Effects of Professional Development on Programming Knowledge and Self-Efficacy

Yolanda J. Reimer
Dept of Computer Science
University of Montana
Missoula, USA
yolanda.reimer@mso.umt.edu

Michael Coe
Cedar Lake Research Group
Portland, USA
michael@cedarlakeresearch.com

Lisa M. Blank
PJW College of Education and Human Sciences
University of Montana
Missoula USA
lisa.blank@umontana.edu

Jeffrey Braun
Dept of Computer Science
Montana Tech
Butte, USA
jbraun@mtech.edu

***Abstract*— This Research Full Paper presents the effects of our weeklong Professional Development class on the programming skills of nineteen high school teachers, their confidence in programming, and their confidence for teaching programming. A primary objective of the CS10K and CS For All initiatives is the education of K-12 teachers in aspects of computer science and computational thinking so they can teach CS courses in their schools. Many of these educators have degrees in disciplines other than computer science, such as math, science, and business, so preparing them to teach CS is a challenge, particularly since most K-12 teachers have limited time to devote to learning new curriculum. This study describes how we managed a short course in computational thinking and programming to a group of high school teachers. We illustrate through survey data assessment and evaluation that significant gains in skill level and self-efficacy can be realized within a short but intensive week of face-to-face training. Five months later we follow-up with the same cohort to see if earlier achievements remain evident over time. Discussion throughout the paper identifies strengths and weaknesses of the training week, which is useful to others planning to undertake similar PD offerings.**

Keywords*— Professional development; CS10K; CS For All; computational thinking; programming skill; K-12 instruction

I. Introduction

Our work within the CS10K initiative focuses on broadening participation in computer science (CS) across the rural state of Montana. Like much of the country, Montana has a strong demand for graduates with technical and computer programming skills; some estimates suggest the number of those graduating with CS degrees from Montana public colleges and universities meets only 10% of statewide demand [1]. Our project unites three flagship institutions along with a Tribal College in a statewide collaboration. The goal is to increase the number of qualified high school CS teachers by providing professional development (PD) opportunities and sustaining resources like curriculum plans, assignments, schedules, etc.

Many high school teachers who are recruited or volunteer to teach CS courses are educated primarily in other disciplines (e.g., Math, Science, Business). While they are a highly motivated group, these teachers often lack prerequisite experience with programming and computational thinking, and so require opportunities to develop their skill sets as well as resources to maintain them over time. Educating more K12 teachers in CS will result in an increase in the number of CS courses taught in high schools, and subsequently more students will be exposed to the area and will give further consideration of its career prospects.

As we prepared to offer our first professional development class for high school teachers from around the state, we looked to the experiences of others. We observed another PD workshop offering the same curriculum we would use, and we attended discussions about how to organize PDs effectively and the importance of logistics and community building [2, 3]. We considered much of what we learned and made some specific decisions about our own PD that ranged from having meals catered on-site rather than going off campus to eliminating certain curriculum topics that seemed too advanced for a one week session.

This paper provides an overview of our PD, including the curriculum we covered and why, who our participants were, and the weeklong schedule we followed. However, the main focus of the report is the impact our PD had on participants' computational thinking abilities, programming competency, and self-efficacy. Some of our findings coincide with results from Price et al.'s study evaluating the impact of the Beauty and Joy of Computing (BJC) CSP PD his team offered [4]. Price shows significant improvement in participants' perceived ability to teach CSP across four categories: Content, Inquiry, Equity and Differentiation. While these categories are geared towards the goals of the BJC curriculum and our are centered differently, there is certain overlap between our two projects. In broad terms, both studies show improvement in the confidence level of participants across various spectrums, and our study further illustrates that these gains can be realized after only one week of face-to-face training, and then sustained somewhat over a longer period of time.

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

II. PD OVERVIEW

Our professional development class covered the Joy and Beauty of Computing (JBC) curriculum. JBC is a semester long 3-credit college course first piloted at Montana State University (MSU) during the fall of 2013 to forty students. It has since grown in popularity and size, and is now regularly taught to hundreds of students around the state. Despite having similar titles, JBC should not be confused with Berkeley's Beauty and Joy of Computing (BJC) course [5]. Two key differences between them include: 1) students learn Python in JBC and Snap*!* in BJC, a visual programming language; 2) JBC incorporates a variety of web resources, including the online text "How to Think Like a Computer Scientist" [6]; in contract, "Blown to Bits" [7] is used to teach students about social implications of computing in BJC.

The goal of JBC is to give students who are interested in exploring computer science a gentle introduction to computational thinking and a high level understanding of the field: where it has been, where it is now, where it is going, and what careers are possible. As such, JBC provides students with an alternative pathway into the curriculum. This multi-pronged approach is necessary as entry-level CS courses are often populated with a somewhat homogeneous group of students who have taught themselves about computers since an early age. When novices and self-taught students are lumped together, the experience can be intimidating to the beginners, perhaps causing them to harbor the misconception that they are significantly behind other students and sowing doubts as to whether they have chosen a viable major.

A. Curriculum

The curriculum for the 15-week version of JBC, including suggested schedule, topics and assignments, is freely accessible online [8]. JBC focuses on computational thinking using the Python programming environment, but it also integrates a wide variety of computing topics into course discussions and assignments in order to engage a more diverse student body. We spent considerable time developing a Moodle supplement for our PD course, which participants relied on heavily throughout the week and can continue to access. This supplement contains curriculum schedules, assignments, text readings, exams, solutions, samples of student work, participant contact information, etc. We also started a Moodle forum for teachers so that they could share tips and resources related to teaching with one another. This rich repository of information is a critical resource for participants as they teach JBC at their own local high schools, now and in the future.

A key challenge of our PD class was not only familiarizing participants with core aspects of computational thinking and programming, but also discussing how participants can teach this course to their own students. Throughout the week we talked about how to assess computer programs and assignments, administer exams, and integrate new, contextually relevant activities. We routinely asked participants to submit their solutions to various exercises covered in class, which we then reviewed together as a group. This allowed us to discuss problem solving techniques, highlight common mistakes, and illustrate how participants might grade student solutions on their own going forward.

Work supported by the National Science Foundation under Grant No. CNS-1639841.

B. Participants

Nineteen K-12 teachers from across the state of Montana enrolled in our weeklong PD class. Participants came from rural communities with extremely small populations (e.g., Stanford, pop. 384), as well as from larger cities and towns, including Missoula (pop. 72,364) where the workshop was held. Some traveled as long as 7 hours to attend the class. Eighteen of our participants teach at the high school level (grades 9-12), and one teaches middle school. Some participant schools are located on Native American reservations. The classes our participants typically teach include Business, Math and Computer Applications.

We were able to offer one thousand dollar stipends to each participant and pay for travel and meal expenses. As part of participation in our PD, we asked participants to take an online Python course offered by Codecademy prior to the June PD, attend a follow-up weekend workshop in November and again during the following spring. Our intent is to sustain this community of teachers over time, scaffold learning between previous and new groups, and expand on both the number of participants and the variety of resources going forward.

At the start of the PD week, participants were asked to assess their level of experience with programming and in teaching programming, both with Python and without. As seen in Table 1, most reported little to no prior experience. This finding foreshadows their initial responses pertaining to self-efficacy, which we report on later in this paper. Both measures indicate that our group of PD participants considered themselves to be novice programmers.

TABLE 1. Participant background experience (*n=19*)

Q1 - Please indicate your level of experience regarding the following statements.	**None**	**A Little**	**A Fair Amount**	**A Lot**
How much programming have you done before your involvement in the CS10K project?	0%	68%	16%	16%
In particular, how much Python programming have you done before your involvement with the CS10K project?	53%	32%	16%	0%
How much teaching of computer programming have you done before your involvement in the CS10K project?	42%	42%	11%	5%
In particular, how much teaching of Python programming have you done before your involvement with the CS10K project?	58%	37%	5%	0%

C. Schedule

Class met from 8:30am-5pm, Monday to Friday with one hour allocated for lunch and a few other small breaks scheduled throughout the day. Covering course materials normally taught over the span of a semester in five days is unquestionably daunting, and because of this many PDs are offered over an extended period of time and in multiple formats [4, 9]. However, we chose to limit our PD to one face-to-face week during the summer followed by two weekends throughout the year in order to maximize teacher participation.

To assist with this tight scheduling, we relied heavily on our prior experience co-teaching JBC at a local high school during the previous year and the schedule developed for previous JBC PDs held at MSU. Having gone through the course at least once with a high school class allowed us to know which aspects of the curriculum to focus on more than others during the PD week. To fit the curriculum into a five day week, we winnowed down some content by skipping or glossing over exercises and activities that we felt were already covered sufficiently given the time frame. We were also careful to minimize instructor lecture time and maximize hands-on activities during the week, and we alternated frequently between activities to avoid remaining on any one topic for too long. At times, this meant participants were not able to complete exercises entirely, but they typically got far enough along to know they could finish if they had more time. Some participants also worked on programs after class hours. Table 2 below provides a high-level overview of the topics and activities we covered during the week.

Still we anticipated it would be challenging for participants to keep pace with the intense schedule. To mitigate this further, we required all attendees to complete Codecademy's Python course [10] as a prerequisite. While some participants reported having difficulties with the material, it at least provided all of them with basic exposure to fundamental programming and computational thinking concepts prior to attending the face-to-face PD.

TABLE 2. PD schedule

	Topics	Hands-on Activities
Mon	• Curriculum overview • Python shell, IDLE editor • Simple python data • Turtle graphics	• Pre-course assessment • Create static business card • Custom business card • Draw initials on screen • Draw star
Tue	• Graphics using loops • Advanced turtle graphics • Practicum reviews • Intro to functions	• Draw Pokemon Go character • Practice practicum • Practicum 1
Wed	• More functions • Selection • Modules • Practicum review	• Factorial function • Repeating song lyrics function • Minecraft drawing using functions • Practice practicum • Practicum 2
Thurs	• Curriculum development • Iteration • Computing topics discussion	• Assignment ideation • Manufactoria functions
Fri	• Strings • (Optional) recursion • Practicum review	• Practicum 3 • Post-course assessment

D. Curriculum Development Model

Throughout the week we stressed to participants that the JBC curriculum was not just about computational thinking and programming, but that a significant focus of it when taught in the high school should be on expanding the diversity of the student body. Various computing topics are listed on the original course syllabus, which instructors can use as a starting point for initiating further projects, classroom discussions, and research. To facilitate discussions of how we might build a more inclusive curriculum and engage a wider body of students in our CS courses, we allocated one full morning later in the week for curriculum discussion and development.

Small participant groups were formed based loosely on similar teaching environments (classes, location, school size, etc.). Each group was asked to come up with alternative assignments and exercises beyond those we had already provided based on what they thought their particular student body would be most interested in. Aside from the ideas generated, many of which involved having students using Python's turtle graphics to design Native American symbols and other images that students might be particularly invested in, we realized the importance of providing this time for teachers to collaborate with other teachers and of involving them directly in curriculum development. Gray et al. also recognize the importance of incorporating teachers in curriculum development; their entire CSP PD is built around the Teacher Leader model whereby more experienced K12 teachers were recruited and tasked with much of the curriculum planning efforts and resource development [11].

III. RESEARCH METHOD

In this study, we focus on the effectiveness of our weeklong professional development (PD) course with particular regard given to changes in participant self-efficacy for programing, and self-efficacy for teaching programming. On the first morning of PD week we asked participants to complete an online survey, the results of which formed a baseline from which we could measure change. During the afternoon of the last day of class participants were asked to complete a post-PD survey that contained many of the same questions as the pre-PD survey. Five months after the weeklong PD, we invited this same teacher cohort to meet for a weekend workshop. Sixteen of the original nineteen participants attended the weekend workshop. We asked participants again at the beginning of the weekend to complete a survey containing many of the same questions as previous surveys. This approach gives us a sense of the extent to which observed changes in participant programming skill and confidence were sustained over time. We use our research findings to determine the extent to which our PD is effective and to engage in continuous improvement.

Throughout the remainder of this paper, we report on the pre and post PD results, along with the five month delayed (weekend workshop) results. Note that not all of the questions on the surveys are shown here. We omitted questions that are either unrelated to the focus of this report, or that involve open-ended responses (e.g., not multiple choice) which are more difficult to include because of the various scoring rubrics required. Also note that although we have numbered the survey questions in each of our tables, we do so only for ease of reference; the ordering shown here is not exactly the same as they appeared on our surveys.

Limitations of our study include a somewhat small sample size and that three PD participants were not available for the follow-up weekend workshop when retention over time was

measured. We also believe that while survey results such as our are useful and meaningful as they can provide an important snapshot in time, they can become even more so with the added context of follow-up participant interviews. Also as discussed further in the paper, our computational thinking questions turned out to be a bit narrow in scope and type. In the future, we will expand on both the number and type of computational thinking questions so we get a fuller picture of the progress our participants are making.

A. Computational Thinking

The computational thinking questions we included on the surveys, as seen in Table 3 below, come from Dehnadi and Bornat's work exploring student mental models [12, 13, 14]. We hoped that after our PD class more participants would be able to solve single and multiple assignment problems (Q4-Q7). Although each of these questions listed many possible multiple choice answers, only a few are shown due to the space constraints of this paper. Each of these assignment questions has only one correct answer, and even though participants were allowed to check more than one box in each case, we only counted an answer as correct if it was the only box selected. In other words, if participants checked more than one box, even if one of them included the correct answer, we still marked the response as incorrect since the others selected were not correct. Although this is a less nuanced and stricter way to grade responses for these questions, and one that does not consider the possible mental models that students might have used in coming up with their answers as Dehnadi and Bornat have done, it does provide a snapshot of participant progress.

TABLE 3. Computational Thinking
(Pre and Post *n=19;* Delayed *n=16*)

	Pre	***Post***	***Delay***

Q4 - Read the following statements and check the box next to the correct answer(s). a = 10, b = 20, a = b. Check all boxes that apply.
`a = 20 and b = 0`
`a = 20 and b = 20`
`a = 0 and b = 30`
`etc. (more options were available, but limited here for space constraints)`
	32%	47%	50%

Q5 - Read the following statements and check the box next to the correct answer(s). a = 40, b = 30, b = a, a = b. Check all boxes that apply.
`a = 0 and b = 30`
`a = 70 and b = 70`
`a = 0 and b = 70`
`etc.`
	32%	37%	38%

Q6 - Read the following statements and select the box(es) next to the new values of big and small. big=10, small=20, big=small.
`b = 0 and small = 30`
`big = 0 and small = 10`
`big = 20 and small = 0`
`etc.`
	42%	47%	44%

Q7 - Read the following statements and select the box(es) next to the new values of a, b and c. a = 0; b = 1; c = 2; b = a; a = c; c = b.
`a = 2 and b = 2 and c = 2`
`a = 2 and b = 1 and c = 3`
`a = 1 and b = 4 and c = 3`
`etc.`
	32%	42%	38%

Table 3 shows the percentage of participants who got the correct answer for each computational question. While some gains are evident from the beginning of the PD week to the end of the week and in delayed performance (i.e., after a five-month period had elapsed), overall movement in these areas was surprisingly small. We discuss possible reasons for this, along with the fact that only 50% or less answered each question correctly, later in the Discussion section.

B. Python Programming

The next set of questions on our surveys evaluated Python programming skill levels. These questions were assessed in the same way as the computational thinking questions, and Table 4 shows the percentage of participants who got each question correct. As previously mentioned, all participants were required to take Codecademy's online Python course prior to arriving at our face-to-face PD class. While some struggled with the Codecademy course and may not have finished parts of it entirely, all participants did report spending time with the course, thus giving them some initial exposure to Python programming and computational thinking.

To derive this set of questions, we looked at Codecademy materials and the content of what we covered during PD. We tried to include one or two basic questions for each of the main topics of our class listed in Table 2. Categories are noted in parentheses prior to the listing of each question in Table 4. Each question was multiple choice and had only one correct response, and we show the number of participants who answered each correctly both pre and post PD as well as after five months had elapsed (Delay column).

There was no decline in average scores from pre to post PD for any of the questions, and many show strong improvement in participant programming skill over the course of the week. Two questions where little or no improvement is apparent involve mathematical computation and syntax (Q8), and strings (Q16). Scores for Q8 likely stay mostly the same because they were relatively high to begin with, and Q16 is arguably a more challenging question that is difficult to answer even after a week of PD. The topic of Strings was the last one we covered, when participants may also have been saturated with new material, but these results indicate that future PDs should spend more time and include more exercises related to both topics, especially given how central they are to programming.

The delayed results in programming skill assessed five months later fell generally between the pre- and post-PD scores. On average, teachers correctly answered 50 percent of these questions at pretest, 70 percent at posttest, and 57 percent at the delayed follow-up. While we would like to see even stronger retention rates over time, less than a third of our teachers (five out of sixteen) were teaching JBC in their high schools when the delayed assessment took place, and thus they likely did not have continued engagement with the material during the interim period. As more and more teachers are able to integrate JBC in their own classrooms, we expect these retention results will improve.

TABLE 4. Python Programming Skill
(Pre and Post *n=19;* Delayed *n=16*)

	Pre	*Post*	*Delay*
Q8 - (Data & Syntax) Evaluate each of the following expressions and write the answer below:			
1. *5 ** 2*	79%	79%	88%
2. *9 * 5*	89%	89%	88%
3. *15 / 12*	63%	68%	56%
4. *15 // 12*	11%	58%	25%
5. *5 % 2*	21%	26%	38%
Q9 - (Data & Syntax) What is the value of the following expression: 16 - 2 * 5 // 3 + 1 (If you do not have an answer, please respond with IDK).	11%	58%	19%
Q10 - (Data & Syntax) What data type is False and True in Python?	79%	100%	81%

Q11 - (Functions) What will the following function return?
`def mystery(n):`
`  answer = 0`
`  for i in range(1,n+1):`
`  answer = answer + i`
`  return answer`
 	
`# Call function and print answer`
`print(mystery(10))`
	53%	68%	44%

Q12 - (Functions) What are the parameters of the following function?
`def drawSquare(t, sz):`
`"""Make turtle t draw a square of with side sz."""`
`  for i in range(4):`
`  t.forward(sz)`
`  t.left(90)`
	53%	95%	69%

Q13 - (Functions) What is a variable's scope?	63%	84%	81%

Q14 - (Selection) True or False: The following two blocks of code are equivalent and produce the same result.
`#####################################`
`#block 1`
`if x < y:`
`  print("x is less than y")`
`elif x > y:`
`  print("x is greater than y")`
`else:`
`  print("x and y must be equal")`
`#####################################`
`#block 2`
`if x < y:`
`  print("x is less than y")`
`else:`
`  if x > y:`
`  print("x is greater than y")`
`  else:`
`  print("x and y must be equal")`
	53%	74%	56%

Q15 - (Loops) How many times does the word "Hello" get printed to the screen with the following code:
`for i in range(5):`
`  print("Hello")`
	47%	74%	63%

Q16 - (Strings) What is printed when the following code runs. If you do not have an answer, please respond with "I don't know".
`somestring = "JBC Rocks!"`
`print(somestring[1], somestring[3], somestring[5])`
	32%	32%	31%

TABLE 5. Self-Efficacy for Programming
(Pre and Post *n=19*; Delayed *n=16*)

	Strongly disagree	Moderately disagree	Slightly disagree	Slightly agree	Moderately agree	Strongly agree
I have a lot of experience with programming to create software, websites, etc.						
Pre	26%	32%	0%	5%	26%	11%
Post	16%	26%	0%	42%	5%	11%
Delay	19%	13%	13%	25%	19%	13%
I am very confident in my current ability to use Python code to accomplish programming tasks and projects.						
Pre	37%	16%	11%	16%	16%	5%
Post	0%	11%	5%	47%	21%	16%
Delay	0%	31%	6%	38%	13%	13%
I am very familiar with and can use more than one programming language.						
Pre	37%	16%	0%	21%	11%	16%
Post	37%	5%	11%	26%	11%	11%
Delay	25%	6%	13%	38%	6%	13%
I regularly use or tinker with code I have written.						
Pre	47%	5%	11%	16%	16%	5%
Post	21%	16%	11%	21%	21%	11%
Delay	13%	31%	0%	25%	25%	6%
I know where to find examples of code that I can repurpose for my own programming projects.						
Pre	16%	42%	0%	16%	11%	16%
Post	5%	0%	0%	16%	32%	47%
Delay	0%	6%	13%	13%	31%	38%

C. Self-efficacy

Perhaps the most important measurement of the success of our PD is the change in confidence that participants felt with regard to their own programming ability and their ability to teach programming and computation to others. Related questions are shown in Tables 5 and 6, along with pre and post PD responses and the delayed responses obtained five months later during our weekend workshop.

Responses to these five questions were averaged to form a scale score for **Self-Efficacy for Programming.** The scale had high internal consistency reliability (Cronbach's alpha = .93 at pretest and .89 at post-test). Change from pre to post was statistically significant ($t_{(18)} = 4.30$, $p = .0004$), with the mean response rising from near the *"slightly disagree"* response at pre-test to near the *"slightly agree"* response at post-test. Change from pre to the delayed follow up remained statistically significant, ($t_{(15)} = 4.05$, $p = .0010$), with the mean response at the delayed follow up remaining near the "slightly agree" level. If we aggregate all response options indicating agreement, of note is that post-PD participants said they were more confident in their current ability to use Python to accomplish programming tasks (37% pre PD vs. 84% post vs. 64% delayed), and they were more familiar with how to locate other examples, resources and code to use for their own programming projects (43% pre PD vs. 95% post vs. 82% delayed).

In terms of self-efficacy for *teaching* programming, participants also reported considerable gains (Table 6). Responses to these ten questions were averaged to form a scale score for **Self-Efficacy for Teaching Programming.** The scale had high internal consistency reliability (Cronbach's alpha = .96 at pretest and .90 at post-test). Change from pre to post was statistically significant ($t_{(18)} = 4.60$, $p = .0002$), with the mean response rising from near the *"slightly disagree"* response at pre-test to above the *"slightly agree"* response at the post-test. Change from pre to the delayed follow up remained statistically significant, ($t_{(15)} = 3.88$, $p = .0015$), with the mean response at the delayed follow up remaining near the "slightly agree" level. Again aggregating all three levels of agreement, confidence doubled or more from pre to post in the belief participants had in:

- learning progressions that help students develop their programming concepts and skills (42% pre, 84% post, 75% delayed);
- how to structure programming concepts and skills so that students can systematically develop their understanding (26% pre, 84% post, 68% delayed);
- employing group CS learning strategies such as pair programming with their students (38% pre, 90% post, 63% delayed);
- fostering students' computational thinking (43% pre, 95% post, 75% delayed).

Also notable is that 100% of post-PD participants expressed some confidence in their ability to apply the programming content they learned to their own CS teaching. The discussion we had during PD about diversity and developing specific curriculum modules (topics, assignments, etc.) that would be of particular cultural relevance to students also made a positive impact on participants. While 32% either moderately or strongly agreed to feeling confident that their CS teaching strategies would be effective with diverse groups of students pre-PD, that number almost doubled to 63% directly after PD before falling back down to 38% five months later.

While all other self-efficacy for teaching programming responses also rose from pre to post PD, smaller percentages were reported relating to experience participants had helping students write code and teaching computational thinking. There was also one interesting anomaly in that participants actually reported a lower level of confidence in their current ability to teach students how to code in the five month delayed survey than they did pre-PD (56% delayed versus 63% pre-PD). We're not sure why this decrease happened, but we believe it may be related to a lack of sustained engagement with the curriculum during the interim period as mentioned previously and discussed further in the next section.

TABLE 6. Self-Efficacy for Teaching Programming
(Pre and Post *n=19*; Delay *n=16*)

	Strongly disagree	Moderately disagree	Slightly disagree	Slightly agree	Moderately agree	Strongly agree
I have a lot of secondary teaching experience in helping students learn how to write code.						
Pre	47%	11%	11%	16%	11%	5%
Post	16%	26%	21%	26%	5%	5%
Delay	6%	25%	13%	44%	6%	6%
I am very confident in my current ability to teach students how to code.						
Pre	26%	11%	0%	37%	21%	5%
Post	11%	0%	5%	47%	26%	11%
Delay	0%	25%	19%	31%	19%	6%
I am very familiar with learning progressions that help students develop their programming concepts and skills.						
Pre	42%	11%	5%	21%	21%	0%
Post	5%	0%	11%	42%	37%	5%
Delay	0%	13%	13%	38%	31%	6%
I know how to structure programming concepts and skills so that students can systematically develop their understanding.						
Pre	47%	5%	21%	16%	5%	5%
Post	11%	5%	0%	32%	47%	5%
Delay	0%	13%	19%	31%	31%	6%
I have used Python with students and I am familiar with how to teach computational thinking.						
Pre	63%	11%	0%	16%	5%	5%
Post	37%	16%	11%	16%	16%	5%
Delay	19%	13%	31%	13%	19%	6%
I am very confident that I will be able to apply the programming content I will learn in this course in my CS teaching.						
Pre	11%	5%	21%	5%	42%	16%
Post	0%	0%	0%	21%	42%	37%
Delay	0%	0%	13%	44%	19%	25%
I am very confident in my current ability to explain essential CS concepts such as functions to my students.						
Pre	26%	11%	11%	16%	32%	5%
Post	0%	0%	16%	47%	11%	26%
Delay	0%	19%	6%	38%	25%	13%
I am ready to employ group CS learning strategies such as pair programming with my students.						
Pre	37%	16%	11%	11%	16%	11%
Post	0%	5%	5%	37%	37%	16%
Delay	0%	25%	13%	25%	25%	13%
I am very confident in my current ability to foster my students' computational thinking.						
Pre	16%	26%	16%	0%	32%	11%
Post	0%	5%	0%	26%	53%	16%
Delay	0%	13%	13%	31%	31%	13%
I am very confident that my CS teaching strategies will be effective with diverse student groups including both male and female learners.						
Pre	26%	16%	5%	21%	21%	11%
Post	0%	5%	0%	32%	42%	21%
Delay	0%	6%	13%	44%	25%	13%

IV. DISCUSSION

As we planned our PD class, we were concerned with the large amount of material we needed to cover in just one week. We also felt, though, that it was not practical to schedule the class for a longer period of time. Even extending it to two weeks would likely cause some teachers to opt out, particularly since many were not local, and it would put additional strain on our budget. We tried to compensate for the tight face-to-face time period by requiring participants to gain exposure to computational thinking and Python programming via Codecademy prior to attending our PD. Despite the difficulty some participants had with that material, particularly as they worked on it independently and without immediate help available, we ultimately think requiring some prior experience with course topics was effective and should be repeated.

Even so, we were still concerned that participants might become overwhelmed and frustrated with the intense schedule and large amount of new material they were expected to absorb during PD week. And, in fact, on final course feedback forms a number of participants did note hitting a wall of sorts about mid-week. However, when we look at the results of pre and post PD surveys of programming skill, self-efficacy for programming, and self-efficacy for teaching programming, we are reassured. There were no instances where participants regressed during the PD week, and in many areas they made significant progress. We are most excited by the gains in self-efficacy participants reported as we believe this is a critical barometer of their willingness to teach JBC in the future at their own schools, and to be successful with it. It is also worth noting that all PD participants completed the entire week of class; none dropped out. Furthermore, many of the gains realized were still evident after a five-month period had elapsed.

Some results of our assessment were not as positive as we would like, however. As noted earlier, we were surprised at the relative lack of improvement with the computational thinking questions, shown in Table 3, and that fewer than half of participants answered these questions correctly. We attribute this in part to the lack of prior programming experience participants reported (Table 1), but perhaps more-so to a slight disconnect between the fairly narrow computational thinking questions we asked and the content and main focus of our PD. While computational thinking was certainly covered during our course, it did not take the same format as the questions we posed on our assessments, all of which were very similar in type. In the future we would look to remedy this on both ends: do a better job covering aspects of variables and how values are stored and changed; and include a broader set of computational thinking questions on assessments.

We are also cautious about steeper drop offs that occurred in some delayed responses. For example, while we would like to see even stronger retention rates over time in programming skill, we also understand that most teachers were not yet teaching JBC in their high schools when the delayed assessment took place, and thus they likely did not have continued exposure to the material during the interim period. As more and more teachers are able to engage regularly with JBC in their own classrooms, we expect the retention results will improve in skill levels, which should also drive up delayed self-efficacy scores. In the meantime, though, to help mitigate proficiency gaps that may appear over time, we must continue to develop and provide additional resources like curriculum plans and assignment solutions and encourage teachers to consistently engage with them.

For others looking for specific suggestions in offering their own PDs, in our case having good support staff on hand greatly contributed to participant learning. We were fortunate that all four of our project PIs were able to assist the class, along with one undergraduate student who helped with the high school offering of JBC the previous fall. Having both the number and the expertise of this support staff proved invaluable. It allowed for different PIs to teach different modules throughout the week, freeing up others to assist students as they worked through the hands-on activities. Many participants commented favorably on this particular aspect of the week on their final course feedback forms.

Lastly, we also mention that a number of our participants expressed interest in obtaining credit for completing the PD class, which we were able to accommodate by setting up a separate course number at our institution. Taking the PD class for credit was entirely optional, and it did cost enrollees $155 if they chose to do so, but it was an added incentive for some. For the seven students that enrolled in the PD for credit, we required them to upload solutions to various exercises throughout the week and write a reflection of their learning process in each case. Having teachers enroll for credit which they had to pay for themselves likely contributed to more serious engagement with the class and better overall results.

V. FUTURE WORK

In our second year of the CS10K project, we continue to extend this work in a number of ways. We are currently supporting teachers who are teaching JBC in their local schools and assessing student outcomes in these classes. We recently piloted the Mobile CSP course at Montana Tech, held a weekend workshop there, and will offer weeklong PDs for both it and JBC during the summer of 2018. We are expanding our online community forum for our teacher cohort, and hope to use it to provide additional resources year-round. We will roll out the Mobile CSP curriculum to high schools during the 2018/19 academic year and continue to support and assess the effectiveness of both courses.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1639841. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Dennison, M. (2013, September 17). Former Bozeman software company exec urges more computer science students. The Missoulian. Retrieved from missoulian.com.

[2] Davis, S. (2017) Session 15N: BoF 559: Improving Effectiveness of CS Teacher Professional Development *SIGCSE 2017*.

[3] Camp, T., Schanzer, E., Goode, J., Astrachan, O., & Campos, E. (2017). CSPd Week: A Scalable Model for Preparing Teachers for CS for All. *Proceedings of SIGCSE '17,* 645-646. doi>10.1145/3017680.3017681

[4] Price, T., Catete, V., Albert, J., Barnes, T., & Garcia, D. (2016). Lessons Learned from "BJC" CS Principles Professional Development. *Proceedings of SIGCSE '16,* 467-472. doi> 10.1145/2839509.2844625

[5] Garcia, D., et al. *The Beauty and Joy of Computing.* Retrieved from https://bjc.berkeley.edu/

[6] Miller, B. & Ranum, D. *How to Think Like a Computer Scientist: Interactive Edition.* Retrieved from: http://interactivepython.org/courselib/static/thinkcspy/index.html

[7] Abelson, H., Ledeen, K., & Lewis, H. *Blown to Bits: Your Life, Liberty, and Happiness After the Digital Explosion.* Retrieved from http://www.bitsbook.com/thebook/

[8] Paxton, J. *The Joy and Beauty of Computing*. Retrieved from http://ou.montana.edu/t2cs10k/jbc/lectures/index.html

[9] Morelli, R., Uche, C., Lake, P., & Baldwin, L. (2015). Analyzing Year One of a CS Principles PD Project. In *Proc. of the 46th Annual ACM SIGCSE Conf.*, 2015.

[10] Codecademy: Learn Python. Retrieved from https://www.codecademy.com/learn/learn-python

[11] Gray, J., Haynie, K., Packman, S., Boehm, M., Crawford, C. & Muralidhar, D. (2015). A Mid-Project Report on a Statewide Professional Development Model for CS Principles. *Proceedings of SIGCSE '15,* 380–385. doi>10.1145/2676723.2677306

[12] Bornat, R., & Dehnadi, S. (2008). Mental models, consistency and programming aptitude. *ACE '08 Proceedings of the tenth conference on Australasian computing education 78,* 53-61.

[13] Dehnadi, S. & R. Bornat. (2006). The camel has two humps. *Little PPIG* 2006. Coventry,UK.

[14] Dehnadi, S. (2006). Testing Programming Aptitude, in P. Romero, J. Good, E. A. Chaparro and S. Bryant, eds, *Proceedings of the PPIG 18th Annual Workshop,* pp. 22--37. URL:http://www.ppig.org/papers/18thdehnadi.pdf