MATHEMATICAL LITERACY DEMONSTRATED IN PROGRAMMING-BASED MATHEMATICAL PROBLEM SOLVING: THE CASE OF COMPOUND INTEREST

Athena Chan, Oi-Lam Ng

The Chinese University of Hong Kong; athenachan59@yahoo.com.hk, oilamn@cuhk.edu.hk

In this paper, we explore students' mathematical literacy in a problem-based digital making course. We report on a task-based interview in which pairs of students were using a block-based visual program "Scratch" to solve mathematical problems related to compound interest. Two pairs of students (one grade 9 group and one grade 11 group) were invited to join the course. Their code files, screen captures of computer work were analysed in terms of their developed mathematical reasoning from the perspective of mathematical literacy.

Keywords: Scratch, programming, problem solving, mathematical literacy

INTRODUCTION

Mathematical literacy in the 21st century includes some aspects of computational thinking. It includes the use of mathematical tools such as a variety of digital and physical equipment, software, and calculation devices. Computational thinking (CT) tools offer students a context in which they can reify abstract constructs by exploring and engaging with math concepts in a dynamic way (OECD, 2018). CT supports the development of mathematical thinking (Ng and Cui, 2021). The combination of mathematical and computational thinking allows students to view realistically how mathematics is practised and used in real world and prepares them for pursuing careers in related fields (OECD, 2018). The Computer Science Teachers Association (CSTA) stated that computational thinking enables students to conceptualize, analyse and solve complex problems better by selecting and applying suitable tools and strategies in virtual and real world. (David et al., 2016). The nature of computational thinking is defining and elaborating mathematical knowledge which can be expressed by programming, allowing students to model mathematical concepts and relationships dynamically (OECD, 2018).

Researchers argued that writing computer program provides a big canvas on which students can sketch half-understood ideas and construct semi-concrete image of mathematical structures that students are building intellectually on the screen (Laura et al., 2017). The connection between abstract mathematical concepts and concrete experiences allows students to use mathematics in a meaningful way. For example, programming at school has the potential of developing higher order of mathematical thinking in relation to numbers linked to mathematical abstraction, including algebraic thinking and problem solving abilities (Ng & Cui, 2021; Ng et al., 2021)

In this paper, students participated in a programming-based mathematical activity of using Scratch to solve mathematical problem. Scratch is a free online community and programming language which students can create their own stories, animations and games by coding. Scratch has been proved of providing opportunities for learning important computational and mathematical concepts and offering space for systematic reasoning, creative thinking and collaborative work (Laura et al., 2017). It is easily readable, composite and browsable alongside its visual, interactivity and dynamic outcomes. Our research questions are as follows: (1) How do secondary students develop and demonstrate mathematical literacy during programming-based mathematical problem-solving? (2)

What opportunities and hinderance for learning does the programming context have on students' mathematical problem-solving?

RESEARCH BACKGROUND

Constructivist learning with programming-based mathematical problem solving

During the past two decades, the introduction of digital technology in schools has given rise to new ways of doing and representing mathematics, with dynamic geometry being a prime example of technology affording an environment in which mathematics is represented and learned in transformative ways. More recently, there has been increased focus on the impact of making, pointing towards new opportunities for engaging learners in constructionist practices with digital technology. Scholars have generally defined Making as the hands-on production of artefacts that are technologically-enhanced (Ng & Cui, 2021). Stemming from the constructivist view of learning as "building knowledge structures", constructionist learning underpins the context whereby the learner is consciously constructing a public entity (Papert & Harel, 1990). Of interests to this study is a subset of constructivist learning with programming-based mathematical problem solving which affords the active creation of both programming-based artefacts during the course of mathematical problem solving.

In recent years, research into integration of programming and mathematics in K-12 education has provided evidence that programming-based mathematical activities supports students' development of critical thinking, creativity and communication and collaboration (Weng et al., 2022). It also develops students' computational thinking and problem solving competences (Ng, Liu & Cui, 2021). According to Feurzeig, Papert and Lawler (2011), programming fosters an experimental approach towards problem solving. It provides students opportunity to discuss mathematics through natural framework, standard vocabulary and personal experiences. Moreover, solving mathematical problem by computational thinking tools can encourage students to practice prediction, reflection and debugging skills (OECD, 2018).

Mathematical literacy

In this paper, we define mathematical literacy as not just doing arithmetic calculation nor solving algebra problems, but as according to PISA 2022 Mathematics Framework, i.e. students' capacity to reason mathematically, and to employ, formulate and interpret mathematics to solve real life problems. It includes concepts, facts, procedures, and tools to describe, explain and predict phenomena. Mathematical literacy can help students understand the role of mathematics in the world and make judgements and decisions needed by engaged, constructive and reflective 21st Century citizens. It comprises two aspects: mathematical reasoning and problem solving. Mathematical reasoning involves situation evaluation, strategies selection, logical conclusion drawn, solution development and description, and solution application recognition. For the purpose of studying students' mathematical reasoning and problem solving competence in the context of programming-based mathematical problem solving, we illustrate the following shortened list of practices addressing how mathematical reasoning can be demonstrated in PISA 2021 Mathematics Framework (OECD, 2018):

1. Drawing simple conclusion
2. Selecting appropriate justification
3. Explaining the reason that a mathematical conclusion or result does or does not make sense with the problem context

4. Utilising and understanding rules, definition and formal systems as well as employing computational thinking and algorithms
5. Explaining, defending and providing a justification for the devised and identified representation of a real-world situation
6. Explaining or defending and providing a justification for the procedures or simulation and processes used to determine a mathematical solution or result
7. Identifying and critiquing the model limits used for solving a problem
8. Reflecting on mathematical arguments, justifying and explaining the mathematical result
9. Interpreting mathematical result back into real life situation in order to explaining the meaning of results
10. Explaining the relationships between context-specific language of a problem and the formal and symbolic language needed to represent it mathematically.
11. Reflecting on mathematical arguments, explaining and justifying mathematical result
12. Reflecting on mathematical solutions and creating arguments and explanations that support, qualify and refute a mathematical solution to a contextualised problem
13. Analysing similarities and differences between a computational model and the mathematical problem that is modelling
14. Explaining how a simple algorithm works, detecting and correcting errors in programs and algorithms

METHODOLOGY

The task-based interview

In this report, four students from Grade 9 and Grade 11, divided into two pairs respectively, attended a task-based interview in which they were tasked with making computer programs in Scratch while solving a mathematical problem related to simple and compound interest. The choice of two different grade levels was to examine the chosen research questions with participants having different levels of prior knowledge in mathematics; unlike the Grade 11 students, the Grade 9 students have not yet learned the formula of determining compound interest, $A=P(1+r)^n$ (A: amount, P: principle, r: interest rate, n: number of deposit period). The interview (first author) posed the students with two problems, one on simple interests and another on compound interests, with several sub-problems in three-hour session. As the students had minimal experience with Scratch prior to the study, they were initially guided to learn some basic functions, such as creating blocks and running the program in Scratch, and were then scaffolded to solve the very first problem posed. After that, they were given ample time to explore the solutions to the subsequent problems with their partners, and explain their thinking processes to the interviewer, who would provide adequate guidance when needed. Students' code files, screen captures of computer work and dialogue between teacher and students were analysed in terms of their developed mathematical literacy. An interview will also be carried out after the lessons.

The design of task

The students were required to compare two different saving plans using Scratch. To make the problems more relevant to real-life, we design scenarios in which different banks use different method to accumulate interests (simple interest and compound interest) and different interest rate to attract investors. To compare interest return from banks, students need to design an interest calculation system using Scratch. The design of calculation system allows students to learn the setting of variables and conditional blocks by understanding the relationship between amount, principal, interest rate per period and number of deposit periods. Besides understanding the meaning of different variables, students also need to understand the quantitative relationship of different deposit methods.

Alejandra, Noemi and Susanna (2019) divided digital making into four stages: introductory stage, design stage, production stage and final evaluation and diffusion stage. During the introductory stage, it is important to introduce the key programming concepts and not focus on learning the programming environment rather than learning through it (Laura et al., 2017). Furthermore, they suggest that the design of Scratch researcherial should be relevant and consistent in order to ensure that intervention is aligned with mathematical curriculum. Researchers should advocate a set of learning guides, best practices and curriculum models to help students encounter the richness of Scratch ideas (Laura et al., 2016). In the design stage, whole-class discussion should be led by teachers through inclusion of reflective questioning. Harel and Papert (1990) mentioned that generating verbal explanations can help clarify ideas. Moreover, the balance between direction (researcherial) and discovery (students' own exploration) is important. Finally, the design of tasks should provide students opportunities to investigate ideas by trying things and debugging errors (Laura et al., 2017). In line with these pedagogical considerations, we design the lesson flow, lesson objectives and content as shown in Table 1.

Prior knowledge	The four basic arithmetic operations Linear equations Inequality	
Learning objectives	1. Solve problem about numbers in Scratch 2. Use iterations and conditionals to model different deposit situations 3. Work with several variables at the same time	
Mathematics and Programming Outcomes	1. Mathematics: quantitative relationship, simple and compound interest 2. 2. Programming skills: variables, conditionals, operator, sequences	
Four stages of digital making	*Introductory stage*	Students learn the setting of variables and conditional blocks by designing a simple interest calculation system. The content is shown below:
	Design stage	Students are divided into two groups. They need to set up an interest calculation system to calculate compound interest offered by two banks and do the comparison.
	Production stage	Students demonstrate their model to other students and invite others to play the Scratch.
	Final evaluation and diffusion stage	Students complete reflection on the learning process by interview

Table 1. The design of the task

Analytical Method

We developed an analytical framework and used deductive coding to analyze students' discourse and programming actions during the task-based interview. This allowed us to observe how students develop their mathematical literacy when solving the task. Specifically, our data analysis took on the following steps. First, we synthesized various practices addressing mathematical reasoning as listed in PISA 2021 Mathematics Framework. Second, we used open coding to identify how students demonstrate these practices mathematical reasoning according to the listed practices. These steps allow us to look for evidence in how mathematical literacy and computational thinking can be meaningfully synergized into mathematics learning and problem solving situations in a programming context. Finally, we focus our attention on the learning opportunities and hinderance that programming had on students' problem solving by delving into the way the students interacted with the Scratch.

RESULTS

We detail students' practices of mathematical reasoning in the following three episodes. We use italics to highlight these practices analyzed.

Episode 1. When setting the simple interest calculation system, students first drew a timeline for creating the function (Figure 1).

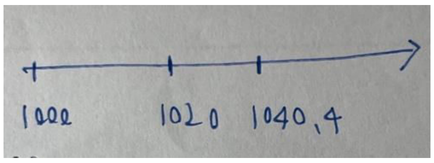


Figure 1. Students' drawing during Episode 1

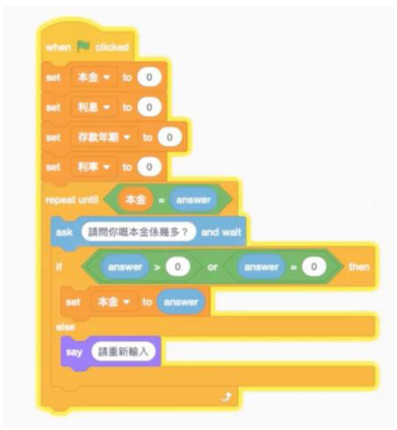
Researcher:	Let's look at the timeline, the interests in each year are different which does not match the question.
Student A:	Maybe we should not add the index.
Researcher:	Could you write the equation for calculating the simple interest?
Student A and B:	……
Researcher:	Let's look at the timeline, what is the interest after 1 year?
Student B:	\$20
Researcher:	After 2 years?
Student B:	\$20 + 20
Researcher:	After 3 years.
Student B:	\$40 + 20
Researcher:	After n years.
Student B:	n + 20
Student A:	No.
Student B:	After 5 years, the interest is 80 + 20 = 100. 5 times 20 is also equal to \$100.
Researcher:	What is the interest after n years?
Student B:	20n.

From Episode 1, we observe that the students were able to explain their calculation by drawing a timeline and deduce interest equation. In doing so, they demonstrate *explaining, defending and providing a justification for the devised and identified representation of a real-world situation.* At first, the students applied the compound interest equation to solve the current problem, which is related to simple interests. They were able to identify variables but found it difficult to derive an algebraic equation for simple interest. Later, upon being guided by the researcher, they observed that the interests increases by \$20 every year and successfully set up the algebraic expression, 20n, for the amount of interests earned after n years. This desmonstrates that the students reflected on *mathematical arguments, explained and justifies mathematical result.*

Episode 2. Students needed to think about inequality prior to using "if…then…else" block for setting requirement of data entered.

Researcher:	What if the data is negative? How to present positive number?
Student C:	The principle is larger than 0.
Student D:	The principle is larger than interest.
Researcher:	Why?
Student D:	No.
Researcher:	You may look at the data here. What do you notice about the data?
Student C:	Not negative.
Researcher:	Can the data be zero?
Student C:	Zero is not a positive number.
Researcher:	Zero is not positive but can the principle be zero?
Student D:	Yes. Can have no interest.
Researcher:	What is the meaning when the principle is zero?
Student C and D:	No money.

In Episode 2, the students explained *how a simple algorithm work* in the sense of presenting an inequality for the variables, which demonstrates their mathematical reasoning. Through solving the problem in a programming context, the need for prompting for input for different variables is observed. In particular, they noticed that the principle, interest rate and number of deposit periods must be non-negative number. As a form of mathematical reasoning, they *explained the relationship between context-specific language of a problem* "principle should be larger or equal than zero" *and the formal mathematical language* (i.e. not negative). The students also *interpreted the mathematical result* of "zero principle" *into real life situation*, stating that no money would be deposited in this situation. When setting up the "if…else…then" and "repeat until" situations, the students *checked their codes with their calculation results*. They *detected and corrected errors in programs* and successfully obtained the desirable result at the end (Figure 2).



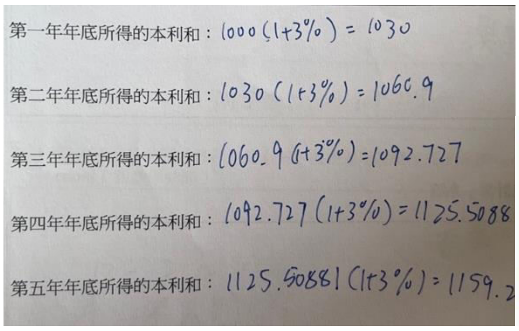
```
when 🏴 clicked
set 本金 ▾ to 0
set 利息 ▾ to 0
set 存款年期 ▾ to 0
set 利率 ▾ to 0
repeat until 本金 = answer
    ask 請問你嘅本金係幾多？ and wait
    if answer > 0 or answer = 0 then
        set 本金 ▾ to answer
    else
        say 請重新輸入
```

Figure 2. Students' codes of setting situations

Episode 3. Before setting the compound interest calculation system in Scratch, students calculated the amount of deposit in the first five years (Figure 3a).

Researcher:	If we would like to design a calculation system, what should we enter?
Students D:	Principle, deposit period, interest rate
Researcher:	What is the equation?
Student D:	All variables should be multiplied together.
Studenst C:	Principle x Interest rate x Deposit Period. "Interest rate x Deposit period" keeps on repeating. Let's add a bracket to "Interest rate x Deposit period". In 1st year, the index should be 1. In 2nd year, the index should be 2 and so on.

	So the index is 5. The equation is Principle ×(interest rate × deposit period)^5.
Student D:	But when we substitute the numbers (Figure 3), the result is wrong.
Studenst C:	We have the index be the "deposit period" so we should not add deposit period inside the bracket. It should be Principle ×(1 + interest rate) $^{\text{deposit period}}$.
Researcher:	The equation is correct but it is difficult to set up index in Scratch.
Student D:	So we may set to multiply "1+interest rate" by 5 times.
Researcher:	Yes you are right.


第一年年底所得的本利和：1000(1+3%) = 1030

第二年年底所得的本利和：1030(1+3%) = 1060.9

第三年年底所得的本利和：1060.9(1+3%) = 1092.727

第四年年底所得的本利和：1092.727(1+3%) = 1125.5088

第五年年底所得的本利和：1125.50881(1+3%) = 1159.2

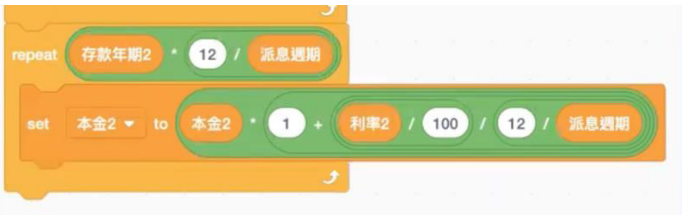

```
repeat (存款年期2 * (12 / 派息週期))
    set 本金2 ▾ to (本金2 * (1 + (利率2 / 100 / (12 / 派息週期))))
```

Figure 3(a) Students' calculation of compound interest. (b) Students' codes of calcuting amount under compound interest calculation

In Episodes 3, the students were able to *explain how a simple algorithm works,* and to *detect and correct errors in algorithms* by setting up codes that perform the calculation of compound interests. They could identify the index of deposit period as repeatedly multiplying (1+interest rate). Also, they *reflected on mathematical arguments, justified, and explained the mathematical result* of the compound interest rate equation.

DISCUSSION

This investigation examined how two pairs of secondary school students engaged in mathematical reasoning related to programming in a programming-based mathematical problem-solving environment. Results suggest that programming in Scratch promoted various practices of mathematical reasoning. They analyzed the situation, and used symbols and representations to make sense of their programming actions, which is contextualized in a real-life scenario. It can be said that Scratch can allow students to make and test conjectures, thereby facilitating mathematical reasoning. When setting the conditions for data entered, students encoded by changing the representation of blocks in a script. They were able to detect and correct errors in program. Also, concept of variables is important in programming. When solving the task, students can performed their CT by setting up variables to calculate the amount of interest. During the process of deducing an algebraic expressions for interest earned, the students explained their thinking and justified the mathematical result. They also gave an explanation that zero principle means no money deposited. Finally, they detected and correct errors of equation by comparing with previous calculation. When setting the formula for calculating the amount at the end of deposit period, they used graphical method like timeline to explain how the formula works and deduced equation.

The programming context provided opportunities and hinderance for students' learning on mathematical problem-solving. In conventional paper-and-pencil lessons, teachers would commonly introduce formula for computing compound and simple interests for the students. Using programming to set up equation can help students better understand the equation since they were required to think about the relationship between different variables. It is important to understand the problem solving process or strategy used rather than just focusing on solution (Laura et al., 2017). However, some students explained that they had difficulties in programming since they needed to consider many conditions.

CONCLUSION

This study supported that students experienced mathematical reasoning in programming-based mathematical problem-solving. Though some students faced in-moment challenges when solving mathematical problem in programming, they learned to see the problem from the perspective of CT and commented that programming could help them understand mathematical context. As stated in PISA 2021 Mathematics Framework, mathematical literacy includes mathematical reasoning and problem solving. Further study can be carried out to examine students' problem solving in programming-based mathematical problem-solving. For example, theeffectiveness of programming in helping them understand mathematical concepts are worthy of future studies.

REFERENCES

Alejandra, B., Noemi, S. & Susanna, T. (2019). Digital Making in Educational Projects. *C E P S Journal, 9*(3). https://doi.org/10.26529/cepsj.629

Constanta, O. (2022). Programming, mathematical reasoning and sense-making. *International Journal of Mathematical Education in Science and Technology,* 53(8), 2046-2064. https://doi.org/10.1080/0020739X.2020.1858199

Cui, Z., & Ng, O. (2021). The Interplay Between Mathematical and Computational Thinking in Primary School Students' Mathematical Problem-Solving Within a Programming Environment. *Journal of Educational Computing Research, 59*(5), 988-1012. https://doi.org/10.1177/0735633120979930

David, W., Elham, B., Michael, H., Kai, O., Kemi, J., Laura, T. & Uri, W. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *J Sci Educ Technol, 25,* 127-147. https://doi.org/10.1007/s10956-015-9581-5.

Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments, 19*(5), 487-501.

Laura, B., Ivan, K., Celia, H. & Richard, N. (2017). Bridging Primary Programming and Mathematics: Some Findings of Design Research in England. *Digital Experiences in Mathematics Education.* https://doi.org/10.1007/s40751-017-0028-x

Papert S., & Harel, I. (1990). Situating constructionism. In I. Harel (Ed.), *Constructionist learning*. Cambridge, MA: MIT Media Laboratory.

Ng, O., & Cui, Z. (2020). Examining primary students' mathematical problem-solving in a programming context: towards computationally enhanced mathematics education. *ZDM Mathematics Education. Advanced Online Publication,* http://doi.org/10.1007/s11858-020-01200-7

Ng, O., Liu, M., & Cui, Z. (2021). Students' in-moment challenges and developing maker perspectives during problem-based digital making. *Journal of Research on Technology in Education.* http://doi.org/10.1080/15391523.2021.1967817

OECD (2018). PISA 2021 Mathematics Framework. https://www.oecd.org/pisa/sitedocument/PISA-2021-mathematics-framework.pdf

Weng, X., Cui, Z., Ng, O., Morris, J. & Thomas, C. (2022). Characterizing Students' 4C Skills Development During Problem-based Digital Making. *Journal of Science Education and Technology.* https://doi.org/10.1007/s10956-022-09961-4