Boerkamp, S., & Sardella, J.. (2023). Re-Designing Coding-Based Mathematics Tasks into Scaffolded and Project Formats: Two Pre-Service Teachers' Perspectives. In Online Proceedings of the *Coding, Computational Modelling, and Equity in Mathematics Education Symposium*, St. Catharines (Canada), April 2023.

Re-Designing Coding-Based Mathematics Tasks into Scaffolded and Project Formats: Two Pre-Service Teachers' Perspectives

Samantha Boerkamp, Jessica Sardella, with Chantal Buteau
Brock University

Introduction

During the Coding, Computational Modelling, and Equity in Mathematics Education Symposium, various mathematics/programming activities were presented. These included those from the PD Day as well as the Working Groups, Keynotes, Poster Presentations, and Panel Discussions. The activities were designed with varying levels of coding and mathematics knowledge in mind, whereby some have also included additional foci, such as equity. The structure and guidance provided for each activity also differed depending on the purpose of its presentation– i.e. to present real, useable math coding activities, to prompt discussion around creation of math coding activities, to discuss implementation of coding in the math classroom, etc.

Within Working Group B: Coding and Computational Modeling in Secondary and University Mathematics Education, the leaders proposed a task for the members to work on throughout their sessions. This task involved selecting and working with a provided coding-based mathematics activity to modify it for a certain group of students. Following the symposium event, we (the two first authors) took on the task that was suggested. For the purpose of this paper, we selected two activities that incorporated coding in different ways, re-designed them in a) a step-wise guided activity ('scaffolded'), and b) a project-based activity, and reflected on our process. Below, we outline the context of our work, including our background, the approach we took, and a brief overview of our process including our selection of activities. We then present the activities as they were provided to us at the symposium and selected aspects of our engagement when working through them, followed by our scaffolded and project versions, and highlights of key changes we made and why. We conclude with some final thoughts comparing our process of task re-design for each activity.

Our Math + Coding Experience at Brock University

As Mathematics/Education majors in Brock University's Concurrent Education program with the intention of becoming math teachers, we have had experience with coding-based mathematics activities. These experiences mostly stem from our completion of a sequence of three programming/math courses, called *Mathematics Integrated with Computers and Applications* (MICA). Consisting of a total of 14 projects, these courses engage students (such as ourselves) in programming-based math investigation projects in which they design, program, and use interactive computer environments to investigate mathematics concepts, conjectures, theorems, or real-world situations (Buteau et al., 2015). The final MICA course is split into two sections: MICA III for math/stats majors, and MICA III* for future teachers such as ourselves. MICA III* provides students with explicit insight into the affordances of programming for math and supports preservice understanding of teaching in relation to the new coding curriculum in Ontario (Ontario Ministry of Education, 2020; 2021). This is done through guided reflections, comparisons between coding languages, and a final project in which students collaborate with in-service teachers to prepare and implement a coding math activity in the classroom.

It is with our experiences in the MICA courses and as pre-service teachers that we engaged in the two task re-design and reflected on our decision-making and thought processes when adapting the activities.

Our Task Design Approach: Two Versions

After the Symposium, we took the opportunity to look further into two of the presented activities and adapt them to create both a 'scaffolded' and 'project' version of each. The scaffolded version of these activities guides the student throughout their learning, giving explicit instructions and explanations so the student can achieve the desired result and gain the appropriate understanding of the concepts (Waite & Grover, 2020). While still structured, the project versions of these activities are more independent and allow the student to apply the concepts while demonstrating the skills and knowledge they have acquired with more opportunity for decision making and problem solving. Along with the activity guidelines, we also included a brief "Teacher's Guide" with each activity outlining necessary prior knowledge and helpful instructional tips. By creating two versions of each activity, we illustrate different ways that programming can be integrated into the mathematics classroom for different purposes or needs.

Selection of the Two Coding-Based Mathematics Activities and Overview of Our Process

The first activity we selected was a 'Cryptography' activity from Working Group B that Jessica (and Chantal) had begun to work with during the Working Group sessions. This activity focused on cryptography methods related to the Caesar cipher and coding with a key. It was provided with some code and base questions in a separate document, with the aim of modifying it to be a suitable activity for a certain audience, e.g. secondary/first-year university students. As presented, we felt it had both a focus on the mathematics and coding aspects, as students would need to understand the mathematics to adapt the code. For this activity, we initially both created our own versions of a scaffolded activity, but due to their similarities we decided to combine them into one version encompassing both of our ideas. We each created our own project versions of the Cryptography activities.

The second activity was presented by Callysto at the PD Day and focused on the probability of flipping a coin with an emphasis on the Law of Large Numbers. This activity was provided in a complete form, ready to implement in a classroom. While it included code, we felt the focus was more on the mathematics learning from *using* the code, rather than the code (or parts of the code) itself. For this activity, we again had both initially created our own versions of a scaffolded activity. This time we both took very different approaches, with Jessica focusing only on modifying the provided activity, and Sam focusing on adding an extension to the task. As we both felt our activities were incomplete, we found combining these two approaches help to create a more complete activity for students' learning. Unlike the 'Cryptography' activity, for this activity we created the project version after combining our scaffolded versions. Because of this, we both had similar ideas of what the project would look like, resulting in only one version.

'Cryptography' Activity

The 'Cryptography' activity provided by Working Group B included an activity guide (particularly useful for the instructor) outlining the programming language, suggested starting level for both math and programming, mathematical and computational concepts, and thinking/habits of mind, as well as brief instructions. The activity involves an exploration of basic coding and decoding techniques such as translation (known as the Caesar code) and coding

with a key which is based on arithmetic modulo 26. The full code was also provided for both cryptography methods.

- See the activity guide here: Cryptography Activity Guide
- See the provided code here: Cryptography Provided Code

When initially looking at this activity, we were both somewhat overwhelmed as it was a lot of code provided at once and including some syntax we did not know. To overcome this, we each decided to run through the code first to see what it outputted. After seeing the output, we both tried to work through each line of the code to try and understand how the output was formed. Essentially, we were breaking down the code into smaller chunks to see what each section did- something that we decided to incorporate into our 're-designed' activities (see 'Scaffolded 'Cryptography' Activity). We both also had to do some of our own research on the Caesar code and coding with a key to better understand what the code (and math involved) was aiming to do, as these were new concepts to us. Sam also tried altering some of the code to try and understand why some of the components were there, including altering the shift in the Caesar code and removing the '%26' to understand why it was used. Jessica researched more about the coding concepts, including what '+=' was used for, which helped in understanding how words could be encoded. Coding concepts we both had to research included the *ord* and *chr* functions, as neither of us had seen this used before. Overall, after doing some basic research, neither the math nor the coding was overly complicated. Because of this, we feel that this activity could be suitable for beginner coders (especially the scaffolded version) or more advanced coders, depending on the code provided. Based on the math and coding involved in this activity, we re-designed this activity with high-school aged students in mind (Grades 9-12).

```python
def letter_to_code(st):
    return ord(st)-65

def code_to_letter(n):
    return(chr(n+65))


# Caesar code and its inverse

def code1(plain):
    cypher=''
    for s in plain:
        if s!=' ':
            cypher+=code_to_letter((letter_to_code(s)+3)%26)
        else:
            cypher+=' '
    return cypher

def code1_inverse(cypher):
    plain=''
    for s in cypher:
        if s!=' ':
            plain+=code_to_letter((letter_to_code(s)-3)%26)
        else:
            plain+=' '
    return plain

plaintext='I SERIOUSLY NEED COFFEE LOTS OF IT'
print('given:  ', plaintext)
cyphertext=code1(plaintext)
print('encoded:', cyphertext)
plaintext=code1_inverse(cyphertext)
print('decoded:', plaintext)
```

```python
## coding with a key

def code2(plain,key):
    cypher=''
    key=(key*len(plain))[0:len(plain)]
    for i in range(len(plain)):
        if plain[i]!=' ':
            formula=(letter_to_code(plain[i])+letter_to_code(key[i]))%26
            cypher+=code_to_letter(formula)
        else:
            cypher+=' '
    return cypher

def code2_inverse(cypher,key):
    plain=''
    key=(key*len(cypher))[0:len(cypher)]
    for i in range(len(cypher)):
        if cypher[i]!=' ':
            inverseformula=(letter_to_code(cypher[i])-letter_to_code(key[i]))%26
            plain+=code_to_letter(inverseformula)
        else:
            plain+=' '
    return plain

plaintext='I SERIOUSLY NEED COFFEE LOTS OF IT'
print('plain text:', plaintext)

key='DUCK'
cyphertext=code2(plaintext,key)
print('cyphertext:',cyphertext)

plaintext=code2_inverse(cyphertext,key)
print('plain text:',plaintext)
```

Figure 1. Left: Code provided by the Working Group leaders for encrypting and decrypting for the "Caesar Cipher" portion of the 'Cryptography' activity. Right: Code provided by the Working Group leaders for the "Coding with a Key" portion of the 'Cryptography' activity.

Scaffolded 'Cryptography' Activity

- Student Copy: Cryptography Scaffolded- Student Copy
- Solution Guide: Cryptography Scaffolded- Solution Guide

Following conversation with other Working Group B members, we wanted to first add some more context behind the mathematical content being explored through this activity. Adding a brief explanation of what the Caesar cipher is and/or how a Cipher Wheel works aims to give students a general understanding of how their program should work and what their goal actually

is. We also added an introduction to modular arithmetic, as this is likely a new concept for students and is necessary in order to shift letters past Z and back to the start of the alphabet.

The main, and arguably most important, change that was made to this activity was breaking it down into smaller steps, and adding explanations. As mentioned, when going through the activity on our own, we were at times overwhelmed and struggled to understand what parts of the code meant. It took us significant time to go line by line and understand what was happening. For beginner coders, we felt like this would be too much all at once. We also added "TRY IT!" tasks to the worksheet to encourage incremental coding and working, in order to help encourage students to understand each aspect before they all come together.

In terms of what is given to the students, we decided to give students the functions to convert between letters and numbers. Python's *ord()* and *chr()* commands were new to us and would also likely be new to students. Providing this code does not take away from students' math learning as it is more of a technical aspect of the coding that they can be introduced to and learn for future use. This also meant introducing the Unicode as it is what the functions are based on and will help students to see why they will need to add or subtract 65 to the functions for easier use. Students can then alter and explore the codes and may find 'limitations' with them, e.g., that this can only be used for individual characters and not words or sentences.

We also decided to guide students through the process of encrypting, but leave the decrypting process to them to work through. Starting with encoding a letter, moving to a word, and then to a sentence helps familiarize the students with the concept of the cipher (shifting) and the syntax (e.g. '+='), while still requiring them to think computationally (through the use of loops, if/else statements, etc.). As some code is provided, students are guided to alter this code for certain goals, and then asked to create their own decrypting function, following a sort of "Use, Modify, Create" model (Waite & Grover, 2020).

For the coding with a key section, we first encouraged students to try this method on paper so they can see how it works before trying to code it. This is also something they use later on when verifying if their code works. Regarding the code, a 'fill-in-the-blank' encryption code is provided to students to encourage them to figure out how to apply the math to the code using their knowledge from the first part of the activity (Caesar cipher) as well as other methods (e.g. trial and error, etc.). Again, students are asked to create the decryption code on their own since this is very similar to that for encryption, before being asked to explore the encryption method furthe– this time with regards to the security of the key.

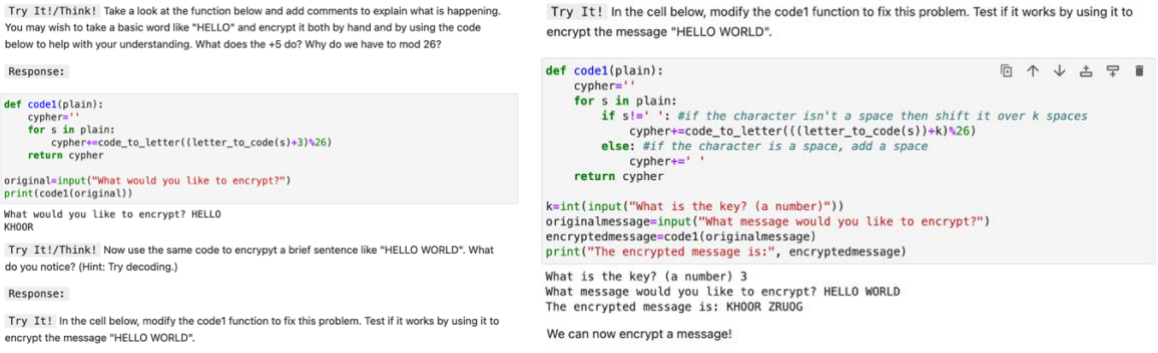


Figure 2. Left: A section of the scaffolded 'Cryptography' activity in which students are guided through programming the encryption process of the Caesar cipher. Students should discover that the provided code does not consider spaces when encrypting and that this will affect the decrypting process. Students are then asked to alter the code to fix this problem. Right: An example of a solution code to encrypt a message.

Jessica's 'Cryptography' Project

- Link here: Jessica's Cryptography Project

When designing my version of the Cryptography project, I relied on my scaffolded version in order to ensure that the main outcomes of the program would be similar. I also referenced some of the MICA course assignments in order to come to a sort of structure for the assignment- having students engage in the mathematical investigation process 'as a mathematician would', starting with research, engaging in the programming and debugging process, and finishing with a sort of discussion on the topic (Buteau et al., 2023). Although this is a project, and less structured in terms of the process, I wanted to ensure clear guidelines on what is expected; since they are being provided a topic, I think it's important they know the goal. By breaking the guidelines for the program into different points, my goal is to encourage students to work incrementally, and to help with the organization of the program.

I decided to provide students with the functions to convert between numbers and letters (for the same reason as in the scaffolded activity), and the code to encrypt a word since beginner coders likely wouldn't know how to move from one letter to the next (i.e. using +=). Since this is a project, I think that providing only this code is reasonable. Students may work in a similar process to that in the scaffolded activity, but they will need to think of these steps and encounter the various challenges (e.g. needing to account for spaces between words) on their own before resolving them.

Samantha's 'Cryptography' Project

- Link Here: Sam's Cryptography Project

When designing the project, I knew it was going to be much less scaffolded than the activities made before. I only wanted to introduce just enough of the mathematics so that the students would understand the concept on paper, but not too much where they wouldn't be able to investigate it on their own. This is why I only gave a brief definition of each encryption method.

I am also assuming that the students would have already been introduced to cryptography and its purpose, as a project is usually considered for an end of unit activity or summary so the students would not need such an elaborate introduction to the concept in order to succeed. While an activity is made for them to build their learning through guided steps, the project is for them to explore their understanding on their own terms without step-by-step guidance telling them what to do. This allows the students to express their understanding in a way that makes sense to them, without a unique right solution. Keeping to the goal of the activity, I asked them to create their own codes representing the Caesar model and encryption with a key model but with very limited code provided to them. I justified that the students would already have a strong understanding of loops and conditionals that they would not need anything else other than what was given to them. After creating the code, I wanted them to be able to explore it, which is why I asked them to try and break another classmate's cipher. This allows them to use their code with a purpose while also using their math knowledge on the subject to try and decipher another student's message. It is an application of how ciphers are used in the real world. The next part of the project makes them reflect on these two models of cryptography by asking how they could make these stronger and less breakable in their own new form of ciphering text.

I found creating the project more difficult than the scaffolded activity because it was something I was not familiar with doing. I was unsure at what level of student I should be creating this activity for or how much work they should be doing. My goal was to try and

incorporate all of the activity's components into the project, just without the scaffolding that was in place for the activity. This meant asking them to do things that incorporated all of the ideas they needed to obtain by the end of it. It also meant giving them more opportunity to explore on their own and creating questions that would encourage them to do so. A key moment was when I decided to include the partner work in this project, because it gave the opportunity for exploration of these models without direct guidance from the teacher of how to do so. The basis of cryptography is sending secret messages between two people, so it made sense to include a partner for them to test their knowledge and capabilities with. After this, it was easier for me to understand what I wanted the students to achieve from this project, and how a project differs from an activity.

Flipping Coins' Activity

The 'Flipping lots of Coins' activity provided by Callysto included a Python notebook with pre-made code and instructions. The full code is included as a part of the activity; i.e. students are not asked to code, rather they are asked to *run* the program to explore the mathematical concepts involved. The activity involved probability, with students exploring the "Law of Large Numbers" through the simulation of flipping a coin. Programming concepts utilized in the code differed from the 'Cryptography' activity, utilizing the random function, graphing, and storing data in tables.

- See the activity provided here: Flipping Coins Provided Code

We both looked at this activity after looking at the 'Cryptography' activity and felt that it was quite different in terms of structure, format, and concepts. The code itself was relatively simple, and since students are not actually doing any of the coding, we felt that the activity was more math-based with potential for the coding aspect to be overlooked. While we still worked through the code to try and understand it, we did not do the same level of research as we did with the 'Cryptography' activity. This is because it is not necessary to understand how the code is working to understand the math, since most of the code is related to graphing. While students may not gain programming skills from this, they are able to benefit from some of the affordances of programming such as automation (i.e. through simulation of a large number of coin flips, e.g. N=100,000). Compared to the 'Cryptography' activity, we felt that this activity was already quite scaffolded and broken down for students, as the code was already divided into individual cells that could be run separately from each other together with guidelines in separate text-cells. For a student in Grade 7-10 (the age range we had in mind when re-designing the activity), the mathematical concept itself is likely not too difficult to understand and could be considered beginner friendly.

Figure 3. Left: A section of Callysto's 'Flipping Coins' activity in which students run the cell to display the results of a given number of coin flips in a table. Right: Students can then visualize the results of the coin flips in a histogram by using the code provided.

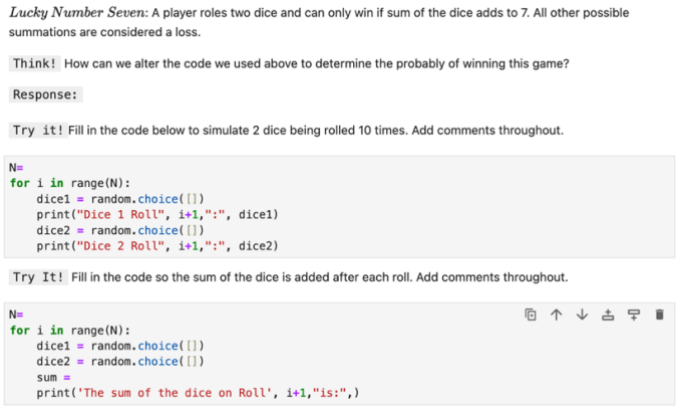
Scaffolded 'Flipping Coins' Activity

- Student Copy: Flipping Coins Scaffolded- Student Copy
- Solution Guide: Flipping Coins Scaffolded- Solution Guide

For the Flipping Coins task, our main goal was to engage students more, including in the programming aspect, rather than just have them run through the code. To do so, we added various questions throughout for students to think about as they go through the activity (e.g. making predictions, what they notice, forming conclusions about changing number of flips, etc.). We also redesigned the actual coding involved in the task to: a) allow students to do some of the coding on their own; b) utilize coding concepts or possible solutions that we feel encourage more computational thinking (e.g. utilizing loops and conditional statements).

For the 'biased coin' part of the task, we decided to provide students with an alternative method for coding an un-biased coin. Since this method, which may be used for a biased coin as well, is slightly more advanced than that they would create on their own we felt it would be good to provide. Students will still need to understand why this code works and modify it to fit the given situation, again using the "Use, Modify, Create" model (Waite & Grover, 2020).

We also added on to the activity, using Callysto's suggestion that this activity lead into the Monte Carlo approach. We felt that the Flipping Coins code provided enough to guide students, looking for challenge, to create their own representation of the Monte Carlo method involving rolling a dice and explore probabilities that they may not already know (i.e. probability of rolling different sums of dice). We also added an extension activity, in which students can take what they learned earlier to simulate a different dice-rolling game. Each of these parts becomes less scaffolded and allows the students to work more independently to develop their coding skills for mathematics learning.


Lucky Number Seven: A player roles two dice and can only win if sum of the dice adds to 7. All other possible summations are considered a loss.

`Think!` How can we alter the code we used above to determine the probably of winning this game?

`Response:`

`Try it!` Fill in the code below to simulate 2 dice being rolled 10 times. Add comments throughout.

```
N=
for i in range(N):
    dice1 = random.choice([])
    print("Dice 1 Roll", i+1,":", dice1)
    dice2 = random.choice([])
    print("Dice 2 Roll", i+1,":", dice2)
```

`Try It!` Fill in the code so the sum of the dice is added after each roll. Add comments throughout.

```
N=
for i in range(N):
    dice1 = random.choice([])
    dice2 = random.choice([])
    sum =
    print('The sum of the dice on Roll', i+1,"is:",)
```

Figure 4. A portion of our scaffolded "Flipping Coins" activity, in which we ask students to use what they have learned to create a program that determines the probability of winning the "Lucky Number Seven" game. A fill-in-the-blank code is provided.

Our 'Flipping Coins' Project

- Link here: Flipping Coins Project

When designing the 'Flipping Coins' project, we again used our scaffolded version as a guide for what we wanted to include. Again, we tried to keep the concepts and content very similar, but left the process of the investigation and coding more to the student. Part 1 (Task A and B) has the students creating their coin flip simulations and discovering the Law of Large Numbers and investigating experimental vs theoretical probabilities. Similar to the scaffolded version, students form predictions, create coin flip simulators, and run them a varying number of

times to compare theoretical and empirical probabilities. The main difference here is that there is very minimal code provided to students. As this is a project, we assume that they have the coding knowledge necessary to do this on their own, and want to encourage them to think computationally to figure out what that code may look like. Part 2 has students wrap up their findings and tie it all together. This includes students forming a conclusion, and creating an adapted version of their program to further validate their learning. This additional simulation also allows students to see a different tool for a probability simulation, showing them different potential outcomes while giving them the freedom to choose their own tool to model. We felt that adding this additional component was necessary to push students forward in their individual thinking since Part 1 was still broken down for them in terms of what they needed to do.

Final Thoughts

The provided activities were both different in terms of the structure, coding concepts, and math. Thus, we found our experiences adapting them progressed differently as well. We found the Cryptography activity easier to adapt than the Flipping Coins activity. This could be because the Flipping Coins activity provided to us was already quite scaffolded with a limited total number of code lines, making us feel limited in our creativity, whereas the Cryptography activity had the code provided (including multiple lines, and control commands) but didn't provide the same kind of guidance for exploration allowing for more creative freedom. The Cryptography activity also included an activity guide (useful for the instructor) with explanations of the concepts that were being covered, making the goal for the activity clearer and providing background on what students should be gaining. The Flipping Coins activity felt limited in terms of where to go with it, although we recognize that there are applications that could be explored (such as those of the Monte Carlo method) but we were not familiar enough with them at the time of creating the activities to explore them further. We also found it useful to start with the scaffolded versions before moving on to the project design, as we had already broken down the concepts and knew what would need to be included. This helped in deciding how much guidance to provide within the project instructions.

Overall, our experience creating these scaffolded and project activities allowed us to think more deeply about different aspects involved when integrating programming with mathematics. We used our own experiences from learning programming in a similar context and as future teachers to think about what students may need to get started, as well as what may provide more of a challenge. Our experiences allowed us to get a feel for what the appropriate amount of guidance should be and helped in our decision making. We hope that we can implement these activities in classrooms in the near future and get both indirect and direct feedback on the design of our activities. From here, we could further adapt the activities to better meet the needs and skills of our students.

References

Buteau, C., Muller, E., & Ralph, B. (2015). Integration of programming in the undergraduate math program at Brock University. Retrieved at http://researchideas.ca/coding/docs/ButeauMullerRalph-Coding+MathProceedings-FINAL.pdf

Buteau, C., Broley, L., Dreise, K., & Muller, E. (2023). Students using programming for pure and applied mathematics investigations. *Épijournal de Didactique et Epistémologie des Mathématiques pour l'Enseignement Supérieur.* https://doi.org/10.46298/epidemes-9190

Ontario Ministry of Education (2020). Elementary curriculum: Mathematics. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics

Ontario Ministry of Education (2021). Mathematics, Grade 9. https://tinyurl.com/3djym45h

Waite, J., & Grover, S. (2020). Worked examples & other scaffolding strategies. In S. Grover (Eds.), Computer Science in K-12: An A-to-Z handbook on teaching programming (240-249). California, CA: Edfinity.