

Computational Numeracy (CN) for Under-Prepared, Novice Programming Students

Carla Coetzee, Machdel Matthee

▶ **To cite this version:**

Carla Coetzee, Machdel Matthee. Computational Numeracy (CN) for Under-Prepared, Novice Programming Students. 20th Conference on e-Business, e-Services and e-Society (I3E), Sep 2021, Galway, Ireland. pp.744-756, 10.1007/978-3-030-85447-8_62 . hal-03648152

HAL Id: hal-03648152

https://inria.hal.science/hal-03648152v1

Submitted on 21 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License



This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

Computational Numeracy (CN) for under-prepared, novice programming students

Carla Coetzee[0000-0002-0624-7384] and Machdel Matthee[0000-0002-6973-1798]

Department of Informatics, University of Pretoria, Pretoria, South Africa
CoetzeeC@tut.ac.za and machdel.matthee@up.ac.za

Abstract. Numeracy has become a critically important skill in data rich environments. A large number of first-year ICT students entering HEIs in South Africa lack computational thinking and problem-solving skills and consequently they are not prepared for programming. Many of these students are not proficient enough in numeracy to solve programming problems that require knowledge and understanding of numeracy concepts. A new concept, Computational Numeracy (CN) for under-prepared, novice programming students, is presented in this article. The purpose of this conceptual article is to show how the components of Computational Numeracy were developed by exploring and reviewing its basic building blocks: computational thinking and numeracy. A critical synthesis of published research related to numeracy and computational thinking related to programming skills to define Computational Numeracy for under-prepared, novice programming students, is presented. Six components of computational thinking were selected and mapped to numeracy in a typical programming problem to demonstrate the links between computational thinking and numeracy and how it can be seen as CN. Future research includes the development of a framework to guide lecturers at HEIs on how to teach CN to under-prepared, novice programming students.

Keywords: Numeracy, Computational Thinking, Programming, Computational Numeracy, Under-prepared ICT students

1 Introduction

To interpret numbers, graphs and other statistical data, numeracy skills are becoming increasingly important in the 21st century economy. In adults' daily lives, essential tasks such as shopping for groceries, using recipes, balancing budgets, and doing home improvements, the importance of numeracy skills cannot be underestimated. One of the more frequently used, concise definitions of numeracy states that numeracy is the *"ability to access, use, interpret and communicate quantitative information and ideas, in order to engage in and manage the quantitative demands of a range of situations in adult life"* [1:48]. Therefore, it can be concluded that adults are dependent on basic numeracy.

In Steen's seminal publication about quantitative literacy (numeracy), they distinguish between numeracy and mathematics and argued that *"Numeracy is not the same*

as mathematics, nor is it an alternative to mathematics. Mathematics is abstract and Platonic, offering absolute truths about relations among ideal objects. Numeracy is concrete and contextual, offering contingent solutions to problems about real situations. Whereas mathematics asks students to rise above context, quantitative literacy is anchored in the messy contexts of real life. Truly, today's students need both mathematics and numeracy." [2:1].

It is widely believed that mathematics is the most appropriate subject for developing problem-solving and numeracy skills, even though this is usually not the case [3], [4],[5]. Traditional mathematics taught in schools, inadequately prepare students for the quantitative nature of life in the twenty-first century [2]. As far back as 1990, it was recommended that mathematics at school level should develop students' numeracy skills, also referred to as quantitative reasoning, to understand the relationships between mathematics and "real-life situations" [6].

Competencies in basic numeracy and mathematics are regarded as essential for computer science students [7]. Learners often leave school, lacking the basic mathematical and numeracy skills that are required in tertiary programming modules forming part of ICT courses [8]. A significant percentage of South African learners lack the mathematical proficiency needed for tertiary studies [9]. It could be argued that the lack of mathematics proficiency ultimately influences students' numeracy skills. The poor quality of mathematics education in South African schools potentially restrict learners' prospects in terms of further education and training, and more specifically, their access to ICT qualifications which includes programming subjects [10], [11]. More importantly, researchers predict that the cognitive strategies or problem-solving skills underlying programming, also called computational thinking, will become pervasive in all disciplines and walks of life [12].

There is a strong relationship between the problem-solving skills needed for programming and computational thinking skills. Problem-solving skills, which include computational thinking skills, are essential for programming students [13]. One of the definitions of computational thinking which is frequently quoted is the following: *"Computational Thinking is the process of formulating problems and transforming them into computational steps and algorithms."* [14:832].

The layout of the paper is as follows: Section two provides the research objective followed by an overview of the concept Computational thinking (section three) and Numeracy (section 4). In section five, the concept CN is developed followed by an illustration of it in section 6. The paper concludes by suggesting further research to be undertaken including the development of a CN teaching framework to empower lecturers to incorporate the concepts in their teaching.

2 Research Objective

This article addresses the following research question: What is meant by CN for under-prepared, novice programming students? A literature review was conducted to answer the research question and to provide a critical synthesis of published research related to numeracy and computational thinking related to programming skills. The

definition of Computational Numeracy for under-prepared, novice programming students, is presented in section 4.

3 Computational Thinking

Research in the field of Computational Thinking (CT) is most often aligned with Computing Education Research (CER), including programming education [15]. The development of CT as a research topic spans several decades, beginning in the 1960s and extending to the present, but in a seminal article, published in 2006, the author reimagined the older definitions of the concept [16]. The majority of CT publications and research originated in the US and were published in two waves over the years between 2006 and 2012 followed by the second wave which started in 2013 [15].

The well-known author, Wing, believes computational thinking is not just for computer scientists but for everyone, and she believes it should be taught at the same level of importance as reading, writing, and doing arithmetic [16].

A group of authors identified 59 definitions for CT and reasons that CT is a 21st century skill which enables individuals to solve problems in their daily lives [17]. One of the definitions in literature states that CT is *"essentially a framework for defining a set of critical reasoning and problem-solving skills"* [18]. Some authors concur that and define CT as the path followed from the original problem description, the development of an algorithm and finally the final stages of finding and evaluating a solution [18], [19].

The components, strategies, or stages, of CT vary in the literature and include the following: Abstraction; Algorithmic Thinking or Algorithms; Automation; Decompositions or Problem Decomposition; Pattern Recognition; Parallelisation; Simulation; Analysis; Debugging; Evaluation; and Generalisation [12], [16], [20], [21], [22], [23], [24], [25], [26], [27]. We must add that CT requires us to use critical thinking and is embedded in the fundamental principles of computer science (programming) that can be applied in a range of subject areas [28], [29]. Some, or all, of the above components of computational thinking could be included in the final operational definition of CN.

4 Numeracy

The main objective of numeracy, also referred to as quantitative literacy, adult numeracy and even quantitative reasoning, is to be able to understand (basic) mathematics in real-life contexts [30]. Furthermore, it can be said that school mathematics competencies should include numeracy concepts to prepare learners for a variety of quantitative contexts in real-life [31].

In South Africa, the NBT Quantitative Literacy (NBT QL) test is written by school-leavers to determine the level of numeracy and to provide HEIs with supplementary information that could be used for selection and placement of students as well as academic support to newcomer students [32]. The tests assess school-leaving higher education applicants with the objective to address the following question: *"What is the*

academic literacy, quantitative literacy [numeracy] and mathematics levels of proficiencies of the school-leaving population, who wish to continue with higher education, at the point prior to their entry into higher education at which they could realistically be expected to cope with the demands of higher education study?" [32:2].

Each one of the three tests were developed, based on constructs relevant to the specific domain, to address the above question with levels of proficiency as the primary focus. The NBT QL test assesses candidates' ability to solve problems in a real context that is relevant to higher education study while using basic quantitative information, the information could be represented verbally, graphically, in tables or by symbols.

The need for numeracy intervention is evident when the NBT QL results in 2020 of 511 first-year ICT (diploma) students, at a South African University of Technology (from now on referred to as University X), are considered. Only one of the 511 students was considered proficient whereas 96% of the students were in the basic (results less than 34%) and lower intermediate bands (results from 34% to 49%). Students in the Lower Intermediate band will not cope if not placed in extended programmes. Students in the Basic band have serious learning challenges and should be discouraged to enroll at universities (testing was done by Centre for Educational Testing for Access and Placement in February 2020). The results of the NBT QL tests, mentioned above, as well as national results, are indicative of the fact that South African HEIs need to address newcomers' lack of numeracy as a matter of urgency.

"Numeracy, not calculus, is the key to understanding our data-drenched society" [33:2]. The Programme for International Assessment of Adult Competencies (PIAAC) refers to numeracy as the "ability to access, use, interpret and communicate mathematical information and ideas, in order to engage in and manage the mathematical demands of a range of situations in adult life" [34], [35]. The PIAAC further refers to numerate behaviour as dealing with problem-solving in real-world contexts where problems are numerical in nature, containing mathematical content that could be represented in multiple ways.

Numeracy skills further include the ability to demonstrate numerate behaviour as described by some, or all, of the following processes [35], [36]:

- Conceptual understanding of mathematics and the relevant mathematical knowledge: In order to reach this point of conceptual understanding in mathematics, called mathematical proficiency, students need to be guided (taught) through relevant mathematical and numeracy concepts [37], [38].
- Adaptive reasoning and problem-solving skills:
  - Adaptive reasoning is the ability to think in a logical way and to understand the relationships between concepts and contexts [38].
  - Problem-solving requires the ability to reason about the appropriate solutions and finding alternative solutions to problems [38].
- Literacy skills (ability to read, write and talk).
- Knowledge of the context of the problem.
- Prior numeracy-related experiences.

In a workshop hosted by the Higher Education Quality Control Council of Ontario (HEQCC), the improvement of numeracy skills of students in postsecondary (higher)

education, was explored. The six principles that emerged from the workshop concur with the concept of numerate behaviour as described above [35] [36], [38], [39]. The principles should therefore be taken into consideration when addressing the improvement of numeracy skills of students enrolled at higher education institutions:

- The need for numeracy skills is evident in all aspects of our lives;
- Numeracy does not require advanced mathematical skills;
- Numeracy skills are developed through all life-stages, and the development thereof is an ongoing process;
- To be numerate, one must be able to engage with quantitative information represented in different ways, for example, graphs, text, diagrams, maps, and tables;
- Basic number sense (numeracy skills) is essential to make informed decisions when faced with quantitative data or information [40].

The view that mathematics and numeracy are inseparable has probably contributed to the fact that numeracy has received less attention in higher education institutions. To enable programming students with the necessary numeracy skills to solve programming problems, academics at University X, where the first author is a lecturer, created a curriculum for a module called Computational Mathematics (CM). The students are studying towards a National Diploma in Informatics and mostly come from rural, resource-deprived schools. Most of the cohort is registered for the extended course due to poor grade 12 results. The objective of this module is to improve students' computational thinking and problem-solving skills, while preparing them for the quantitative nature of programming problems. The theoretical knowledge obtained from this extended module (CM) is expected to enable students to solve problems of a quantitative nature when programming while contributing to the development of students' reasoning and problem-solving skills.

Based on the six principles mentioned above [40], the concept of numerate behaviour [31], [33], [35], [36], [40], [41] the topics of the NBT QL test, as well as personal experience, numeracy concepts included in the Computational Mathematics module at University X, are: Number Sense, Number System, Fractions and Percentages, Ratios, Rates and Proportions, Basic Algebra, Measurements and Elementary Statistics.

First-year (novice) programming students must be able to solve problems, reason about solutions and then write algorithms to solve the problems. Real-world, quantitative problems must be solved in the known context from where these are translated into a program or a programming concept that can be applied in a real-world problem-solving scenario [42].

It is critical to understand that programming activities do not solely include the syntax and semantics of a programming language, it also requires several other skills and competencies, including computational thinking and numeracy [43]. Therefore, students who study programming must be able to solve and apply problems of a numerical nature.

5 Towards defining Computational Numeracy.

The focus of this article is to identify the components of CT and Numeracy relevant to CN, specifically for under-prepared, novice programming students, by exploring and reviewing both computational thinking and numeracy. The only reference to the term "Computational Numeracy" in literature, was found in a conference paper in the year 2000 where the term computational numeracy was used to refer to computational fluency [44]. This author regards a student as computationally fluent when he/she is fluent in the basic facts of addition, subtraction, multiplication, and division. Computational fluency is regarded as a goal for mathematics education to enable learners (students) to cope with the cognitive demand of more challenging mathematical problems [44].

The term Computational Numeracy will be used to underpin the computational thinking and numeracy skills that novice programming students need to successfully solve programming problems**.** Due to the nature of both numeracy and programming, six concepts of computational thinking were selected from literature to define CN for under-prepared, novice programming students (see Table 1). Referring to the PIAAC assessment domains of the Survey of Adult Skills, CT skills can be considered as cognitive strategies [45]. The cognitive strategies need to be applied to specific content, within a specific context, therefore the inclusion of Numeracy Content and Students' Profile, also referred to as the Context [45].

Table 1. The Domains of Computational Numeracy (headings adapted from [45])

Content (Numeracy)	CT Skills (Cognitive strategies)	Context (Students' Profile)
Number Sense; Number Systems; Fractions, Percentages; Ratios, Rates and Proportions; Basic Algebra; Measurements; and Elementary Statistics.	Problem Decomposition; Pattern Recognition; Abstraction; Algorithmic Thinking; Evaluation; Generalisation.	Poor quality schooling; Lack of mathematical and numeracy proficiency; In need of extensive support, preferably in extended courses.

The majority of the first-year ICT students at the said University X are from disadvantaged backgrounds and had little or no exposure to computers and/or programming. The challenge is to teach the students computational thinking skills which includes reasoning and problem-solving skills. Applying the concepts of CN in the modules, Computational Mathematics and Principles of Programming at University X, is expected to contribute to the development of students' reasoning and problem-solving skills which in turn will enable them to solve problems of a quantitative nature when programming. The developers of the modules are constantly integrating the concepts of numeracy and programming in both modules trying to prevent working in silos. Students are made

aware of the links between the Computational Mathematics and Principles of Programming modules, and of the fact that they need the problem-solving skills they acquire in both modules to succeed in the rest of their ICT careers.

A typical, basic problem which is used in both the Computational Mathematics and Programming Principles modules at University X, will now be discussed to further explain the relevancy of CN for under-prepared, novice programming students.

6 Computational Numeracy Illustrated

During lectures of the module Programming Principles, the principles of programming are first demonstrated by using Scratch 3.0 (block-based visual programming language) after which Java (class-based, object-oriented programming language) is introduced. *"Scratch software appeared to be an engaging and relatively easy to use space for problem solving…Scratch provided a worthwhile and motivating programming environment to explore some mathematical ideas"* [46:55].

By the time students are expected to start using Java, they are familiar with planning their programs, using input-processing-output (IPO) charts, and writing algorithms in pseudocode. The students must write a very basic Java program as one of their first programming assignments; a similar problem is presented to students during their Computational Mathematics classes to reinforce the numeracy concepts of volume and surface area, see Table 2.

Table 2. Computational Numeracy Applied

Numeracy question *(Module: Computational Mathematics)*	**Programming question** *(Module: Programming Principles)*
A local gym built a new indoor swimming pool. The dimensions of the pool are as follows: The length 25 m and the width of 15 m. • Calculate the number of litres that will be needed to fill the pool for the first time if the pool is 1.8 metres deep. • The inside of the pool needs to be waterproofed. 1 litre of waterproofing paint covers 7.75 m^2. Calculate how many 5 litre containers of paint will be needed for 2 coats. Only full tins of paint can be bought.	A fish farmer built a new dam on his farm. The known dimensions of the dam are: Length is 15 metres and width is 10 metres. Open the file called FishDam.java that you downloaded from the LMS. Add code to create a Java program to calculate and display the following: • The number of litres that will be needed to fill the dam for the first time if the dam is a certain depth. Tip: $1m^3 = 1000$ litres • The inside of the dam needs to be waterproofed. 1 litre of paint covers 5.5m^2. • The program should ask the user how many coats of paint should be applied. • Then calculate and display the numbers of tins of paint to be bought. Only full tins of paint can be bought.

- Format the number of litres using a thousand separator.
- Use constant values where possible.

What are the main differences between the above problem statements?

- The specific values are given.
- The students use known formulae to calculate the results, using given values.
- Calculations are done on a calculator (or on paper).

- The students must understand and apply the concept of variables.
- The user decides which values must be used.
- The formulae are incorporated in the algorithm (processing).

What are the similarities between the above problem statements?

Selection of the "appropriate arithmetic operation", correct use and execution of the formulae in numeracy and programming could be considered as a computational skill [47], also described as algorithmic thinking. Students must understand numeracy concepts to apply formulae correctly. The numeracy concepts could include the arithmetic operations (addition, subtraction, multiplication, division) and the relevant rules, for example, order of operations.

Following a problem-solving process [48] ***the components of CN can be linked to the above problem statements:***

Identify the Problem:

Problem Decomposition: Programming requires complex problems to be broken down into smaller, more manageable problems, it is also referred to as "divide and conquer" [22]. Similar skills are needed for solving numeracy problems. The student must break up the problem in smaller sections:

- To calculate the number of litres (of water), the students must identify that volume must be calculated first. To calculate how much paint is needed, the students must identify that total surface area must be calculated first.

Abstraction: Abstraction is a process of identifying and removing redundant information from a problem, resulting in a problem without unnecessary detail [23]. What is the redundant information in the above problem statements?

- "A fish farmer built a new dam on his farm" and "A local gym built a new indoor swimming pool".

It is only necessary for students to know that it is a dam or a pool and what a dam or a pool is (dams and pools are usually filled with water and has certain measurements called dimensions).

Gather data:

Pattern Recognition: When solving a problem, it is valuable to look for patterns and/or trends within the problem, or patterns previously recognised in other problems [23]. Students are (supposed to be) familiar with the concepts of volume and total surface area from school mathematics and the Computational Mathematics course. They should recognize the similarities and apply their numeracy skills in this programming context.

Plan the Solution:

Algorithmic Thinking: When expressing the problem in sequential steps, it becomes an algorithm. Understanding and writing calculations in steps, is algorithmic thinking, whether in pseudo code, or in terms of numbers and mathematical operations. Following an algorithmic thinking approach, simplifies problem-solving [22]. When writing the pseudocode, numeracy principles are incorporated, values are unknown, and the user should provide values:

- To calculate the volume and the total surface area, the length, width, and depth is needed. The depth is unknown and therefore the student must understand that the user must provide the depth for the program to produce the correct solution (output). The user may also decide (and capture) how many coats of paint he/she wants to paint.

Implement and Assess the Solution:

Evaluation: It is crucial to judge (test) whether a solution is effective or correct, an effective solution could also be generalised [18], [19]. Students are encouraged to reflect whether the output of their programs correlate with the calculations done with a calculator. They must experiment and try with different values as well as with null values to test possible logical errors.

Generalisation: Students may recognise similarities with problems they previously solved and then apply their prior knowledge to solve a new problem [26]. In the "Report of a Workshop of Pedagogical Aspects of Computational Thinking", Kolodner emphasised that when a student is able to reflect on a problem and its solution, and then teach a peer how to solve the problem, the student demonstrates computational thinking [49]. Students should recognise the similarities of exercises done in the Computational Mathematics class and apply it to the given scenario in programming.

7 Conclusion

First-year ICT students at University X are under-prepared for tertiary studies, and even more so under-prepared for the problem-solving nature of programming. Students

qualify for entrance to the extended ICT diplomas with an average mark of 50% to 59% for Grade 12 Mathematics or 80% to 89% for Grade 12 Mathematical Literacy. Unfortunately, most of the numeracy concepts are only taught in primary school (Grades 1 to 7) and by the time learners reach Grade 12, most of it has been forgotten. The students' knowledge of basic numeracy has therefore proven to be inadequate for solving programming problems of a quantitative nature. Therefore, students find ICT studies challenging without adequate numeracy skills. Numeracy is commonly rooted in real life scenarios, which is reflected in programming in a wide variety of contexts.

This paper presented Computational Numeracy as a new concept to be used in the teaching of under-prepared programming students. An initial definition of the concept includes different components: content (numeracy), cognitive strategies (computational thinking) and context (students with low numeracy skills levels due to different environmental factors). Computational Numeracy teaching strategies need to take these three components into account. The concept was illustrated by showing how these components are used in the teaching of numeracy and programming to students at a university of technology in South Africa.

Although ICT lecturers in South Africa are well-qualified in their field of study, most of the ICT lecturers at University X do not have teaching qualifications and are therefore not well equipped to teach under-prepared students. Teaching under-prepared, novice students to program is challenging, especially if a lecturer is a seasoned programmer. Going back to basics and teaching students problem-solving, numeracy and computational thinking skills, while teaching programming, is a challenge, and a skill. As part of a larger study, a framework will be developed, providing guidance to lecturers to assist novice, under-prepared programming students in developing a solid foundation for their ICT studies.

8 References

1. OECD: Skills Matter. OECD (2016). https://doi.org/10.1787/9789264258051-en.
2. Steen, L.A.: Mathematics and Numeracy: Two Literacies, One Language. The Mathematics Educator. 6(1), 10-16. (2001).
3. Heymann, H.W.: Why Teach Mathematics? A Focus on General Education. Kluwer Academic, Dordrecht (2010).
4. Resnick, L.B.: Nested learning systems for the thinking curriculum. Educational Researcher. 39, (2010). https://doi.org/10.3102/0013189X10364671.
5. Spaull, N., Taylor, S.: Access to what? Creating a composite measure of educational quantity and educational quality for 11 African countries. Comparative Education Review. 59, (2015). https://doi.org/10.1086/679295.
6. Steen, L.A., National Research Council (U.S.). Mathematical Sciences Education Board.: On the shoulders of giants: new approaches to numeracy. National Academy Press (1990).

7. Oddie, A., Hazlewood, P., Blakeway, S., Whitfield, A.: Introductory Problem Solving and Programming: Robotics Versus Traditional Approaches. Innovation in Teaching and Learning in Information and Computer Sciences. 9, (2010). https://doi.org/10.11120/ital.2010.09020011.
8. Barlow-Jones, G., Chetty, J.: The Effects of a Social Constructivist Pedagogy on At-risk Students Completing a Computer Programming Course at a Post-Secondary Institution. (2012).
9. Spaull, N., Kotze, J.: Starting behind and staying behind in South Africa. The case of insurmountable learning deficits in mathematics. International Journal of Educational Development. 41, 13–24 (2015). https://doi.org/10.1016/j.ijedudev.2015.01.002.
10. Marnewick, C.: The mystery of student selection: Are there any selection criteria? Educational Studies. 38, (2012). https://doi.org/10.1080/03055698.2011.567041.
11. Barlow-Jones, G.: High school mathematics marks as an admission criterion for entry into programming courses at a South African university. (2015).
12. Wing, J.M.: Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 366, 3717–3725 (2008). https://doi.org/10.1098/rsta.2008.0118.
13. de Jong, I.: Teaching Computational Thinking with Interventions Adapted to Undergraduate Students' Proficiency Levels. In: Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE. pp. 571–572. Association for Computing Machinery (2020). https://doi.org/10.1145/3341525.3394001.
14. Aho, A. v.: Computation and computational thinking. Computer Journal. 55, 833–835 (2012). https://doi.org/10.1093/comjnl/bxs074.
15. Saqr, M., Ng, K., Oyelere, S. sunday, Tedre, M.: People, Ideas, Milestones: A Scientometric Study of Computational Thinking. ACM Transactions on Computing Education. 21, 1–17 (2021). https://doi.org/10.1145/3445984.
16. Wing, J.M.: Computational Thinking. Communications of the ACM. 49, 33–35 (2006).
17. Haseski, H.I., Ilic, U., Tugtekin, U.: Defining a New 21st Century Skill-Computational Thinking: Concepts and Trends. International Education Studies. 11, 29 (2018). https://doi.org/10.5539/ies.v11n4p29.
18. Hunsaker, E.: Computational Thinking. (2018).
19. Sethi, R.J.: Essential Computational Thinking Computer Vision-Group Analysis in Video View project Scientific Workflows for Visual Stylometry: Digital Tools for Exploring the Nature of Artistic Style in the Visual Arts View project.
20. Angeli, C., Voogt, J., Fluck, A.: A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. (2016).
21. Barr, V., Stephenson, C.: Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads. 2, 48–54 (2011). https://doi.org/10.1145/1929887.1929905.

22. Cansu, F.K., Cansu, S.K.: An Overview of Computational Thinking. International Journal of Computer Science Education in Schools. 3, 17–30 (2019). https://doi.org/10.21585/ijcses.v3i1.53.
23. Csizmadia, A., Curzon, P., Humphreys, S., Ng, T., Selby, C., Woollard, J.: Computational Thinking - A Guide for Teachers. (2015).
24. Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., Werner, L.: Computational thinking for youth in practice. ACM Inroads. 2, 32–37 (2011). https://doi.org/10.1145/1929887.1929902.
25. Hello World issue 4 — Hello World, https://helloworld.raspberrypi.org/issues/4, last accessed 2021/05/10.
26. Selby, C.C., Woollard, J.: Computational Thinking: The Developing Definition. (2010).
27. Wing, J.M.: Computational Thinking: What and Why? (2010).
28. Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., Wilensky, U.: Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology. 25, 127–147 (2016). https://doi.org/10.1007/s10956-015-9581-5.
29. Yadav, A., Hong, H., Stephenson, C.: Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. TechTrends. 60, 565–568 (2016). https://doi.org/10.1007/s11528-016-0087-7.
30. van Peursem, D., Keller, C., Pietrzak, D., Wagner, C., Bennett, C.: a Comparison of Performance and Attitudes Between Students Enrolled in College Algebra vs. Quantitative Literacy. Mathematics & Computer Education. 46, (2012).
31. Jablonka, E.: The evolvement of numeracy and mathematical literacy curricula and the construction of hierarchies of numerate or mathematically literate subjects. ZDM - International Journal on Mathematics Education. 47, 599–609 (2015). https://doi.org/10.1007/s11858-015-0691-6.
32. Centre for Educational Testing for Access and Placement: The National Benchmark Tests National Report 2018 Intake Cycle. (2018).
33. Steen, L.A.: Numeracy: The New Literacy for a Data-Drenched Society. Educational Leadership. 57, 8-13. (1999).
34. OECD: Literacy, Numeracy and Problem Solving in Technology-Rich Environments: Framework for the OECD Survey of Adult Skills. (2012).
35. Alatorre, S., Close, S., Evans, J., Kingdom, U., Johansen, L., Maguire, T.: Piaac Numeracy: a Conceptual Framework. Education. (2009).
36. Curry, D., Soroui, J.: Using the PIAAC Numeracy Framework to Guide Instruction: An Introduction for Adult Educators. (2017).
37. Pei, C. (Yu), Weintrop, D., Wilensky, U.: Cultivating Computational Thinking Practices and Mathematical Habits of Mind in Lattice Land. Mathematical Thinking and Learning. 20, 75–89 (2018). https://doi.org/10.1080/10986065.2018.1403543.
38. Jeremy Kilpatrick, Jane Swafford, B.F.: Adding It Up: Helping Children Learn Mathematics Jeremy. Society. II, (2001).

39. OECD: Literacy, Numeracy and Problem Solving in Technology-Rich Environments. OECD Publishing., Paris (2012). https://doi.org/10.1787/9789264128859-en.
40. Brumwell, S., Macfarlane, A.: Improving Numeracy Skills of Postsecondary Students: What is the Way Forward? The Higher Education Quality Council of Ontario Cite this publication in the following format.
41. Peursem, V.: A Comparison of Performance and Attitudes Between Students Enrolled in College Algebra vs Quantitative Literacy. Mathematics and Computer Education. Vol. 46, Issue. 2, 107-118. (2012).
42. Rogalski, J., Samurçay, R.: Acquisition of Programming Knowledge and Skills. In: Psychology of Programming. pp. 157–174. Elsevier (1990). https://doi.org/10.1016/b978-0-12-350772-3.50015-x.
43. Attallah, B., Ilagure, Z., Chang, Y.K.: The Impact of Competencies in Mathematics and beyond on Learning Computer Programming in Higher Education. In: ITT 2018 - Information Technology Trends: Emerging Technologies for Artificial Intelligence. pp. 77–81. Institute of Electrical and Electronics Engineers Inc. (2019). https://doi.org/10.1109/CTIT.2018.8649527.
44. Australian Council for Educational Research (ACER), Australian Council for Educational Research (ACER). Research Conference (2000 : Brisbane): Improving numeracy learning : research conference 2000 : proceedings.
45. OECD Skills Outlook 2013. OECD (2013). https://doi.org/10.1787/9789264204256-en.
46. Calder, N.: Using scratch to facilitate mathematical thinking. Waikato Journal of Education. 23, (2018). https://doi.org/10.15663/wje.v23i2.654.
47. Millians, M.: Computational Skills. In Encyclopedia of Child Behavior and Development. (2011). https://doi.org/10.1007/978-0-387-79061-9.
48. Kalelioğlu, F., Gülbahar, Y., Kukul, V.: A Framework for Computational Thinking Based on a Systematic Research Review. (2016).
49. Report of a Workshop on the Pedagogical Aspects of Computational Thinking. National Academies Press (2011). https://doi.org/10.17226/13170.