Work-in-Progress: Research Plan for Introducing Problem Solving Skills through Activities to an Introductory Computer Science Course

Stephany Coffman-Wolph, Kimberlyn Gray, and Marcia Pool

Department of Computer Science, The University of Texas at Austin
2317 Speedway, Austin, TX, 78712, USA
E-mail: sscw@cs.utexas.edu

Department of Chemical Engineering, West Virginia University Institute of Technology
512 S Kanawha St, Beckley, WV, 25801, USA
E-mail: Kimberlyn.Gray@mail.wvu.edu

Department of Bioengineering, University of Illinois at Urbana Champaign
1304 W. Springfield Avenue, Urbana, IL, 61801, USA
E-mail: mpool@illinois.edu

Abstract

This work-in-progress research plan paper describes the process of developing and planning an introductory computer science course utilizing fundamental problem-solving skills in combination with hands-on visual activities to explain various Computer Science (CS) concepts. Problem solving skills, as observed by the authors of the paper, are challenging for students across multiple STEM disciplines, but those who develop these skills perform better within their STEM courses. The authors hypothesize that introduction of these skills within a first-year computer science course will benefit a student's successful completion of a STEM degree and their future STEM career [1]. The goal of this research is to integrate fundamental problem-solving skills into the existing course material and in-class activities. The research project will use two-sections of the same course taught during the same semester with approximately 200 students in each section. Nine hands-on activities, each covering a fundamental programming concept, were created to explain these concepts to students with a visual, real-world component. Both sections will cover the same computer science material, but some activities will be different between the two sections to allow for comparison of performance. There are nine planned activities: three will be performed with both sections; three will be performed only in section 1; and the remaining three will be performed only in section 2. Student performance on exams and programming assignments for these topics will be same and compared across both courses. This paper details the similarities and differences between the two sections of the course in terms of setup, activities planned, targeted problem-solving skills, and learning objectives. Additionally, the paper explains the evaluation plan and assessment tools/measures to be used (including pre- and post-surveys and assessment of student performance).

1. Introduction

Problem solving skills have been shown to be extremely important for successfully completing a degree in a STEM field and becoming a successful practitioner [1]. Below, the authors describe the activities which will be used in the course to integrate problem-solving into the curriculum while teaching the programming course concepts required for an introductory computer science class. The course is taught with a high-level of active learning as is shown in the descriptions of activities offered. Using two sections of the introductory course, two test groups will be created in which one group will utilize certain real-world, hands-on activities and the other group will not. Researchers will compare student performance on exams, assignments, and

other course work between the two groups. Additionally, all students in the course will be surveyed to identify their views on the in-class activities.

2. Course Learning Objectives

This introductory computer science course is entitled "Elements of Programming and Problem Solving" (CS303e). The students will learn the basics of a high-level programming language, Python, and problem-solving techniques for both numerical and scientific problems. CS303e is part of the "Elements of Computing" certificate program in the Computer Science Department at The University of Texas at Austin. The certificate program courses are designed to allow students from a variety of majors to develop an understanding of the technologies they will encounter in life and to gain important computer skills valuable in the job market. Thus, the student population is highly diverse and includes students from the business school, the communication school, other STEM fields, several engineering fields, and computer graphics. Given CS303e is an introductory course, it assumes no prior technical or computer science background and has no prerequisites. Learning objectives designed to guide the content delivered are as follows. By the end of the course students will be able to (1) define, in their own words, the basics of computer architecture; (2) explain, to a classmate, the basics of the special features of the Python language; (3) explain, to a classmate, the basics of the Python Syntax and how they are used; (4) explain and use correctly the basics of programming (i.e., variables, arrays/lists, if statements, loops, functions, classes, etc.); (5) create Python programs to solve problems using the correct syntax; and (6) use Python programming knowledge to create Python programs to solve numerical and scientific problems. The measurement instruments used to evaluate the learning objectives will be used as part of the assessment process for this study.

3. Course Design

3.1 Summary of Proposed Study

Nine, hands-on activities have been developed and used to aid college students in learning introductory programming concepts [2,3]; these activities were inspired by the work of "CS Unplugged" [4,5]. These activities have been used in introductory classes to supplement traditional lecture, with anecdotal positive feedback from students. To evaluate the effectiveness of these activities, the authors propose to use different activities in two sections of an introductory programming course, taken by students minoring in computer science or receiving a certificate in computer science, and compare the student performance on these topics.

Both sections (known as A and B) of the course are offered as MWF, fifty-minute sessions in the afternoon (1:00 pm and 3:00 pm) and will share the same instructor, four graduate teaching assistants (TAs), and two undergraduate proctors. In other words, the teaching staff will be consistent across both sections and grading of student work will be consistent between sections as well (e.g. question 2 on variables will be graded by the same person on all exams for both sections). Students enroll in sections by their preference or as permitted by their schedule, and the composition of each class by student background and major will be examined at the end of the course. Course attendance will also be recorded, as Friday afternoon sessions have a history of higher number of absences.

Of the nine total activities, three will be used in both sections (A and B), three will be used in only section A, and three will be used in only in section B. Lecture material for these topics will be the same in both sections of the class. An overview of the activities and placement of the activity in the course schedule is shown in Table 1. Both sections will use the "Understanding Variables and Arrays with Paper Bags" activity in the first two weeks of the course so that students in each section can become comfortable with the idea of a real-world activity to explain a programming concept. Then section A will have activities that focus on arrays, try/catch statements, and sorting algorithms while section B activities will focus on if statements, loops, and classes. Both groups will have the same activity on for and while loops in week four as well as a recursion activity in week thirteen.

For each activity, students will have one or two assignments, one low stakes quiz, and one exam to measure their skill development. Exam one will cover four activities: two of the shared activities and one separate activity for each section. Exam two will cover three activities: one for section A and two for section B. Exam three will cover two activities: one shared activity and one for section A. Table 1 illustrates the specific activities and the section(s) that the activity will be performed for each exam. The course content and topics covered for both sections will be the same. The only difference will be which activities are performed in each section (Table 1). Additionally, the exam topics and types of questions will be the same for both sections with multiple versions of the exam distributed in both sections.

3.2 Selection of Activities for Each Course Section

As stated above nine total activities will be used during the course with each section completing six of these activities (three overlapping (both sections A and B) and three different (three unique to section A and three unique to section B)). Each of the activities attempts to use real-world problems or examples students are familiar with to help illustrate and explain the computer programming concept. Table 1 provides a summary of each activity, the activity type, the skills covered by each activity, the section(s) in which the activity will be performed, the week during the semester, and the exam that will cover the topic.

The activities completed in both sections of the introductory programming course were selected for a particular

reason. The first joint activity, which is performed in the second week, sets the tone of the course and allows students to become familiar with the active learning concept. The second joint activity "Loops with Music" was selected because loops and conditional statements are important fundamental programming concepts. The third joint activity, recursion, was selected because the authors' experiences teaching have found this to be an extremely difficult topic for students to grasp. The remaining activities were divided between section A and B in order to divide skills and spread the activities throughout the semester.

3.2 Summary of Activity Types

The materials for each activity are low/minimal cost and use items that are readily available or easily attainable. There are three types of activities: Demo, Groups in Lecture, and Hybrid (demo and groups in lecture). All three types incorporate elements from active learning. The demos tend to need fewer supplies for the entire classroom, and most of the "action" happens in the front of the lecture style classroom. During the demos, students will attempt to do related problems/questions individually and then work with the students near them in pairs or in small groups. With the groups in lecture activities, students will be placed into groups for the duration of the activity with the instructor, TAs, and proctors visiting individual groups throughout the activity to provide guidance, feedback, and at times to just check-in. In other words, the "action" happens completely within the group (and each group will need a set of supplies). With the hybrid model, the majority of the "action" still happens at the front of the room (as with the demo), but the students will be placed in groups (like a group in lecture activity) to complete related tasks while receiving feedback from the instructor, TAs, and proctors visiting groups throughout the activity.

Of the nine activities, five are demos, three are groups in lecture, and one is a hybrid. The demo activities are the easiest to do within large lecture hall style classrooms. The activities that require groups requires more pre-planning and resources. (This pre-planning includes deciding how to divide the students, where they will sit, how many of the TAs or proctors can attend, prepackaging the supplies for each group, etc.).

4. Summary

It has been the authors' observation that students often have difficulty understanding fundamental programming concepts because they cannot relate the new information to their experiences. These activities were designed to help bridge the gap and, hopefully, help the students become better programmers. Up to this point, the authors have been working on anecdotal evidence only. In order to determine if the activities are having the desired outcome, the experiment outlined within this paper was designed to allow for formal assessment. The authors will use the outcomes of this experiment to alter the already designed activities or create new activities.

Table 1:

Name	Type	Summary	Skills	Class
Understanding Variables and Arrays with Paper Bags [2]	Demo	The instructor demonstrates the various types and sizes of variables with labeled bags or bins. Additionally, demonstrate that only one value can be stored in any bag/bin at a time.	Variables Arrays Syntax	Both (Week 2) Exam 1

References

[1] J. Wai, D. Lubinski, and C.P. Benbow, "Spatial Ability for STEM Domains: Aligning Over 50 Years of Cumulative Psychological Knowledge Solidifies Its Importance," Journal of Educational Psychology, vol. 101, no. 4, pp. 817-835, November 2009.

[2] Coffman-Wolph, S., Innovative Activities to Teach Computer Science Concepts Inside the Classroom and at Outreach Events Paper presented at 2016 ASEE Annual Conference & Exposition, New Orleans, Louisiana. 10.18260/p.25715, June 2016.

[3] Coffman-Wolph, S., Fun, Innovative Computer Science Activities for the Classroom and Outreach Paper presented at 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/28394, June 2017.

[4] T. Bell, et al., "Computer Science Unplugged: School Students Doing Real Computing Without Computers," Computing and Information Technology Research and Education, New Zealand (CITRENZ), vol. 13, no. 1, pp. 20-29, 2009.

[5] T. Bell, et al., CS Unplugged: Computer Science without a Computer. www.csunplugged.org., 2015.

Arrays with Household Goods [2]	Demo	Students determine what the various objects have in common and, thus, learn the concepts of arrays. Time is spent discussing individual elements within the array.	Various objects that have an “array-like” structure	A (Week 2) Exam 1
Branching and Looping Statements with Starburst Candies [2]	Groups in lecture	Students practice branching to determine who is the team leader of the group for the day. In the second part of the activity, students pass the paper of candies until they find a red Starburst (i.e., while loop).	If Statements While Loops	B (Week 3) Exam 1
Loops with Music [3]	Demo	The entire class participates by doing certain actions during various conditions (i.e., loop).	While loops For Loops Conditional Statements	Both (Week 4) Exam 1
Monsters Hate Chocolate: Learning Try/ Catch Blocks [2]	Groups in Lecture	Each group tries to “feed” the monster (i.e., the paper sack). The statements that contain an exception (i.e., the chocolate candies) do not fit. The statements that are valid code match the Starburst candies and fit.	Try/Catch Statements Exceptions	A (Week 7) Exam 2
General Class Structure with Bags, Boxes, and a Bin [2]	Demo	The large storage bin represents a class in a computer language. Develop the contents of the class by adding bags/bins (variables) and boxes (functions).	Classes Functions Class Variables	B (Week 8) Exam 2
Dr. Doolittle’s Vet Office: Learning Classes with Stuffed Animals [2]	Demo	Using a variety of stuffed animals, students practice designing a class (variables and functions) for animals in a vet program.	Classes Functions and Variables Class Design	B (Week 9) Exam 2
Sorting Algorithms with Paper Bags [3, 4, 5]	Hybrid (demo and groups in lecture)	Students develop their own algorithms for sorting a series of paper bags with numbers on each. The bags are then used to demonstrate the basic sort algorithms.	Bubble, Selection, and Insertion Sort Algorithm Development	A (Week 11) Exam 3
Recursion Introduction: Simple Tower of Hanoi with Colored Paper [3]	Groups activity	Students develop the algorithm for the Tower of Hanoi using 3, 4, and 5 disks to work out the similarities in movement.	Recursion Algorithm Development	Both (Week 13) Exam 3