PATCH-ED

Creating Curricula to Effectively Transition Young Computer Science Students from Block-Based to Text-Based Programming

Duncan Johnson, Laidlaw Scholar

Dr. Ethan Danahy, Laidlaw Mentor

Tufts University, Medford, MA

September 2023

ABSTRACT

Introducing middle school students to text-based programming languages like Python is difficult, especially in such a way that facilitates student creativity. Patch (codepatch.org) is a coding platform that helps students bridge the gap between block-based and text-based programming languages, but no curricula existed for instructors to utilize Patch in their classroom. I developed two versions of a curriculum for Patch, experimenting to find effective lessons and teaching methods. In the summer of 2023, I taught my curricula in two computer science workshops for middle school students in Columbus, Ohio. While many lessons had significant flaws, I found promising instances of student creativity and made improvements to the curriculum after each workshop. To allow for global access to the Patch-Ed curriculum, I'm hosting the curriculum online (patch-ed.org) for any educator to freely use in their classroom.

INTRODUCTION

Scratch (scratch.mit.edu) is the largest free coding platform for children, with over 113 million users since its release in 2007 (Scratch Foundation). Targeted at students ages 8 to 16, Scratch offers visual coding blocks that users can drag and snap together to create scripts that move characters, play sounds, and much more. Users can easily build creative Scratch projects by adding and customizing characters, backdrops, and sounds.

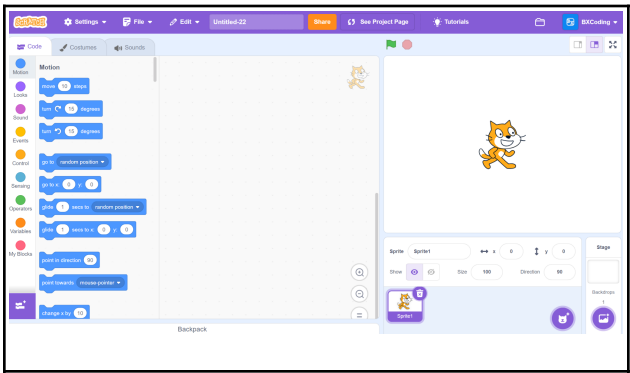
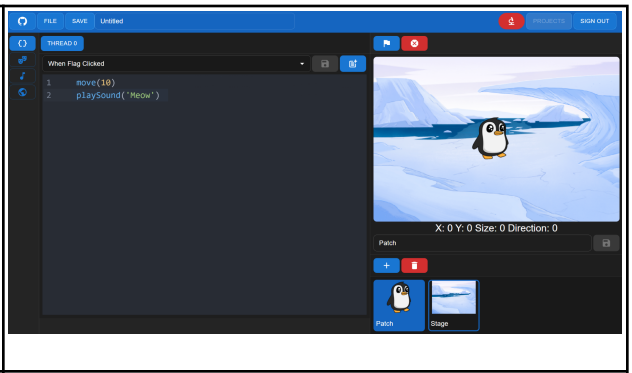
Python is a widely used programming language with applications in software development, data science, and machine learning. To create even simple programs in text-based languages like Python, users must memorize a significant amount of vocabulary and syntax, which can be difficult for new users (Liao, 2022).

I've taught both Scratch and Python at week-long summer workshops operated by BX Coding, the nonprofit I founded with Elliot Roe in 2019. Frequently, I found students creatively building and exploring with Scratch, but rarely did I find the same level of creativity in students' Python projects. Since Python doesn't easily allow for the usage of sounds, images, or animations in programs, students coding in Python had much less freedom with the projects they could create than those coding in Scratch.

Identifying that students were having difficulties transitioning from Scratch to Python, BX Coding decided to create Patch, a coding platform to be a midpoint between Scratch and Python. Patch's interface is similar to Scratch (Table 1), allowing users to easily add characters, backdrops, and sounds to their projects, but has the key difference that students have to type their code rather than dragging blocks together. Instead of asking students to memorize hundreds of Python commands, Patch has

commands that exactly mirror many Scratch blocks, allowing for students' Scratch knowledge to transfer into Patch. For example, the "Play Sound Meow" block in Scratch is simply "playSound("Meow")" in Patch. However, the key programming concepts of variables, functions, loops, indentation, and parameters are identical in Patch and Python, meaning proficiency in Patch gives students a strong foundation in Python and other text-based languages.

Table 1

	
Scratch Editor https://scratch.mit.edu/projects/editor	Patch Editor https://codepatch.org

There weren't any curricula available for educators who wanted to use Patch to assist in the transition from block-based to text-based programming, so my research project was to build a curriculum that introduces middle school students to Python by first building skills in Scratch and Patch. After initial research and interviews with experts, I created a curriculum that I implemented in two different BX Coding summer workshops, iterating upon the curriculum after each week-long workshop. I'm treating the two workshops as different experiments: a half-day workshop and a full-day workshop.

EXPERIMENT 1

This half-day, week-long workshop in June 2023 was three hours a day, Monday through Friday, in a computer lab in Bexley, Ohio. There were 11 students between the ages of 11 and 14 who learned about the workshop from the Bexley Recreation and Parks Department's summer activity brochure and signed up for $150.

METHODOLOGY

I set out to develop a curriculum that emphasized student creativity and agency, utilizing the freedom of exploration that Scratch and Patch provide. I started by researching methods for facilitating student-directed projects, reading two papers (Brennan et al., 2021; Brennan, 2021) and discussing the topic with Dr. Ethan Danahy, my Laidlaw Mentor. Strongly connected to the goal of self-directedness is the concept of failure. Middle school students, many with little to no programming experience, were bound to fail when creating projects without step-by-step instructions. Dr. Danahy introduced me to "Importance of Failure in Young Students' Engineering," a presentation by Dr. Chelsea Andrews of the Tufts University Center for Engineering Education and Outreach (CEEO). Dr. Andrews emphasized that students could be trusted to effectively test and iterate and that instructors should work to create a safe space for failure.

While researching potential lesson structures, I came across the PRIMM model (Sentance et al., 2019), which is an acronym for "Predict, Run, Investigate, Modify, and Make". In the PRIMM model, students are first given code and asked to *predict* what will happen when the code is executed. Then, they *run* the code, observing and recording what happens. Next, students are asked questions that prompt them to closely

investigate important parts of the code. Students are then given ideas for ways to *modify* the provided code by customizing the program or adding new functionality. Finally, students open a blank code file and are given an open-ended prompt to *make* a new program that utilizes a concept from the instructor-provided code explored in the first four steps to solve a different problem (Sentance et al., 2019). The PRIMM model fit well with my goals of self-directedness and failure, since students were able to *modify* and *make* projects with few restrictions, and had ample opportunities to fail, iterate, and learn when making their own program.

Dr. Danahy connected me with David Zabner, a PhD student in Tufts University's STEM Education program, who is a proponent of the Use-Modify-Create (UMC) lesson structure, which is very similar to the PRIMM model. I spoke to David, and he argued that the UMC structure teaches fearlessness, which is the most important trait for young programmers. Students are taught to tackle code they've never seen and extract its concepts to solve new problems. David believes students who learned from the UMC/PRIMM structures will be better equipped to analyze and utilize code they don't understand than students who learned via traditional lectures.

Using what I learned, I created a curriculum for the half-day workshop, building 15 hours of lessons, including lecture slides, student handouts, and activity plans. I used a wide variety of lesson structures, from traditional lecture-based lessons to self-directed PRIMM projects and even "unplugged" activities that taught core computer science concepts without using any technology. The first day and a half of the workshop were taught using Scratch, the next two days utilized Patch, and in the final day and a half, students coded in Python.

I taught the first workshop alongside fellow BX Coding directors Elliot Roe and Ava Wandersleben, taking notes about the successes and failures of each day and informally interviewing Elliot and Ava to hear their thoughts. I also examined the work produced by students to get a better idea of whether each lesson effectively conveyed the concepts being taught. I used these sources of data to improve my curriculum for workshop two.

Finally, I surveyed each student at the beginning of the workshop, asking them to identify their Scratch and Python experience on a scale that included responses “none,” “a little,” “some,” “a lot,” and “tons” so that I could compare the prior experience of students across the two different workshops.

RESULTS

Day three of the workshop started with a Patch warm-up, where students were given a short prompt and I planned for students to spend just a few minutes coding in Patch. Shockingly, the class worked on Patch projects for a full 30 minutes before I cut them short to start the scheduled lesson. Before day three, students had only used Patch for 90 minutes and only learned a few basic concepts. After an hour and a half of text-based programming instruction, students were creatively exploring. During this warm-up activity, one student created a project that included character cloning without ever being taught the Patch code required to create and control clones. She knew how the character cloning blocks worked in Scratch and applied her knowledge to the new environment of Patch.

However, not every activity was a success. In each PRIMM lesson, students have to write short responses for the *predict*, *run*, *investigate*, and *modify* steps. For some PRIMM lessons, I printed out worksheets where students would handwrite their responses, and in others, I had students type their responses directly into the code as non-executed “comments”. I found that students were unlikely to type responses as comments in their code, but were more likely to write responses onto physical worksheets. Regardless of the delivery medium, I felt PRIMM activities were effective at teaching concepts and allowing students to be creative in how they applied new skills.

In the unplugged activities, I found that students were standing around unengaged for too much of each activity. For example, in the robot maze activity, where teams had to write commands that guided a student acting as a “robot” through a maze created by drawing chalk on the ground, once each team of three completed the maze, they didn’t have another task to complete while other teams completed the maze and quickly lost interest in the activity.

EXPERIMENT 2

This full-day, week-long workshop in July 2023 was six and a half hours a day, Monday through Friday, in a science lab in Gahanna, Ohio. There were 15 students, rising 6th through 9th graders, who learned about the workshop from Columbus Academy Summer Experience’s brochure and signed up for $340. Accounting for lunch and recess time, there were roughly 20 hours of instruction.

METHODOLOGY

Using the data collected from the first workshop, I created the curriculum for the second workshop. Since the second workshop was full-day rather than half-day, I had more instruction time to allocate for additional Patch and Python activities that reinforced concepts I felt students didn't completely grasp in workshop one. After seeing in workshop one that students would be able to creatively explore in Patch, I also budgeted more time for students to freely code projects that interested them. Additionally, to attempt to remedy the engagement issues with unplugged activities, I changed many of them to allow an instructor to pull out a small group of students at a time. For the outdoor robot maze game, I changed the activity between workshops one and two by adding a related indoor component and then having one instructor bring four or five students outside at a time to complete the maze.

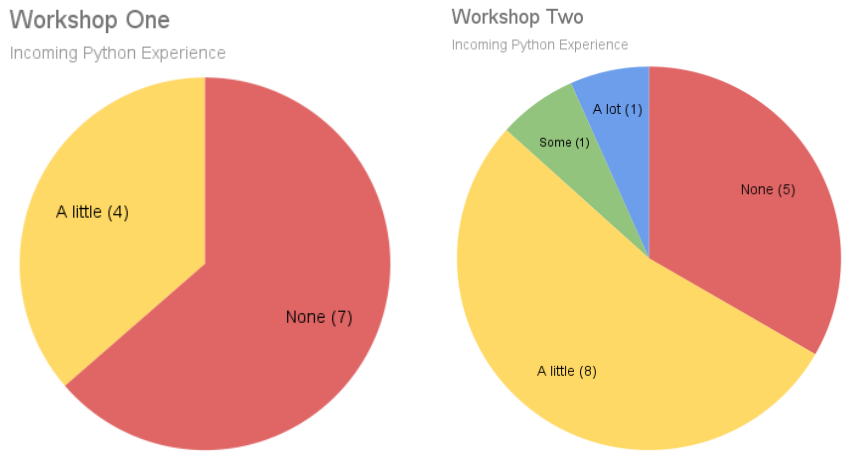
After implementing PRIMM activities during workshop one and finding that students preferred writing each prompt's short responses on physical worksheets rather than typing responses throughout the code, I adapted all the PRIMM activities to be paper handouts. I even decided to switch some lecture-based lessons to PRIMM activities. For example, in workshop one, I taught students to code a trivia game in Python by presenting the concepts required before asking students to code their own themed trivia game, but for workshop two, I taught trivia as a PRIMM activity.

As with the first workshop, I gave an entrance survey to all students to gauge their incoming experience with Scratch and Python.

RESULTS

From my notes, informal interviews with Elliot and Ava, and student artifacts, I felt the second workshop was overall clearer and more engaging for students than the first workshop. As expected, having several more hours of instruction allowed for more time to thoroughly explain and practice core concepts, leading to more complex student-directed projects. Yet, the improvement from one workshop to another cannot be fully explained by changes in the curriculum, as the students in the second workshop had significantly more incoming experience in Python than the students in the first workshop, as shown from the responses to the entry survey (Figure 1).

Figure 1



After the changes made to some unplugged activities from workshop one, I found that workshop two's small group unplugged activities were much better. For the robot maze activity, with most students engaged inside, an instructor was able to work with a small group of students outside to efficiently complete the maze activity without significant downtime.

For the *predict*, *run*, *investigate*, and *modify* steps of the newly-created trivia PRIMM activity for workshop two, I gave students the code for an addition and subtraction calculator in Python, then asked them to *modify* the calculator by adding the multiplication and division operators. For the *make* step, I simply asked students to create a trivia game, hoping that students would apply the observed Python concepts of user input, conditionals, and control flow to a completely different task. Ambitiously, I placed the calculator-to-trivia PRIMM activity as the first purely Python lesson in the curriculum. While some students struggled and required significant assistance, other students were able to successfully create a trivia game in Python. After three and a half days of Scratch and Patch, a five-minute presentation on Python, and being shown the Python code for a calculator, some students could independently code a trivia game in Python from the simple instruction “create a themed trivia game.” The results of this activity point to the idea that for some students, instruction in Scratch and Patch builds knowledge that can be quickly transferred to Python.

DISCUSSION

Observations from both weeks point to the idea that Patch and the Patch-Ed curriculum allow students to creatively explore text-based coding after only a short period of instruction time. Especially in the second workshop, where there was more time to build text-based programming skills in Patch, some students were able to pick up and create in Python very quickly.

However, there were a few limitations that affected the results. First, the workshop instructors were not representative of the average computer science

instructor. As the curriculum developer, my perspective as an instructor is different from a more detached instructor. Elliot and Ava, the two other instructors, had an intimate understanding of Patch since they both did significant work to code the Patch platform. Second, I taught the Patch-Ed curriculum in week-long summer workshops, but that's not the average environment where middle school students are introduced to text-based programming. Many middle school computer science courses meet for less than an hour each day or even just once a week. Future work is required to adapt the Patch-Ed curriculum to effectively fit a less compact lesson schedule.

One of my goals for this project is to publish the curriculum online so that instructors can use the entire curriculum or just specific lessons in their classrooms. Currently, I'm working to package the curriculum and host it on patch-ed.org, and I hope to observe other instructors teaching my curriculum so I can learn from how more experienced educators implement each lesson.

CONCLUSION

Utilizing Scratch, a block-based programming platform, and Patch, a coding platform halfway between block-based and text-based programming, I created curricula that taught middle school students to code in Python. After initial research into computer science pedagogy, I created a curriculum and implemented it in two week-long summer workshops for middle school students, iterating and improving upon the curriculum after each workshop. In the two workshops, I found that some students were able to creatively explore text-based programming in Patch without significant instruction. I also found instances of skills built in Scratch transferring into Patch and Patch knowledge

transferring into Python. I'm now working to host the curriculum on patch-ed.org so that other instructors can freely and easily use the Patch-Ed curriculum in their classrooms.

ACKNOWLEDGEMENTS

Dr. Ethan Danahy, for his invaluable mentorship, guidance, and support.

Elliot Roe, Ava Wandersleben, and the entire BX Coding team, for their hard work building Patch and teaching at workshops.

The Laidlaw Foundation, for creating the wonderful Laidlaw Scholars program to support the research and leadership development of myself and many others.

The Tufts-Laidlaw Team of Andrew Singleton, Dr. Dawn Terkla, and Diana Capone, for their flawless facilitation of all aspects of the Laidlaw Scholars program.

CONFLICT OF INTEREST DISCLOSURE

Duncan Johnson, the principal investigator of Patch-Ed, is the Executive Director of BX Coding, a nonprofit dedicated to expanding access to computer science education. BX Coding created Patch, a free and open-source computer science education software utilized by the Patch-Ed curriculum.

REFERENCES

Brennan, K. (2021). How kids manage self-directed programming projects: Strategies and structures. *Journal of the Learning Sciences*, *30*(4–5), 576–610. https://doi.org/10.1080/10508406.2021.1936531

Brennan, K., Blum-Smith, S., Peters, L., & Kang, J. (2021). Designing for student-directedness: How K–12 teachers utilize peers to support projects. *ACM Transactions on Computing Education*, *22*(2), 1–18. https://doi.org/10.1145/3476515

Liao, S.-M. (2022). Scratch to R: Toward an inclusive pedagogy in teaching coding. *Journal of Statistics and Data Science Education*, *31*(1), 45–56. https://doi.org/10.1080/26939169.2022.2090467

Scratch Foundation. (n.d.). *Community Statistics*. Scratch. https://scratch.mit.edu/statistics/

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, *29*(2–3), 136–176. https://doi.org/10.1080/08993408.2019.1608781