Pythons and Martians and Finches, Oh My! Lessons Learned from a Mandatory 8th Grade Python Class

Amal Nanavati*
amaln@cs.uw.edu
University of Washington
Seattle, Washington

Aileen Owens
amowens@southfayette.org
South Fayette School District
South Fayette, Pennsylvania

Mark Stehlik
mjs@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania

ABSTRACT

As computing technologies continue to have a greater impact on daily life, it becomes increasingly important for the K-12 education system to prepare students for the computerized world. In this paper, we present the curriculum design, implementation, and results from a one-trimester introductory Python course that is mandatory for all 8th graders in our school district. This course is a crucial component of the K-12 computational thinking pathways we are developing at our school district, which take students from block-based programming and computational thinking (elementary school) to text-based programming and applications of computer science (high school). Our mandatory 8th grade course serves as a bridge between these two components. We present qualitative results that highlight the challenges that arose from teaching a course for all students – not just those with a prior interest in computing – and how the instructor overcame those challenges. We also present quantitative results that demonstrate the course's positive impact on students' attitudes towards computer science, their intent to re-engage with computer science in the future, and the gender gap with regards to confidence in computer science.

CCS CONCEPTS

• **Social and professional topics** → **Computational thinking**; **Model curricula**; **Computing literacy**; **K-12 education**.

KEYWORDS

K-12 computer science education, computational thinking, robotics

ACM Reference Format:
Amal Nanavati, Aileen Owens, and Mark Stehlik. 2020. Pythons and Martians and Finches, Oh My! Lessons Learned from a Mandatory 8th Grade Python Class. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366906

*Research conducted while at Carnegie Mellon University.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366906

1 INTRODUCTION AND PRIOR WORK

As computing becomes woven into the economic, political, and social fabric of our lives, it becomes increasingly important for the K-12 education system to teach students computer science (CS) and computational thinking (CT) skills. There have been many efforts to integrate block-based programming into elementary schools and there are multiple text-based programming courses in high schools, but there are few explicitly designed transitions between them. Further, one approach to introducing students to CS and CT has been integrating computing activities into non-technical high school subjects, which requires that students have some baseline CS and CT knowledge upon entering high school. To address both these gaps, we developed an introductory middle school Python course, themed around Ridley Scott's *The Martian* [16], and made it mandatory for *all* 8th graders in our school district. The class was intended to teach students text-based programming as well as CT paradigms, and has been taught in our school district since 2017. This paper presents the curriculum for the course, qualitative results about lessons learned from teaching the course, and quantitative results about how the course impacted students' perceptions of CS. Our qualitative results point to ways to overcome the difficulties of teaching a course to *all* students, not just those with prior interest in CS. Our quantitative results reveal that taking the course positively increased students' interest in CS and openness to re-engaging with CS in the future, while lowering the gender gap in students' confidence in learning CS. The course also nearly tripled the number of 9th graders who requested high school CS elective(s).

Prior work on CS and CT courses in late-elementary or middle school often focused on activities that occurred outside the regular school day [2, 8, 14] or were an elective course [17]. As a result, students opted into the programs, skewing the student population towards those who were already interested in CS. One work investigated a brief (six lesson) robotics experience for *all* 500 students in a K-5 elementary school [12]. One of their findings was that students enjoyed using the robot during unstructured time, an approach we utilized by providing extra time for students to be creative. Another project focused on integrating a robotics experience into various non-technical K-12 classes [5], which resulted in *all* students in the class participating in the experience. They found that students' technical knowledge and perceptions about technology increased along multiple scales, and that students identified teamwork as a reason they enjoyed and learned during the lessons.

Many of the efforts to introduce CS and CT concepts in middle school use block-based programming languages such as Scratch [8], App Inventor [10], or Blockly [3]. Block-based programming languages have the benefit of allowing students to learn CS and CT concepts without worrying about complicated syntax, but have the

drawback of not preparing students for text-based programming in high school or beyond. Some programs began with block-based programming and transitioned to text-based programming [1, 2, 15]; they found that teaching block-based programming first helped increase students' motivation and allowed them to more easily understand concepts in the text-based programming language. In terms of curriculum design, prior work used robots [5, 12], Hollywood themes [6, 8], or unplugged activities [11], to name a few. We integrated many similar activities into our curriculum to achieve a balance between teaching useful content, teaching conceptually accessible content, and maintaining student interest and motivation.

2 SETTING

The work described in this paper came out of a collaboration between South Fayette School District, a suburban K-12 school district, and Carnegie Mellon University, a nearby research university in a mid-sized American metropolitan area. Our school district has 3,318 students, divided into an elementary school (K-2), intermediate school (3-5), middle school (6-8), and high school (9-12). The student body is 77% Caucasian, 2% African-American, 1% Hispanic, 16% Asian/Pacific Islander, and 4% Multiracial. In addition, 12% of students are eligible for free and reduced lunch, 8% qualify for special education, and 1% are English language learners.

Our school district has defined computational thinking as a new literacy for all students. Over the last nine years, we have worked to embed computational thinking, project-based learning, and human-centered design thinking into our K-12 curricula. In elementary and middle school, students learn to: (1) make interactive games and mobile apps using Scratch and App Inventor; (2) explore deeper learning in math and science using Lego EV3, VEX IQ, and Arduino boards; and (3) explore electrical circuitry using Squishy Circuits, LittleBits, and MaKey MaKey. From 9th-12th grade, students can take Python, Java, AP CS Principles, and/or AP CS A courses as electives. However, we have noticed a gap between the K-8 block-based programming and the 9-12 text-based programming, which negatively impacts students' attitudes towards, and willingness to engage with, CS in high school. Further, we would like to expand our K-12 computational thinking pathway by integrating computing into non-technical high-school courses. The 8th grade Python course we present was created to address both needs.

The 8th grade Python course was initially created by Teknowledge, a CS educational outreach student group at Carnegie Mellon University, and taught as an after-school incubator course at South Fayette School District in Spring 2016. Incubator courses are an important part of our school district's computational thinking pathway, since they allow us to test new ideas and leverage external talent in lower-stakes environments where only interested students take the course. We then worked with members of the school district to perfect and adapt the curriculum to fit into a one trimester course (30-days in a block schedule) for all students. This course has been taught as a mandatory 8th grade course since 2017, with roughly 25 students per class and 270 per year. After each trimester, we worked with the teacher and school district to further adapt the course to student needs and interests. This paper focuses on the course as it was offered at the end of the 2019 school year, and the results of student pre- and post-surveys from the 2018-2019 year.

Figure 1: *Left:* In "People Pathing," one student closes their eyes while the other gives them specific instructions – as if they were instructing a robot – to follow a path. *Right:* In "Catch the Bug," students are given a printout of Python code and have to fix the code as well as predict what it will do.

3 CURRICULUM DESIGN

The goal of this one-trimester course (30 classes, 40 minutes per class) is to provide students with baseline text-based programming and CT skills to enable them to effectively utilize CS in high school. Therefore, it covers (in order): printing, strings, string manipulation, variables, numbers, user input, mathematical operators, conditionals (if-elif-else), and iteration (while and for loops). The course also covers many Computer Science Teachers Association middle-school standards[1]. Each class begins with a brief period of instruction, followed by guided practice. During guided practice, the teacher interactively writes code on her computer (connected to a projector) while students follow along on their computers. Each class ends with students working on challenges or activities. The class has no required homework, and students are encouraged to meet with teachers for extra help during a flexible study hall period.

In order to expose students to the diverse ways in which computer code can interact with the world, the curriculum covers three input-output modalities: the command line, turtle bots, and Finch robots[2]. During the command line unit, students create Mad Libs and a Trivia Game to share with friends. During the turtle bot unit, students create geometric shapes and patterns and use the turtle bot to simulate Finch robot motion. During the Finch unit, students complete multiple realistic challenges, such as making the robot navigate a maze. The curriculum also incorporates unplugged activities, which help students understand key CT concepts and test students' abilities to read/write code without a computer (Figure 1).

The entire curriculum is themed around Ridley Scott's *The Martian* [16], a story about an astronaut, Mark Watney, who gets stranded on Mars. At the beginning of the course, students are told that they are Watney's tech specialists, and it is up to them to help Watney survive. When students learn printing, they must send messages to NASA to indicate that Watney is alive – but they must pick their messages carefully, since they have limited bandwidth. When they learn variables, they must create variables

[1]The course covers standards for Computing Systems, Data & Analysis, Algorithms & Programming, and Impacts of Computing: 2-CS-01, 2-CS-03, 2-DA-08, 2-AP-10, 2-AP-11, 2-AP-13, 2-AP-14, 2-AP-16, 2-AP-17, 2-AP-18, 2-AP-19, 2-IC-20, 2-IC-21.
[2]https://www.finchrobot.com/

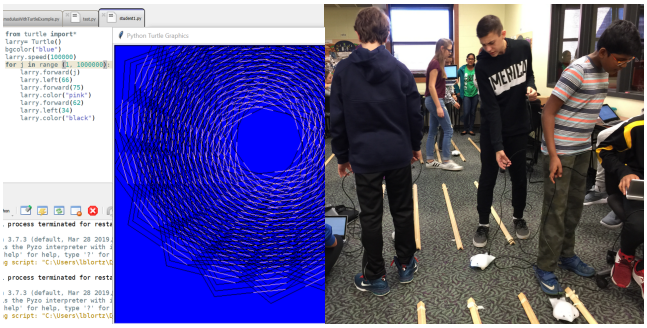

Figure 2: *Left:* A student-made geometric pattern from the turtle bots unit. *Right:* Students navigate the Finch robot through Mars' terrain to find the lost satellite.

for quantities like the oxygen level in Watney's settlement. When they learn string concatenation, they must create journal entries that document Watney's experiences for future space travelers. For mathematical operators, students must calculate how many potatoes Watney needs to survive. Students use turtle bots to create sand patterns that could indicate to passing satellites that Watney is alive. They use loops to print the decreasing fuel levels in Watney's spaceship. Finally, they use the Finch robot as Watney's Mars rover, to go out to retrieve the communication satellite that blew away. But when a sandstorm occludes the rover, students must learn to use the Finch's lights and sounds so Watney can track it. And after a hard day's work, students must navigate the Finch into a cave to park, by detecting light levels in the environment and turning on the Finch's light when it gets too dark. Figure 2 shows sample turtle bot and Finch robot activities. A majority of the activities are done in pairs, to teach students the importance of teamwork. Finally, students are encouraged to use Code Combat[3] for supplementary instruction outside of class, if they finish their activities early, or if there is a substitute teacher.

4 TEACHER EXPERIENCES

This course marked the first time our school district taught a *mandatory* text-based programming course, which made it new for the teacher and students. To understand the on-the-ground experiences of teaching the course, we held a semi-structured interview with the teacher who taught most sections of this course since it began in 2017. This teacher did not have prior experience with programming; she learnt the necessary skills by sitting in on the incubator course and the first trimester of the mandatory course (taught by a high school CS teacher). The teacher felt that this class had unique challenges since she was teaching *all* 8th graders as opposed to just those who opted into the class. As a result, there was a lot of variation in students' abilities, their interest in CS, and their predilection for CS concepts. This section recounts strategies the teacher used to address challenges arising from that fact, as well as other teacher experiences.

The teacher found that as students lost ground in the course they began giving up, and once they gave up it was very difficult to rekindle their interests. Therefore, the teacher worked hard to support students so they didn't lose interest in the first place. This included relying on study hall, a 40 minute period at the end of the day where students could meet with any teacher they desired. Study hall gave the teacher the chance to work with students individually: to re-engage students who did not understand the content, to support students who may have been too shy to raise their hand in class, and to push students who wanted further CS enrichment. The teacher estimated that she met with about 15% of her students during study hall. Importantly, she noticed that the conditionals unit was the most difficult for students, and was the time they were most likely to lose interest in the course.

The teacher felt that guided practice was a key to effectively teaching the content. Firstly, it kept students engaged and attuned to the multiple facets of each coding concept. Secondly, it allowed students to experience errors in a supportive environment. As the teacher encouraged students to debug by comparing their code to hers, she found that not only did students become more adept at fixing their errors, but they also became more forthright at helping others. Thirdly, guided practice helped the content seem more manageable; instead of solving one large challenge, students were solving multiple smaller challenges sequentially. Fourth, guided practice allowed the teacher to highlight the code's logic by first reasoning about a challenge in English, then converting it to pseudocode, and finally converting it to Python code. Lastly, guided practice helped demystify coding. Students would see the teacher writing code, see her make mistakes (sometimes intentionally), and see her thought process as she corrected those mistakes. In the teacher's view, this helped students realize that coding has less to do one's innate ability to be good at CS, and more to do with the thought process and reasoning one employs.

In terms of curriculum, the teacher felt that a few components were particularly important to keep students engaged and effectively teach CS concepts. Firstly, she felt that theming the whole course around *The Martian* was crucial to making the course interesting even for students who were uninterested in CS. Secondly, she felt that the interactive graphics (i.e., turtle bots) unit was significant, since students enjoyed the creativity associated with making their own graphics. Further, interactive graphics were more similar to previous forms of coding the students had experienced (e.g., moving a turtle on the screen is similar to moving a sprite in Scratch). Finally, she felt that the robotics unit was important to engage students and to let them see the real-world impact of their code. Further, since the robotics unit of the course included plenty of extra time for students to be creative, it allowed students who were behind on their work from other units to catch up. This was especially important since most students' parents were not familiar with programming, and therefore they could not get help at home for unfinished work. In addition to the main activities, the teacher felt that Code Combat was especially useful for her special needs students, due to its real-time hints and engaging graphics.

The teacher also noticed that gender played a role in how students interacted with the course material. Although the girls tended to be less excited by the course content, the teacher found that they participated more in class and were more focused during classroom activities. On the other hand, boys tended to be more excited by the course, but were more agitated and easily distracted in class. However, when they began the robotics unit, boys were more focused

[3]https://codecombat.com/

and interested in the activities. The teacher felt that this was due to an abundance of energy that boys of that age had, which dissipated as they stood up and moved around during the robotics unit.

Finally, the teacher felt that being only one trimester long was an important aspect of the class, because it helped prevent those who were not interested in CS from giving up on the course. She also felt that students came into the course expecting to have fun, since it wasn't a serious, one-year long class.

5 RESULTS

Every trimester, students took pre- and post-surveys to gauge their experiences with, and attitudes towards, CS and CT. The surveys were designed based on Hoegh's CS attitudes survey [13], shortened to account for students' attention spans, and modified to add questions that the school district was particularly interested in investigating. In the end, the survey analyzed seven constructs – Confidence, Interest, Gender Perception, Usefulness, CS for Future, CS Perception, and Problem Solving Approaches – and a few individual questions. Unless otherwise noted, all questions used a 5-point Likert scale from "Strongly Disagree" (−2) to "Strongly Agree" (2). The questions for each construct are listed in Table 1. We additionally collected demographic data, including students' gender and their prior familiarity with Python (a self-reported yes/no).

Overall, we gathered both pre- and post-survey responses from 180 students (87 M, 93 F; 29 with prior familiarity in Python), a 71% completion rate. We first ran Cronbach's alpha reliability analysis [4] to determine whether the intended constructs were, in fact, measuring the same concept (Table 1). All constructs except Usefulness were found to be reliable ($\alpha \geq 0.7$), and were averaged into one value each. The Usefulness questions were analyzed individually. We analyzed each construct using a repeated-measure ANOVA [9] with a type 2 sum of squares, using time-of-survey (pre- or post-survey) as the within-subjects factor and gender, prior familiarity with Python, and the trimester they took the course in as between-subject factors. Post-hoc testing was done using a t-test (paired when the independent variable was time-of-survey), with the Bonferroni correction [7] to address family-wise error rate. For each factor, we removed students who did not respond to all questions from both the pre- and post-data. We present all significant ($p < 0.05$) main effects and two-factor interaction effects.

Confidence: Time-of-survey had a significant effect on students' confidence ($F(1, 162) = 5.61$; $p = 1.91\text{e}{-2}$), where student confidence was higher in the post-survey (paired difference M = 0.13, SD = 0.73). Prior familiarity also had a significant effect on confidence ($F(1, 162) = 25.26$; $p = 1.31\text{e}{-6}$), where students who had prior familiarity with Python had a higher confidence (M = 0.88, SD = 0.76) across both surveys than those who did not (M = 0.14, SD = 0.81). There was also a significant interaction effect between gender and time-of-survey ($F(1, 162) = 6.83$; $p = 9.85\text{e}{-3}$), with no significant change in males confidence but a significant increase in females confidence ($p = 2.9\text{e}{-4}$; paired difference M = 0.26, SD = 0.68). However, females began with a significantly lower confidence than males on the pre-survey ($p = 5\text{e}{-3}$; female M = 0.02, SD = 0.89; male M = 0.38, SD = 0.91), which means that while the course did not end with equal confidence across genders, it significantly lowered the gender gap.

Interest: Time-of-survey had a significant effect on students' interest ($F(1, 163) = 7.53$, $p = 6.72\text{−}3$), where student interest was higher in the post-survey (paired difference M = 0.12, SD = 0.60). Gender also had a significant effect on students' interest ($F(1, 163) = 11.46$, $p = 8.9\text{e}{-4}$), where males were more interested in CS (M = −0.01, SD = 0.98) across both surveys than females (M = −0.47, SD = 0.90). Prior familiarity also had a significant effect on students' interest ($F(1, 163) = 16.63$, $p = 7.1\text{e}{-5}$), where students who had prior familiarity with Python were more interested in CS (M = 0.35, SD = 1.14) than students who did not (M = −0.36, SD = 0.88). There was also a significant interaction effect between trimester-taken and prior familiarity ($F(2, 163) = 3.82$, $p = 2.4\text{e}{-2}$), but given that the differences occurred only amongst the small group of students with prior familiarity with Python per trimester, they were likely due to arbitrary scheduling factors.

Gender Perception: The only significant effect was between gender and gender perception and CS ($F(1, 157) = 19.95$, $p = 1.51\text{e}{-5}$), with females more strongly believing that both genders were equally capable of pursuing CS (M = 1.47, SD = 0.60) than males (M = 1.08, SD = 0.73). Notably, taking the course had no significant impact on students' views about the relationship between gender and CS. This may be because students began the course already agreeing with the idea that females and males are equally capable of pursuing or studying CS (pre-survey M = 1.31, SD = 0.70).

Usefulness: Time-of-survey did not have a significant impact on any of the usefulness questions, which indicates that we could do a better job stressing the importance of CS and CT skills to future careers. Prior familiarity had a significant effect on responses to the question "Knowledge of computing will allow me to secure a good job" ($F(1, 166) = 14.88$, $p = 1.64\text{e}{-4}$), with students with prior familiarity with Python agreeing with the statement more (M = 1.02, SD = 0.94) across both surveys than students without it (M = 0.43, SD = 0.94). Prior familiarity also had a significant effect on responses to the question "My career goals do not require that I learn computing skills" ($F(1, 166) = 7.19$, $p = 8.07\text{e}{-3}$), with students with prior familiarity with Python disagreeing with it more (M = −0.07, SD = 1.32) across both surveys than those without (M = 0.34, SD = 0.98). The remaining significant effects for the usefulness questions had to do with trimester-taken or the interaction between trimester-taken and prior familiarity, which may have been the result of scheduling or subtle ways the teacher changed how she talked about the usefulness of CS across trimesters.

CS For Future: Time-of-survey had a significant effect on students' views about the role CS might play in their futures ($F(1, 164) = 23.00$, $p = 3.62\text{e}{-6}$), with students in the post-survey being more inclined to re-engage with CS (paired difference M = 0.25, SD = 0.70). Gender also had a significant effect on CS For Future ($F(1, 164) = 15.93$, $p = 9.89\text{e}{-5}$), with males being more inclined to re-engage with CS (M = −0.08, SD = 1.08) across both surveys than females (M = −0.64, SD = 0.94). Trimester-taken also had a significant effect on CS For Future ($F(2, 164) = 4.25$, $p = 1.58\text{e}{-2}$), which again likely had to do with scheduling factors beyond our control. The final significant main effect was students' prior familiarity with Python ($F(1, 164) = 14.35$, $p = 2.13\text{e}{-4}$), with students who had prior familiarity being more open to re-engaging with CS (M = 0.20, SD = 1.19) than those who didn't (M = −0.48, SD = 0.98). Finally, there was a significant interaction effect between prior familiarity

Construct	Questions	α
Confidence	• I have little self-confidence when it comes to computing courses.† • I am confident that I can solve problems by using computer applications. † • I do not think that I can learn to understand computing concepts. † • I am comfortable with learning computing concepts.†	0.75
Interest	• I hope that my future career will require the use of computer science concepts.† • I would voluntarily take additional computer science courses if I were given the opportunity.† • The challenge of solving problems using computer science does not appeal to me.† • I think computer science is boring.† • I like to use computer science to solve problems.† • (Post: After taking this course,) I am interested in working on my own to teach myself Python or another programming language.	0.91
Gender Perception	• Men are more likely to excel in careers that involve computing than women are.† • Men and women can both excel in computing courses.† • Women produce the same quality work in computing as men.† • Computing is an appropriate subject for both men and women to study.† • Men are more capable than women at solving computing problems.†	0.78
Usefulness	• Developing computing skills will not play a role in helping me achieve my career goals.† • Knowledge of computing will allow me to secure a good job.† • My career goals do not require that I learn computing skills.†	0.65
CS for Future	• I am considering pursuing a career in computer science. • I am considering taking computer programming courses in high school. • I am interested in pursuing courses in robotics in high school.	0.89
CS Perception	• I think computer-programming (Pre: sounds like / Post: can be) fun. • I think computer programming is easy to learn if I have help. • I think computer programming is too difficult for me to learn.	0.70
Problem Solving Approaches	• I try to find new ways of doing things even if they might not work. • When I am successful completing challenges or problems, I want to keep learning more. • I try to learn new things even if I might make mistakes.	0.72

Table 1: The questions included in each construct the survey investigated, as well as the Cronbach's alpha value indicating construct reliability. Text in parenthesis differed between the pre- and post-surveys. † questions came from [13].

and time-of-survey ($F(1, 164) = 4.76$, $p = 3.05e{-2}$), with people without prior experience in Python significantly increasing their consideration to re-engage ($p = 3e{-7}$; paired difference M = 0.30, SD = 0.70) while those with prior experience had no significant change. Therefore, taking the course significantly lessened the gap between students with and without prior experience in Python in terms of considering re-engaging with CS in the future.

CS Perception: Time-of-survey had a significant effect on students' perceptions of CS ($F(1, 164) = 8.23$, $p = 4.66e{-3}$), with students having a better perception of CS on the post-survey (paired differences M = 0.15, SD = 0.71). Prior familiarity also had a significant effect on students' perceptions of CS ($F(1, 164) = 12.41$, $p = 5.53e{-4}$), with students who had prior familiarity with Python having a better perception of CS (M = 0.99, SD = 0.81) across both surveys than students who did not (M = 0.43, SD = 0.82).

Problem Solving Approaches: Gender had a significant effect on students' problem solving approaches ($F(1, 164) = 5.70, p = 1.81e{-2}$), with males agreeing more with the survey prompts (M = 0.97, SD = 0.66) across both surveys than females (M = 0.75, SD = 0.69). Prior familiarity with Python also had a significant main effect on students' problem solving approaches ($F(1, 164) = 9.83$, $p = 2.03e{-3}$), with students who had prior familiarity with Python agreeing more with the survey prompts (M = 1.17, SD = 0.73) across both surveys than those who did not (M = 0.80, SD = 0.66). Notably, there was no significant effect between time-of-survey and problem solving approaches, which indicates that taking the course likely had little to no impact on the measured problem solving skills.

Post-Survey Specific Questions: Based on students' self-reported assessment of their Python abilities in the post-survey, the course seems to have been successful in teaching them Python. 84% of students answered "Yes" to the question "Upon completing this course, I am familiar with the Python programming language", compared to 16% for a similar question on the pre-survey. Further, 50% of the students answered "Agree" or "Strongly agree" to the question "I think I can write code at home now even without the teacher's help," 36% answered "Neither agree not disagree," and only 24% answered either "Disagree" or "Strongly disagree." In addition to self-reported proficiency, we also asked students which components of the course they wanted to spend more time on. Figure 3 shows the results, divided by gender. As can be seen, students tended to prefer the fun activities over the process of sitting down and learning code; the graphics, Finch, and Trivia Game activities were very popular across genders, and print/input and conditionals were very unpopular across genders. The unit on loops was an interesting exception to this; it was popular across genders, more so for males, despite being a section where students had to sit down and learn a

Figure 3: The parts of the course that students wanted to spend more time on, divided by gender.

new coding construct. The reason for this could lie in the teacher's experiences; she found that students quickly grasped the concept of loops and then moved on to being creative with it by, for example, creating infinite loops or creating loops that printed their names.

High School Course Requests Finally, we analyzed the number of students who requested CS courses as their high school elective(s) from 2015 onwards. High school CS courses included Python, Java, AP CS Principles, AP CS A, Data Science, and Cybersecurity. Only AP CS A and Java were offered in 2015-2016, and Data Science and Cybersecurity are new courses in 2019-2020 that will build on students' 8th grade Python knowledge. As Figure 4 Top shows, we found that that number of 9th graders who requested high school CS courses doubled after the introduction of the Python course, and is currently triple of what it was before the 8th grade course was offered. The number of 10th graders requesting CS courses two years after the start of the mandatory course also roughly doubled. This indicates that the 8th grade Python course increased student's desire to re-engage with CS in high school. Figure 4 Bottom shows the number of 9th graders who requested high school CS courses partitioned by gender. Despite the increase in the number of students who requested CS courses since the 8th grade course began, the gender proportion stayed roughly the same (1 female for every 2 males). This indicates that we have more work to do in the 8th grade Python course and our whole K-8 computational thinking pipeline to close the gender gap in CS.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented a one-trimester (30-day) introductory Python course that has been taught since 2017 as a mandatory 8th grade class. The goal of the curriculum is to provide all students baseline experiences with text-based programming and computational thinking skills that they can further develop and apply in high school courses. In order to balance skill learning with fun activities that keep students interested in the class, we themed the class around Ridley Scott's *The Martian* and included various unplugged activities, creative writing activities (i.e. Mad Libs and Trivia Game), graphics activities, and robotics activities. Our qualitative results showed that the fact that this course was mandatory for *all* 8th grade students presented unique challenges for the teacher. She found that a flexible study hall period where she could work one-on-one with select students was crucial to handling those challenges, as well as the large variety of activities that appealed to the diverse student body. Our quantitative results showed significant improvements in students' confidence and interest in CS, their intent to

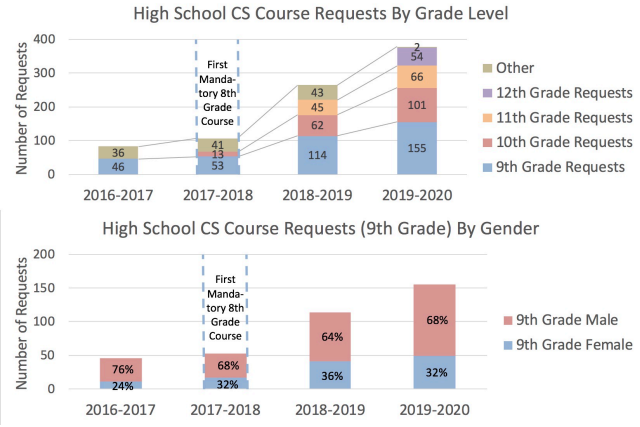

Figure 4: *Top:* The number of students who requested high school CS courses, by grade level. "Other" refers to either students who have graduated and therefore no longer have data stored at the school district, or advanced middle school students. *Bottom:* The number of 9th grade students who requested high school CS courses, partitioned by gender.

re-engage with CS in the future, and their perceptions of CS. We also found that the course significantly lowered the gender gap in students' confidence in CS, and the prior experience gap in students' openness to re-engaging with CS. We found that the students most wanted to spend more time on the graphics, robotics, and Trivia Game activities, while least wanting to spend more time on print/input and conditionals. Finally, we found that student requests to take high school CS courses were much higher amongst students who took the 8th grade course (triple for 9th graders), although the gender proportion remained roughly the same.

This course is a work-in-progress; it evolves with the field of computer science and with our school district's pathways for K-12 computational thinking. As such, we intend to integrate more data science into the course – by, for example, having students analyze the Finch's light sensor readings to determine whether the Finch is in a cave or not – as well as discrete math – by, for example, having students reason about how many colors it takes to color a map of Mars. In addition, this paper brings up exciting new directions of research inquiry for the CS education community. For example, future work could study the *long-term* impacts of a mandatory CS course on student outcomes, or develop high-school CT modules that build upon text-based programming and CT skills that students would have gained in middle school through this course.

We invite interested school districts and community partners to contact our school district to learn how they can access, adapt, and teach this curriculum and other K-12 CS curricula in their classes.

ACKNOWLEDGMENTS

Sincerest thanks to Lynette Lortz and Dawn McCullough for taking on the adventure of teaching this class, the students for making it enjoyable, and our friends and colleagues at South Fayette School District and Carnegie Mellon University, especially the Teknowledge team, for their support, guidance, and collaboration.

REFERENCES

[1] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. 2015. From Scratch to &Ldquo;Real&Rdquo; Programming. *Trans. Comput. Educ.* 14, 4, Article 25 (Feb. 2015), 15 pages. https://doi.org/10.1145/2677087

[2] Caelin Bryant, Yesheng Chen, Zhen Chen, Jonathan Gilmour, Shyamala Gumidyala, Beatriz Herce-Hagiwara, Annabella Koures, Seoyeon Lee, James Msekela, Anh Thu Pham, et al. 2019. A Middle-School Camp Emphasizing Data Science and Computing for Social Good. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 358–364.

[3] Philip Sheridan Buffum, Kimberly Michelle Ying, Xiaoxi Zheng, Kristy Elizabeth Boyer, Eric N Wiebe, Bradford W Mott, David C Blackburn, and James C Lester. 2018. Introducing the Computer Science Concept of Variables in Middle School Science Classrooms. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 906–911.

[4] Lee J Cronbach. 1951. Coefficient alpha and the internal structure of tests. *psychometrika* 16, 3 (1951), 297–334.

[5] Jennifer L Cross, Emily Hamner, Lauren Zito, and Illah Nourbakhsh. 2017. Student outcomes from the evaluation of a transdisciplinary middle school robotics program. In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[6] Amber Dryer, Nicole Walia, and Ankur Chattopadhyay. 2018. A Middle-School Module for Introducing Data-Mining, Big-Data, Ethics and Privacy Using RapidMiner and a Hollywood Theme. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 753–758.

[7] Olive Jean Dunn. 1958. Estimation of the means of dependent variables. *The Annals of Mathematical Statistics* (1958), 1095–1111.

[8] Russell Feldhausen, Joshua Levi Weese, and Nathan H Bean. 2018. Increasing Student Self-Efficacy in Computational Thinking via STEM Outreach Programs. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 302–307.

[9] Andy Field, Jeremy Miles, and Zoë Field. 2012. *Discovering statistics using R*. Sage publications.

[10] Shuchi Grover, Satabdi Basu, and Patricia Schank. 2018. What we can learn about student learning from open-ended programming projects in middle school computer science. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 999–1004.

[11] Shuchi Grover, Patrik Lundh, and Nicholas Jackiw. 2019. Non-programming activities for engagement with foundational concepts in introductory programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 1136–1142.

[12] Cecily Heiner. 2018. A Robotics Experience for All the Students in an Elementary School. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 729–734.

[13] Andrew Hoegh and Barbara M Moskal. 2009. Examining science and engineering students' attitudes toward computer science. In *2009 39th IEEE Frontiers in Education Conference*. IEEE, 1–6.

[14] Chrystalla Mouza, Alison Marzocchi, Yi-Cheng Pan, and Lori Pollock. 2016. Development, implementation, and outcomes of an equitable computer science after-school program: Findings from middle-school students. *Journal of Research on Technology in Education* 48, 2 (2016), 84–104.

[15] Farzana Rahman. 2018. From App Inventor to Java: Introducing Object-oriented Programming to Middle School Students Through Experiential Learning. In *2018 ASEE Annual Conference & Exposition*. ASEE Conferences, Salt Lake City, Utah. https://peer.asee.org/30539.

[16] Dir. Ridley Scott, Prod. Drew Goddard, Teresa Kelly, Simon Kinberg, et al. 2015. *The Martian*. 20th Century Fox.

[17] Jennifer Tsan, Kristy Elizabeth Boyer, and Collin F Lynch. 2016. How early does the CS gender gap emerge?: a study of collaborative problem solving in 5th grade computer science. In *Proceedings of the 47th ACM technical symposium on computing science education*. ACM, 388–393.