Interpreting Student Code: Prospective Teachers' Scripted Responses to a Scratch© Coding Task

Dov Zazkis
Montclair State University

Ami Mamolo
Ontario Institute of Technology

We explore prospective intermediate and secondary school teachers' strategies for supporting students in coding and troubleshooting errors while working in Scratch ©. Participants were asked to create an imagined dialogue that addressed student-generated "starter" code, which included bugs, in order to explore the probability of different sums of two dice rolls and present these sums as a graph. Our analysis shows that participants attended to different possible student difficulties with the coding, yet few of them attended to the given starter code as an instance of student thinking when suggesting viable resolutions or ways forward. Instead, participants tended to introduce their own approaches and trajectories for how to advance the code. We suggest that while participants demonstrated useful teaching strategies to support student-characters, their attention was focused on advancing any code, rather than building with or on ideas presented by the student-characters.

Keywords: Probability, Prospective teachers, Computer programming, Scripting, Scratch©.

Recognizing and interpreting student errors as a way to advance student learning and understanding is advocated for in research and policy documents (e.g., NCTM, 2000; Son, 2013). A wide variety of learning theories and studies have emphasized the importance of learning trajectories that build with student thinking (e.g., Hershkowitz, et al., 2001; Simon, 2006). However, researchers have found that prospective teachers commonly prefer providing teacher guidance that mirrors their own thinking as opposed to building on students' thinking. Mamolo and Pali (2014) argued that robust mathematical knowledge is linked with prospective teachers' flexibility in adopting a variety of guidance strategies to support and advance students' thinking, even when the approaches are different from those teachers' own preferences or prior experiences as students. In this research, we are interested in how prospective teachers provide guidance that builds on a hypothetical student's thinking in relation to content for which participants had no analogous prior experiences as students - developing mathematical ideas through coding. Specifically, we address the following research question:

What can a scripting task reveal about how prospective teachers work with student ideas while debugging a coding simulation for rolling two dice?

Background Literature

Block-based coding in programming languages such as Scratch © have become increasingly common in teaching mathematics (e.g., Benton et al., 2017; Burke, et al., 2012; Passey, 2016). Francis and Davis (2018; see also Pei et al., 2018) describe coding as a competency that "can complement and co-amplify mathematics learning," emphasizing its "deep roots in and emergent powers for mathematics" (p.14). When used as a tool to address mathematics problems, coding has been described as transformative to learning (Papert, 1980), with benefits including fostering resilience, creativity, and learner-driven activity. Research has also pointed to benefits such as: self-motivation; experimentation; development of mathematical intuitions and approaches; critical reflection; and working with abstraction and different representations (e.g., Bailey & Borwein, 2011; Henderson, Cortina, & Wing, 2007; Howson & Kahane, 1986; King et al., 2001; Weintrop et al., 2016).

Notwithstanding the positive potential of teaching mathematics through coding, Resnick and Rusk (2020) noted a concern about teachers treating coding with an instrumental instructional approach – such as memorizing definitions, or copying lines of code – instead of fostering student reasoning, experimentation, and active engagement. They warn that the positive potential will be lost if coding is reduced to a memorization exercise. In the jurisdiction where this research took place, coding is newly embedded within the K-12 mathematics curriculum, and teacher education programs have sought ways to support prospective teachers in learning both related content and pedagogy (e.g., Gadandis, et al., 2017). Yet, there is limited research on how to best support prospective teachers in adopting this new-for-them approach (e.g., Gleasman & Kim, 2018, 2020; Mamolo, et al., 2022). Mamolo et al. (2022) suggest that "before coding can be thought of or used as a tool through which to develop and promote mathematics learning, there is a need to establish a foundation of coding practices, content, and confidence" (p.975). They highlight practices related to experimenting and iterating, remixing and reusing, testing and debugging, and stepwise refinement, amongst others.

The need for research in coding education for teachers, along with the concern raised by Resnick and Rusk (2020), compelled us to take a closer look at how prospective teachers attend to student thinking when developing teaching competencies and content knowledge related to coding. We chose the mathematical context of probability since this is a topic for which coding can serve as a useful tool (e.g., Franklin et al., 2015) and is a topic in which students teachers struggle (e.g., Chernoff & Russell, 2012; Shaughnessy, 1977; Stohl, 2005).

Theoretical Underpinning

A theoretical position that supports our research can be found in the work of Carpenter, Fennema, and colleagues, who advocate for cognitively guided instruction that builds on the knowledge of students (e.g., Carpenter & Fennema, 1992; Carpenter, Fennema, & Franke, 1996; Carpenter et al., 1989). In this perspective, teaching is conceptualized as a student-centered activity in which students' ideas, intuitions, and experiences are built upon in the pursuit of new learning. The premise is that by engaging with student thinking, teachers can uncover and expand students' mathematical understanding beyond procedural or rote learning. Researchers working from this perspective primarily research teacher training that focuses on teachers building on students' thinking. We instead use cognitively guided instruction as an impetus for asking, "Is this prospective teacher working to build on student thinking, and if so, how?" This parallels our research question.

Methods and Task

A total of 24 prospective intermediate and secondary teachers participated in this research by developing scripts of imagined dialogues as part of their regular classroom assignments for a methods course in coding education. Scripting or role-playing tasks in which participants imagine an exchange between a teacher-character and student-characters have been used in prior research to explore and develop mathematical knowledge (e.g., Bergman et al., 2023; Kercher et al., 2023; Zazkis & Zazkis, 2014) and pedagogical awareness (e.g., Campbell & Baldinger, 2022; Lajoie & Maheux, 2013; Mamolo, 2021, 2017). In this study, we extend the applicability of these tasks to explore how prospective teachers engage with student thinking in a coding context. Our data includes the scripts of fourteen prospective teachers who elected to work in groups of 2-3 and handed in joint work (referred to as Groups 1-7), and 10 participants who composed their scripts individually (referred to by pseudonyms). The scripting task had two core components: 1) an initial student-generated code, which was to be the starting point for the script (Figure 1), and

2) a goal of building on the provided background to debug the existing code and develop a future plan (Figure 2).

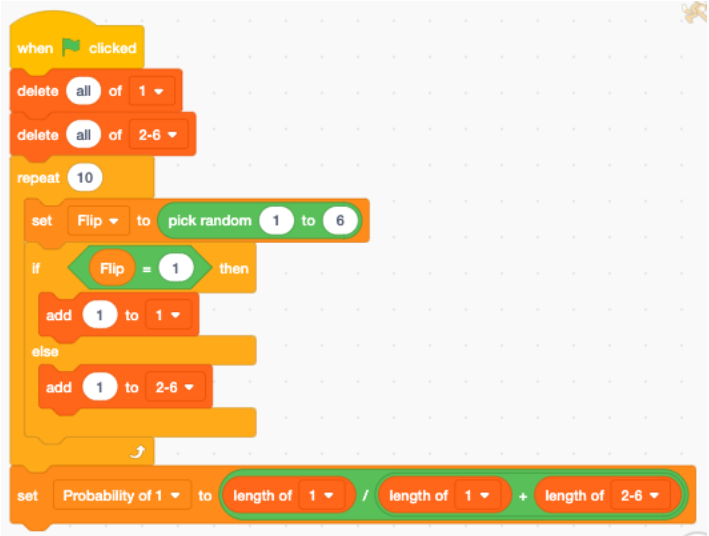


Figure 1. Initial student scratch code for scripting task

Background: A few students are creating a dice simulation for a probability assignment. They want to repeatedly roll two dice, record the sum and then graph the results. They have made a great start with their code: https://scratch.mit.edu/projects/324585345/editor
Bugs: The group notices the program is not recording the sum (it is only recording the most recent sum). They also realize that the program is repeating 10 times- they want the user to choose how many times to roll the die.
Future Plans for Project: The group wants to graph their results but does not know how to do this.

Figure 2. Two Dice Scripting Assignment

In analyzing the data, the authors, first individually and then jointly, carefully documented what occurred in each script. This included the trajectory the code followed from the initial code in Figure 1 to the finalized code that the student-characters in the script created. We also documented which benchmarks were accomplished by the time the script concluded (e.g., was the user prompted for a number of rolls? Did the final code have two dice? Did the code graph the results, and, if so, how?). Finally, we documented which difficulties the student-characters encountered and how these were overcome.

To better acquaint the reader with the scripting task and familiarize the reader with coding in Scratch, it is worthwhile to parse the initial code in Figure 1. The code begins by deleting the content of previously saved lists named "1" and "2-6." Then there is a loop that repeats 10 times. With each repetition, a variable named "flip" is assigned a random integer value between 1 and 6. Then if "flip" is 1, a 1 is added to the list named "1", otherwise, a 1 is added to the list named "2-6." The result is two lists of 1's with a total length of 10. The length of the first list represents the number of times a 1 occurred, while the length of the second list represents the number of times a value other than 1 was rolled.

There are several things to note about the code in Figure 1, given that the scripting task asks that it be interpreted as initial student work. First, only two outcomes are documented within the code, whereas the problem scenario encompasses 11 possibilities, as rolling two dice can yield any integer sum between 2 and 12, inclusive. Next, there is only one variable assigned to a random value. We would expect two variables since there are two dice in the problem scenario. Finally, one generally flips a coin and rolls a die. So, it is unusual to name the variable "flip", as opposed to "roll." A reasonable conclusion to draw is that the code in Figure 1 started as code for

flipping one coin. This explains all three of the above observations (number of variables, number of possibilities documented, and the variable name). So, a natural interpretation of this initial student code is that students started with code for documenting coin flips, began remixing it to account for the differences between coins and dice, and got stuck.

Results and Analysis

Only 4 of the 18 completed scripts included an assessment of the motivation behind the initial student code. All four of those paralleled the "started out as code for a coin flip" explanation discussed above. The other 14 scripts did not acknowledge the origins of the code. Those scripts proceeded by either pointing out the initial code's flaws relative to the task to make progress (8) or replacing the initial code with entirely different code to use as a starting point (6). Below, we exemplify each of these three observed approaches.

Motivating the initial scratch code

As mentioned, only 4 of the 18 scripts included motivation for why students might have generated the initial Scratch© code in the scripting task, as illustrated in Group 5's script:

> *Teacher*: It looks like you have some elements in your program that you don't use. Did you make this code or is it remixed from someone else?
> *Student1*: We used a code that showed the probability for flipping a coin. I made some modifications so that it would work for dice, but we can't get it to repeat 10 times.
> *Teacher*: Can you explain to me what the code is doing?
> *Student1*: Well, it starts with deleting all the previous results, then it flips the die by picking a random number from 1 to 6. Then, if the roll is 1, it adds to column 1, and if it's something else, it adds to column 2.
> *Teacher*: Before you go further with the code, you need to think about the problem you are trying to solve. If you want to remix this code, you need to think about the differences between calculating the probability of a coin flip and the sum of a pair of dice.

The student-character's motivations for the initial code in Figure 1 are evident when he states that the code started as "code that showed the probability for flipping a coin," and that he "made some modifications" so it would work for dice. This simultaneously provides a plausible explanation for why students working on the sum dice task might have generated the code in Figure 1, and provides the teacher-character with a starting point for guiding their progress. In Group 5's script, the teacher-character directs the student-characters to think about the differences between the single coin flip scenario the initial code addressed and the rolling two dice scenario they are working on. Consistent with the tenets of cognitively guided instruction, the teacher-character is building on the imagined students' work, and helping to direct them to the kinds of questions one might ask when trying to self-direct their coding progress.

Notably, a prospective teacher recognizing the code in Figure 1 started as code for flipping a coin does not necessarily mean that their approach is consistent with cognitively guided instruction. For instance, consider Felicity's script below:

> *Teacher:* Ok that sounds like a great idea. Let's see how we can work through the problems you are having. What is the code you currently have doing?
> *Student1:* Well we kind of used a code we found for the flip of a coin and tried to change it around and that didn't seem to work.

Student3: Right now it seems like the dice is rolled 10 times and if the roll is a 1 it gets added to the list of ones and if the roll is between 2 and 6 a 1 gets added to the 2-6 list.
Student1: But we don't want to know if the roll was a 1 or not we want to know exactly what the roll was and do that for 2 dice.
Student2: Does that mean we need to start over and try something else?
Student3: I think we should. [S2 and S1 nod in agreement.]

In the above script, the scriptwriter, Felicity, notes that the initial given code started as code for flipping a coin, when Student 1 indicates that "we kind of used code we found for flipping a coin." So, she has a viable explanation for why the code in Figure 1 might have reasonably emerged from students' work on the two dice task. However, instead of building on this code, Felicity uses a different code that she, as the scriptwriter, generated as a starting point for guiding the student-characters' progress. Although this idea is presented as coming from the student-characters when Student 2 says, "Does that mean we have to start over and try something else?", it is important to keep in mind that this is the work of a scriptwriter, not a student. By presenting the abandonment of the initial code as a student-character's decision, Felicity is, in a sense, avoiding the teacher-character being responsible for not building on the initial code. Felicity is instead using her own code as a starting point. As we discuss later, this is a common move in scripts that do not build on the given code.

Leveraging the flaws in the initial scratch code

A majority of scripts did not provide an explanation for the origins of the initial code in Figure 1. However, 8 of these scripts found other ways to leverage the initial code without such an explanation. This was primarily achieved by analyzing the initial code's functionality in relation to the rolling of two dice task. For instance, Group 6's script included this excerpt:

Teacher: What is happening in this small part of your code?
Student1: That's where we roll the dice.
Teacher: How many dice are you rolling?
Student1: Well, we want to roll 2 dice…
Teacher: Okay, so when you roll 1 die, what are the possible outcomes?
Student2: We can get 1, 2, 3, 4, 5 or 6.
Teacher: Exactly, so in the code here how many dice are you rolling?
Student1: Oh, we're only rolling once! Okay, so how do we roll twice?
Teacher: You've already rolled once, so do that again and combine them however you want. You could also pick a single random number that represents a roll of two dice, but I'd suggest just getting the sum.

Notice that in this excerpt, the teacher-character uses the initial student code as a starting point for guiding student-characters toward code that successfully addresses the rolling two dice task. This is achieved by highlighting the differences between the two dice task and what the initial code does. The above excerpt was used to motivate adding a second variable to record dice rolls, as well as replacing the two lists "1" and "2-6" with eleven lists named 2-12, each for recording one of the eleven possible outcomes of rolling two dice. This, in our view, is similar to Group 5's script, which focused on the differences between flipping a coin and rolling two dice to motivate the same changes to the initial code. Notable in Group 6's work is that the variables

used to record dice rolls are named “flip1” and “flip2”. This suggests that the scriptwriters overlooked the potential inappropriateness of those variable names in relation to the dice task.

Group 6 and Felicity’s scripts present an interesting contrast with respect to how the scriptwriters versus how the teacher-characters appeared to work with student ideas. While Felicity’s teacher-character lets the student-characters take the lead on deciding to shift away from the initial code, as a scriptwriter, Felicity does not attend to building on the student ideas that are represented in the initial code (e.g., the thinking which may have led to the code in the first place). In contrast, Group 6’s teacher-character offers direct instruction in response to a student’s question of “how do we”, suggestive of a model of teacher-centered instruction. Yet, we found Group 6 was attentive to the student ideas represented in the initial code. We suggest that Group 6’s consideration of the thinking that could have resulted in the initial bugs informed their approach in directing the students toward advancing their code, which is consistent with cognitively guided instruction. This contrast helps highlight the tension between how a scriptwriter represents a teacher-student interaction and that scriptwriter's actual attention to the presented student’s thinking.

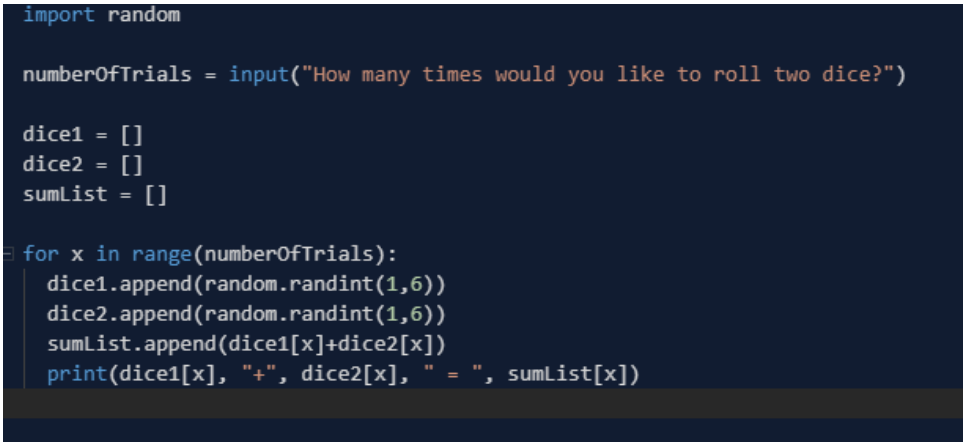
Replacing instead of building on the initial code

Some prospective teachers in our study avoided building on the initial code in Figure 1 entirely. This was commonly done by replacing the code in Figure 1 with scriptwriters own code and using that new code as a starting point for addressing the rolling two dice task. As alluded to in relation to Felicity’s script above, we view this as the scriptwriters avoiding engaging in the thinking and motivation behind the initial code in favor of interjecting their own thinking. Below is an excerpt from Sadiya, which does this in an exaggerated way:

Teacher: Yeah, this code has elements that are necessary for this type of project but it feels like you may have not thought all the way through the problem before starting to code. Good job though on starting early! I like the initiative.
Student1: May we use Python instead of Scratch for this assignment? I feel like for this assignment it would be easier, I remember a lot of how to use it from grade 11 functions and I really like making graphs on it.
Teacher: Sure you can use it. However, Python is not as friendly as Scratch and you might have to google questions if I am not available as well as be prepared to debug things.
Student1: Okay yeah I am prepared for that.
Teacher: Are you okay with Student1 wanting to use Python instead?
Student2: Yes, we already talked about it. I can use either language but I think Student1 might be right that Python could be more straightforward for this project.

In the above, the initial code in Figure 1 is abandoned in favor of a different starting code written in an entirely different programming language, Python. We interpret the teacher-character’s utterance, “it feels like you may have not thought all the way through the problem before starting to code,” as a way to avoid addressing student ideas. It may be that Sadiya was either unable to, or uninterested in, making sense of why a student might have generated the particular code in Figure 1. The Python code Sadiya introduced to side-step the initial code can be found below in Figure 3. As can be seen below, this code already receives user input for how many pairs of dice are to be rolled and records the sum of those pairs of dice rolls in a list called “sumList.” So, all of the “bugs” outlined in Figure 2 have already been addressed by the time the script begins in earnest. The remainder of Sadiya's script is focused on adding features and

graphical representations to the Python code (the "future directions" in Figure 2). We interpret this as Sadiya avoiding building on student thinking entirely. Instead, Sadiya and other similar scriptwriters completed the task themselves without leveraging the initial code or the thought process behind it at all. Then they represented their own personal coding journey in the form of a dialogue. This, in our view, is scriptwriter behavior inconsistent with the cognitively guided instruction perspective which underpins this paper.



```
import random

numberOfTrials = input("How many times would you like to roll two dice?")

dice1 = []
dice2 = []
sumList = []

for x in range(numberOfTrials):
    dice1.append(random.randint(1,6))
    dice2.append(random.randint(1,6))
    sumList.append(dice1[x]+dice2[x])
    print(dice1[x], "+", dice2[x], " = ", sumList[x])
```

Figure 3. Sadiya's Python code

Discussion and Future Directions

This paper provides an illustration of how scriptwriting can be used as a tool for researching prospective teachers' engagement with student thinking. We demonstrated that the scriptwriters engaged in leveraging student thinking did so by directing student-characters to the difference between where they are and their goal. This was done by either discussing the differences between a coin flip and rolling dice, as was seen in Group 5's script, or the differences between the initial code and the code the sum of dice tasks warranted, as was seen in Group 6's script. The prospective teachers who avoided engaging in this student thinking commonly did so by replacing the initial code in the assignment with scriptwriters' own code. This was seen in Sadiya and Felicity's scripts. Commonly, this decision was attributed to student-characters, to maintain the facade that student thinking was being leveraged.

We believe an additional contribution this paper makes is in helping illustrate the utility of parsing code when teaching. Increasingly, teachers are expected to integrate digital technologies and coding into their mathematics teaching. Yet most of these teachers have little if any experience coding and have not experienced coding in their own K-12 education. Tasks that engage teachers in parsing the thinking behind student mathematics have a long history in mathematics education. We suggest that coding tasks, with a similar flavor, which task teachers with parsing, not just what code does, but also the student thinking that might have generated it and how that thinking can be built upon, need to be further researched and studied in relation to teacher education and professional development. The scripting task we discussed here is one such task, and there are myriad others that can include or not include scripting, depending on the specific goals of teacher educators.

Acknowledgments

This research was supported by Social Sciences and Humanities Research Council Grants 435-2021-1089 and 430-2019-0671.

References

Bailey, D. & Borwein, J.M. (2011). Exploratory experimentation and computation. *Notices American Mathematical Society, 58*(10), 1410-1419.

Benton, L., Hoyles, C., Kalas, I. & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education, 3*(2), 115-138.

Bergman, A., Gallagher, K., & Zazkis, R. (2023). Prospective teachers' responses to students' dialogue on fractions: Attribute substitution and heuristic approaches. *Research in Mathematics Education, 25*(1), 85-104.

Campbell, M.P. & Baldinger, E.E. (2022). Using scripting tasks to reveal mathematics teacher candidates' resources for responding to student errors. *Journal of Mathematics Teacher Education, 25*, 207-531.

Carpenter, T.P. & Fennema, E. (1992). Cognitively guided instruction: Building on the knowledge of students and teachers. *International Journal of Educational Research, 17*(5), 457-470.

Carpenter, T.P., Fennema, E., & Franke, M. (1996). Cognitively guided instruction: A knowledge base for reform in primary mathematics instruction. *Elementary School Journal, 91*(1), 3-20.

Carpenter, T.P., Fennema, E, Peterson, P.L., Chiang, C., & Loef, M. (1989). Using knowledge of children's mathematics thinking in classroom teaching: An experimental study. *American Educational Research Journal, 26*(4), 499-531.

Chernoff, E. J., & Russell, G. L. (2012a). The fallacy of composition: Prospective mathematics teachers' use of logical fallacies. *Canadian Journal of Science, Mathematics and Technology Education, 12*(3), 259–271. doi:10.1080/14926156.2012.704128

Francis, K., & Davis, B. (2018). Coding Robots as a Source of Instantiations for Arithmetic. *Digital Experiences in Mathematics Education*.

Gadinidis, G., LeSage, A., Mamolo, A., & Namukasa, I. (2017). Re-designing K-12 teacher education: A focus on computational and mathematical thinking. *CATE Polygraph Series: Initial Teacher Education in Ontario*.

Gleasman, C & Kim, C (2020). Preservice Teachers' use of Block-based programming and computational thinking to teach elementary mathematics. *Digital Experiences in Mathematics Education 6*(3), 52-90.

Gleasman, C. & Kim, C (2018). Use of blocked-based coding in teaching conceptual mathematics. *Paper presented at Association for Educational Communication and Technology Conference*, Kansas City: MO.

Henderson, P. B., Cortina, T. J., & Wing, J. M. (2007, March). Computational thinking. In ACM SIGCSE *Bulletin (Vol. 39, No. 1, pp. 195-196)*. ACM.

Hershkowitz, R., Schwartz, B., & Dreyfus, T. (2001). Abstraction in context: Epistemic actions. *Journal for Research in Mathematics Education, 32*(2), 195 - 222.

Howson, A.G., & Kahane, J.P. (eds). (1986). *The influence of computers and informatics on mathematics and its teaching. ICMI Study Series (Vol. 1)*. Cambridge University Press.

Kercher, A., Bergman, A. & Zazkis, R. (2023). What is geometric about geometric sequences? Informal justifications for mathematical terminology. *Canadian Journal of Science, Mathematics and Technology Education, 23*, 48-65.

King, K., Hillel, J., & Artigue, M. (2001). Technology – A working group report. In D. Holton (ed.) *The Teaching and Learning of Mathematics at University Level: An ICMI Study, (pp. 349-356)*. Dordrecht: Kluwer Academic Publishers.

Lajoie, C. & Maheux, J.F. (2013). Richness and complexity of teaching division: Prospective elementary teachers' roleplaying on a division with remainder. *Proceedings of the Eighth Congress of European Research in Mathematics Education (CERME 8).*

Mamolo, A. (2021). Transfer of Mathematical Knowledge for Teaching as elicited through scripted role-play. In (Eds.) J. Lobato & C. Hohensee, Transfer of Learning: *Progressive Perspectives for Mathematics Education and Related Fields (pp.389-403)*. Springer

Mamolo, A. (2017). Eyes, ears, and expectations: Scripting as a multi-lens tool. In (Eds.) R. Zazkis and P. Herbst. *Scripting Approaches in Mathematics Education: Mathematical Dialogues in Research and Practice, (pp.229-248)*. Springer.

Mamolo, A., Tepylo, D., Ruttenberg-Rozen, R., & Rodney, S. (2022). Learning math through coding and learning coding through math: Two sides of the same coin. *Canadian Journal of Science, Mathematics, and Technology Education, 22*, 974–985.

Mamolo, A. & Pali, R. (2014). Factors influencing prospective teachers' recommendations to students: Horizons, hexagons, and heed. *Mathematical Thinking and Learning, 16*(1), 32-50.

National Council of Teachers of Mathematics (2000). *Principles and standards for school mathematics*. Reston, VA. NCTM.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Passey, D. (2016). Computer science (CS) in the compulsory education curriculum: implications for future research. *Education and Information Technologies*, 1-23.

Pei C., Weintrop, D., & Wilensky, U. (2018). Cultivating computational thinking practices and mathematical habits of mind in lattice land. *Mathematical Thinking and Learning, 20*(1), 75-89.

Resnick, M. & Rusk, N. (2020). Coding at a crossroads. *Communication of the ACM 63*(11), 120 – 127.

Shaughnessy, J. M. (1977). Misconceptions of probability: An experiment with a small-group, activity based, model building approach to introductory probability at the college level. *Educational Studies in Mathematics, 8*, 285–316.

Simon, M. A. (2006). Key developmental understandings in mathematics: a direction for investigating and establishing learning goals. *Mathematical Thinking and Learning, 8*(4), 359–371

Son, J.W. (2013). How preservice teachers interpret and respond to student errors: Ratio and proportion in similar rectangles. *Educational Studies in Mathematics, 84*, 49-70.

Stohl, H. (2005). Probability in teacher education and development. In G. Jones (Ed.), *Exploring probability in schools: Challenges for teaching and learning* (pp.345-366). Springer.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*, 3717-3725.

Zazkis, R., & Zazkis, D. (2014). Script writing in the mathematics classroom: Imaginary conversations on the structure of numbers. *Research in Mathematics Education, 16*(1), 54–70.