Computational Thinking Dashboard

For learners in Jupyter notebooks

by

Bhoomika Agarwal

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday October 29, 2021 at 10:00 AM.

Student number:	5018463	
Project duration:	March 24, 2021 – October 29, 2021	
Thesis committee:	Prof. Dr. M. Specht,	TU Delft, supervisor
	Dr. M. Aniche,	TU Delft

This thesis is confidential and cannot be made public until October 20, 2021.

TU Delft Delft University of Technology

Preface

"Students must be taught how to think, not what to think."

- Margaret Mead

Acknowledgements

Firstly, I would like to offer my heartfelt thanks to my supervisor, Dr. Marcus Specht for his continued guidance and support through this project. Your feedback helped me gain direction and put my thoughts and ideas into action.

I would also like to thank Manuel Valle Torre, Ioana Jivet and Xiaoling Xhang for their guidance, patience and critical feedback that helped me chisel my initial ideas for this project into a completed product. Your insights always pushed me in the right direction and kept me anchored.

I would like to express my gratitude to my parents for their unwavering support and love that kept me going all my life and continues to do so. I owe special thanks to my sister for keeping my spirits up and helping me see the light at the end of the tunnel. Without her support, I would be like a boat at sea without a compass. Last, but not the least, I am forever indebted to my housemates, my partner and my friends for making me feel at home in Delft while I was away from home. I could not have asked for a better surrogate family.

- Bhoomika Agarwal
October 2021

Contents

1 Introduction

With the rapid integration of computers and technology into our daily lives in the 21st century, we are in the midst of the technology revolution. While it is not yet necessary to learn to program or code, most of us use computers on a daily basis. We need to learn to think like them to get the best of this revolution. This process of thinking like a programmer or a computer scientist is called Computational Thinking(CT). As per the National Research Council of the National Academy of Sciences in the United States of America, CT is a skill that everyone should acquire, not just programmers[13][14].

Computational Thinking is a concept that lacks an agreed-upon definition[46][10][6][14]. It was first mentioned by Wing in 2006 in an article in the Communications of the ACM calling for the integration of CT beyond just computer scientists[51]. Wing later defined Computational Thinking as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent"[50, p. 1]. Barr and Stephenson defined CT in the context of K-12 education in terms of the core computational thinking concepts and capabilities, providing examples of how they can be applied in different disciplines. Brennan and Resnick defined CT with respect to design-based learning activities in Scratch[1] - a block-based programming language - in terms of three dimensions: computational concepts, computational practices, and computational perspectives. Weintrop et al. defined CT in terms of 4 major categories and 22 sub-skills: data practices , modeling and simulation practices, computational problem-solving practices, and systems thinking practices[49]. Selby and Woollard defined CT as the thought processes of abstraction, decomposition, algorithmic design, evaluation, and generalization[43]. Among all of these varied definitions, this research uses an adapted version of the definition by Brennan and Resnick, considering 2 dimensions - computational concepts and computational practices[10] - and adapting them to the Python programming language.

Unlike programming skills, CT skills are transferable and can be used to solve problems in everyday life. Wing further elaborates on this by giving everyday examples of how we use pre-fetching, caching, backtracking, online algorithms, independence of failure and redundancy in design, and other CT concepts in everyday tasks[51]. The International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA) also elaborate on how CT skills can be used to solve everyday problems in different domains and different grade levels[47]. Section 2.1 elaborates on the importance of CT skills in domains besides computer science and the need to integrate them into curricula as CT skills have an immense potential to transform other domains.

Even though there is no consensus on the definition of CT in research, there is a definite consensus on the potential of CT skills to transform the way students learn at educational institutions[6][51][46]. This consensus has led to multiple attempts to integrate CT into classrooms in the form of game design[10], introductory courses on CT[25], physics courses[3][20], mathematics courses[38][49] and courses in other domains[49][2]. With this increasing integration of CT into classrooms, there is a need for CT assessment tools to evaluate acquisition of CT skills[18][46]. Section 2.2 looks at various tools to assess CT skills in education, grouping them into four types - traditional test, portfolio assessment, interviews and surveys. This research aims to develop a CT assessment framework that allows self-regulated learning through feedback in the form of learning analytics dashboards.

[1] https://scratch.mit.edu

"Learning analytics is the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs"[44]. Suthers and Verbert call learning analytics (LA) as the "middle space" between learning and analytics where they come together. LA aims to enhance the learning process with analytics in the form of computational data about it. One of the ways to do this is through visualization of the learning data in the form of LA dashboards.

Learning analytics (LA) dashboards are learning tools that support students and teachers by helping them understand LA data and use it to improve learning[23]. One of the main aims of such dashboards is to support students in improving their learning and performance by empowering them by presenting them with learning patterns that can motivate them and improve learning strategies[21]. Designing LA dashboards with support for SRL strategies helps learners support awareness and trigger reflection[22]. Self-regulated learning (SRL) is the theoretical framework used for the design of Learning Analytics dashboards most frequently[22].

Self-regulated learning is the ability to control, manage and plan one's own learning process with an aim to increase attainment of goals[21]. With the increase in online learning, the need for independent learning and self-direction by students has increased[11]. This has also resulted in a decrease in completion rates of online courses. Self-regulated learning strategies can help solve this as they can increase success in problem solving, allow higher academic achievement, intrinsic motivation and interest in learning tasks[40]. This research enables self-regulated learning by providing feedback to students through CT dashboards. Thus, the students can regulate their learning process themselves.

My research aims to answer the research question: *How can computational thinking be assessed through detection of user micro-interactions in a university-level self-paced Python beginners course integrated into Jupyter notebooks?* This is done by integrating a CT assessment framework that uses user micro-interactions via CT dashboards in a university-level self-paced Python beginners course. This research question is further broken down into 3 research sub-questions :

1. *RQ1: Did the users acquire Computational Thinking (CT) skills in the form of both CT concepts and CT practices?*
2. *RQ2: Was there a significant improvement in the self-reported CT skills of users after taking the Python basic programming course?*
3. *RQ3: How do the self-reported survey responses correspond to the actual user micro-interaction data?*

The rest of this report is structured as follows: chapter 2 identifies the gaps in existing computational thinking assessments and how my research bridges these gaps. Following this, chapter 3 lays down the design and implementation details of my research and the chapter 4 elaborates on the user evaluation study conducted to verify the same. Lastly, chapter 5 draws the conclusions of the research.

2

Learning and Computational Thinking

This chapter highlights the gaps in the existing research by examining the existing literature and shows and how my research addresses those gaps. The chapter begins by establishing the importance of computational thinking (CT) in learning. Then it examines the available Computational thinking assessment tools and highlights their shortcomings. Finally, it looks at Learning Analytics (LA), LA Dashboards and Self-regulated Learning(SRL) and how they can be used to enhance the learning process.

2.1. Computational Thinking

The concept of Computational thinking was brought to the limelight in 2006 when Wing suggested that thinking computationally was a fundamental skill for everyone, not just computer scientists, and argued for the importance of integrating computational ideas into other subjects in school[51]. Since then, there have been numerous efforts to teach CT to students at various levels.

Computational thinking has been shown to be a valuable skill for other domains and disciplines such as mathematics and science. Multiple studies have looked at CT skills as a transfer skill and how it can be applied in other domains. Tang et al.[46] show that there is a lag in the implementation of CT into non-CS classrooms.

Weintrop et al.[49] define a taxonomy for CT and argue "for the inclusion of computational thinking in mathematics and science classrooms"[49, p.17]. Their research states 3 main benefits of integrating CT into mathematics and science curricula -

1. building upon the reciprocal learning relationship between CT and science
2. practical concerns regarding reaching all students and making teachers proficient are addressed
3. aligns science and mathematics education with the field's current professional practices

The authors are working upon enhancing high school STEM curricula through activities based on their taxonomy that can be easily incorporated.

Ahamed et al.[2] conducted a workshop for science teachers working with grades 9-12 to introduce computational thinking to high school teachers to change long-term perceptions about computer science and its relationship with other disciplines. They not only designed modules addressing physics, biology, chemistry, and mathematics, but also designed modules for common areas of overlap, including probability, simulations, and computational thinking. Their language of choice was VPython, a visual extension to the popular Python programming language.

Kafura, Bart, and Chowdhury[25] elaborate on the design process for an 'Introduction to Computational Thinking course' that must be completed by all graduating students. The early results and quantitative and qualitative assessments show that students gained "a deeper appreciation for the use of computation in their own disciplines"[25, p.6]. The study uses NetLogo programming environment, Blockly (block-based programming language) and a Python IDE for its 3 topics, with a progressive organization of the topics.

Aiken et al.[3] integrated computation in an existing 9th-grade physics course for 32 students and assessed the development of transfer CT skills via 3 methods - proctored programming assignment, written essay and think-aloud interviews that involved creation and discussion of a computational model of a baseball in motion using VPython. The results were that about a third of all the students successfully completed

the programming assignment, demonstrating how they "synthesized their knowledge of physics and computation"[3, p. 2].

With a focus on Physics, Jaipal-Jamani and Angeli[20] examined the self-efficacy, understanding of science and CT in elementary pre-service teachers as they interacted with LEGO WeDo robotics kits in a science methods course. The main aim of this research is to investigate the preparedness of teachers to teach STEM. The results showed that "engaging with robotics in an existing science methods course can enhance preservice teachers' self-efficacy beliefs to teach with robotics, their science knowledge, and their computational thinking skills", calling for more quantitative studies on robotics[20]. [32] also showed similar results of increase in teachers' self-efficacy, CT understanding and dispositions by using robotics and game design to broaden STEM participation in rural communities.

With a focus on CT and mathematics, Pei, Weintrop, and Wilensky looked at 'mathematical habits of mind' using Lattice Land - "a mathematical microworld where learners explore geometrical concepts by manipulating polygons drawn with discrete points on a plane" [38, p.2]. These mathematical habits of mind were associated with CT concepts using 4 categories defined in [49]: data, modeling and simulation, computational problem-solving, and systems thinking. Using the example of Lattice Land, the authors show that embedding CT practices in mathematics can have broad impacts by providing the required context that allows students to meaningfully engage in CT. In this way, CT practices can motivate students to learn powerful mathematical concepts.

A majority of the cross-disciplinary research mentioned above makes use of visual and block-based programming languages such as Scratch[1], Logo[2] and VPython[3]. Block-based programming languages are ones that are comprised of 'blocks' of instructions or actions that can be dragged and dropped into the editor and run easily. This graphical representation of code makes it easier to learn the basics of programming, especially for K-12 students. The ease of learning provided by block-based programming languages also makes it suitable for integrating it into curricula in other domains. On the downside, the functionality of block-based programming language is limited by the available blocks and they do not offer the flexibility that text-based programming languages provide. Tang et al. show that a majority of studies related to CT are focused on elementary and middle school grade levels and emphasize on the need for more studies for high school and college students so that the complete development trajectory for CT skills in students can be mapped[46].

2.2. Computational Thinking Assessment

While Computational thinking (CT) is being integrated into curricula rapidly, there is a need for methods to assess and evaluate learning of CT concepts[18][46]. The lack of an agreed-upon definition of CT, lack of assessment mechanisms for CT and lack of usage of CT in classrooms are the major roadblocks in the integration of CT into curricula[33].

This chapter examines the available Computational thinking (CT) assessment tools and highlights their shortcomings.

A variety of tools and frameworks have pioneered the assessment and analysis of CT in educational environments. Tang et al.[46] observed 4 types of assessment -

1. Traditional test composed of selected- or constructed-response questions
2. Portfolio assessment
3. Interviews
4. Surveys[46]

While portfolio assessments are used widely for holistic assessment of the CT skills gained through projects, time-consuming manual methods such as traditional paper-pencil tests, surveys and interviews form a huge chunk of the assessment methods. A majority of the CT assessments lacked evidence for reliability and validity, making it challenging to use them with confidence in classrooms[46].

2.2.1. Traditional test

Traditional tests are the usage of multiple choice questions or open-ended questions to evaluate the acquisition of CT concepts summatively[46]. Their frequent usage implies that CT skills are considered a learning product and this type of assessment can be quite time-consuming due to the manual effort involved in collecting the data and grading the results.

[1] https://scratch.mit.edu
[2] https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html
[3] https://vpython.org/

González[17] presents the design of a Computational Thinking Test (CT-test) aimed at Spanish students between 12 and 13 years (grades K-7 & K-8). They provide a detailed description of the design guidelines underlying the test and the content validation via expert judgement. However, this test cannot be extended to other programming languages as it is designed for Scratch and focuses on programming concepts of Scratch.

The Bebras challenge[4] is an online international initiative that strives to promote CT among school students through a set of fun and engaging short problems called the Bebras tasks. Araujo et al. evaluate the Bebras challenge as a tool for CT assessment, finding that "the performance on Bebras is only moderately correlated to the student grade"[5, p. 2]. The study was conducted with 138 students from two Brazilian universities. The authors take into account the grades from an introductory programming course and the students' performance in two simulated Bebras challenge - one before the course and one after. The conclusion is that CT measures cannot be derived from the current design of the Bebras challenge. This conclusion is based on the analysis of Item Curve Characteristics by considering Item Response Theory, which shows that the Bebras tasks are not useful in discriminating students with high and low levels of CT ability. Based on this result, my research rules out Bebras challenge as an evaluation for CT skills through programming.

2.2.2. Portfolio assessment

Portfolio assessment is a systematic process of collecting and evaluating students' projects, notes or other direct observations to evaluate CT skills. These assessment tools can capture a holistic view of obtained CT skills through projects or code submitted to programming or computing platforms[46]. Portfolio assessments can measure the quality of the programming product quite well but cannot capture the learning process and affective outcomes.

One of the most well-known portfolio assessments tools for CTA is Dr. Scratch [36, 37] - a free and open-source web tool, powered by Hairball, that analyzes Scratch projects to automatically assign a CT score using 8 different parameters. The authors randomly downloaded and analyzed 100 projects from the Scratch web repository, rating them into 3 levels - Basic, Developing and Proficient - based on a set of rules. While they measured some of the aspects of CT, code analysis cannot throw light on aspects such as debugging and remixing that are more real-time. The mapping developed between programming concepts and aspects of CT by this research serves as a foundation for CT assessment(CTA) as it serves as a bridge between CT and programming.

Ambrósio, Xavier, and Georges proposed 'Digital Ink' - a cognitive assessment test for Computational Thinking for freshman university students taking introductory programming [4]. This research studies the cognitive processes associated with CT based on the Cattell-Horn-Carroll (CHC) framework of intelligence[12]. The test evaluates spatial reasoning, time spent on each task and number of corrections made as parameters for a map navigation application. The second digital assessment is the 'traditional Rey-Osterrieth's Complex' for testing long-term visual memory and visual perception. However, the implemented applications of Digital Ink in this research have not been extended to text-based programming languages but are limited to applications that store traces drawn by subjects, followed by searching this data for relevant patterns. While the authors do mention the easy reproduction of hand-written materials using Digital Ink, there is not much elaboration or stated application of this. In addition, the implemented version of the Computational Thinking Test using Digital Ink is yet to be verified as an adequate assessment tool for CT.

Koh et al.[29] evaluate the extent to which visual programming languages help students learn CT concepts using the visual evaluation tool developed by them - Computational Thinking Pattern (CTP) graph."The CTP graph illustrates the amounts and kinds of computational thinking patterns implemented in a given game"[29, p. 5]. While visual programming languages motivate students, the CTP graph evaluates the extent of knowledge transfer within computer science and other disciplines. The main limitations of the CTP graph are its arbitrary nature, difficulty in interpretation, accuracy and amount of chosen CT patterns. Koh et al. use the CTP graph in REACT[28] - a real-time online assessment system for Scalable Game Design (SGD) projects that focus on teaching CT to students at an early age via the creation of playable games. This tool enables teachers to see the students' understanding of CT concepts by analysing CT patterns for visual language learning. The downside of this system is that it is web-based and is embedded in the SGD arcade. This approach is based on visual language learning and is not easily extended or ported to traditional programming languages, confining the scope greatly. Also, the dashboard implemented is preliminary in that it only displays the student progress and enables identification of the students who require scaffolding. Further data analysis is required to understand the underlying factors. My research will focus on creating a portable ex-

[4] https://www.bebras.org/

tension that is applicable to traditional programming languages for CT analysis. Owing to the holistic nature of portfolio assessments, my research will use this assessment method to evaluate CT skills.

2.2.3. Interviews

Interviews are used by researchers to examine the participants' understanding of CT skills which are then coded using pre-developed protocols. This approach is useful for capturing the processing behind learning and applying CT skills. The main downside is that it is time-consuming for both the researcher and the participant and requires a lot of manual effort on behalf of the researcher.

Yuen and Robbins[55] conduct a qualitative case study to understand the thought processes of students in a beginners' data-driven computer science course for biology students. The majority of the data is collected via three clinical interviews per student where the student has to work upon or review their own solutions for a given MATLAB-based programming project[55]. This data was then transcribed and analyzed based on a grounded theory approach. "The findings of this study showed that data computation is an ongoing process in which participants refine their understanding of the data through organizing and concurrent restructuring of both their programs and visualizations."[55, p.14]

Wong and Cheung[52] study the impact of programming education on creative thinking, critical thinking and problem solving. A computer programming curriculum was first introduced in the primary school and students learnt to program interactive games. Data was collected via questionnaires before the intervention and via interviews after the intervention. Structured interviews were conducted where the students were asked about their perception of the relationship between programming and other courses taught at school. They were also interviewed about the skills involved in learning. The results show an improvement in the creative thinking, critical thinking and problem solving capabilities of the students' after the intervention. Due to the time-consuming nature of interviews and the manual effort involved, my research does not use interviews as an assessment method.

2.2.4. Surveys

Surveys are useful to assess affective learning outcomes such as motivation and attitude towards learning CT skills via quantitative items and open-ended questions. While they are an efficient and convenient data collection instrument, surveys cannot capture spontaneous attitudes and might not be too accurate for young children due to the questions being misinterpreted.

Yağcı[53] develops a scale to measure CT skills in high school students using 42 items with a five-point Likert scale. These questions assessed 4 factors - Problem Solving, Creative Thinking, Critical Thinking and Algorithmic Thinking[53]. The authors proved the validity and reliability of the scale statistically as a tool to measure CT skills in high school students. This scale has not yet been tested for different grade levels and with other CT concepts.

Korkmaz, Çakir, and Özden[30] developed a scale to determine the levels of CT skills of students via a five-point Likert scale consisting of 29 items. This scale considers five factors - Creativity, Cooperativity, Algoritmic- Critical Thinking and Problem Solving[30]. The authors also conduct some analyses to show the validity and reliability of the scale as a tool to measure the computational thinking skills of the students. Korucu, Gencturk, and Gundogdu investigate CT skills in secondary school students in terms of various variables using the scale developed by Korkmaz, Çakir, and Özden and considering the usage of technology(both mobile and remote) and the internet. Their findings reveal that there is a difference among the CT skills of the participants in terms of the class levels and duration of possession of mobile technologies but do not differ in terms of gender, weekly internet usage durations, mobile device usage competence situations[31].

Benakli et al.[7] promote CT skills in STEM students via hands-on computer experiments. The study presents nine experiments and suggests more that teach CT throughs visualizations, simulations and data analysis. A survey was conducted to understand the experience of the student about using R in the classroom. The authors believe that "there is evidence for the pedagogical effectiveness of using R as a scientific programming language for stochastic and deterministic modelling and simulation, data analysis, visualization, statistical computing, etc".[7, p.31]. The authors conclude that CT can be encouraged in students by using hands-on computational projects that enable them to solve problems using technology and programming.

Kılıç, Gökoğlu, and Öztürk developed a survey with 33 questions using a five-point Likert scale to assess the programming-oriented CT skills among university students[26]. This research considers 3 aspects of CT - conceptual knowledge, algorithmic thinking, and evaluation - based on their common occurrence in various studies examined that focus on the programming concepts in CT. The validity of this scale is tested by

exploratory factor analysis using principal component analysis and the reliability of the scale was proven to be satisfactory using item-total correlations and Cronbach Alpha[26]. As surveys are automated and can be used to assess attitudes towards CT, my research will use an adapted version of the survey developed by Kılıç, Gökoğlu, and Öztürk[26] as an evaluation method.

2.2.5. Combinations

Each of the assessment types stated above have their own downsides. In order to get the best results, a number of studies use a combination of multiple assessment types. The usage of a combination of assessment tools allows the examination of the CT learning experience of the user from different aspects. The shortcomings of one type of CT assessment tool can be solved by using it alongside another type that has a stronger focus on the shortcoming.

Brennan and Resnick developed a foundational CT framework for Scratch (a block-based programming language) based on their studies on interactive media design[10]. It contains 3 key dimensions: "computational concepts (the concepts designers employ as they program), computational practices (the practices designers develop as they program), and computational perspectives (the perspectives designers form about the world around them and about themselves)"[10, p.3]. This paper also discusses methods to assess CT skills: project portfolio analysis, artifact-based interviews and design scenarios. One of the major downsides of these assessment techniques is that they are too time-consuming and not all of them are real-time.

Computational Thinking Assessment System (CoTAS) [54] enables assessment of CT skills, both summative and formative, for high school students, thereby supporting teachers to shape the continuous learning process. It can be used to teach programming and other disciplines that require CT skills integration and are based on Python. The evaluation is done based on problem-based tests, problem-based assignments, surveys and interviews. It cannot be integrated into current existing CT curricula and acts as an evaluation extension to them instead.

Owing to the advantages of a combination of assessments, my research will use a combination of a portfolio assessment and an adapted version of the survey scale developed by Kılıç, Gökoğlu, and Öztürk[26] to assess the programming-oriented CT skills of undergraduate students. By using this combination, the attitudes of the users towards CT skills can be measured using the scale and a holistic view of the users' CT skills can be gained through the portfolio assessment.

2.3. LA and LA Dashboards

With the increasing usage of online learning environments and educational platforms such as Learning Management Systems and Massive Open Online Courses, there has been a tremendous amount of learning data generated in the last decade. There is a need to harness the potential of this learner data and make the most of it through the usage of data analytics. This is where Learning Analytics (LA) comes into the picture. LA is a domain that acts as the intersectional space between learning sciences and data analytics[45]. The Society for Learning Analytics(SoLAR)[5] defined LA as "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs"[44]. The main aim of LA is to use the data from learning sciences to augment and enhance the learning process.

With the availability of huge amounts of learner data, it is necessary to transform this data into knowledge that can be useful to the learner. Durall and Gros elaborate on the importance of information visualization as a tool for sense-making in the form of goal-oriented visualizations like dashboards through synthesis of complex information and facilitation of comparisons and inferences[15]. LA can act as a tool for reflection and enable learners to filter their visualizations to focus on their learning goals[15].

LA dashboards are learning tools that can help learners and teachers harness the power of LA use it to improve their learning[23]. Schwendimann et al. define LA dashboards as "a single display that aggregates different indicators about learner(s), learning process(es) and/or learning context(s) into one or multiple visualizations"[42, p. 8]. LA dashboards can help students improve learning and performance by presenting them with learning patterns that can help motivate them and improve their learning strategies[21]. By making the learner aware of their progress and triggering self-reflection[22], LA dashboards can help users regulate their own learning.

[5] https://www.solaresearch.org

2.4. Self-regulated learning

Self-regulated learning (SRL) can be understood as the "ability to control, manage, and plan learning actions and behavioral processes that increase goal attainment" [21, p.6]. Students who apply SRL strategies in their learning are more successful in problem solving, and have higher academic achievement, intrinsic motivation, and task interests[40].

Online learning provides flexibility and accessibility to students through increased learning opportunities, access to learning resources and opportunities for collaboration[11]. The downside of online learning is that its success relies heavily on independent learning and the students' autonomous engagement in the course[11]. SRL strategies can help learners to gain and retain knowledge methodically and systematically[11].

Self-regulated learning is the theoretical framework mostly used for designing Learning Analytics Dashboards[22] in general. Jivet et al. investigate the effect of learner goals and SRL skills on dashboard-sense making. The results found 26 factors that students use to interpret dashboards ranging from indicators to visualizations to contextualizing features to action taking features. The authors further conducted an exploratory factor analysis on these items to obtain 3 sense-making factors: transparency of design, reference frames and support for action[23]. Using multiple regression analyses on these sense-making factors, this research found that there are specific relations between SRL and these three sense-making factors. Through these relations, they show that "students could use the support for action offered by the dashboard in order to adjust their goals and plans" for SRL[23, p.12].

Boekaerts identify SRL as "a series of reciprocally related cognitive and affective processes that operate together on different components of the information processing system"[9, p.3]. The authors identify 2 key aspects of self-regulation:

1. regulation of the learning process
2. use of metacognitive knowledge and skills to direct one's learning

Keeping in line with these key aspects, my research aims to regulate the learning process and direct the student learning process with knowledge and feedback in the form of an SRL dashboard that includes the sense-making factors and support for action for SRL.

3

Implementation

This chapter outlines the requirements analysis, design and implementation of the Computational Thinking (CT) assessment framework developed for this research. Firstly, the requirements are analysed in terms of the research gap and based on the requirements analysis in section 3.1. Then the design of the framework is described in section 3.2. Section 3.3 then lays down the details of the implementation.

For the design of the CT assessment framework, the software engineering process is used. The waterfall model is followed wherein each phase follows the other in a linear way[48]. Each phase is started after the previous one is completed, just like a waterfall. The main phases of this cycle are:

1. Requirements Analysis: Based on the research gaps in the existing literature and the platform analysis, the requirements for the product are collected in this phase
2. Design: The design of the product and the features is done in this phase
3. Implementation: Based on the outcomes of the previous phases, the product is implemented
4. Verification: The implemented product is evaluated with a user study, further explained in chapter 4

3.1. Requirements Analysis

To collect the requirements, the course setup is first analysed and then the research gaps based on the literature study from chapter 2 are analysed. The requirements are then grouped into functional and non-functional ones.

3.1.1. The Python basic programming course

The CT assessment framework is integrated into the the Python basic programming. This course is a self-paced course to teach Python programming without any pre-requisite knowledge to university students. It is comprised of 4 modules -

1. Variables
2. Control flow
3. Code Organization
4. Basic plotting

The course is based on Jupyter notebooks to allow for active leaning and experimentation and uses nbgrader[1] for releasing the exercises. The code in these notebooks is runnable, producing output, and can be modified by the student, so as to learn all the details and study the effects of changes and variations.

3.1.2. Jupyter notebooks

As CT is being integrated into curricula at various levels, tools such as Jupyter notebook provide an easy interface to teach these concepts in an interactive way[27]. Jupyter notebooks allow creation of assignments while integrating instructions and code easily. Owing to the world-wide adoption of Python as a programming language for Science, Technology, Engineering, and Mathematics (STEM) and non-computer science courses, Jupyter notebooks have gained popularity as a easy-to-use tool. nbgrader[8] enables instructors to use Jupyter notebooks to grade assignments without excessive effort, thereby improving communication and

[1] https://nbgrader.readthedocs.io/en/stable/

the learning experience for students and instructors both. Two tools were found in the literature that create platforms for CT analysis but these cannot be integrated with courses that use Jupyter notebooks and use specific platforms. Farah et al. develop Code app - a web-based platform that uses online activity traces and surveys for a computational thinking course on the the Graasp open education platform[16]. Koh et al. proposed REACT (Real Time Evaluation and Assessment of Computational Thinking) to allow formative assessment of programming tasks in real-time[28]. However, not much has been done to integrate this analysis into Jupyter notebooks due to the lack of appropriate infrastructure for the same.

Another major shortcoming in Jupyter notebooks is the lack of support for continuous feedback. While it supports code checking and validation using nbgrader, there is no support for real-time feedback. Nbgrader allows creation of auto-graded assignments that can be released to the students by the instructor. The students downloads a copy of these assignments on their user environment and submits the assignment upon completion. The instructor then auto-grades these assignments and provides manual feedback if required. While this framework setup by nbgrader allows for auto-grading of assignments, real-time feedback cannot be provided.

3.1.3. Research gaps in CTA and research question

Based on the literature review in chapter 2 and the system analysis done above, this research identifies and addresses the following research gaps:

1. The need to map the complete trajectory of CT skills[46] by assessing CT skills among university students using Python.
2. Addressing the lack of usage of CT in classrooms[46] through easy integration of CTA that uses user micro-interaction logging in Python courses using Jupyter notebooks.
3. Addressing the downsides of online learning by applying SRL strategies in the form of an SRL dashboard that includes the sense-making factors investigated by Jivet et al.[23]

These research gaps lead to the following research question: *How can computational thinking be assessed through detection of user micro-interactions in a university-level self-paced Python beginners course integrated into Jupyter notebooks?*

3.1.4. Requirements

Requirements can be classified as functional requirements and non-functional ones. The functional requiremens describe the fundamental and essential subject matter of what the product does. On the other hand, the non-functional requirements are properties that the product must possess such as data privacy and usability. Based on the requirement analysis in the previous section, the requirements for the CT assessment framework are grouped into the following functional and non-functional requirements.

The functional requirements for the Computational Thinking (CT) assessment framework are:

1. The user should learn about the Python programming language and underlying transferable CT skills.
2. The user should be able to learn interactively by experimenting live with the code snippets and seeing the corresponding outputs.
3. The CT assessment should be easy to integrate into Jupyter notebooks for any Python beginners course.
4. The CT assessment should provide feedback based on user micro-interaction data in the form of a dashboard to help the user improve their CT skills.
5. The user should be able to track their progress and gain actionable feedback through the CT dashboards.

The non-functional requirements for the Computational Thinking (CT) assessment framework are:

1. The user should be able to access the Python beginner course without doing any setup of the environment or the required libraries.
2. The user data should be anonymized before it is sent to the server to protect data privacy.
3. The user should be able to access the learning materials from any device connected to the internet at any hour of the day.
4. The CT dashboards should be easy to understand and use.

3.2. Design

3.2.1. Adapted definition of Computational Thinking

For this research, an adapted definition of Computational Thinking(CT) that combines those by Brennan and Resnick[10] and Yeni and Hermans [54] is used. Brennan and Resnick define CT for Scratch with 3 key dimen-

sions: “computational concepts (the concepts designers employ as they program), computational practices (the practices designers develop as they program), and computational perspectives (the perspectives designers form about the world around them and about themselves)”[10, p.3]. Yeni and Hermans adapt this definition to Python by modifying the CT concepts list to one that is better suited to Python. The CT concepts used by Yeni and Hermans are:

1. Data structures
2. Operators
3. Conditionals
4. Sequences
5. Loops
6. Functionals

Visualization, also referred to as ‘Simulation’ or ‘Modelling’ is an important CT concept that is missing in the above definition. Hambrusch et al. develop a course to introduce CT to science majors and they believe that although visualization is an important aspect of computing, it has not been effectively used in teaching CT concepts[19]. The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) define CT as a problem-solving process that includes representation of data via abstractions - models and simulations[47]. Barr and Stephenson[6] include simulation in their definition of CT for K-12. Weintrop et al. and Yuen and Robbins consider a data-based approach to CT and find that there is a link between the computational and visualization tasks[49][56]. Thereby, based on the structure of the Python programming course and relevant literature, my research adds ‘Visualizations’ to the list of CT concepts proposed by Yeni and Hermans. Thereby, the revised list of CT concepts used in my research is, along with their definitions are:

1. Data structures: Data involves storing, retrieving, and updating values
2. Operators: Operators provide support for mathematical, logical, and string expressions
3. Conditionals: Conditionals allow making logical decisions based on a condition
4. Sequences: Sequences are an activity/task expressed as a series of individual steps
5. Loops: Loops allow repetition of a statement multiple times based on a condition
6. Functionals: Functionals are blocks that a program is divided into
7. Visualizations: Visualization involves viewing data using graphs or charts

Brennan and Resnick[10] identify 4 CT practices as part of their CT definition, which are identified by micro-interactions (explained in section 3.3.1):

1. Being Incremental and Iterative
2. Testing and Debugging
3. Reusing and Remixing
4. Abstracting and Modularizing[10]

My research uses these 4 CT practices and detects them through the user’s micro-interactions. This is further explained in section 3.3.1.

3.2.2. CT concepts mapping

This research uses 7 CT concepts and maps them to the 4 learning modules in the Python basic programming course. This mapping is used for the design of the module-wise dashboards described in section 3.3.3. The CT concepts are mapped to the learning modules as shown in Table 3.1.

Table 3.1: Module mapping to CT concepts

Module	CT concepts
Variables	Data, Operators
Control flow	Loops, Conditionals
Code Organization	Sequences, Functionals
Basic plotting	Visualization

3.2.3. Software architecture

For this study, 2 servers are required as per the design requirements. The software architecture is shown in Figure 3.1. The first server is the JupyterHub server where the Jupyter notebook files are hosted as assignments using nbgrader[8]. This server is hosted on Amazon Web Services (AWS) Elastic Compute Cloud (EC2)

on an Ubuntu server and contains a build of The Littlest JupyterHub (TLJH) [2]. The 4 learning modules are hosted here along with the LogUI server configuration files, nbgrader library CT dashboards and the LogUI interaction logs[1]. The link to this server was sent to all the registered users for the user evaluation study. The second server is the LogUI server that is also hosted on AWS EC2 on an Ubuntu server. It contains a dockerised image of the LogUI server and its components. The address to this server was configured into the LogUI server configuration file in the JupyterHub. The link to the JupyterHub was configured in the client configuration here correspondingly.

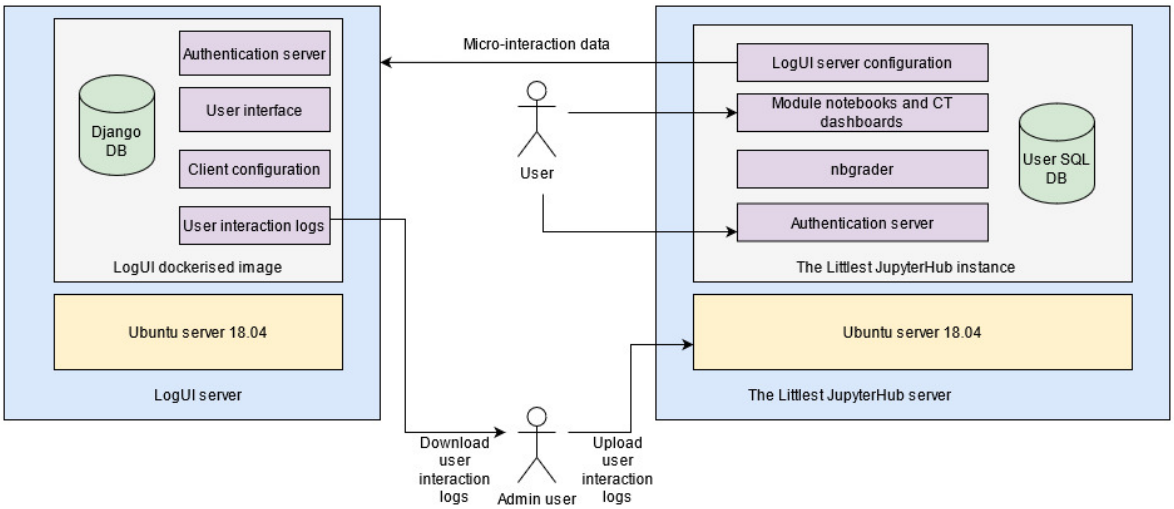

Figure 3.1: System architecture

3.2.4. User activity flow

The flow of activities for the user is as shown in Figure 3.2. The user begins by filling in the pre-evaluation survey and logging in to the JupyterHub server. They then fetch the modules from the server as assignments and complete the learning modules one at a time. The CT concepts dashboard is to be viewed after each module and provides feedback about whether the progress is satisfactory and if the module needs to be repeated. Once the user completes all the modules, they view the global CT practices dashboard for further overall feedback. Following that, the user fills in the post-evaluation survey to assess their CT skills after learning basic programming.

3.3. Implementation

3.3.1. Micro-interactions

Micro-interactions are the small-scale interactions that the user does with a platform such as keypresses, mouse button presses, copy and paste, etc. They can be useful to track the user behavior in real-time and provide feedback about their learning process. Micro-interactions can be aggregated and grouped to provide learning indicators that can help users with self-regulation of their learning process[34]. This research collects micro-interaction data and processes them to form indicators of CT skills. There are two sources of the micro-interaction data - LogUI and notebook metadata.

LogUI[35] is a framework-agnostic client-side JavaScript library developed by Maxwell and Hauff for logging user interactions on webpages. This research uses LogUI integrated into Jupyter notebooks together with Jupyter notebook metadata to detect micro-interactions such as the time spent on a cell, copy and paste. These micro-interactions are then aggregated to learning paramaters. For example, the number of copy-paste actions can indicate reuse of code in learning. These micro-interactions are then used as input for a global SRL dashboard.

The second source of micro-interaction data is the metadata from Jupyter notebooks. Jupyter notebook stores its cells as an array of JavaScript Object Notation(JSON) objects. This contains metadata about the

[2] https://tljh.jupyter.org/en/latest/

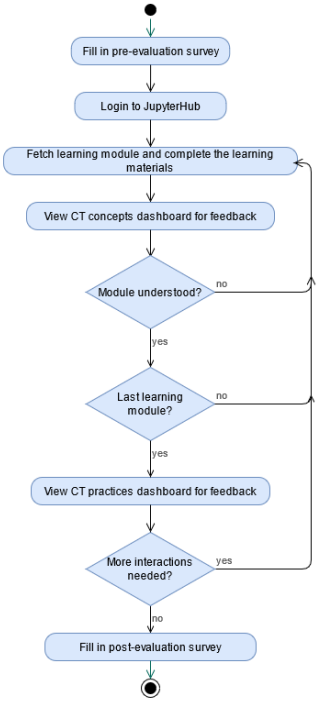

Figure 3.2: User flow diagram

number of cell runs, errors, cell source and more. This is processed to obtain some of the micro-interaction parameters.

3.3.2. Mapping of micro-interactions to CT practices

A combination of LogUI and notebook metadata is used to track micro-interactions.

The CT practices are defined as:

1. Being Incremental and Iterative: This is an adaptive process involving approaching a solution in small steps and working on it iteratively
2. Testing and Debugging: The practice of developing strategies for anticipating and dealing with problems
3. Reusing and Remixing: The practice of building upon existing code to help users find ideas to build upon
4. Abstracting and Modularizing: It is the practice to the building something large by putting together collections of smaller parts

These CT practices are mapped to micro-interactions as shown in Table 3.2. 'focusin' and 'focusout' refer to the user entering and exiting a code cell respectively. The difference between these two micro-interactions is used to compute the time spent on a cell. The count of keystrokes, cell runs in a notebook, errors in the output, new functions added and modules imported is used for their respective actions. 'copy' and 'paste' refers to the number of times a user copies and pastes something from and to a code cell respectively. Each of these micro-interactions is recorded per notebook and then analyzed.

3.3.3. SRL Dashboards

Dashboards are tools that support both students and teachers by helping them make sense of the learning analytics data such that it can be used to improve the learning process[23]. Dashboards can be used to trigger learners to think about the effort invested in learning and the subsequent outcomes of these activities[22]. Dashboards are used in this research to provide feedback to students per module and also about how the micro-interaction data can be used to improve the learning process. In this way, the students can regulate their learning process themselves.

Table 3.2: Micro-interactions mapping to CT practices

Micro-interaction	Action	Source	CT practice
focusin + focusout	Time spent on a cell	LogUI	Being Incremental and Iterative
keystrokes	additions	LogUI	Being Incremental and Iterative
Cell run count	-	Notebook metadata	Testing and Debugging
Errors in output	Errors	Notebook metadata	Testing and Debugging
copy	Copying from the notebook	LogUI	Reusing and Remixing
paste	Pasting to the notebook	LogUI	Reusing and Remixing
adding new functions	-	Notebook metadata	Abstracting and Modularizing
module import	-	Notebook metadata	Abstracting and Modularizing

3.3.4. CT concepts dashboard

The user is provided feedback for self-regulated learning via Computational Thinking(CT) concepts dashboards per module. This dashboard uses metadata tags for the cells and checks the completion using certain conditions. Additionally, the user is provided with actionable suggestion for iterative self-regulated learning, as shown in Figure 3.3.

Figure 3.3 shows the dashboard for the module 'Code Organization', covering 2 CT concepts - Sequences and Functionals. The progress of each concept is shown by a progress bar. This progress is computed by the ratio of the number of cells tagged with a concept that have been completed by the user against the ratio of the total cells tagged with a concept, scaled to a CT concept score of 1-10. The color of the progress bar is red if the progress is less than 60% of this ratio, as can be seen for the concept Functionals. The user is advised to revisit the module if the progress bar is red or else proceed to the next one. This way, the user can track their progress and can decide their next step based on quantitative data.

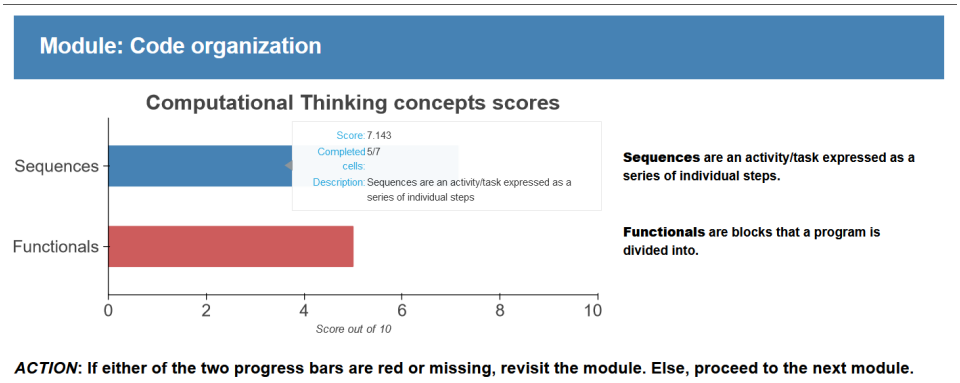

Figure 3.3: Module-wise CT dashboard

3.3.5. CT practices dashboard

The micro-interactions of the user are tracked using LogUI and notebook metadata and are mapped to the CT practices, as per Table 3.2. These are shown in 4 sections corresponding to the CT practices and each micro-interactions is displayed module-wise. A screenshot of the dashboard for one of the CT practices is shown in Figure 3.4.

3.3.6. Integration and Reproducibility

The framework created for CT assessment in this research can be integrated and reproduced easily for any Python beginners course that uses Jupyter notebooks. The detailed instructions can be found on the Github repository[1]. The steps to reproduce this CT assessment are:

1. Setup a LogUI server following the documentation[3]
2. Add metadata tags to the course cells

[3]https://github.com/logui-framework/server

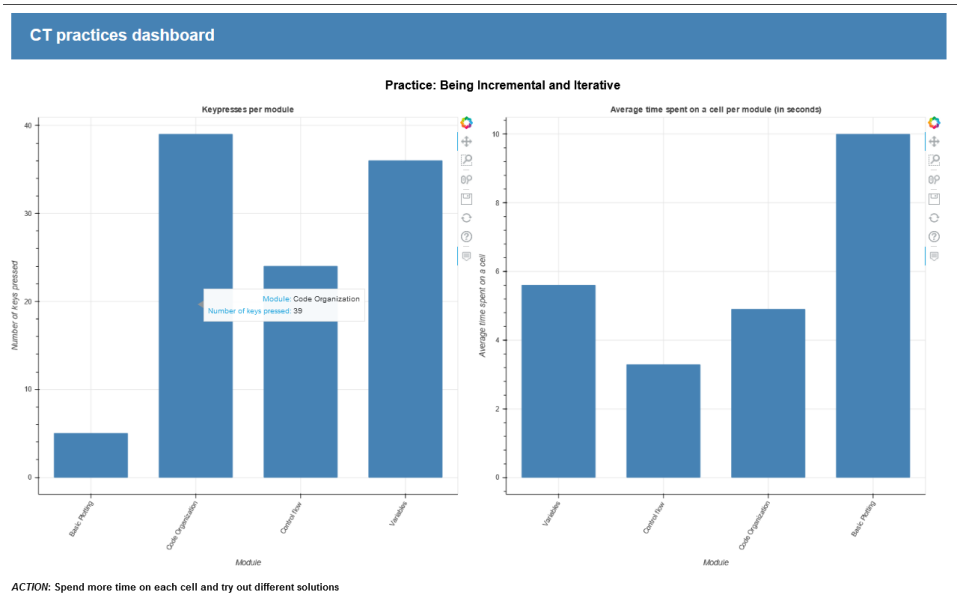

Figure 3.4: Global CT practices dashboard

3. Add the code for logging the micro-interactions into each notebook
4. Add the LogUI client files and configure the LogUI server link and authorisation token (follow LogUI client instructions[4])
5. Add the CT concepts dashboards to the modules and the overall CT practices dashboard(user ID to be configured here)

[4]https://github.com/logui-framework/client

4

Methods

To test the effectiveness of the Computation Thinking Assessment (CTA) framework developed, a user evaluation study was conducted for a period of length of 20 days.

4.1. Participants

25 participants signed up via an open call for participation, out of which 48% of the participants (12 participants) completed the study. Among the 13 participants who dropped out, 5 logged in but did not make much progress due to time constraints while 8 of them did not log in to the JupyterHub server at all. Only the 12 participants who completed the course are considered for further results and conclusions, thereby setting the sample size to 12. To ensure the data privacy of the participants, the results presented have been anonymized.

The characteristics of the participants who completed the study are quite equalized, as can be seen in Table 4.1.

Table 4.1: Participant characteristics, Sample size = 12

Characteristic	Values	Participant count
Gender	Female	5
	Male	7
Age range	20 to 25	5
	25 to 30	7
Highest education level	Bachelors degree	5
	Masters degree	7
Current domain of work or study	Computer science	2
	Non-Computer science STEM	7
	Non-STEM	3
Prior Python programming experience	1	5
	2	3
	3	3
	5	1
	6	1

These participant characteristics help establish the target audience for the CT assessment framework and for the Python basic programming course. From Table 4.1, it can be seen that the gender ratio is quite equalized. Table 4.1 shows the age range of the users - with all of them falling in the age range of 20-30 years. This age range corresponds to the approximate age range of university students. Table 4.1 further establishes the target audience as university students by considering the current or highest level of education of the users - finding all of them to be either currently pursuing or completed a Bachelors' degree or a Masters' degree. As CT skills are transferable, it is also important to consider the current domain of work or study of the users to ensure that the participants are from different domains. Table 4.1 shows the domain distribution of the

users to be quite wide-spread across non-CS Science, Technology, Engineering and Mathematics (STEM) and non-STEM domains.

As part of the call for participation, the participants were asked to report their prior Python programming experience on a scale of 1-10, with 1 signifying 'no knowledge' and 10 signifying 'master'. This self-reported prior Python programming experience of users (on a scale of 1-10) is shown in Table 4.1. 2 of the participants have moderate prior Python programming experience while 10 of them have no knowledge to little knowledge. Based on these characteristics, the user evaluation study considered participants who are beginners to Python programming at the university level from different domains.

4.2. Materials

The CTA framework is integrated into the Python basic programming course as described in subsection 3.1.1. This course comprises of 4 modules - variables, control flow, code organization and basic plotting. Each module has learning materials and the CT concepts dashboards. There is a global CT practices dashboard in the main folder. The course uses Jupyter notebooks and nbgrader as the main tools for the environment setup. As decribed in section 3.2.3, a JupyterHub server (hosted on Amazon EC2 on an Ubuntu server) was set up to host all the user data, learning materials and dashboards. Each participant had an independent login and local storage on this server where their progress was stored.

4.3. Procedure

First, participants were asked to sign up by filling out an open call for participation through a survey. Then, a user login was created to each participant and communicated to them via email on the day that the user evaluation study began. The participants were then asked to fill in the pre-evaluation survey before logging in to the server for the first time. Once they logged into the server and completed the course materials and followed the feedback from the CT dashboards, they were asked to fill in the post-evaluation survey as the last step.

An experimental design is used to measure the improvement in CT skills of participants before and after taking the Python basic programming course with the CTA framework integrated. The self-reported CT skills of users before and after taking the course are the dependent variable. A within-subjects design is chosen to measure the change in CT skills of each participants before and after taking the Python basic programming course. Based on this experimental design, the null hypothesis H_0 and alternate hypothesis for the experiment H_1 respectively are:
H_0: There is no difference in the CT skills users before and after taking the course
H_1: There is a difference in the CT skills of users before and after taking the course
These hypotheses are tested through 3 sub-research questions in Section 4.4.

To measure the effect of the user evaluation study on CT skills of the participants, a pre-evaluation survey was conducted before taking the course and a post-evaluation survey was conducted after taking the course. Both of these surveys have the same 24 questions with a five-point Likert scale (1=Strongly Disagree, 2=Disagree, 3=Neutral, 4=Agree, 5=Strongly Agree). The scores between the two surveys are compared to see the change in CT skills. Additionally, the participants are asked 5 questions in the post-evaluation survey to evaluate the usability of the SRL dashboards. The questions in the surveys are presented in Appendix A. The survey used is an adapted version of the survey created by Kılıç, Gökoğlu, and Öztürk[26]. This survey is used as it has been designed to evaluate the programming-oriented CT skills at the university level. As my research associates programming concepts with CT skills, it was necessary to find a survey scale that measures this correspondence same. This survey was found to be the best-suited to this purpose.

4.4. Results

This research aims to answer the research question: *How can computational thinking be assessed through detection of user micro-interactions in a university-level self-paced Python beginners course integrated into Jupyter notebooks?*
To answer this research question, the results of the study are analysed under 3 research sub-questions :

1. *RQ1: Did the users acquire Computational Thinking (CT) skills in the form of both CT concepts and CT practices?*
2. *RQ2: Was there a significant improvement in the self-reported CT skills of users after taking the Python basic programming course?*

3. *RQ3: How do the self-reported survey responses correspond to the actual user micro-interaction data?*

To check for the usability and accessibility, the users are asked 5 questions in the post-evaluation survey. These questions are a paraphrased version of the short version user-experience questionnaire UEQ-S developed by Schrepp, Hinderks, and Thomaschewski and adapted to this research[41]. These questions are used to measure the experience of the users towards the CT dashboards developed in this framework.

The number of participants who selected each of the 5-point Likert values per question is shown in Table 4.2. The mode values - the most frequent responses - for each of the questions are also shown in Table 4.2. These modal values also correspond to the median values, that is, the central values. Based on the modal values and the distribution of answers in these survey feedback questions, it can be concluded that the CT dashboards were usable and accessible.

Table 4.2: Survey values and modal values for CT dashboard feedback questions

Question	**Strongly Disagree**	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**	**Modal value**
I found the dashboards easy to use	0	0	3	5	4	Agree
I could follow the course materials	0	0	4	6	2	Agree
The dashboards helped me track my learning progress	0	0	2	4	6	Strongly Agree
The dashboards gave me valuable feedback and action points	0	0	1	6	5	Agree
The dashboards were transparent and easy to understand	0	0	1	4	7	Strongly Agree

4.4.1. RQ1: Did the users acquire Computational Thinking (CT) skills in the form of both CT concepts and CT practices?

For RQ1, the mapping of the survey questions from Kılıç, Gökoğlu, and Öztürk[26] to the survey questions used by this research is examined. Kılıç, Gökoğlu, and Öztürk divides the questions into 3 subscales: conceptual knowledge, algorithmic thinking, and evaluation[26]. Conceptual knowledge is defined as basic programming structures and data concepts and the ability to use them effectively[26] - mapping to the CT concepts in the adapted definition of CT presented in Section 3.2.1. Question 15-24 presented in Appendix A thereby maps to CT concepts and the questions 1-14 map to the CT practices. To analyse the acquisition of self-reported CT skills, the post-evaluation survey was used. The count of each of the options of the Likert scale was aggregated per question for the 12 completed users, as shown in Figure 4.1. Following this, the mean (taken by encoding the Likert option values) and standard deviation (SD) was computed per question to get the final score per question, shown in Figure 4.1. Then, the average value of the mean for the CT concepts questions(15-24) and CT practices questions(1-14) was computed and was found to be 4.35 and 4.27 respectively. The standard deviation for the CT concepts questions(15-24) and CT practices questions(1-14) are both found to be in the range of 0.53-1.13, signifying a short deviation from the average value. Based on these values, it can be concluded that the users acquired CT skills in the form of both CT concepts and CT practices.

4.4.2. RQ2: Was there a significant improvement in the self-reported CT skills of users after taking the Python basic programming course?

To analyze the change in the self-reported CT skills of users before and after the course, 2 analysis methods were used - a data-based approach and a statistical approach. In the data-based approach, the change in the survey results was analyzed per question and per user. The difference between the Likert-style responses in the post-evaluation survey and pre-evaluation survey was calculated per question for each user. This change in the survey answers can be seen in Figure B.1 in Appendix B.

The change values were further grouped as follows:

- Large Increase = 4 or 3
- Increase = 2 or 1
- No change = 0

The breakdown per user using this grouping can be seen in 4.2. The percentage increase per user according to this grouping is broken down in Table 4.3. As can be seen in this table, most of the users have a higher

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Mean	SD
I can detect errors related to coding on a subject I know.	0	0	1	6	5	4.33	0.82
I can detect and clean unnecessary code structures in a program.	0	0	4	5	3	3.92	0.58
I can detect the similarities and differences between two different programs.	0	0	3	5	4	4.08	0.59
I can understand how the result changes when different values assigned to the variables in the program.	0	0	0	4	8	4.67	1.13
I can continue the coding on a subject where I left off.	0	0	0	3	9	4.75	1.28
I can interpret the causes of the errors I encountered during the coding process.	0	0	2	4	6	4.33	0.75
I can detect errors in syntax (if, for, operators, etc.)	0	0	3	4	5	4.17	0.60
I can decide on the operations that will create my code structures or code blocks in the program	0	0	2	5	5	4.25	0.68
I can break a problem into smaller parts and work on them independently.	0	0	2	5	5	4.25	0.68
I can fix errors in the mathematical operations of the program.	0	1	0	6	5	4.25	0.85
I can understand an existing program and reuse it in my code.	0	0	1	5	6	4.42	0.84
I can detect and fix a logical error in the flow of a program.	0	0	4	4	4	4.00	0.53
I can sort the solution steps of the complex program appropriately.	0	0	2	6	4	4.17	0.71
I can visualize the processing steps of the program in my mind before I start coding.	0	0	2	5	5	4.25	0.68
I can use the loop structures (for, while, etc.) appropriately.	0	0	1	6	5	4.33	0.82
I can use the decision structure (if-else, switch-case) appropriately	0	0	2	4	6	4.33	0.75
I can determine the suitable data type (string, int, char, float, etc.) for a variable.	0	0	2	3	7	4.42	0.88
I can use mathematical (,=, == etc.) and logical operators (and, or, etc.) appropriately.	0	0	1	3	8	4.58	1.08
I can use general methods of programming languages [write(), read(), wait(), move(), scanf(), printf(), etc.]	0	0	3	5	4	4.08	0.59
I can code programs in which decisions (if-else, switch-case) and loops (for, while) can be used together.	0	0	0	6	6	4.50	0.98
I can make sections that require mathematical operations in the program.	0	0	0	7	5	4.42	0.99
I can determine which of the similar structures (if-switch, for-while) would be more appropriate.	0	0	1	5	6	4.42	0.84
I know which variables I would use before I start coding.	0	0	2	4	6	4.33	0.75
I can decide how to split complex programs into smaller pieces.	0	0	2	7	3	4.08	0.83

Figure 4.1: Mean and SD values per question

percentage of increase than no change, showing that there was an improvement in the CT skills of users after taking the Python basic programming course.

To further analyze the significance of the change in CT skills before and after the Python basic programming course, a statistical approach is also used by conducting a paired t-test for the population. A paired t-test is a statistical test used to compare the difference between a pair of variables (called dependent variables) for the same subject. The paired t-test was done by considering the average of the responses in the pre-evaluation survey for each user and the average of the responses in the post-evaluation survey for each user as the pair of dependent variables. The null hypothesis H_0 and alternate hypothesis H_1 respectively are:
H_0: There is no difference in the self-reported CT skills users before and after taking the course
H_1: There is a difference in the self-reported CT skills of users before and after taking the course

The significance level α is set to a value of 0.05. If the two-tailed $p - value < 0.05$, the null hypothesis H_0 is rejected. As seen in Figure 4.3, the p-value is less than α. Thereby, the null hypothesis H_0 is rejected for the group - showing a significant improvement in self-reported CT skills. Based on the above results, it can be concluded that there is a significant change in the self-reported CT skills of users before and after taking the course.

4.4.3. RQ3: How do the self-reported survey responses correspond to the actual user micro-interaction data?

To answer RQ3, the average scores of the increase in self-reported CT skills were computed and compared to the user micro-interaction data and dashboard usage data. The average of the change in the self-reported CT skills of users is computed for the CT concepts as CTC_S_avg and the CT practices as CTP_S_avg by taking the average of the numbers presented in Figure B.1 in Appendix B. The change is score is computed by taking the difference per question and per user between the post-evaluation survey and the pre-evaluation survey. Then, the average score for this change is obtained by averaging over the CT concept questions and CT practices questions for each user. The scale for these scores is thereby 0-4 as the maximum possible change is 4.

Table 4.3: Breakdown of user-wise CT skills change percentage pre-post

User	Increase group	Percentage value
User 1	Large Increase	87.5%
	Increase	12.5%
User 2	Large Increase	4.2%
	Increase	45.8%
	No Change	50%
User 3	Large Increase	8.3%
	Increase	91.7%
User 4	Increase	70.8%
	No Change	29.2%
User 5	Large Increase	66.7%
	Increase	33.3%
User 6	Large Increase	75%
	Increase	25%
User 7	Large Increase	91.7%
	Increase	8.3%
User 8	Large Increase	87.5%
	Increase	12.5%
User 9	Large Increase	8.3%
	Increase	70.8%
	No Change	20.9%
User 10	Increase	50%
	No Change	50%
User 11	Large Increase	70.8%
	Increase	29.2%
User 12	Large Increase	62.5%
	Increase	37.5%

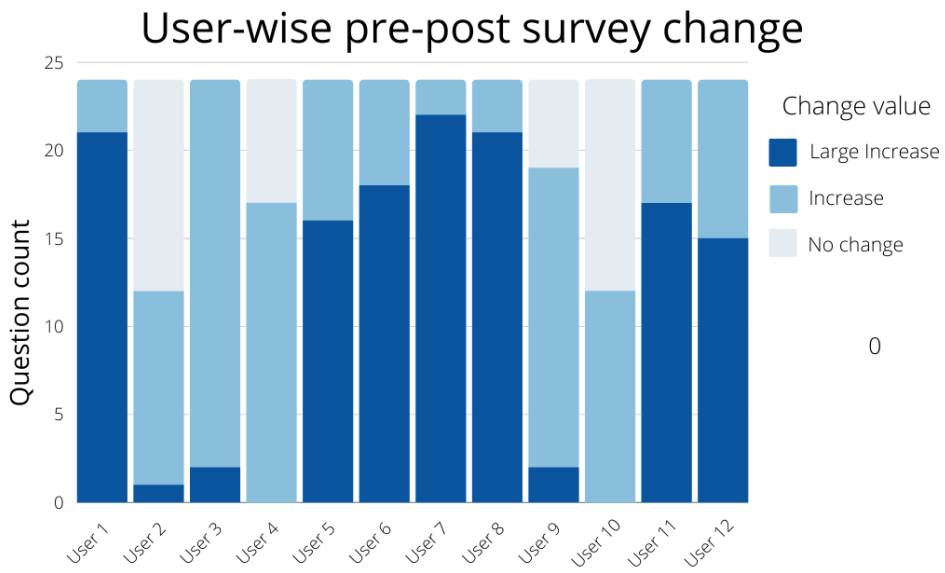

Figure 4.2: Breakdown of user-wise CT skills change pre-post

Paired t-test

Sample size	12
Difference Mean	2.18403
t	7.09089
Df	11
P-value (one-tail)	1.00839e-05
P-value (two-tail)	2.01679e-05
Lower 95.0%	1.50611
Upper 95.0%	2.86194

Figure 4.3: Paired t-test result

To compare this to actual user interaction data, the CT concepts dashboard scores for each CT concept(on a scale of 0 to 10) is averaged to get the CT concepts average score as CTC_DB_avg. The CT concept dashboards contain a score for each CT concept on a scale of 1-10. This score per concept is averaged to get a single score so that the comparison is easy. The usage of the CT dashboards is measured as the average of the number of times the CT concepts dashboard is run by each user as CTC_DB_runs. The number of runs for the dashboard of each module is averaged over the 4 modules to get this score. For the CT practices, the user-wise scores for each of the micro-interactions in the CT practices is averaged over the modules to get a single average score for each micro-interaction as CTP_DB_avg.

This data is presented in Table 4.4. The columns in this table are:

- User: User ID
- CTC_S_avg: Average change in the self-reported CT concepts of user as reported in the survey (averaged over questions 15-24 of survey questions in Appendix A)
- CTP_S_avg: Average change in the self-reported CT practices of user as reported in the survey (averaged over questions 10-14 of survey questions in Appendix A)
- CTC_DB_avg: Average of the CT concepts dashboard scores of the user for each CT concept (averaged over the 4 module-wise CT concepts dashboards)
- CTC_DB_runs: Average number of times that the user runs each the CT concepts dashboard (averaged over the 4 module-wise CT concepts dashboards)
- CTP_DB_avg: Average score for each micro-interaction of the user in the global CT practices dashboard (averaged over the 4 modules in the dashboards)

As can be seen from Table 4.4.3, the change in CT skills reported by the users roughly corresponds to the user micro-interaction data. For example, User 1 reports a high change of 3.4 and 3.2 in CT concepts and CT practices and this is reflected accordingly in the high values of the average CT concepts dashboard scores and runs and the values of the CT practices dashboard. On the other end of the spectrum, low self-reported scores correspond to low values in the micro-interaction data. An example of such a user is User 2.

From the data in Table 4.4.3, it can be seen that Users 4 and 10 report a low change in the CT skills. These users have a pretty good prior knowledge of the Python programming language(5 out of 10 and 6 out of 10) and thereby did not gain much added value from the course. These users also score highly on the CTC_DB_avg, signifying a good knowledge of the programming constructs and spend quite less time on the course, as is seen in the low 'Time spent' and 'Cell runs' in the CTP_DB_avg.

Based on the correspondence between the self-reported survey responses and the actual user micro-interaction data, it can be concluded that they reflect quite strongly on each other, thereby implying honest responses to the survey questions.

Table 4.4: User micro-interaction scores and survey response changes

User	CTC_S_avg	CTP_S_avg	CTC_DB_avg	CTC_DB_runs	CTP_DB_avg
User 1	3.4/4	3.2/4	8/10	2	Keystrokes: 438 Time spent: 113s Cell runs: 21.75 Errors: 0 Copy: 1 Paste: 1 Functions: 1.5 Modules imported: 3
User 2	1/4	0.93/4	7/10	1	Keystrokes: 19 Time spent: 6.5s Cell runs: 9.75 Errors: 0.75 Copy: 0.5 Paste: 1 Functions: 1.25 Modules imported: 3
User 3	1.9/4	1.7/4	7.68/10	1.25	Keystrokes: 96 Time spent: 21s Cell runs: 8 Errors: 0.25 Copy:0.5 Paste: 1 Functions: 1.25 Modules imported: 3
User 4	0.7/4	1.07/4	8.32/10	1	Keystrokes: 176 Time spent: 44s Cell runs: 5.75 Errors: 0.75 Copy: 0.5 Paste: 2 Functions: 1.5 Modules imported: 3
User 5	3/4	2.57/4	7.9/10	2.25	Keystrokes: 590 Time spent: 106s Cell runs: 32 Errors: 0 Copy: 1 Paste: 8 Functions: 2 Modules imported: 3
User 6	3/4	2.57/4	8.88/10	2	Keystrokes: 145 Time spent: 114s Cell runs: 10.75 Errors: 0.5 Copy: 0.5 Paste: 3 Functions: 1.25 Modules imported: 3

Continued on next page

Table 4.4 – *Continued from previous page*

User	CTC_S_avg	CTP_S_avg	CTC_DB_avg	CTC_DB_runs	CTP_DB_avg
User 7	3.3/4	3.29/4	8.84/10	2	Keystrokes: 139 Time spent: 104s Cell runs: 10.25 Errors: 0.5 Copy: 1.25 Paste: 1 Functions: 1.5 Modules imported: 3
User 8	3.5/4	3.36/4	9.5/10	2.5	Keystrokes: 488 Time spent: 221s Cell runs: 22 Errors: 0 Copy: 1 Paste: 1 Functions: 2 Modules imported: 3
User 9	1.3/4	1.29/4	7.84/10	1	Keystrokes: 28 Time spent: 37s Cell runs: 15.25 Errors: 0.5 Copy: 1 Paste: 1 Functions: 1.25 Modules imported: 3
User 10	0.7/4	0.43/4	7.63/10	1	Keystrokes: 54 Time spent: 14s Cell runs: 8.75 Errors: 0.5 Copy: 2.25 Paste: 1.75 Functions: 1.5 Modules imported: 3
User 11	2.6/4	3/4	8.22/10	2.25	Keystrokes: 958 Time spent: 119s Cell runs: 17.75 Errors: 0 Copy: 4.75 Paste: 8.25 Functions: 1.25 Modules imported: 3
User 12	2.79/4	2.3/4	7.8/10	3	Keystrokes: 92 Time spent: 61s Cell runs: 12.75 Errors: 0.25 Copy: 0.5 Paste: 1 Functions: 1.25 Modules imported: 3

5

Conclusion

This research aimed to answer the research question - *How can computational thinking be assessed through detection of user micro-interactions in a university-level self-paced Python beginners course integrated into Jupyter notebooks?* To answer this research question, a framework for computational thinking (CT) assessment using detection of micro-interactions was developed and integrated in a university-level self-paced Python beginners course in Jupyter notebooks. A user evaluation study is conducted to show that this framework can be used to improve the acquisition of CT skills via programming.

To assess CT, a combination of a survey and portfolio assessment method are used in this research. The portfolio assessment is done by detecting user micro-interactions and using them as indicators of CT - providing a holistic view of the users' CT skills. As the portfolio assessment cannot capture the users' attitudes towards learning and affective outcomes, a survey is used before and after the programming course to assess these. The results of the user evaluation study are summarised here. Firstly, the accessibility and usability of the CT dashboards was found to be sufficient through feedback questions in the post-evaluation survey. To test the acquisition of CT skills by the users, the mean and standard deviation of the post-evaluation survey responses was calculated over all users and it was found that the users acquired CT skills in the form of both CT concepts and CT practices. The improvement in the self-reported CT skills of users after taking the course was computed by comparing the self-reported survey scores before and after taking the course. The significance of this change in self-reported CT skills was computed using a paired t-test and it was concluded that there is a significant change in the CT skills of users before and after taking the course. Lastly, the correspondence between the self-reported survey responses and the actual user micro-interaction data was checked and found to have a strong correspondence across a majority of the users. Thus, the results show an improvement in CT skills of the users and an accurate assessment of the same through this framework.

The results of the user evaluation study show that the developed framework for computational thinking (CT) assessment using detection of micro-interactions can be easily integrated in a university-level self-paced Python beginners course in Jupyter notebooks and this framework is effective in improving CT skills among users. In addition, a mapping of CT skills to the micro-interactions is developed in this research and this is used to create CT dashboards that provide feedback for self-regulation to users. The guidelines to integrate this framework easily in other introductory Python programming courses using Jupyter notebooks are provided in section 3.3.6.

As with every research, this one has its limitations too. There are 3 main limitations. Firstly, the results of the micro-interactions logging and the dashboard are not available to the user in the form of the global CT practices dashboard at all points of time. As the logging library - LogUI - is still in the development phase, it does not currently have the functionality to stream or access the user interaction logs in real time. The logs for all users and all sessions are appended to a single log file that can be download from the LogUI server and uploaded to the JupyterHub server when a user would like to view the CT practices dashboard. This is not a major issue for a small number of users as the logs can be downloaded at regular intervals of time or on-demand when a user would like to view the CT practices dashboard after completing an iteration of the course. However, this could cause issues in scaling as the number of users increases. The LogUI development team is currently working to resolve this issue and implement this functionality

The second limitation is that the auto-graded entrance test and exit test modules could not be integrated due to a currently unresolved issue with Jupyter client[24]. Future versions or re-runs of the course could

include these modules when the issue with Jupyter client is resolved.

The third limitation is that the assessment of self-regulated learning - Motivated Strategies for Learning Questionnaire (MSLQ)[39] - could not be fully integrated in this research owing to the time constraints of the user evaluation study. MSLQ is a self-reported questionnaire used to assess the cognitive view of motivations and learning strategies in a college course. Adding the MSLQ validation would help assess the self-regulated learning among students through this course. Owing to this limitation, the self-regulation aspect of this CT framework could not be fully assessed in this research.

In conclusion, a framework to assess CT skills was developed for a university-level self-paced Python beginners course and micro-interaction data was used to provide feedback to improve the acquisition of CT skills by the user. This framework can be integrated easily into other courses that teach CT skills through Python programming using Jupyter notebooks. While the user evaluation study conducted validates the CT assessment framework developed for a basic programming course, the results might differ for an advanced programming courses and courses that do not teach programming. Future work aimed at testing the applicability of this framework to other non-programming courses and to advanced programming courses should be carried out to validate the results of this CT assessment framework to them. In addition, integration of the MSLQ validation framework would enable validation of the complete theoretical design of this CT assessment framework.

A

Survey questions

The questions used in the pre-evaluation survey are:

1. I can detect errors related to coding on a subject I know.
2. I can detect and clean unnecessary code structures in a program.
3. I can detect the similarities and differences between two different programs.
4. I can understand how the result changes when different values assigned to the variables in the program.
5. I can continue the coding on a subject where I left off.
6. I can interpret the causes of the errors I encountered during the coding process.
7. I can detect errors in syntax (if, for, operators, etc.)
8. I can decide on the operations that will create my code structures or code blocks in the program
9. I can break a problem into smaller parts and work on them independently.
10. I can fix errors in the mathematical operations of the program.
11. I can understand an existing program and reuse it in my code.
12. I can detect and fix a logical error in the flow of a program.
13. I can sort the solution steps of the complex program appropriately.
14. I can visualize the processing steps of the program in my mind before I start coding.
15. I can use the loop structures (for, while, etc.) appropriately.
16. I can use the decision structure (if-else, switch-case) appropriately
17. I can determine the suitable data type (string, int, char, float, etc.) for a variable.
18. I can use mathematical (,=, == etc.) and logical operators (and, or, etc.) appropriately.
19. I can use general methods of programming languages [write(), read(), wait(), move(), scanf(), printf(), etc.]
20. I can code programs in which decisions (if-else, switch-case) and loops (for, while) can be used together.
21. I can make sections that require mathematical operations in the program.
22. I can determine which of the similar structures (if-switch, for-while) would be more appropriate.
23. I know which variables I would use before I start coding.
24. I can decide how to split complex programs into smaller pieces.

The post-evaluation survey contains all the questions in the pre-evaluation survey and 5 additional questions to evaluate the research. These 5 additional questions are:

1. I found the dashboards easy to use
2. I could follow the course materials
3. The dashboards helped me track my learning progress
4. The dashboards gave me valuable feedback and action points
5. The dashboards were transparent and easy to understand

B

Detailed results

B.1. User-wise change

	0	1	2	3	4	5	6	7	8	9	10	11
I can detect errors related to coding on a subject I know.	3	2	2	2	3	2	3	3	3	0	3	3
I can detect and clean unnecessary code structures in a program.	3	0	1	2	3	3	4	2	2	0	2	3
I can detect the similarities and differences between two different programs.	3	1	2	0	4	2	3	2	0	0	3	4
I can understand how the result changes when different values assigned to the variables in the program.	4	0	2	1	2	3	4	2	1	1	4	1
I can continue the coding on a subject where I left off.	4	1	1	2	3	3	4	4	3	0	4	3
I can interpret the causes of the errors I encountered during the coding process.	4	2	2	1	2	2	3	4	1	1	3	2
I can detect errors in syntax (if, for, operators, etc.)	2	1	1	1	2	3	4	3	0	0	3	3
I can decide on the operations that will create my code structures or code blocks in	3	1	2	1	3	2	3	4	2	1	2	3
I can break a problem into smaller parts and work on them independently.	2	3	2	0	3	3	3	4	2	1	3	3
I can fix errors in the mathematical operations of the program.	3	0	2	1	1	3	3	4	1	0	4	2
I can understand an existing program and reuse it in my code.	3	1	1	0	3	2	4	4	1	1	3	2
I can detect and fix a logical error in the flow of a program.	3	0	2	1	1	2	2	4	1	1	2	3
I can sort the solution steps of the complex program appropriately.	4	1	2	1	3	3	2	3	1	0	3	4
I can visualize the processing steps of the program in my mind before I start coding.	4	0	2	2	3	3	4	4	0	0	3	3
I can use the loop structures (for, while, etc.) appropriately.	2	0	2	0	2	3	4	3	2	0	3	2
I can use the decision structure (if-else, switch-case) appropriately	4	0	2	0	1	3	4	3	2	0	3	3
I can determine the suitable data type (string, int, char, float, etc.) for a variable.	3	0	1	0	4	3	4	4	0	0	2	1
I can use mathematical (,=, == etc.) and logical operators (and, or, etc.) appropriately.	3	0	1	0	3	3	3	4	1	1	2	2
I can use general methods of programming languages [write(), read(), wait(), move(), scanf(), printf(), etc.]	4	0	1	2	2	3	3	4	0	1	3	1
I can code programs in which decisions (if-else, switch-case) and loops (for, while) can be used together.	4	0	3	1	4	3	3	3	1	1	3	3
I can make sections that require mathematical operations in the program.	3	0	2	1	4	3	3	4	1	2	3	2
I can determine which of the similar structures (if-switch, for-while) would be more appropriate.	4	1	3	1	4	3	3	3	2	1	3	3
I know which variables I would use before I start coding.	4	1	2	2	3	3	3	4	2	1	2	3
I can decide how to split complex programs into smaller pieces.	3	2	2	1	3	3	3	3	2	0	2	3

Figure B.1: Change in user-wise CT skills pre-post

Bibliography

[1] Bhoomika Agarwal. *ct_dashboards*. URL: https://github.com/bhoom10/ct_dashboards.

[2] Sheikh Iqbal Ahamed et al. "Computational thinking for the sciences: a three day workshop for high school science teachers". en. In: *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*. Milwaukee, Wisconsin, USA: ACM Press, 2010, p. 42. ISBN: 978-1-4503-0006-3. DOI: 10.1145/1734263.1734277. URL: http://portal.acm.org/citation.cfm?doid=1734263.1734277 (visited on 06/08/2021).

[3] John M. Aiken et al. "Understanding student computational thinking with computational modeling". In: *AIP Conference Proceedings* 1513.1 (2013), pp. 46–49. DOI: 10.1063/1.4789648. eprint: https://aip.scitation.org/doi/pdf/10.1063/1.4789648. URL: https://aip.scitation.org/doi/abs/10.1063/1.4789648.

[4] Ana Paula Ambrósio, Cleon Xavier, and Fouad Georges. "Digital ink for cognitive assessment of computational thinking". In: *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE. 2014, pp. 1–7.

[5] Ana Liz Souto O Araujo et al. "Exploring computational thinking assessment in introductory programming courses". In: *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2017, pp. 1–9.

[6] Valerie Barr and Chris Stephenson. "Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?" In: *ACM Inroads* 2.1 (Feb. 2011), pp. 48–54. ISSN: 2153-2184. DOI: 10.1145/1929887.1929905. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/1929887.1929905.

[7] Nadia Benakli et al. "Introducing computational thinking through hands-on projects using R with applications to calculus, probability and data analysis". In: *International Journal of Mathematical Education in Science and Technology* 48.3 (2017), pp. 393–427. DOI: 10.1080/0020739X.2016.1254296. eprint: https://doi.org/10.1080/0020739X.2016.1254296. URL: https://doi.org/10.1080/0020739X.2016.1254296.

[8] Douglas S Blank et al. "nbgrader: A tool for creating and grading assignments in the Jupyter Notebook". In: *The Journal of Open Source Education* 2.11 (2019).

[9] Monique Boekaerts. "Self-regulated learning: where we are today". In: *International Journal of Educational Research* 31.6 (1999), pp. 445–457. ISSN: 0883-0355. DOI: https://doi.org/10.1016/S0883-0355(99)00014-2. URL: https://www.sciencedirect.com/science/article/pii/S0883035599000142.

[10] Karen Brennan and Mitchel Resnick. "New frameworks for studying and assessing the development of computational thinking". In: *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*. Vol. 1. 2012, p. 25.

[11] Jaclyn Broadbent and Walter L Poon. "Self-regulated learning strategies & academic achievement in online higher education learning environments: A systematic review". In: *The Internet and Higher Education* 27 (2015), pp. 1–13.

[12] John B Carroll et al. *Human cognitive abilities: A survey of factor-analytic studies*. 1. Cambridge University Press, 1993.

[13] National Research Council et al. *Report of a workshop on the scope and nature of computational thinking*. National Academies Press, 2010.

[14] National Research Council et al. *Report of a workshop on the pedagogical aspects of computational thinking*. National Academies Press, 2011.

[15] Eva Durall and Begoña Gros. "Learning Analytics as a Metacognitive Tool." In: *CSEDU (1)*. 2014, pp. 380–384.

[16] Juan Carlos Farah et al. “Bringing Computational Thinking to non-STEM Undergraduates through an Integrated Notebook Application”. In: *15th European Conference on Technology Enhanced Learning*. CONF. 2020.

[17] Marcos Román González. “Computational thinking test: Design guidelines and content validation”. In: *Proceedings of EDULEARN15 conference*. 2015, pp. 2436–2444.

[18] Roxana Hadad and Kimberly A Lawless. “Assessing computational thinking”. In: *Encyclopedia of Information Science and Technology, Third Edition*. IGI Global, 2015, pp. 1568–1578.

[19] Susanne Hambrusch et al. “A Multidisciplinary Approach towards Computational Thinking for Science Majors”. In: *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*. SIGCSE ’09. Chattanooga, TN, USA: Association for Computing Machinery, 2009, pp. 183–187. ISBN: 9781605581835. DOI: 10.1145/1508865.1508931. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/1508865.1508931.

[20] Kamini Jaipal-Jamani and Charoula Angeli. “Effect of robotics on elementary preservice teachers’ self-efficacy, science learning, and computational thinking”. In: *Journal of Science Education and Technology* 26.2 (2017), pp. 175–192.

[21] Ioana Jivet. “The Learning tracker: a learner dashboard that encourages self-regulation in MOOC learners”. In: (2016). URL: http://resolver.tudelft.nl/uuid:f6c2ede4-a4e3-4ff0-b681-b0d057854e3c.

[22] Ioana Jivet et al. “Awareness Is Not Enough: Pitfalls of Learning Analytics Dashboards in the Educational Practice”. In: *Data Driven Approaches in Digital Education*. Ed. by Élise Lavoué et al. Cham: Springer International Publishing, 2017, pp. 82–96. ISBN: 978-3-319-66610-5.

[23] Ioana Jivet et al. “From students with love: An empirical study on learner goals, self-regulated learning and sense-making of learning analytics in higher education”. In: *The Internet and Higher Education* 47 (2020), p. 100758. ISSN: 1096-7516. DOI: https://doi.org/10.1016/j.iheduc.2020.100758. URL: https://www.sciencedirect.com/science/article/pii/S1096751620300348.

[24] Jupyter. *Release 6.1.13 nbconvert failing with TypeError: coroutine object is not subscriptable | Issue 637 | jupyter/jupyter_client*. URL: https://github.com/jupyter/jupyter_client/issues/637.

[25] Dennis Kafura, Austin Cory Bart, and Bushra Chowdhury. “Design and preliminary results from a computational thinking course”. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. 2015, pp. 63–68.

[26] Servet Kılıç, Seyfullah Gökoğlu, and Mücahit Öztürk. “A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking”. In: *Journal of Educational Computing Research* 59.2 (2021), pp. 257–286.

[27] Thomas Kluyver et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows*. Vol. 2016. 2016.

[28] K. H. Koh et al. “Real Time Assessment of Computational Thinking”. In: *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2014, pp. 49–52. DOI: 10.1109/VLHCC.2014.6883021.

[29] Kyu Han Koh et al. “Towards the automatic recognition of computational thinking for adaptive visual language learning”. In: *2010 ieee symposium on visual languages and human-centric computing*. IEEE. 2010, pp. 59–66.

[30] Özgen Korkmaz, Recep Çakir, and M. Yaşar Özden. “A validity and reliability study of the computational thinking scales (CTS)”. In: *Computers in Human Behavior* 72 (2017), pp. 558–569. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2017.01.005. URL: https://www.sciencedirect.com/science/article/pii/S0747563217300055.

[31] Agah Tugrul Korucu, Abdullah Tarık Gencturk, and Mustafa Mucahit Gundogdu. “Examination of the computational thinking skills of students”. In: *Journal of Learning and Teaching in Digital Age* 2.1 (2017), pp. 11–19.

[32] Jacqueline Leonard et al. “Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching”. In: *Journal of Teacher Education* 69.4 (2018), pp. 386–407.

[33] Joseph A Lyon and Alejandra J. Magana. “Computational thinking in higher education: A review of the literature”. In: *Computer Applications in Engineering Education* 28.5 (2020), pp. 1174–1189.

[34] Wannisa Matcha et al. “Analytics of learning strategies: the association with the personality traits”. In: *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*. 2020, pp. 151–160.

[35] David Maxwell and Claudia Hauff. “LogUI: Contemporary Logging Infrastructure for Web-Based Experiments”. In: *Advances in Information Retrieval (Proc. ECIR)*. 2021, pp. 525–530.

[36] Jesús Moreno-León, Gregorio Robles, et al. “Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills”. In: *Scratch conference*. 2015, pp. 12–15.

[37] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. “Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking”. In: *RED. Revista de Educación a Distancia* 46 (2015), pp. 1–23.

[38] Christina (Yu) Pei, David Weintrop, and Uri Wilensky. “Cultivating Computational Thinking Practices and Mathematical Habits of Mind in Lattice Land”. In: *Mathematical Thinking and Learning* 20.1 (2018), pp. 75–89. DOI: 10.1080/10986065.2018.1403543. eprint: https://doi.org/10.1080/10986065.2018.1403543. URL: https://doi.org/10.1080/10986065.2018.1403543.

[39] Paul R Pintrich et al. “A manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ).” In: (1991).

[40] Paul R Pintrich. “The role of goal orientation in self-regulated learning”. In: *Handbook of self-regulation*. Elsevier, 2000, pp. 451–502.

[41] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. “Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S)”. In: *International Journal of Interactive Multimedia and Artificial Intelligence* 4 (Jan. 2017), p. 103. DOI: 10.9781/ijimai.2017.09.001.

[42] Beat A Schwendimann et al. “Perceiving learning at a glance: A systematic literature review of learning dashboard research”. In: *IEEE Transactions on Learning Technologies* 10.1 (2016), pp. 30–41.

[43] Cynthia Selby and John Woollard. “Computational thinking: the developing definition”. In: (2013).

[44] SoLAR. *What is Learning Analytics?* Mar. 2021. URL: https://www.solaresearch.org/about/what-is-learning-analytics/.

[45] Dan Suthers and Katrien Verbert. “Learning Analytics as a "Middle Space"”. In: *Proceedings of the Third International Conference on Learning Analytics and Knowledge*. LAK ’13. Leuven, Belgium: Association for Computing Machinery, 2013, pp. 1–4. ISBN: 9781450317856. DOI: 10.1145/2460296.2460298. URL: https://doi.org/10.1145/2460296.2460298.

[46] Xiaodan Tang et al. “Assessing computational thinking: A systematic review of empirical studies”. In: *Computers & Education* 148 (2020), p. 103798. ISSN: 0360-1315. DOI: https://doi.org/10.1016/j.compedu.2019.103798. URL: https://www.sciencedirect.com/science/article/pii/S0360131519303483.

[47] International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA). *Operational Definition of Computational Thinking*. 2011. URL: https://cdn.iste.org/www-root/ct-documents/computational-thinking-operational-definition-flyer.pdf.

[48] *Waterfall model*. Sept. 2021. URL: https://en.wikipedia.org/wiki/Waterfall_model.

[49] David Weintrop et al. “Defining computational thinking for mathematics and science classrooms”. In: *Journal of Science Education and Technology* 25.1 (2016), pp. 127–147.

[50] Jeanette Wing. “Research notebook: Computational thinking—What and why”. In: *The link magazine* 6 (2011), pp. 20–23.

[51] Jeannette M Wing. “Computational thinking”. In: *Communications of the ACM* 49.3 (2006), pp. 33–35.

[52] Gary Ka-Wai Wong and Ho-Yin Cheung. “Exploring children’s perceptions of developing twenty-first century skills through computational thinking and programming”. In: *Interactive Learning Environments* 28.4 (2020), pp. 438–450. DOI: 10.1080/10494820.2018.1534245. eprint: https://doi.org/10.1080/10494820.2018.1534245. URL: https://doi.org/10.1080/10494820.2018.1534245.

[53] Mustafa Yağcı. “A valid and reliable tool for examining computational thinking skills”. In: *Education and Information Technologies* 24.1 (2019), pp. 929–951.

[54] Sabiha Yeni and Felienne Hermans. “Design of CoTAS: Automated Computational Thinking Assessment System”. In: *perspectives* 23 (2019), p. 28.

[55] Timothy T. Yuen and Kay A. Robbins. “A Qualitative Study of Students’ Computational Thinking Skills in a Data-Driven Computing Class”. In: *ACM Trans. Comput. Educ.* 14.4 (Dec. 2014). DOI: 10.1145/2676660. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/2676660.

[56] Timothy T. Yuen and Kay A. Robbins. “A Qualitative Study of Students’ Computational Thinking Skills in a Data-Driven Computing Class”. In: *ACM Trans. Comput. Educ.* 14.4 (Dec. 2014). DOI: 10.1145/2676660. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/2676660.