-stakes introductory computer programming course. While the goals of this course were similar to those of introductory courses offered in other universities (increased retention of CS majors, increasing the interest of non-majors in computing, making computing more accessible) the scope was more limited. We wanted to provide an experience that informed without being overwhelming. Furthermore, the intentional inclusion of education majors was an unusual twist.

The informal results suggest that students agreed that the class was indeed a gentle, yet informative introduction. They believed that their personal goals for taking the class were met. Many students felt that the beginning of the course moved too slowly, so that will be addressed in future offerings.

It remains to be seen whether the potential CS majors who took the course will start the regular Introduction to Computer Programming course with less anxiety and whether the education majors will feel comfortable introducing computing to their students as a result of the course. We intend to follow these students, but the preliminary results seem promising.

Another goal of the course was to reduce the population of the traditional Introduction to Computer Programming course by encouraging those who simply want to know what programming is to take the new course. We will carefully watch the enrollment trends over the next couple of years to see if this goal is met.

REFERENCES

[1] L. Bradford , "8 Jobs That Are Easier To Land If You Can Code",June 2,2016, https://www.forbes.com/sites/laurencebradford/2016/06/02/8-jobs-that-are-easier-to-land-if-you-can-code/2/#239c0fef72ae Retrieved April 14, 2018

[2] D. Restuccia, “Five Careers Where Coding Skills Will Help You Get Ahead,” http://www.burning-glass.com/blog/five-careers-where-coding-skills-will-help-you-get-ahead/, Jul 18, 2016, Retrieved April 14, 2018

[3] A. Petersen, M. Craig, J. Campbell, and A. Tafliovich. 2016.” Revisiting why students drop CS1.” In Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16). ACM, New York, NY, USA, 71-80. DOI: https://doi.org/10.1145/2999541.2999552

[4] CRA CS enrollment report, https://cra.org/data/generation-cs/R. Retrieved April 10, 2018.

[5] G. Flores,. Learning to Learn and the Navigation of Moods. Pluralistic Networks Publishing, 2016

[6] H. M. Walker. 2017. ACM RETENTION COMMITTEE: Retention of students in introductory computing courses: curricular issues and approaches. ACM Inroads 8, 4 (October 2017), 14-16. DOI: https://doi.org/10.1145/3151936

[7] FACT SHEET: President Obama Announces Computer Science For All Initiative, Press Release, White House, January 30, 2016, https://obamawhitehouse.archives.gov/the-press-office/2016/01/30/fact-sheet-president-obama-announces-computer-science-all-initiative-0, Retrieved April 10,2018

[8] Hour of Code Learning retrieved from https://hourofcode.com/us April 10, 2018

[9] CodeHS retrieved from https://codehs.com/ April 10, 2018.

[10] Girls Who Code retrieved from https://girlswhocode.com/ April 10, 2018

[11] M. Guzdial, and A. Forte. "Design process for a non-majors computing course." ACM SIGCSE Bulletin. Vol. 37. No. 1. ACM, 2005.

[12] R. H. Sloan and P. Troy. "CS 0.5: a better approach to introductory computer science for majors." ACM SIGCSE Bulletin. Vol. 40. No. 1. ACM, 2008

[13] T. J. Cortina. "An introduction to computer science for non-majors using principles of computation." ACM SIGCSE Bulletin. Vol. 39. No. 1. ACM, 2007.

[14] C. Alvarado, Z. Dodds, and R. Libeskind-Hadas. 2012. Increasing Women’s Participation in Computing at Harvey Mudd College. Inroads 3, 4 (2012), 55–64.

[15] T. Berger-Wolf, B. Igic, C. Taylor, R. Sloan, and R. Poretsky. 2018. A Biology-themed Introductory CS Course at a Large, Diverse Public University. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). ACM, New York, NY, USA, 233-238. DOI: https://doi.org/10.1145/3159450.3159538

[16] M. D. LeBlanc. 2017. Bringing computational thinking to the digital humanities: introducing students to explorations of digitized texts. J. Comput. Sci. Coll. 32, 6 (June 2017), 10-10.

[17] Cornell University: Choosing your First Computer Science Course, Retrieved from https://www.cs.cornell.edu/undergrad/firstcscourse on April 10, 2018

[18] S. Mishra, S. Balan, S. Iyer, and S. Murthy. 2014. Effect of a 2-week scratch intervention in CS1 on learners with varying prior knowledge. In Proceedings of the 2014 conference on Innovation & technology in computer science education (ITiCSE '14). ACM, New York, NY, USA, 45-50. DOI: http://dx.doi.org/10.1145/2591708.2591733

[19] T. Lorenzen and A. Sattar. 2008. Objects first using Alice to introduce object constructs in CS1. SIGCSE Bull. 40, 2 (June 2008), 62-64. DOI: https://doi.org/10.1145/1383602.1383636

[20] J. Carver, L. Henderson, L. He, J. Hodges, and D. Reese. .Increased retention of early computer science and softwareengineering students using pair programming. In Proceedings of the 20th Conference on Software Engineering Education & Training, p. 115-122, Washington, DC, USA, 2007. IEEE Computer Society.

[21] J. A. Cottam, S. Menzel, and J. Greenblatt. 2011. Tutoring for retention. In Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11). ACM, New York, NY, USA, 213-218. DOI=http://dx.doi.org/10.1145/1953163.1953227

[22] Purdue University: The Bridge Program retrieved on April 10, 2018 from https://www.cs.purdue.edu/undergraduate/bridge/ The Bridge Program

[23] D. J. Mir, S. Mishra, P. Ruvolo, L. Pollock, and S. Engen. 2017. How do faculty partner while teaching interdisciplinary CS+X courses: models and experiences. J. Comput. Sci. Coll. 32, 6 (June 2017), 24-33.

[24] Trinket: Code is Your Canvas. Retrieved from https://trinket.io/ July 10, 2017

[25] M. L Kutscher, “The Effects of Digital Technology on Reading”, Psychology Today, Jan 15, 2017. Retrieved from https://www.psychologytoday.com/us/blog/your-childs-brain-and-behavior/201701/the-effects-digital-technology-reading on April 20, 2018