DOI: 10.34641/ctestem.2022.469

Computational Thinking Dashboard: For learners in Jupyter notebooks

Bhoomika Agarwal

TU Delft, Netherlands

bhoomika10@gmail.com

ABSTRACT

Computational Thinking (CT) - the process of thinking like a programmer or computer scientist - is a skill that that has the potential to transform the way students learn at educational institutions in different domains and different grade levels. With the increasing integration of CT in classrooms, there is a growing need for CT assessment tools to evaluate the acquisition of CT skills. This research develops a framework for CT assessment that detects user micro-interactions in a university-level self-paced Python beginners course integrated into Jupyter notebooks. The users can improve their learning with the help of feedback via CT dashboards as part of this framework. A user evaluation study was conducted which showed that this framework can be used to improve the acquisition of CT skills via programming. The main contributions of this framework are the mapping between CT skills and user micro-interactions and development of the CT dashboards to help the user self-regulate their learning of programming. The framework developed can be easily integrated into any course that teaches Python programming using Jupyter notebooks and is yet to be extended to other programming courses.

KEYWORDS

computational thinking, programming education, python, jupyter notebook

1. INTRODUCTION

With the rapid integration of computers and technology into our daily lives in the 21st century, we are amid 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 (National Research Council et al., 2010) (National Research Council et al., 2011).

Computational Thinking is a concept that lacks an agreed-upon definition (Tang et al., 2020)(Brennan & Resnick, 2012)(Barr & Stephenson, 2011)(National Research Council et al., 2011). Brennan & Resnick, (2012) defined CT with respect to design-based learning activities in Scratch - a block-based programming language - in terms of three dimensions: computational concepts, computational practices, and computational perspectives. The concept of Computational thinking was brought to the limelight in 2006 when Wing, (2006) 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. 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 (Weintrop et al., 2016)(Pei et al., 2018)(Leonard et al., 2018)(Jaipal-Jamani & Angeli, 2017).

A majority of the cross-disciplinary research makes use of visual and block-based programming languages. This graphical representation of code makes it easier to learn the basics of programming, especially for K-12 students and 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., (2020) 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.

While Computational thinking (CT) is being integrated into curricula rapidly, there is a need for methods to assess and evaluate learning of CT concepts (Hadad & Lawless, 2015)(Tang et al., 2020). 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(Lyon & Magana, 2020). 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, (2021) 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.

LA dashboards are learning tools that can help learners and teachers harness the power of LA use it to improve their learning (Jivet et al., 2020). Schwendimann et al., (2016) 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". By making the learner aware of their

progress and triggering self-reflection, LA dashboards can help users regulate their own learning (Jivet et al., 2017). Online learning provides flexibility and accessibility to students through increased learning opportunities, access to learning resources and opportunities for collaboration (Broadbent & Poon, 2015). The downside of online learning is that its success relies heavily on independent learning and the students' autonomous engagement in the course (Broadbent & Poon, 2015). SRL strategies can help learners to gain and retain knowledge methodically and systematically (Broadbent & Poon, 2015). 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.

2. DESIGN AND IMPLEMENTATION

2.1. The Python Programming Course

The CT assessment framework is integrated into 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 – Variables, Control flow, Code Organization, Basic plotting. The course is based on Jupyter notebooks to allow for active leaning and experimentation and uses nbgrader for releasing the exercises. The code in these notebooks is runnable, producing output, and can be modified by the student, to learn all the details and study the effects of changes and variations.

2.2. Adapted definition of CT

For this research, an adapted definition of Computational Thinking(CT) that combines those by Brennan and Resnick, (2012) and Yeni and Hermans, (2019) is used. Brennan and Resnick, (2012) define CT for Scratch with 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)". Yeni and Hermans, (2019) adapt this definition to Python by modifying the CT concepts list to one that is better suited to Python. Visualization, also referred to as 'Simulation' or 'Modelling' is an important CT concept that is missing in the above definition(Hambrusch et al., 2009)(Weintrop et al., 2016)(International Society for Technology in Education & Computer Science Teachers Association, 2011)(Yuen & Robbins, 2014). Thereby my research adds 'Visualizations' to the list of CT concepts proposed by Yeni and Hermans, (2019). Thereby, the revised list of CT concepts used in my research is: data structures, operators, conditionals, sequences, loops, visualization.

Brennan and Resnick, (2012) identify 4 CT practices as part of their CT definition in the form of micro-interactions: being incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing. My research uses these 4 CT practices and detects them through the user's micro-interactions.

2.3. 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. The CT concepts are mapped to the learning modules as shown in Table 1.

Table 1. CT Concepts Mapping

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

2.4. 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 Matcha et al., (2020). 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.

Table 2. Micro-interactions mapping.

Micro-interaction	Action	Source	CT practice
focusin + focusout	Time spent on a cell	LogUI	BII
keystrokes	Additions	LogUI	BII
Cell run count	-	Notebook metadata	TD
Errors in output	Errors	Notebook metadata	TD
copy	Copy	LogUI	RR
paste	Paste	LogUI	RR
Add functions	-	Notebook metadata	AM
Module import	-	Notebook metadata	AM

LogUI is a framework-agnostic client-side JavaScript library developed by Maxwell and Hauff, (2021) for logging user interactions on webpages. Jupyter notebook stores its cells as an array of JavaScript Object Notation(JSON) objects. This contains metadata about the number of cell runs, errors, cell source and more. 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 as shown in Table 2. The four CT practices defined by Brennan and Resnick, (2012) are: Being Incremental and Iterative(BII), Testing and Debugging(TD), Reusing and Remixing(RR), Abstracting and Modularizing(AM). 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.

2.5. CT 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 (Jivet et al., 2020). Dashboards can be used to trigger learners to think about the effort invested in learning and the subsequent outcomes of these activities (Jivet, 2016). Dashboards are used in this research to provide feedback to students per module and 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. As the course is in Jupyter notebooks, the dashboards are also integrated into Jupyter notebooks so that the user does not have to use any additional tools or environments.

2.5.1. CT concepts dashboard

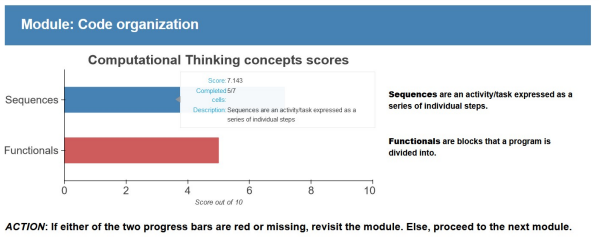

Figure 1. Module-wise CT Concepts dashboard

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 1. Figure 1 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.

2.5.2. 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 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 2.

2.6. Integration and Reproducability

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 (Agarwal, 2021). The steps to reproduce this CT assessment are:

1. Setup a LogUI server following the documentation (Maxwell and Hauff, 2021)
2. Add metadata tags to the course cells
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)
5. Add the CT concepts dashboards to the modules and the overall CT practices dashboard (user ID to be configured here)

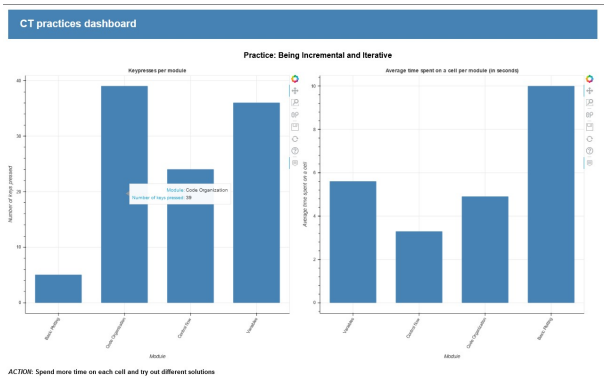

Figure 2. Global CT Practices dashboard

3. 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.

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. All the participants are in the age range 20-30 years. 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'. 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.

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.

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. Both 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. 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 H0 and alternate hypothesis for the experiment H1 respectively are: *H0 = There is no difference in the CT skills users before and after taking the course, H1 = There is a difference in the CT skills of users before and after taking the course.*

The survey used is an adapted version of the survey created by Kılıç, Göko˘glu, and Öztürk, (2021). 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. 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?

4.1.1. RQ1: Did the users acquire CT skills in the form of both CT concepts and 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 3. 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 3. 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. As the self-reported survey questions pertain directly to the acquisition of Python programming skills, the value of the mean and SD also imply an improvement in Python programming skills of the user.

	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 3. Mean and SD values per question

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

To analyze the significance of the change in CT skills before and after the Python basic programming course, a statistical approach is used by conducting a paired t-test for the population. 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 H0 and alternate hypothesis H1 respectively are: *H0 = There is no difference in the self-reported CT skills users before and after taking the course, H1 = 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 H0 is rejected. As seen in Figure 3, the p-value is less than α. Thereby, the null hypothesis H0 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.

4.1.3. 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.

As can be seen from Figure 5, 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 Figure 5, it can be seen that User 4 reports a low change in the CT skills. This user has a good prior knowledge of the Python programming language(5 out of 10) and thereby did not gain much added value from the course. This user also scores 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.

```
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. Paired t-test result

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

Figure 5. User micro-interaction scores and survey response changes

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 an improvement in Python programming skills. 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 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.

There are 2 main limitations of this research. 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. 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 assessment of self-regulated learning - Motivated Strategies for Learning Questionnaire (MSLQ) (Pintrich et al., 1991) - 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.

6. REFERENCES

National Research Council & Others. (2010). Report of a workshop on the scope and nature of computational thinking. National Academies Press.

National Research Council & Others. (2011). Report of a workshop on the pedagogical aspects of computational thinking. National Academies Press.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. Computers & Education, 148, 103798. doi:10.1016/j.compedu.2019.103798

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada, 1, 25*.

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community*? ACM Inroads, 2*(1), 48–54. doi:10.1145/1929887.1929905

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

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

Pei, C. (yu), Weintrop, D., & Wilensky, U. (2018). Cultivating Computational Thinking Practices and Mathematical Habits of Mind in Lattice Land. *Mathematical Thinking and Learning, 20*(1), 75–89. doi:10.1080/10986065.2018.1403543

Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching. *Journal of Teacher Education, 69*(4), 386–407.

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology, 26*(2), 175–192.

Hadad, R., & Lawless, K. A. (2015). Assessing computational thinking. In *Encyclopedia of Information Science and Technology, Third Edition* (bll 1568–1578). IGI Global.

Lyon, J. A., & J. Magana, A. (2020). Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education, 28*(5), 1174–1189.

Kılıç, S., Gökoğlu, S., & Öztürk, M. (2021). A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking. *Journal of Educational Computing Research, 59*(2), 257–286.

Jivet, I., Scheffel, M., Schmitz, M., Robbers, S., Specht, M., & Drachsler, H. (2020). From students with love: An empirical study on learner goals, self-regulated learning and sense-making of learning analytics in higher education. *The Internet and Higher Education, 47,* 100758. doi:10.1016/j.iheduc.2020.100758

Schwendimann, B. A., Rodriguez-Triana, M. J., Vozniuk, A., Prieto, L. P., Boroujeni, M. S., Holzer, A., … Dillenbourg, P. (2016). Perceiving learning at a glance: A systematic literature review of learning dashboard research. *IEEE Transactions on Learning Technologies, 10*(1), 30–41.

Jivet, I. (2016). *The Learning tracker: a learner dashboard that encourages self-regulation in MOOC learners*. Opgehaal van http://resolver.tudelft.nl/uuid:f6c2ede4-a4e3-4ff0-b681-b0d057854e3c

Jivet, I., Scheffel, M., Drachsler, H., & Specht, M. (2017). Awareness Is Not Enough: Pitfalls of Learning Analytics Dashboards in the Educational Practice. In É. Lavoué, H. Drachsler, K. Verbert, J. Broisin, & M. Pérez-Sanagustín (Reds), *Data Driven Approaches in Digital Education* (bll 82–96). Cham: Springer International Publishing.

Broadbent, J., & Poon, W. L. (2015). Self-regulated learning strategies & academic achievement in online higher education learning environments: A systematic review. *The Internet and Higher Education, 27*, 1–13.

Yeni, S., & Hermans, F. (2019). Design of CoTAS: Automated Computational Thinking Assessment System. *perspectives, 23*, 28.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A Multidisciplinary Approach towards Computational Thinking for Science Majors. *SIGCSE Bull., 41*(1), 183–187. doi:10.1145/1539024.1508931

in Education (ISTE), I. S. F. T., & (csta), C. S. T. A. (2011). *Operational Definition of Computational Thinking*. Opgehaal van https://cdn.iste.org/www-root/ct-documents/computational-thinking-operational-definition-flyer.pdf

Yuen, T. T., & Robbins, K. A. (2014). A Qualitative Study of Students' Computational Thinking Skills in a Data-Driven Computing Class. *ACM Trans. Comput. Educ., 14*(4). doi:10.1145/2676660

Matcha, W., Gašević, D., Jovanović, J., Uzir, N. A., Oliver, C. W., Murray, A., & Gasevic, D. (2020). Analytics of learning strategies: the association with the personality traits. *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, 151–160.

Maxwell, D., & Hauff, C. (2021). LogUI: Contemporary Logging Infrastructure for Web-Based Experiments. *Advances in Information Retrieval (Proc. ECIR)*, 525–530.

Pintrich, P. R., & Others. (1991). *A manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ)*.

Agarwal, B. (n.d.). ct_dashboards. *GitHub*. Opgehaal van https://github.com/bhoom10/ct_dashboards