Supporting Co-Regulation and Motivation in Learning Programming in Online Classrooms

LAHARI GOSWAMI, University of Lausanne, Switzerland
ALEXANDRE SENGES, University of Lausanne, Switzerland
THIBAULT ESTIER, University of Lausanne, Switzerland
MAURO CHERUBINI, University of Lausanne, Switzerland

Self-regulation of learning in programming has been extensively investigated, emphasising an individual's metacognitive and motivational regulation components. However, learning often happens in socially situated contexts, and little emphasis has been paid to studying social modes of regulation in programming. We designed Thyone, a collaborative Jupyter Notebook extension to support learners' programming regulation in an online classroom context with the overall aim to foster their intrinsic motivation toward programming. Thyone's salient features - *Flowchart*, *Discuss* and *Share Cell* - incorporate affordances for learners to co-regulate their learning and drive their motivation. In an exploratory quasi-experimental study, we investigated learners' engagement with Thyone's features and assessed its influence on their learning motivation in an introductory programming course. We found that Thyone facilitated the co-regulation of programming learning and that the users' engagement with Thyone appeared to positively influence components of their motivation: interest, autonomy, and relatedness. Our results inform the design of technological interventions to support co-regulation in programming learning.

CCS Concepts: • **Software and its engineering → Collaboration in software development**; • **Human-centered computing → Empirical studies in collaborative and social computing**; • **Social and professional topics → CS1**.

Additional Key Words and Phrases: Collaboration, co-regulation, motivation, programming

ACM Reference Format:
Lahari Goswami, Alexandre Senges, Thibault Estier, and Mauro Cherubini. 2023. Supporting Co-Regulation and Motivation in Learning Programming in Online Classrooms. *Proc. ACM Hum.-Comput. Interact.* 7, CSCW2, Article 298 (October 2023), 29 pages. https://doi.org/10.1145/3610089

1 INTRODUCTION

Computer programming is a complex activity and is often difficult for novices to learn and master [37]. Mastery of the subject requires beginners to develop a diverse set of knowledge and skills pertaining to programming concepts as well as awareness and development of the domain-specific cognitive processes. One of the fundamental process skills crucial to the success of programming learning, as established by prior research, is *self-regulation* of learning [6, 29, 70]. Self-regulation is the process by which an individual, driven by a learning goal, strategically adapts and orients their meta-cognitive, motivational, and behavioural strategies involved in their learning processes

Authors' addresses: Lahari Goswami, lahari.goswami@unil.ch, University of Lausanne, Lausanne, Switzerland; Alexandre Senges, senges.alex@gmail.com, University of Lausanne, Lausanne, Switzerland; Thibault Estier, thibault.estier@unil.ch, University of Lausanne, Lausanne, Switzerland; Mauro Cherubini, mauro.cherubini@unil.ch, University of Lausanne, Lausanne, Switzerland.

2573-0142/2023/10-ART298
https://doi.org/10.1145/3610089

[83]. Studies on self-regulation in programming have frequently emphasised meta-cognitive aspects involved in this process, like the importance of meta-cognitive knowledge and task-specific awareness [22], strategies for efficient self-regulation [23], the role of meta-cognitive monitoring on self-regulation [29], and more recently facilitating this meta-cognitive awareness in novice's programming problem-solving [42]. In addition to meta-cognition, learner's motivation is critical in initiating and sustaining self-regulatory behaviours. While motivation in programming has been studied to identify its predictors such as value in task, intrinsic motivation, self-efficacy and how they affected self regulation and performance [6, 82], only few studies have explored interventions in programming teaching pedagogy to foster this motivation [40, 44, 76].

Introductory programming courses in university classrooms comprise a social setting for learning, involving a large number of students, usually with similar pedagogical experiences. The consequences of the COVID-19 pandemic have also led many classroom instructions to adapt to online modes of learning. Online programming classrooms in universities typically follow synchronous learning formats with integrated tasks and teachers' instructions, such that students are in socially present learning contexts despite being remotely located. This format differs from other online learning settings, like MOOCs, where following a course and doing related activities is learner-paced and asynchronous in nature, with limited social presence [26]. Pedagogical strategies in online programming classrooms necessitate emphasising the sociocultural paradigm of learning [39] unique to this context and further enable students to succeed in these new learning settings.

Collaborative learning is a prevalent pedagogical approach to facilitate synergy among peers and to develop their critical thinking and programming abilities in classrooms or remote learning environments [43]. In the context of collaboration, contemporary education psychology grounds learning regulation to be a social phenomenon and defines three primary modes of social regulation: *self-regulated learning*, *socially shared regulation of learning* and *co-regulated learning* [30] (see Sec. 2.3). An individual's learning process in collaborative settings such as programming classrooms does not occur in isolation. Their learning process is influenced by dynamic internal, social, and environmental conditions that act as affordances and constraints to shape the learner's self-regulation [79] – a process skill crucial for novices learning to program [23]. Therefore, collaborative programming learning interventions in classrooms must recognise and support learners' social regulatory processes.

Learning to program in a collaborative context has received a great deal of attention in research, and has paved the way for various systematic environments and computer-supported collaborative learning (CSCL) interventions. Several CSCL interventions have been designed and investigated to support enabling social processes in learning to program [58], facilitating collaboration through programming editors [56], enabling remote pair-programming [8], using gamification to aid problem-solving by collaboration [71], and so on. Although these CSCL interventions devise different methods of collaborative interactions for productive learning and engagement, they seldom emphasise studying the social modes of regulation in programming. A few recent studies exploring social regulation in programming include Arciniegas-Mendez et al.'s model based on the three social regulations for collaboration in software engineering projects catered towards working professionals [3], and ongoing research (a doctoral consortium) on the socially shared regulation of learning in programming education context [72].

Our work aims to support learners' social regulations in learning programming in online classrooms, with an overall goal to foster their intrinsic motivation[1] viewed through the lens of Self Determination Theory (SDT) [64]. We present Thyone, a collaborative extension for Jupyter Notebooks with three features : *Flowchart*, *Discuss* and *Share Cell*, which provides remote learners with

[1]Intrinsic motivation is doing an activity for its inherent satisfaction rather than for some separable consequence [64, p. 56].

opportunities to regulate their learning at individual and social levels and to drive their learning motivation (Sec. 3). These features of Thyone have been designed to support students' sense of autonomy through meta-cognitive reflection in programming and to facilitate their co-regulatory behaviours while also fostering their relatedness through peer reviews and peer feedback on code, error, and output. We intend to explore and understand the behavioural outcomes of introducing remote classroom learners to Thyone and whether it impacts their learning motivation and experiences. In this study, we pose the following research questions:

RQ1. Does the availability of the salient features of Thyone result in learners co-regulating remotely?

RQ2. Does learner's interactions with the salient features of Thyone affect their SDT motivational constructs?

We conducted an exploratory quasi-experimental analysis to uncover how learners engaged with the features of Thyone in their learning processes and evaluated its influence on their programming motivation. We found that the features embedded in Thyone facilitated learners' remote co-regulation. Also their engagement with these features is positively associated with their respective motivational constructs of interest, autonomy, and relatedness. Although the features of Thyone positively shaped learners' motivational constructs, an overall effect of the treatment manipulation on intrinsic motivation was not observed. We contribute to research by presenting an artifact that informs the design of affordances to support learners' social regulation of programming in online classroom environments. We further provide empirical evidence that design of these situated interactions in online classrooms stimulates learners' social regulation and contributes toward supporting their basic psychological needs that are posited to foster students' intrinsic motivation.

2 LITERATURE REVIEW

2.1 Programming learning in novices

Learning to program is inherently a complex process. It entails various cognitive activities and mental representations related to program design, comprehension, modification, and debugging. Even at the most basic level of computer literacy, it requires acquiring and constructing conceptual knowledge, developing strategies and structuring basic operations into schemas and plans for execution [62]. This complex process presents many challenges for novices who start to learn programming in introductory courses. The difficulties that novice programmers experience are multifaceted. Several studies highlight the cognitive dimension difficulties that learners face when beginning to learn to program; a few of them being difficulties in understanding different programming concepts [47, 62], issues with applying programming knowledge to problem-solving [61, 75], and the lack of demonstrating problem-solving skills [74, 80]. While grasping programming concepts and articulating practical programming skills are central to programming learning, another critical component contributing to a novice's programming learning is the ability to administer their cognitive processes while learning and doing programming. Unlike more experienced programmers, beginners are confronted with a significant barrier in their awareness and the monitoring of their cognitive processes [22], which are critical to problem-solving and computational thinking [9]. Allwood [2] identified that novice programmers struggle to approach programming tasks efficiently due to their limited knowledge of both domain and problem-solving techniques and that they usually employ hasty, general problem-solving approaches rather than specialised strategies used by experts. Research analysis on effective programming teaching and learning suggests that process skills, particularly *self-regulation*, to be a critical factor in programming learning success [70], which we discuss in the next section.

2.2 Self-regulation in learning programming

Theories of self-regulation of learning entail understanding how learners learn. Zimmerman defines self-regulated learning as the extent to which a learner actively participates in the adaptation and strategic orientation of one's own metacognitive, motivational, and behavioural strategies while working toward a learning goal [83]. Learners who self-regulate take proactive control of their meta-cognition by iteratively planning, monitoring, and adapting their thoughts, beliefs, and strategies in a given context, thereby shaping their learning and outcomes [84]. Programming is essentially a cognitive activity, and as Falkner et al. [23] emphasises, "crucial to successful mastery is the development of discipline specific cognitive and metacognitive skills, including self-regulation." Their research highlighted the importance of self-regulation for successful programming by identifying instances of self-regulatory strategies that influence and assist students' learning processes.

In programming, self-regulation has been studied from different perspectives, primarily emphasising the meta-cognition dimension of learning [55]. From a social cognitive point of view, "meta-cognitively self-regulated learners are the ones who plan, organise, self-instruct, self-monitor, and self-evaluate at various stages during the learning process" [83]. Griffin et al. [29] asserted the importance of this aspect in programming learning by pointing out that accurately assessing one's current state of comprehension during a cognitive task is at the core of self-regulation, which influences learning for that task as well as transcends to future ones. Many studies have established that successful learners and programmers demonstrate meta-cognitive self-regulation behaviours like self-explanation, explicitly monitoring their progress, comprehension and misunderstandings and reflecting on the effectiveness of their problem-solving strategies [14, 22, 41, 60]. Clearly, meta-cognitive awareness about where one is at their problem-solving process and self-reflection on it is critical in programming learning self-regulation. One of the most relevant comprehensive studies conducted on facilitating this meta-cognitive awareness in novices' programming problem-solving is Loksa et al. [42]. They defined meta-cognition involved in programming solving as one's iterative ability to have a mental representation of the problem at hand, being able to refine and reflect this mental representation while solving it and finally being able to converge towards a solution and express it as code. Centered around this notion of meta-cognition, they proposed a programming teaching intervention that explicitly taught key steps of programming problem-solving to students and was able to conclusively promote meta-cognitive awareness. This intervention also promoted learners' affective aspects like productivity, independence, programming self-efficacy and growth mindset.

Along with meta-cognition, the social cognitive view of self-regulated learning also emphasises the role of learners' motivation as a key factor influencing their cognitive control involved in self-regulation. Learners need to be motivated in order to self-regulate their learning process and be successful in programming [27]. Studies on motivation reveal that certain predictors like students' perception of the value they associate with the task, self-efficacy, intrinsic motivation have a positive influence on their regulatory behaviours and as well as their performance [5, 6, 59, 82]. The implications of these studies pave the way for investigating motivational interventions to facilitate programming learning. For example, Bergin et al. [6] and Ruf et al. [63] suggest opportunities for pedagogical interventions that promote learner's self-regulation and foster intrinsic motivation for better programming learning experiences and outcomes.

In our research, we aim to foster learners' programming intrinsic motivation by leveraging their regulatory processes in a social learning context. To do this, we investigate the social aspects of programming regulation which we discuss in the following section.

2.3 Collaborative learning in programming

Collaborative learning encompasses a variety of pedagogical practices in which interactions between learners are the most crucial factor in their learning [20]. When it involves large numbers of students, this pedagogical approach leans on the sociocultural paradigm of learning [20], which emphasises that learning is socially situated. It explains that learners mediate their actions and relations across individual and social planes and cultural contexts to construct practices, meanings, and roles that constitute new learning [39, 49]. Contemporary educational psychology also grounds the regulation of learning as a social phenomenon and defines the regulatory process responsible for effective learning in a collaborative context. In their most recent theory on social modes of regulation in collaboration, Hadwin et al. [30] describe three primary modes:

(1) Self-regulated learning (SRL): This refers to the regulatory process of individual learners in a collaborative context and is the precursor for optimal productive collaboration to occur in the context of a group task.
(2) Socially shared regulation of learning (SSRL): This refers to the processes of group regulation, which entail groups taking metacognitive control of the task collectively through negotiated, iterative fine-tuning of cognitive, behavioural, motivational, and emotional states as required.
(3) Co-regulated learning (CoRL): This refers to the transitional and flexible stimulation of regulation through interpersonal interactions and exchanges between learners in a collaborative context. It focuses on creating affordances and constraints for productive self-regulated learning for individual or shared regulation of learning in a group.

Introductory programming courses in universities usually cater to a large cohort of students in the first year of a Bachelor's program. In formal online classrooms, the collaborative interactions among students and their regulation of learning differ from those in other online learning settings, like MOOCs [54]. Learners in MOOCs have the freedom to pursue the courses on volition, following them asynchronously, in a self-paced manner, without the commitment to finish them. Attributed to the fact that learners do not follow courses at the same time [81], and other contextual factors, garnering social interactions in MOOCs is dissimilar to online classrooms and currently it mostly relies on mediums like forums [26]. In contrast, a formal education environment, such as an online classroom, primarily follows a synchronous online learning setting, typically with students from similar pedagogical backgrounds who are present remotely, but engaged in tasks and classroom instructions at the same time. It constitutes a complex socio-cultural context, shaping simultaneous interactions across teachers, students, their changing roles and tasks embedded within the classroom [10]. This enviroment necessitates the design of pedagogical interventions in online classrooms that recognise the classroom aspect [73], and can support effective peer-related collective learning strategies in remote settings. We scope our research particularly to online classroom learning contexts.

In the context of programming learning, collaboration has been extensively researched. It has prompted the design and investigation of several collaborative interventions and Computer-Supported Collaborative Learning (CSCL) technologies to support and analyse programming learning. A few of the systematic collaborative learning environments designed and researched are SCRATCH programming environment [58] to foster students' interest, creativity and social synergy while programming learning, the RIPPLE (Remote Interactive Pair-programming Learning Environment) environment [8] to facilitate distributed synchronous programming in classrooms, promote motivation and retention within students, PRAISE [16], an online peer-review system to facilitate anonymous review and prompt feedback, Flipped Classroom [32], a pedagogical model which reversed the lecture and homework structure to promote learning motivation, and Pyrus, a collaborative educational game for novices to foster the problem-solving skill of planning while programming [71]. Collaborative interventions have also been integrated into existing environments

like BlueJ IDE[25], Alice [1] and more recently, in computational notebooks like Codestrates [56] which has been used to explore collaborative programming learning in novices [7]. Even though these interventions enabled social processes in learning, scaffolded programming solving strategies, and explored collaborative behaviours to facilitate learning, they rarely addressed fostering of regulatory processes involved in collaboration. In a few recent studies, this has been addressed by Arciniegas-Mendez et al. in their model based on the three social regulations for collaboration in software engineering catered towards expert stakeholders[3], and on-going research in exploring socially shared regulation of learning in programming education context [72]. However, we found ample scope to design collaborative intervention for programming learning by integrating social regulatory learning strategies.

Through our study, we design technological affordances to facilitate co-regulated learning amongst students in online classrooms, therefore also fostering their self-regulatory behaviours and developing intrinsic motivation to program in a collaborative learning context.

2.4 Motivation in Programming: Self-Determination Theory

Self-Determination Theory (or SDT) [64] is especially different from other theories of human motivation because it emphasises the different types and sources of motivation that impact the quality and dynamics of behaviour. SDT posits that people have different *levels* or *amounts* of motivation to perform a specific activity. It also states that people have different types of *orientation of motivation* i.e. the underlying attitudes, goals and values that give rise to action [17]. These types are classified as *intrinsic* and *extrinsic* motivation. Different types of motivation differ in the sources that initiate them, in magnitude, in affects, and in the experiences of the individual and their behavioural consequences [67, p. 14]. Moreover, intrinsically motivated behaviours are autonomous and experienced as being volitional. In contrast, extrinsically motivated behaviours can *vary widely in the degree to which they are controlled versus autonomous* [66]. For example, a student may be extrinsically motivated to study for an exam to avoid the punishment of parents, but could also be motivated because they observe a valued outcome (i.e., getting a degree). Additionally, the SDT explains that these –previously mentioned– extrinsic motivation types can urge a person to behave a certain way in the short-term, but will fail to maintain the behaviour over more extended periods [17]. As a result, behaviour-change interventions designed for extrinsic motivation types may not sustain the new behaviour after the intervention ends. Mainly, SDT describes three Basic Psychological Needs (or BPN), that when satisfied by the contextual conditions, lead to a self-determined action:

(1) *Autonomy* "refers to feeling willingness and volition with respect to one's behaviour. The need for autonomy describes the need of individuals to experience self-endorsement and ownership of their actions." [67, p. 86]
(2) *Competence* "refers to feeling effective in one's interactions with the social environment–that is, experiencing opportunities and support for the exercise, expansion, and expression of an individual's capacities and talents." [67, p. 86]
(3) *Relatedness* "refers to both experiencing others as responsive and sensitive and being able to be responsive and sensitive to them–that is, feeling connected and involved with others and having a sense of belonging." [67, p. 86]

From an SDT perspective, self-regulation of learning is apparent when individuals are intrinsically motivated or internalise their external motivation [19]. It further stresses that social contexts that allow the fulfilment of autonomy, competence and relatedness help sustain this intrinsic or internalised motivation. Classrooms that adequately support these BPNs have been shown to foster students' learning engagement. [57]. Incorporating SDT in online learning contexts has also been shown to promote learners' success by increasing their motivation [33]. Hence in this study we

use the motivational constructs of SDT to investigate students' learning regulation and motivation in online programming classrooms.

This study aims to explore aiding the regulation of programming learning at social levels in an online classroom setting and leverage that to foster learners' intrinsic motivation. Therefore the umbrella research question we aim to answer is *What collaborative interventions may be devised to support social modes of regulation and facilitate intrinsic motivation for novices learning to program in online classrooms?*

3 THYONE EXTENSION

We designed and developed a collaborative extension for Jupyter Notebooks, named Thyone[2] to help students regulate their process of learning programming.

3.1 Design Goals

Effectively solving a programming problem not only necessitates focus on the syntax, semantics and constructs of the programming language, but also for a learner to be aware of their cognitive processes and monitor errors and strategies while performing the task [22]. Thyone's design decisions were driven to promote this regulation required in the programming learning process by supporting individual *meta-cognitive reflection* and *co-regulatory interactions* amongst peers. We define our design goals as the following : (1) To aid metacognitive reflection involved in programming problem-solving, and (2) To facilitate co-regulated learning through peer-reviews and feedback.

Loksa et al. state that to develop metacognitive reflection in programming problem solving, a learner must have a mental representation of the problem at hand and be able to refine and reflect on this mental representation as they progress towards a solution iteratively [42]. In their work, they have further proposed a set of six stages that represent self-regulated progress while solving programming problems. For teaching programming to novices, Soloway has also defined a set of design strategies to help them learn programming by formulating goals, plans, mechanisms and explanations as the products of the programming process [74]. We abstract from these two sets of programming problem-solving strategies, and define the cornerstones of Thyone's design rationale to be: (i) plan actions and explore problem representation, (ii) write code (iii) provide or seek feedback by interpreting output and errors. From an SDT perspective, interactions with Thyone should therefore assist students' basic psychological needs of *autonomy* by facilitating their learning regulation and *relatedness* by instilling a sense of community through collaborative interactions — thereby supporting enhancement of their intrinsic motivation.

In the context of learning, regulation is a multi-faceted, socially situated phenomenon [30]. This concept essentially extends to the regulation process involved in programming learning as well. In programming problem-solving, not only should the learner's cognition and meta-cognition be monitored, but also their motivation, affect, and behaviour, which is in turn shaped by the socio-contextual conditions to which they are subjected. Co-regulation facilitates learning by dynamic shifting and internalising regulatory processes through productive interpersonal interactions amongst learners [31]. Feedback is essential for regulation in both individual and social learning contexts [30]. In learning programming, when faced with errors in code or difficulties in understanding, it can be frustrating for beginners to progress in their learning. Having the possibility to discuss with a peer or seek and provide feedback on each other's code allows for the opportunity to reflect on and regulate one's learning while also influencing the other. We devised Thyone to promote co-regulatory behaviour among students through the design of opportunities

[2]Pronounced /θaɪˈoʊniː/, the extension has been named Thyone after one of the planet Jupiter's moons.

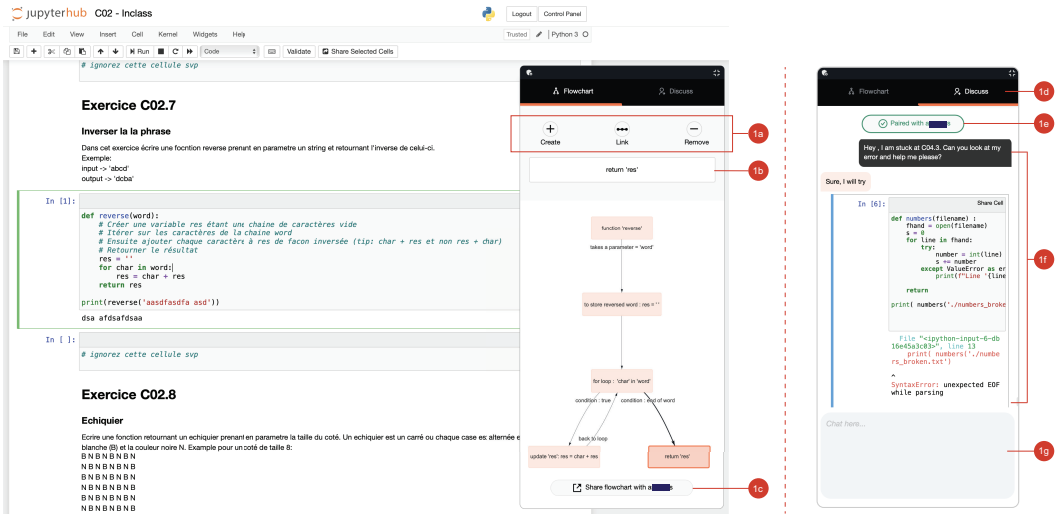

Fig. 1. *Flowchart*: (1a) Buttons to create, connect and modify nodes and edges in flowchart. (1b) Text-box to modify node and edge labels of a flowchart. (1c) Option to share a flowchart with the paired peer. *Discuss*: (1d) Notification broadcasted through blinking of this tab. (1e) Paired with a random peer (anonymised). (1f) Chat history with the peer showing shared text messages and code cells with error. (1g) Chat medium to send text messages to the paired peer.

and affordances that support this dynamic exchange of ideas and peer feedback on code and output to aid their programming learning process.

3.2 Thyone Features

Thyone is a collaborative Jupyter Notebook extension which is available to a user when they open a notebook.[3] It remains anchored to the right side of the notebook and can be minimised. This extension has been developed using `ReactJS` and `TypeScript` in the front-end and `NodeJS` in the back-end. Thyone has three main features named *Flowchart*, *Discuss* and *Share Cell* to accomplish the design goals we stated earlier.

(1) *Flowchart*: The Flowchart feature aims to aid metacognitive awareness involved in programming problem-solving by allowing students to externalise and reflect on their mental representations of a problem. It has been designed as an affordance to let students plan their actions before solving a problem. It includes provisions for creating basic rectangular nodes using the Create button (Fig. 1a) allowing students to objectively represent their actions. To construct a flow of activities, each node can be linked to other nodes as well as to itself using the 'Link' button (Fig. 1a). All the created nodes and edges can be modified and labelled using a text field (Fig. 1b). The Flowchart feature is kept simple and straightforward so that students feel free to abstract and plan the provided problem in any way they see fit, as well as explore and iterate on it. The primary focus of the feature is to encourage students to use it as a mental workout space where they can plan and explore solutions to a programming problem. We suggest that, by externalising their thought process while working on a problem students would, reflect on their comprehension states and regulate accordingly by using the flowchart. This feature is

[3]The code for Thyone is open-source at GitHub. See https://github.com/petlab-unil/thyone-extension-frontend for front-end and https://github.com/petlab-unil/thyone-extension-backend for back-end code.

primarily intended to be used individually, with the optional provision of sharing it for feedback later (using the button in Fig. 1c). Once students save their flowcharts they are stored in their individual notebooks, and remain persistent until the notebook is deleted.

(2) *Discuss* : The Discuss feature (Fig. 1g) serves as a chat medium. It allows bi-directional communication via `Socket IO`[4] between two Jupyterhub users who both have the Thyone extension enabled. It should be noted that the connection is established between users and is not dependent on the notebooks. This enables users who are working on different notebooks to be paired with one another. If a user opens multiple notebooks in separate tabs, he will still be connected to the person with whom he was initially paired, provided that person has not disconnected (Fig. 1e). Students are paired automatically based on the order in which they launch their individual Jupyter servers. When students log into their Jupyterhub and start their servers, they are put in a queue to be paired with the next person who logs in right after. As a result, the pairing mechanism is randomised, and students have no choice in selecting their partner. We devised this random pairing strategy to increase the chances that students engage in learning-related exchanges with peers while also ensuring that they do not feel stuck if communication with a specific pair did not work out.
If no one is available to connect at a given time, or if the paired user disconnects their server, the other student is placed in the waiting queue for the next available connection. When students are paired, they are notified and can start communicating with one another. The notifications rely solely on visual cues, through the blinking of extension's Discuss tab (Fig. 1d). Notifications are broadcast every time a person connects with a peer or receives a message from them. Hence the audio cue is intentionally avoided so as not to disturb the students during lectures, or if the student is using the service in a study room or library.
Sharing content through the Discuss feature is also limited to text and code cells which we will discuss in the next section. The content shared between two people remains persistent. Therefore, if the same people are paired in another instance, their chat history will be visible (Fig. 1f). However, the extension does not allow students to copy or paste text on or from the chat window or from the extension. We explain the rationale for this in the next point.

(3) *Share Cell* : Jupyter Notebook consists of code cells in which students write and execute their code. The Share Cell feature allows students to directly share their code cells along with output and errors from their notebook to the paired classmate (Fig. 2a,b). Ucan and Webb identified that specific events, such as when students express a lack of understanding or a misconception, can frequently initiate co-regulation of metacognitive processes [77]. Furthermore, enabling peers' feedback sharing reinforces students' self-regulatory learning behaviours [50]. Therefore, sharing feedback on each other's code enables students to monitor and reflect on their understanding in relation to their established goals, as well as adopt and adjust regulatory changes in learning mediated through emergent inter-personal interactions. From the first phase of our study, we found that students used WhatsApp and Discord to share code exercises, errors, doubt clarification as copied texts, screenshots and images. However, these modes of sharing limit the readability of codes and errors and allow copy-paste solutions without understanding them. The share cell feature (2) has been designed as a co-regulatory affordance to stimulate learning regulation and reduce student frustration when they are stuck and seek feedback. It aims to enable peer-reviewing through accurate representation of code, output and error, thereby encouraging meaningful code and idea exchanges. Online environments may be conducive to digital plagiarism. One study found that, even if students completed less obligatory tasks, those who copied exercises from peers performed worse than those who copied fewer

[4]See https://socket.io/, last accessed April 2023.

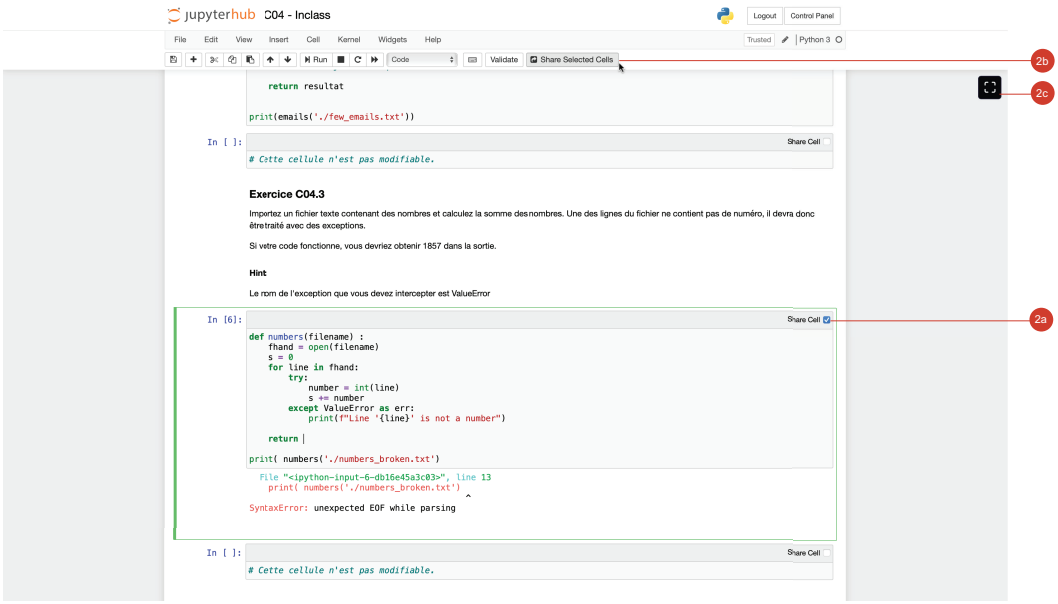

Fig. 2. *Share Cell*: (2a) Option to select cell(s) to share. (2b) Button to share selected cells. (2c) Thyone extension in minimised format.

exercises, even if they took on more voluntary exercises [21]. Furthermore, copying favors rote learning and memorisation while giving minimal support for learning internalisation, which has been found to hinder the performance of programming students [34]. Hence, the shared code or output has been rendered read-only in the chat platform to prevent copying and pasting. This is specifically different from popular editors such as `Google Colab`[5] that allows synchronous collaborative editing, which does not support co-regulation.

3.3 Instrumentation of the Extension

Thyone has been designed to serve as a data logging tool in addition to the primary features that support students' learning. This feature has been developed to collect objective measurements of students' learning behaviour for later analysis (Sec.4.5). In the back-end, regardless of whether the extension is enabled, this data-logging tool of Thyone logs the user's Jupyter activity to our database. If a user has the extension enabled, it will log their interactions with Thyone in addition to their Jupyter usage. Thyone interactions that are logged include interactions with the flowchart feature and instances of sharing messages, cells, and flowcharts along with their content. All data collected by this tool is stored in a database hosted on our institution's server, ensuring data protection policy. Data is anonymised and retained for up to two years.

4 METHODOLOGY

To answer our research questions (presented in Sec. 1) we conducted an exploratory quasi-experimental study to gain an understanding of learners' behaviours of using different features of Thyone and investigate its influence on their learning experiences. The evaluation study was conducted in a first-year programming course (see Sec. 4.1) with a cohort strength of several

[5]See https://colab.research.google.com/, last accessed April 2023.

hundreds of students in an online setup. The study took place from February 2021 end to July 2021 end, which included an initial setup phase of three weeks. We collected both quantitative and qualitative data. The study was approved by the IRB of our institution.

4.1 Study Context

The pedagogical aim of the introductory programming course in which the study had been conducted is to introduce Computer Science (CS) programming syntax and semantics to the first year students pursuing a Bachelor of Science in either Economics or Management. The course is mandatory for the students. The goal of this course is to convey the basics of the Python Programming language and the most common programming concepts (e.g., variables, conditional instructions, loops), exposing them to algorithmic thinking, and enabling them to create simple data-analysis tools to parse large data sets.

This course uses `Jupyter Notebook`[6], which is hosted on `JupyterHub`[7], to teach and practice Python programming. JupyterHub, the multi-user version of Jupyter notebooks, makes the distribution and monitoring of pedagogical materials and notebooks easily scalable and maintainable for a large number of users.

The course took place during the COVID-19 pandemic of 2021, and was therefore entirely taught in online format. At the beginning of the course, the university administration distributed the registered students into three different groups. These three groups were assigned to three separate sessions per week; each session was conducted live via Zoom and taught by three different teachers. To let the students follow the course, they were given early access to the course resources before each weekly session. The resources included the session slides, videos of theoretical explanations (typically 3 to 5, of ~10 minutes each, every week), mandatory exercises to be practised during the class, and non-mandatory homework exercises to be completed before the following session. The Zoom sessions were devoted to classroom lectures, discussion of guided examples and facilitating the students to practice and correct mandatory In-class exercises along with the teacher and peers for that particular session synchronously. An example of the coursework is provided in Supplementary 1 (Supp. 1) [8]. At the end of the semester, students were evaluated with a final exam, which consisted of multiple-choice questions that required them to solve programming problems.

4.2 Study Design and Procedure

We designed a quasi-experiment for the evaluation study and employed a *pretest-posttest nonequivalent control group design* [15]. For research in natural settings, such as educational research in the classroom and clinical research with real clients, quasi-experiments are often used when randomised controlled experiments become difficult or unethical to conduct[28]. Given that our intervention could influence students' learning, and possibly their grades, we relied on a quasi-experimental design. All students in the course were invited to participate in the study via email invitation that contained a link to a consent form and a questionnaire. This questionnaire contained the IMI instrument introduced in Sec.4.4 (available in Supp. 2). Out of 683 students enrolled in the course, N=281 agreed to participate in the study and filled the enrollment questionnaire. We provided them with a description of Thyone and let students decide whether to activate the extension in Jupyter, or not. Once enabled, the students could not uninstall Thyone by themselves and had to opt out by requesting the researchers, but they could minimise the extension (see Fig. 2c). This was done to avoid jeopardising the setup of the evaluation study. The Jupyter activity logging tool implemented

[6]See https://jupyter.org/, last accessed April 2023.
[7]See https://jupyter.org/hub, last accessed April 2023.
[8]All the supplementary materials are available in the Open Science Framework (OSF) Repository. See here https://doi.org/10.17605/osf.io/axdbt, last accessed July 2023.

Table 1. Participants' demographics information

	No. of participants	Age (in years)	Gender	Pre-test Interest
Participants who took part in conversation sessions	N =136	M = 20.3 (SD = 2.6)	81m / 55f	M = 4.47 (SD = 1.22)
Participants who completed the study	control group : N = 44	M = 20.8 (SD = 1.7)	17m / 27f	M = 4.23 (SD = 1.46)
	treatment group : N = 98	M = 20.6 (SD = 2.9)	57m / 41f	M = 4.38 (SD = 1.30)
Participants who attended interview sessions	N = 13	M = 20.9 (SD = 3.0)	8m / 5f	M = 4.73 (SD = 1.05)

through Thyone was installed for all the students who participated in the study, as specified in the consent form. This resulted in N=211 students who chose to enable Thyone, henceforth our treatment group, and N=70 students who did not choose to enable Thyone, henceforth our control group.

During the treatment phase, we recorded *conversation sessions* (explained in Sec.4.4) between peers using Thyone to qualitatively analyse learners' regulatory behaviours (see Sec. 5.1). A total of N=136 students with access to Thyone took part in these conversation sessions. Since these conversations were recorded during the treatment phase, it consists of participants who were not funnelled through the exit survey at the end of the study, explained next.

At the end of the course, we invited all students in the evaluation study to fill in an exit survey, presenting them with the same IMI instrument they had answered at the beginning of the study. Not all students participating filled in the exit questionnaire. We received responses from N=153 of them, out of which five did not participate in any activity during the course, and six did not take the final exam. Since we were unable to capture and associate the objective behaviour data and learning outcomes of these 11 students, we decided to eliminate them from the quantitative analysis. Therefore, even though we conducted our study with 281 participants, we performed our qualitative analysis of conversations with N=136 treatment group participants, and performed the quantitative analysis with a total of N=142: 98 were in the treatment group, and 44 in the control group. Six participants, including three respondents each from pre- and post-test IMI questionnaire selected through a raffle draw, received a monetary incentive of USD 55 to participate in the questionnaires.

Finally, we invited N=13 of these participants for an interview. We selected these participants based on their activity in Thyone and their initial self-reported motivation in programming. We conducted all of the interviews online via Zoom, and the participants' audio, video and screen-shares were recorded. The interview questions (available in Supp. 3) were designed to understand students' motivation and behaviour when using Thyone, and to gather feedback on their perception of using Thyone in their learning processes. Each of the interview participants received a monetary incentive of USD 33. The interviews lasted M = 45.53 (SD = 9.50) minutes. At the end of the interviews 9 hours and 52 minutes of total recording were captured.

4.3 Participants

An overview of the participant demographics is present in Table 1. To analyse the conversation sessions and identify instances of regulatory behaviour, the participants (N = 136) were 81 males and 55 females, with a mean age of M = 20.3 (SD = 2.6) and a mean initial interest of M = 4.47 (SD = 1.22) — all of whom took part in conversation sessions using Thyone during the treatment phase. For the quantitative analysis, the treatment group members (N = 98) were M = 20.6 (SD = 2.9) years old, with 57 males and 41 females, and had a M = 4.38 (SD = 1.30) initial programming interest.[9] The control group participants, aged M = 20.8 (SD = 1.7), with 17 males and 27 females, had a mean initial interest of M = 4.23 (SD = 1.46). The 13 interview participants (for details see Supp. 4) were 8

[9]Interest is computed averaging several Likert measures taken on a 7-point scale (1:Not at all true – 7:Very true), more in Sec. 4.4.

males and 5 females, with a mean age of M = 20.9 (SD = 3.0), and a mean initial interest of M = 4.73 (SD = 1.05). Twelve were from the treatment and one from the control group.[10]

4.4 Metrics

Thyone Usage. Each user's Thyone usage behaviour is defined by their Flowchart Interactions and Conversation Intensity. The *Flowchart interactions* of a student signifies the aggregate of their interactions with the Flowchart feature that resulted in a change in its state at any given time. A change in a Flowchart's state includes adding or deleting a block, adding or deleting an edge, editing the block's content, and editing the edge label. In Thyone, when at least one back and forth message exchange occurs between the connected pairs, we refer to these chats as *conversation sessions*. All the conversations took place using the local language. The Conversation Intensity of each user constitutes their (i) total word count of shared text messages, denoted as *Messages Word Count*, (ii) total number of their shared code-cells, *Shared Cells* and (iii) total number of shared flowchart, *Shared Flowcharts*, for all the conversation sessions that one has participated in during the treatment phase.

IMI measures. Intrinsic Motivation Inventory (IMI) [65] is a multi-dimensional measurement device grounded in SDT used to assess participants' subjective experiences of an activity. This instrument determines levels of intrinsic motivation using the following set of sub-scales: Interests/Enjoyment, Perceived Competences, Efforts, Values/Usefulness, Pressure/Tension, Perceived Choices, and Relatedness. The IMI has been used in several studies focused on intrinsic motivation and self-regulation [18, 24]. We also validated the robustness of this scale in capturing learning motivation by modeling its predictive relationship with students' performance (see Sec. 5.4). The authors of this scale pointed out that, it is necessary to adjust the instrument according to specific tasks and fields. Thyone affordances aim to promote co-regulation, which entails learners demonstrating autonomy and self-regulation, as well as, interaction with peers to regulate and influence each other's learning processes. Thus we used the following sub-scales in our study: the *Perceived Choice* sub-scale is theorised to be a positive predictor of intrinsic motivation and relates to the autonomy BPN posited by the SDT; the *Relatedness* sub-scale reflects the relatedness BPN in SDT; and finally the *Interest* sub-scale represents the most direct measure of self-reported intrinsic motivation [65]. SDT emphasises that higher intrinsic interest in activities reflects higher self-endorsed motivation and intrinsic regulation [64]. This has also been validated by other studies investigating motivation in the context of learning [48]. Hence we use the *Interest* measure as one of the motivational constructs in our analysis. According to SDT, for a learner to be intrinsically motivated, their BPNs of autonomy and relatedness needs to be satisfied. Therefore, to comprehensively measure students' intrinsic motivation, we use the three sub-scales, each contributing to determining learners' intrinsic motivation. The sub-scales of *Interest* and *Perceived Choice*, each comprise 7 rating items, and the *Relatedness* sub-scale has 8 rating items - all on a 7-point scale (1:Not at all true – 7:Very true).

Student Performance. We assessed student performance by calculating the number of questions each student correctly answered in their final exam. The exam consisted of 21 multiple-choice questions, each with 6 answer-items, for which there could either be one or multiple correct response. The question was considered correct only if all correct options were selected by the student. On average students identified 11.23 correct answers (3.78 SD).

[10]One participant was chosen from the control group to compare his experience to those who had access to Thyone.

4.5 Analysis

We used both qualitative and quantitative approaches to analyse the data collected during the study as explained below.

4.5.1 Analysis of conversations in Thyone. To address RQ1 and identify occurrences of learning co-regulation, we qualitatively coded and analysed students' conversations recorded in Thyone during the treatment phase. This is because conversations in Thyone are appropriated via the Discuss feature along with the ability to share Flowchart and code cells using the Share Cell feature – therefore providing the collaborative setting of Thyone for social modes of learning regulation to emerge.

To guide our analysis of the conversation sessions in Thyone, we defined the learning regulatory behaviours of these sessions as follows. Co-regulation is appropriated by self-regulating peers who transiently mediate regulation amongst self and others in a social environment by recognising regulatory support distributed amongst peers, tasks, and environments [30]. Hence, we described *co-regulatory* behaviours in Thyone's conversation sessions as instances of learning-related exchanges between peers to help one another understand programming concepts or problems, solve errors, or discuss problems and lessons, such that either or both of the peers in the conversation received some support in their programming comprehension or implementation ability. On the other hand, we defined the *self-regulatory* behaviours in these conversations as instances when one of the connected peers initiated a learning-related exchange like seeking help, feedback or discussion to aid their programming learning, but their processes were not supported by the connected peers (i.e., did not lead to co-regulation).

Each of the conversation sessions was first coded using *Process* coding [69, p. 96]. We identified all instances of students' discussions that depicted learning exchanges, articulated learning challenges, and related responses and then assigned each instance a code describing the interaction. Guided by the above definitions of regulation, we assessed which codes reflected co-regulatory behaviours in each of these instances. We distinguished these from the instances that were self-regulatory behaviour but were not fulfilled into becoming co-regulation. We also identified conversations that only had social exchanges, distinguishing them from learning-related ones. This analysis was followed by *Pattern* [69, p. 212] coding on the learning regulatory behaviour codes generated from the first phase. We analysed code commonality and identified the emergent patterns for co-regulatory and self-regulatory behaviours. The first author and a research assistant independently carried out the entire coding process on randomly sampled 25% (n=58) of the conversation sessions. The two coders then discussed and achieved a strong agreement level of 91%. The agreement level was computed following the equation proposed by Miles et al. [46]: *agreement level = no. of agreements / (total no. of agreements + disagreement)*. The remaining conversation sessions were then coded by the first author.

4.5.2 Quantitative Analysis. Several statistical analyses were performed on the quantitative data to highlight its main characteristics. The motivational constructs of IMI, were measured by computing the sub-scales of *Interest*, *Perceived Choice* and *Relatedness*. To address RQ2 our goal was to first statistically investigate the relationship between students' Thyone usage behaviour and their motivational constructs. To do this we used Thyone usage data from all the participants (i.e. both control and treatment groups) to analyse its relationship with their post-test motivational constructs by estimating multiple linear regression models for each IMI sub-scale. The post-test measures of *Interest*, *Perceived Choice* and *Relatedness* sub-scales formed the dependent variables for their respective models. The constituents of Thyone's usage, i.e. the flowchart interactions, conversation sessions' message word count, cell-shared count, flowchart shared count, respectively, formed the multiple independent variables for each model. The predictors were checked for collinearity assumptions

using Variation Inflation Factor analysis [45], and multi-collinearity was not determined to be a concern. It should be noted that a certain degree of individual interest primarily resides within an individual for a particular activity [52]. So, to reduce bias and any confounding influence, we used learners' pre-test interest measures as a control variable along with the independent variables of Thyone's usage to predict the model for post-test Interest. The best fit model for each case was selected using the Akaike Information Criterion (AIC).

Next, we employed the treatment group (N = 98) and control group (N = 44) to statistically interpret the two group's difference in their post-test programming interest, perceived choice, and relatedness. We assessed for potential selection bias at the onset of the study by comparing pre- and post-test measures of users' reported Interest between the two groups. The pre- and post-test measures in each of the three IMI sub-scales were assessed for parametric evaluation using the Shapiro-Wilk normality test and homogeneity of variance using the Bartlett's test. Except for pre-test relatedness measures, the evaluation was not satisfied for the rest of the IMI measures. For all the measures the Bartlett test confirmed assumption of equality of variance. Hence to compare the data between the two groups, parametric data were subjected to t-test and non-parametric data to Mann-Whitney's U Test.

In addition to this, we ran an exploratory sub-group analysis to explore participants' co-regulatory behaviours with their interest measures. For this, we first considered all the 98 participants from the treatment group. During this analysis, out of these 98, we found that N=24 people had a conversation intensity of zero, that is, they did not share any messages, code-cells, or flowcharts using Thyone, over the entire treatment phase. This means they had not used Thyone for any interactions with peers during the entire time and were thus not included in this analysis. Hence we ran the exploratory sub-group analysis with N=74 treatment group participants. We grouped these 74 participants based on whether or not they took part in co-regulatory conversation instances, depicted as blue boxes in Fig. 3. We found 53 participants took part in at least one successful co-regulatory instance, forming our co-regulating group. And 21 participants, who were not a part of any successful co-regulatory instance, formed our non-co-regulating groups. The data for the groups was assessed for parametric evaluation using the Shapiro-Wilk normality test which was not satisfied. Hence Mann-Whitney's U Test was used for the analysis. All the tests were run in R, and the graphs were created with the `ggstatsplot` [53] and `ggplot2` packages.

4.5.3 Analysis of Interviews. Finally, we also analysed interviews with *Initial* coding [69, p. 100] to attune ourselves with users' thoughts and opinions on Thyone and comprehend what aspects of the extension they appreciated or found frustrating. This analysis was conducted to bring further evidence to quantitative findings and was coded by the first author.

5 RESULTS

Our results [11] show that features embedded in Thyone facilitated learners' remote co-regulation and have the potential to sustain their motivational constructs. Its use appeared to foster learners' programming interest, autonomy, and relatedness (Fig. 4) — although an overall significant influence on intrinsic motivation was not observed. Next, we review these results in detail.

5.1 Conversation behaviour in Thyone

A total of 231 conversation sessions were recorded during the treatment phase (Fig. 3). Each of these conversations took place between unique pairs randomly formed from a total of N=136 Thyone users during the treatment phase. These were analysed to address RQ1, by understanding the types of social learning regulatory behaviours reflected in the peer interactions using Thyone.

[11] All analysis scripts, including R workbooks, detailed outputs, etc are available in Supp. 5.

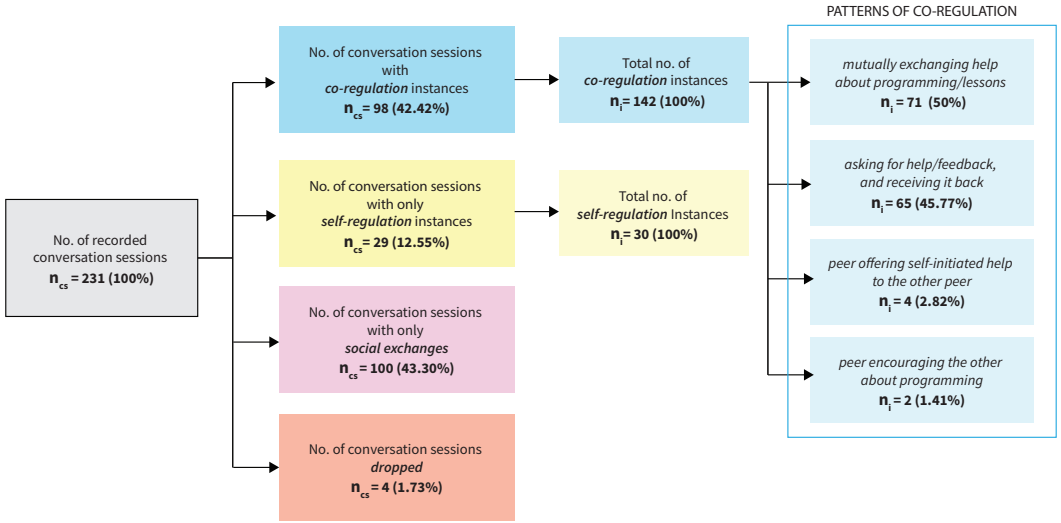

Fig. 3. Conversation Behaviour : Summary of the regulatory instances and social exchanges identified in the conversation sessions recorded during the treatment phase. n_{cs} represents the number of conversation sessions in which respective regulatory and social exchanges took place. n_i represents the number of occurrences of the different types of exchanges corresponding to their n_{cs}.

5.1.1 Co-regulatory behaviours. Firstly, across 98 out of the 231 conversation sessions (42.42 %), we identified a total of 142 instances of co-regulatory exchanges between the pairs in these conversations ($M = 1.45$, $SD = 1.08$). Four different co-regulatory behaviour patterns were reflected across all the instances, explained below:

- *mutually exchanging help about programming/lessons*: The most common pattern identified in these interactions is *mutually exchanging help about programming/lessons*, occurring across 71 instances (or 50%). In all of these instances peers reciprocally shared cells to communicate their errors or to show correct answers and discuss. Six of these instances also involved sharing Flowcharts. They individually worked on their respective exercises or lessons while actively discussing, explaining and assisting each other with challenges they faced in solving programming problems or understanding parts of the lesson, as seen in the example in Table 2 (right).
- *asking for help/feedback, and receiving it back*: This is the second most common pattern we identified which took place 65 times (or 45.77%). In the instances of *asking for help/feedback, and receiving it back*, conversations were initiated to seek help or feedback in understanding a concept or to solve problems and were mostly accompanied with shared cells. The responses from the peers to provide support were, however, varied. In many cases, the responding peers explained the problem and shared their answer cells. Sometimes they did not have the solution but shared what they had been trying to solve. In a few cases, the peer only shared the answer without explaining. Fifty-three out of these 65 co-regulation instances noted at least sharing of one cell in the conversation either to express a problem or provide help and five of these instances also included sharing of flowcharts.
- *peer offering self-initiated help to the other peer*: In four different instances (or 2.82%), we found a peer offering self-initiated help to the other peer. It resulted in sharing cells and related discussion amongst each other in all of these instances.

Self-regulatory behaviour in a conversation session	*Co-regulatory behaviour in a conversation session*
	S16 : "Hey, tell me if you ever want to compare" **S16** : "I have a result but I'm not sure if I did it right" **S126** : <cell shared> **S126** : "I have this" **S16** : "Yes I must have done wrong" **S16** : <cell shared> **S16** : "You have a very different code?" **S126** : "ahh I see, instead of putting triSelection (sizes), I put triSelection(l)" **S16** : <cell shared>
S141 : <cell shared> **S141** : "How do you add to a list? " **S141** : "towards the end of the prog" **S24** : "no idea, sorry :("	**S16** : "I have an even weirder graph ahahhaha" **S126** : "ahh yeah" **S126** : <cell shared> **S16** : "ahhhh " **S126** : "you forgot the 2nd random.shuffle right?" **S16** : "yes precisely " **S16** : "the graph is less weird but it's still not right" **S16** : "I would see the correction I think but it's weird I don't have the same thing" **S16** : "I'm looking but thanks a lot!" **S126** : "okay, no worries!" **S16** : "ahhh voila" **S16** : "I had put for size in range(len(size))" **S16** : "while I have to iterate on the sizes list"

Table 2. Examples of regulatory behaviour in a *conversation session*: (left) an example of a communication attempt that demonstrated only self-regulation; (right) an example of co-regulation where peers mutually exchange help.

- *peer encouraging the other about programming*: We observed this less common occurrence of one peer encouraging the other about programming on two occasions. When a peer expressed their dissatisfaction with being stuck and lost in programming, their connected peer addressed the issues and motivated them by suggesting that with practice, things would get better.

5.1.2 Self-regulatory behaviours. Across 29 (12.55%) conversation sessions, we recorded a total of 30 instances of self-regulatory behaviours, which, however, did not appropriate to co-regulation in these conversation exchanges. Self-regulatory behaviours in these conversations imply instances in which one of the connected peers initiated a learning-related exchange seeking help, feedback or discussion to adapt their own regulatory activity in programming learning. However, their processes were not supported by the connected peers — either because the peer was in a different lesson, did not know the solution, diverted to a non-lesson related discussion, or did not respond — therefore did not lead to co-regulation. From the example in Table 2 (left), it is evident that participant S141 is actively self-regulating own learning to overcome a knowledge gap. Even though S141 initiated a co-regulatory exchange with the connected peer, it was not fulfilled.

5.1.3 Social interactions. In 100 out of 231 conversation sessions (43.30%), we recorded only social interactions from sharing course-related frustrations to non-lesson related discussions. These interactions did not include any learning related exchanges. Lastly, even though one of the peers initiated a conversation, the remaining four conversation sessions dropped because the other peer's response time was not prompt.

5.1.4 Summary. This section provides insight into students' behaviour patterns of socially regulating online learning using Thyone. The behavioural data suggests that the Discuss feature of Thyone afforded social exchanges, and along with sharing cells and flowcharts, it also allowed for the emergence and sustenance of learner's co-regulatory behaviours remotely.

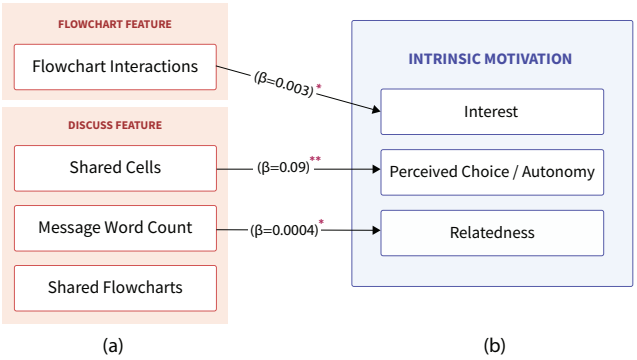

Fig. 4. Results: (a) Thyone usage behaviour represented by Flowchart Interactions, Shared Cells, Message Word Counts and Shared Flowcharts. (b) IMI sub-scales of post-test Interest, Perceived Choice/Autonomy and Relatedness contribute to determining Intrinsic Motivation. The arrow symbol (→) defines significant relationship (∗ = $p < .05$, ∗∗ = $p < .01$) from predictors to outcome variable along with their estimates (β).

5.2 Thyone usage behaviour

To address RQ2, we first explore the relationship between students' objective interactions with Thyone and their post-test motivational constructs. Participants with access to Thyone had a mean flowchart interaction count of $M = 66.29$ ($SD = 50.99$). The summary of their conversation intensity constitute an average messages word count of $M = 298.60$ ($SD = 488.21$) words, an average shared code-cells of $M = 2.30$ ($SD = 5.0$), and an average flowchart shared count of $M = 0.30$ ($SD = 0.92$). Below, we describe how these interactions with Thyone contributed to students' motivational constructs (Fig. 4).

5.2.1 Flowchart Interactions. We found that the Flowchart Interactions component of Thyone's usage behaviour had a significant and positive association with participant's report of greater post-test interest ($\beta = 0.003$, $t_{138} = 2.050$, $p < .05$). In the overall model, Flowchart Interactions and the shared cells count (explained in the next paragraph) demonstrated a significant relationship with post-test interest, while keeping their pre-test interest as a controlling factor in the regression model ($F_{3,138} = 101.5$, $p < .001$, $R^2_{adj} = 0.68$). Since we have taken learners' pre-test interest into account, this finding suggests that participants who were already interested in programming interacted with Flowchart to further self-regulate their learning, potentially contributing to fostering their interest. However, interviews highlighted that Flowchart was used infrequently and only exploratorily because creating a flowchart did not intuitively correspond to learners' programming learning process. They preferred to go step-by-step rather than abstracting the problem and planning before solving. S4 *"So, I tried using a flowchart, but I didn't really understand the point. That's not the way I work to solve a problem, I go one step at a time."*

5.2.2 Shared Cells. The Shared Cells count in participant's usage of Thyone was found to be significantly and positively associated with their higher post-test perceived choice ($\beta = 0.087$, $t_{138} = 3.291$, $p < .01$). In an overall model having a significant relationship between messages word count, shared cells count, and Flowchart Interactions with reported post-test perceived choice ($F_{3,138} = 5.24$, $p < .01$, $R^2_{adj} = 0.08$), only the shared cells count was found to be a significant predictor. Our interview findings also highlighted instances in which students were able to reflect on their mistakes and understanding using this feature. S4 *"More during the course, I was struggling with exercises, I didn't understand where my mistake was and we were able to share our cells. It helped*

me to move forward and understand my mistakes." Another participant mentioned that while trying to help others he was able to reflect on his understanding. S11 *"When someone asked me a question in chat and I was able to answer them, that also helped me progress."* This suggests that the Share Cell feature allowed students to reflect on and regulate their learning, thus feeling more autonomous. Furthermore, even though the effect was not significant, the Shared Cells count was also associated with the post-test interest ($\beta = 0.030$, $t_{138} = 1.758$, $p = .08$, *ns*) of participants, along with Flowchart Interaction being a significant predictor and the pre-test interest being a controlling factor in the overall significant regression model, which analysed their effect on interest.

5.2.3 Messages Word Count. We discovered that participant Message Word Counts for Thyone's text conversations had a significant and positive relationship with their greater post-test relatedness ($\beta = 0.0004$, $t_{140} = 2.409$, $p < .05$). It was the only predictor in an overall significant model for assessing Thyone's usage effect on post-test relatedness ($F_{1,140} = 5.80$, $p < .05$, $R^2_{adj} = 0.03$). This suggests that the more participants engaged in text message conversations, the more it contributed to their sense of relatedness in learning programming. Our interview findings further add to this result by highlighting that five participants used the Discuss feature to foster connections with peers not only for learning-related interactions, but also for socialising. S10, *"I used Thyone to help me or to meet people. In the beginning, I used it more to discuss and get help in errors, and as time went by, I met people."* They also stated that Thyone provided them with the opportunity to connect with their classmates, whom they had not previously met due to the online classes. S6, *"It was good to be able to chat because I didn't know all of the department or those who are in programming course in the morning. So this is a way to socialise and meet people virtually."* However, the Discuss feature also led to certain frustrations induced by lack of flexibility in pairing with random peers who limited productive interactions by being unresponsive. S11, *"There are people who sometimes don't even answer, so there's not much point in being paired with them. And then, you can't easily change the person you're talking to, so you're a bit stuck."* Another source of frustration was that when one disconnected from their Jupyter server, their paired connection was reset when they logged back in. S3, *"With the last person I was talking to, it abruptly got changed to a new pair as he got disconnected, which is an issue since if we found a good partner to work with, it would be nice to stay in the same conversation."*

5.2.4 Summary. We see a positive relationship between student interactions with different features of Thyone and how it shapes their motivational constructs (Fig. 4). It is evident that having the ability to share cells is an autonomy-supportive feature, while being able to exchange messages is a relatedness-supportive feature. Furthermore, even though the Shared Cells count is not a statistically significant predictor, the sharing cell behaviour along with the exploration of flowcharts indicate that the provision to regulate learning processes positively correlates with students' interest in programming.

5.3 Learning motivation

To answer RQ2 more comprehensively and further understand whether the use of Thyone drove students' learning motivation, we compared our control and treatment groups' motivational constructs. Table 1 in Sec.4.3 depicts that the two groups regarding demographic characteristics are balanced.

5.3.1 Interest measures. We assessed for possible selection bias at the onset of the experiment and did not find any significant difference between the control (N=44) and the treatment (N=98) groups on their pre-test interest measures ($U = 2077.5$, $p = .73$, *ns*). Furthermore, at the end of the teaching period, post-test interest measures of the control group and the treatment group did not

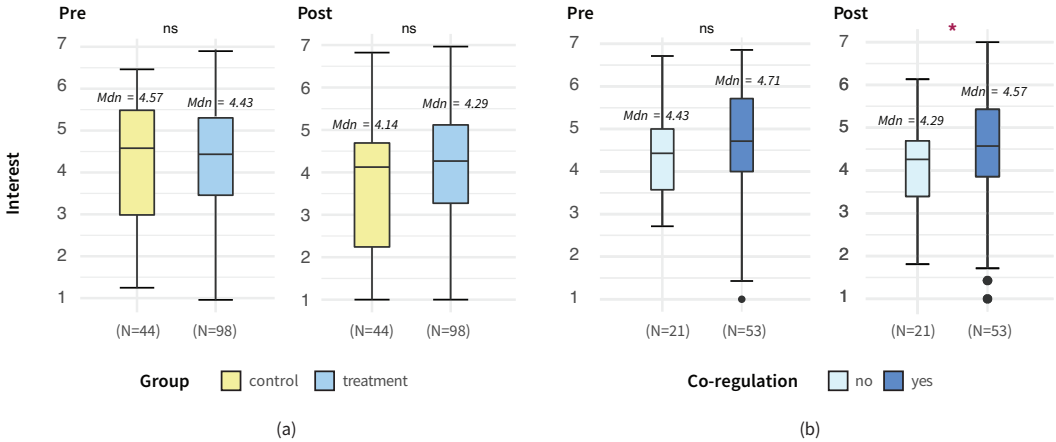

Fig. 5. Interest sub-scale: (5a) Pre and post-test interest measures between the control and treatment groups. (5b) Subgroup analysis with $N = 74$ from treatment group: Pre and post-test interest measures between the users in the treatment condition who did not co-regulate (i.e., "no" group, $N = 21$) and those who co-regulated (i.e., "yes" group, $N = 53$). The remaining 24 treatment group participants were not included in this analysis since they did not use Thyone for any conversation throughout the teaching session. Significant and non-significant differences between groups are denoted by $*$ and ns respectively. Only the medians are reported in both the diagrams.

show a significant difference ($U = 1734$, $p = .06$, *ns*). Even though significant differences in pre- and post-test interests were not found, we observed a negative trend in the changes of interest from pre- to post-test conditions for both groups (Fig. 5a). This negative trend was expected and predicted by the SDT, given the coercive nature of the course, where students are obliged to take the course and comply with a number of requirements and assignments each week.

5.3.2 Perceived Choice and Relatedness measures. On assessing perceived-choice for the two groups, no significant difference was identified in its pre-test measures ($U = 1835$, $p = .157$, *ns*). There was also no significant difference in post-test perceived choice measures found between the control and treatment groups ($U = 1756.5$, $p = .078$, *ns*). Similarly, the pre-test relatedness measures did not significantly differ between the groups ($U = 1745$, $p = .069$, *ns*). Additionally, no significant difference between the groups was found in the post-test relatedness ($U = 2080$, $p = .739$, *ns*).

5.3.3 Sub-group analysis for Interest measures. To get better insights into how co-regulation played a role in driving students' motivation, in an exploratory subgroup analysis (N = 74, explained in sec. 4.5), we compared the non-co-regulating (N=21) and co-regulating (N=53) students on the interest sub-scale. No significant difference was found between the two groups on the pre-test interest sub-scale ($U = 461.5$, $p = 0.257$, *ns*). At the end of the teaching period, a significant difference in their post-test interest between the two groups was found ($U = 391.5$, $p < 0.05$, $r = 0.230$). Fig. 5b shows that the changes of interest from the pre- to post-test phase reflect that higher measures of interest are sustained for participants who co-regulated, even though the trend is negative for both groups.

5.3.4 Students' perception of Thyone. Our findings from the interviews show evidence that Thyone supported the students' remote programming learning process. Eleven out of the 12 participants stated that they primarily used Thyone to receive or provide assistance; six of them said that Thyone assisted them in engaging in meaningful exchanges and debates with their peers to assist

in their programming learning. S6, *"With the extension I was able to talk to some people, exchange ideas and understand some of the problems in the course. I used the Discussion side to debate things with my classmates."* Four of the participants mentioned that Thyone helped them in self-reflecting and understanding their own mistakes. S13, *"Just by saying what I understood, I realise what I didn't understand as well and it helps me improve. So Thyone, for that very reason, is an incredible tool with an amazing potential."* It was also mentioned that Thyone enabled participants to form synergy between peers by making them feel related and connected. One of the participants (S13) said, *"I found it good when all of a sudden we could share with classmates "ah yeah, it's a pain to solve this". It's the fact of realising that we are not alone in having difficulties or we are not completely lost, that makes us move forward."* Another participant could also establish relatedness with a peer that extended to helping each other in other subjects as well. S8, *"I also connected with a peer from the extension and shared some answers and later we tried to work together on other subjects. So it was really helpful."*

5.3.5 Summary. These results indicate that the interest measures and the two SDT constructs of perceived choice and relatedness were not significantly modified during the teaching periods. A plausible explanation for not observing an impact of Thyone on overall motivational constructs could be due to the coercive nature of the course, which possibly overshadowed the effects of Thyone usage. This is also reflected in the exploratory analysis, in which we see higher levels of interest being sustained for co-regulating Thyone users despite a decreasing trend of interest across the treatment phase. We further discuss this finding in detail in Section 6.2.

5.4 Student Performance

To understand whether the self-reported interest of participants at the end of experiment had an influence on their performance, we modelled the two variables in a linear regression. We found that the performance was significantly and positively associated with their self-reported post-test interest in programming ($\beta = 0.80$, $t_{140} = 3.67$, $p < .001$) with an overall model which was also statistically significant ($F_{1,140} = 13.52$, $p < 0.001$, $R^2_{adj} = 0.08$). This suggests that participants with a higher level of interest in programming at the end performed better in their assessment.

5.4.1 Summary. This finding is consistent with SDT's theory that enhancing learners' intrinsic motivation and internalisation leads to higher achievement [68] — further validating the use of IMI scale to capture one's motivation in an educational context.

6 DISCUSSION

This research demonstrates supporting learners' co-regulatory behaviours and in turn their SDT motivational constructs in an online classroom setting for learning programming. Below we discuss the results of our study and provide design implications for collaborative programming learning platforms.

6.1 Supporting co-regulation in online classrooms

Given an online collaborative programming learning setting, we see that learners socially situate their programming self-regulation and mediate interactions with peers to co-regulate their learning processes. Our results depict that regulation in programming is indeed shaped by both self and socio-contextual settings through the dynamic interplay between peers, tasks, and contexts [38]. The presented results provide insights into how learners' programming co-regulation has been enabled and sustained by sharing cells, flowcharts, and productive learning exchanges through the features embedded in Thyone. Furthermore, learners' objective behaviours of using Share Cell facilitated their autonomy, while the Discuss feature enabled them to feel more related when learning to

program remotely. These two features are orthogonal to each other. Interpersonal interactions using the Discuss feature are context-specific. The Discuss feature indeed serves as a chat for socialising conversations and also supports learners' co-regulatory process to unfold - especially when coupled with the sharing of cells or flowcharts that provide a deictic reference to what they refer to in the chat [13]. Therefore, together, these features provide an affordance that enables learners to reflect on their programming errors and comprehension through remote discussion with peers, allowing them to self-regulate, feel related and influence each other's learning. This implies that these two Thyone features together afforded learners' co-regulatory behaviours in programming to emerge and be sustained.

This finding has implications for the design of collaborative pedagogical interventions for introductory programming courses. Incorporating affordances to assist learners' regulation at social levels should be explored in the future design of introductory programming courses in both classroom and online contexts. Programming platforms should integrate the design of collaborative features, like sharing and discussing codes, that allow peers to communicate their understanding, code errors, code outputs, and explanations concisely and clearly. One example would be the live peer code review feature of the tool Puzzle Me [78], which lets students in a group see group members' code and provide reviews in a chat widget during in-class exercise activities. Sharing code in online programming classes can also be facilitated by sharing learners' code editors and running output in the programming environment, as designed in Cocode [11]. This will let learners reflect on their programming learning and co-construct their understanding of the subject with their peers, facilitating opportunities for successful co-regulation. In exploring co-regulation, we concentrated on peer-to-peer interactions in this study. However, different approaches might promote co-regulation through interactions with teachers or teaching assistants and should also be examined in the design of pedagogical interventions for programming. Interactions with teachers enable students to receive structured guidance in their programming learning process, which not only helps in code completion but also stimulates code comprehension. Coding platforms can incorporate deictic code reference features for instructors to provide contextual responses and explanations tied to specific parts of students' codes when they seek help, as demonstrated in EdCode [12]. Furthermore, designs of collaborative pedagogical tasks, coupled with this opportunity to share codes, can also be extended to investigate and support learners' socially shared regulation in programming. For example, in project-based programming activities, where students collaboratively walk-through instructions, debug errors and solve problems, having a shared or synchronous view of their codes can assist in maintaining mutual awareness of their problem-solving approach.

6.2 Supporting intrinsic motivation in programming in formal educational setups

SDT [64] posits that for a learner to be intrinsically motivated, their BPNs of autonomy and relatedness need to be satisfied, along with fostering interest and enjoyment in the task. Our results show that learners' usage of Thyone supported certain degrees of their interest, autonomy and relatedness in programming. However, an overall impact of the treatment manipulation on these constructs to drive intrinsic motivation was not observed. One possible explanation is that Thyone's usage effect was overshadowed by the coercive nature of the course. The course was compulsory and included mandated assignments for students to complete weekly, as described in Section. 4.1. SDT posits that when users act out of pressures that are experienced as controlling, they experience *heteronomy* (the opposite of autonomy) [67, p. 86]. This pushes people's motivation toward the extrinsic end of the SDT motivation continuum, which proceeds from extrinsic motivation at one end to more self-determined and intrinsic motivation on the other. This was likely captured by the negative differences between the pre- and post-scores of the standardised instruments deployed in this study. Specifically, the higher the scores on the sub-scales, the higher the level of *intrinsic*

motivation towards the target activity (i.e., programming). We suspect that the adverse effect of the heteronomous forms of motivation due to the classroom structure thwarted the positive effect Thyone had on users' motivation. This interpretation is supported by the following observation: at the end of the treatment phase, we see that students who co-regulated using Thyone during the teaching period maintained higher levels of *interest* (i.e. the most direct indicator of self-reported intrinsic motivation) in programming. Given the quasi-nature of our experimental setup, we cannot say that the treatment manipulation impacted higher results of interest. However, we observe that learners found support in Thyone for co-regulation, and those who did co-regulate sustained higher *interest* levels during the teaching period, despite the coercive nature of the course.

Formal educational setups in universities are often constrained by classroom instructions, mandated evaluations, and extrinsic rewards like grades. It is impossible to tease apart these extrinsic motivators from classic educational contexts. SDT argues that extrinsic forms of motivation can entice a person to behave in a certain way in the short term to achieve their immediate learning goals. However, to sustain inherent interest for long-term benefits, one's external motivation must be internalised and integrated to become congruent with their core values and be shifted towards the intrinsic end of the motivation continuum. This can be enhanced by providing support for students' BPNs in the learning environment [68]. Therefore, programming learning technologies must consider shaping learners' extrinsic motivation to be driven towards more intrinsic forms. Interventions can incorporate multiple affordances to provide basic psychological needs supportive contexts for learners. For example, solving programming tasks is usually a mandatory activity in introductory courses. In such controlling contexts, interventions should enable learners to progress freely in their tasks and be in ownership of their problem-solving process. In line with our findings, providing opportunities to co-regulate can be an approach to support students' autonomy during problem-solving in classrooms. The social environment of a classroom inherently provides opportunities for students to form connections and feel related. However, it may be challenging in online classrooms to sustain this sense of belonging because of learners' remote presence. Establishing communication channels between peers is imperative to support relatedness in online classes. Nevertheless, to build mutual reliance remote learners could be presented with continuous opportunities of collaborative actions to solve programming problems. Effective peer interactions can be fostered by scaffolding programming activities with Jigsaw scripting [4] or by providing optimal group challenges [68] - particularly if the groups remain consistent over time, it can deepen the potential for relatedness.

6.3 Challenges and Opportunities of Thyone's Features

Thyone's design has certain constraints that might have reduced the effect of the interventions. Firstly, although the use of the Flowchart feature fostered students' interest in programming, it did not significantly contribute to their sense of autonomy. Our qualitative analysis reflected that using the Flowchart feature required students to adopt an active change in their learning behaviour and construct high levels of abstraction from the problem statement independently, without going through a guided abstraction process. The necessity of explicit instructions to teach novices the abstractions of programming problems, along with composing programming plans and goals, has been emphasised in prior work [74]. Even though the Flowchart feature allowed to plan and reflect on goals, it did not have explicit instructions to guide the abstraction process. Furthermore, Flowchart creation, being agnostic to the exercises incorporating it into learning behaviour, posed a challenge for beginners in a span of four months of study period.

Programming necessitates an individual to self-regulate, even in a collaborative context. Using flowcharts to visualise a problem have the potential to support students' self-reflection process. Our findings suggest that designing flowcharts interventions for programming learning should facilitate

guided abstraction of the problem, possibly in an adaptive way based on their evolving expertise. Future iterations of this Flowchart feature may investigate adopting a similar format to Parsons' problem [51]. Rather than providing a blank space to create a flowchart, the design can scaffold concepts linked with a particular programming problem into relevant concept fragments as nodes for the flowchart. These fragments can be presented to students in a drag-and-drop manner, as a puzzle or by filling in the blanks to complete the flowchart. Further, the level of obscurity can be adjusted based on students' progress through the lessons until they are able to do the abstractions themselves.

Secondly, although using the Discuss feature allowed students to connect with peers, it still posed a constraint on relatedness. This is attributed to the random pairing assignment and discreteness of notifications that led to frustrations when students could not engage in productive interactions, missed a message, or when their peers changed. Forming pairs or groups at random may result in individuals exhibiting unequal engagement, off-task behaviour, and reluctance to group work [35]. Furthermore, prior work has shown that familiarity among group members leads to a more favourable and joyful collaborative experience for group members [36]. Randomly pairing does not guarantee to pair members with desired characteristics, therefore constraining students from sharing productive exchanges and a notion of relatedness.

It is evident from our findings that the role of pairing students had an influence on facilitating co-regulation. Therefore group formation deserves attention while designing collaborative affordances for programming. Depending on the scale of a collaborative setting, designers can explore grouping aspects across multiple social planes by incorporating strategies like orchestration [20] or by allowing more flexibility in group formation when designing co-regulatory affordances. Subsequent iterations of the extension can explore intelligent group-forming mechanisms to construct heterogeneous learner groups based on their evolving competence to provide an optimal situation for productive co-regulation. We plan to explore some of these avenues in future work.

6.4 Study Limitations

Our study has several limitations. Firstly, given the ethical constraints, we implemented a quasi-experimental methodology in our study, and students were not randomly assigned to groups. Although we did not find a difference in their programming interest at the onset, there might be other biases in the selection process which we did not consider. Secondly, the course in which the study was conducted was *mandatory* and had an overall constraining effect on students' intrinsic motivation. As discussed in Sec. 6.2, these constraints are usually ingrained in formal education setups and are inevitable. Other non-formal learning environments, could also include extrinsic motivators in their pedagogical setup, such as fees to access courses or to obtain certificates on completion. Therefore, the design of pedagogical technology must recognise the controlling aspects in learning settings and provide support to help students internalise their extrinsic motivation towards becoming more self-determined and to instill long-term values. Thirdly, because the deployment of Thyone was only for a four month period, we believe that this data could only capture the influence of Thyone to a limited extent. If the study had been conducted for a longer length of time, it could have reflected students' behaviour with Thyone more comprehensively and students could have better appropriate the intervention in their practices, allowing us to gain a thorough grasp of its effect. It is important to notice that most university courses typically span only a few months. Therefore it is important to design interventions that could yield appreciable effects in short time periods. Fourthly, we could not fully monitor (and control) co-regulation instances that might have been possible through other communication channels like WhatsApp, Discord, etc. Therefore, we look at co-regulation only through the lens of Thyone and might have overlooked co-regulatory interactions that unfolded in physical or other learning settings. Finally, students

usage of Thyone and its effect on their interest and relatedness may have been influenced by the study's unique time and context, which took place during the COVID-19 lockdown when students were compelled to attend classes remotely.

7 CONCLUSION

In this paper, we explored assisting students' social learning regulation, specifically co-regulation and their learning motivation, in an online introductory programming classroom. We present Thyone, a collaborative extension for Jupyter Notebooks, to support students' meta-cognitive reflection through flowcharts and promote their co-regulatory learning behaviours through sharing and discussing code, errors, and outputs with their peers. We provide empirical insights into how Thyone's embedded features shape learners' motivational constructs of interest, autonomy, and relatedness and foster their co-regulatory behaviours. We discuss the implications of our findings to inform the design of CSCL for learning programming. Our study highlights that pedagogies for teaching introductory programming online should not only focus on nurturing learners' self-regulation, but also on fostering their learning regulations at social levels and integrating suitable interventions to assist these social regulations in programming. Co-regulation has the potential to spawn more proficient self-regulation, as well as social regulation [30] making it an essential process in co-learning and in co-working. Therefore, designing for co-regulation should be extensively adopted not only in computing education, but also in broader applications in other scientific domains to facilitate effective collaboration.

ACKNOWLEDGMENTS

We sincerely thank Marc-Olivier Boldi for providing insightful feedback on the quantitative data analysis of the paper and Pegah Sadat Zeinoddin for helping with the qualitative coding process of the conversation data. We also sincerely thank Kavous Salehzadeh Niksirat and Pooja Rao for providing valuable feedback on the earlier versions of the paper and we thank Kévin Huguenin, Patrick Jermann and Yamane El Zein for reading earlier versions of the paper. Finally, we thank Vincent Vandersluis for proofreading the article.

REFERENCES

[1] Ahmad Al-Jarrah and Enrico Pontelli. 2014. "AliCe-ViLlagE" Alice as a Collaborative Virtual Learning Environment. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE, Madrid, Spain, 1–9. https://doi.org/10.1109/FIE.2014.7044089

[2] C. M. Allwood. 1986. Novices on the Computer: A Review of the Literature. *Int. J. Man-Mach. Stud.* 25, 6 (Dec. 1986), 633–658. https://doi.org/10.1016/S0020-7373(86)80079-7

[3] Maryi Arciniegas-Mendez, Alexey Zagalsky, Margaret-Anne Storey, and Allyson Fiona Hadwin. 2017. Using the Model of Regulation to Understand Software Development Collaboration Practices and Tool Support. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. Association for Computing Machinery, New York, NY, USA, 1049–1065. https://doi.org/10.1145/2998181.2998360

[4] Elliot Aronson, Nancy Blaney, Cookie Stephan, Jev Sikes, and Matthew Snapp. 1978. *The Jigsaw Classroom*. CA: Sage Publishing Company, Beverly Hills.

[5] Susan Bergin and Ronan Reilly. 2005. The influence of motivation and comfort-level on learning to program. In *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group, PPIG 05*. Psychology of Programming Interest Group, Brighton, UK, 293–304. https://mural.maynoothuniversity.ie/8685/

[6] Susan Bergin, Ronan Reilly, and Desmond Traynor. 2005. Examining the Role of Self-Regulated Learning on Introductory Programming Performance. In *Proceedings of the First International Workshop on Computing Education Research* (Seattle, WA, USA) *(ICER '05)*. Association for Computing Machinery, New York, NY, USA, 81–86. https://doi.org/10.1145/1089786.1089794

[7] Marcel Borowski, Johannes Zagermann, Clemens N. Klokmose, Harald Reiterer, and Roman Rädle. 2020. Exploring the Benefits and Barriers of Using Computational Notebooks for Collaborative Programming Assignments. In *Proceedings*

of the 51st ACM Technical Symposium on Computer Science Education (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 468–474. https://doi.org/10.1145/3328778.3366887

[8] Kristy Elizabeth Boyer, August A. Dwight, R. Taylor Fondren, Mladen A. Vouk, and James C. Lester. 2008. A Development Environment for Distributed Synchronous Collaborative Programming. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (Madrid, Spain) *(ITiCSE '08)*. Association for Computing Machinery, New York, NY, USA, 158–162. https://doi.org/10.1145/1384271.1384315

[9] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting*, Vol. 1. American Educational Research Association, Vancouver, Canada, 25.

[10] John Seely Brown, Allan Collins, and Paul Duguid. 1989. Situated Cognition and the Culture of Learning. *Educational Researcher* 18, 1 (1989), 32–42. https://doi.org/10.3102/0013189X018001032

[11] Jeongmin Byun, Jungkook Park, and Alice Oh. 2021. Cocode: Providing Social Presence with Co-Learner Screen Sharing in Online Programming Classes. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2, Article 300 (oct 2021), 28 pages. https://doi.org/10.1145/3476041

[12] Yan Chen, Jaylin Herskovitz, Gabriel Matute, April Wang, Sang Won Lee, Walter S Lasecki, and Steve Oney. 2020. EdCode: Towards Personalized Support at Scale for Remote Assistance in CS Education. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Dunedin, New Zealand, 1–5. https://doi.org/10.1109/VL/HCC50065.2020.9127260

[13] Mauro Cherubini and Pierre Dillenbourg. 2007. The Effects of Explicit Referencing in Distance Problem Solving over Shared Maps. In *Proceedings of the 2007 ACM International Conference on Supporting Group Work* (Sanibel Island, Florida, USA) *(GROUP '07)*. Association for Computing Machinery, New York, NY, USA, 331–340. https://doi.org/10.1145/1316624.1316674

[14] Michelene T.H. Chi, Miriam Bassok, Matthew W. Lewis, Peter Reimann, and Robert Glaser. 1989. Self-explanations: how students study and use examples in learning to solve problems. *Cognitive Science* 13, 2 (April 1989), 145–182. https://doi.org/10.1207/s15516709cog1302_1

[15] T.D. Cook and D.T. Campbell. 1979. *Quasi-experimentation: Design & Analysis Issues for Field Settings*. Houghton Mifflin, Boston, MA, USA. https://books.google.ch/books?id=BFNqAAAAMAAJ

[16] Michael de Raadt, David Lai, and Richard Watson. 2007. An Evaluation of Electronic Individual Peer Assessment in an Introductory Programming Course. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88* (Koli National Park, Finland) *(Koli Calling '07)*. Australian Computer Society, Inc., AUS, 53–64.

[17] Edward Deci and Richard M. Ryan. 1985. *Intrinsic Motivation and Self-Determination in Human Behavior*. Springer US, New York, USA. 372 pages. https://doi.org/10.1007/978-1-4899-2271-7

[18] Edward L. Deci, Haleh Eghrari, Brian C. Patrick, and Dean R. Leone. 1994. Facilitating Internalization: The Self-Determination Theory Perspective. *Journal of Personality* 62, 1 (1994), 119–142. https://doi.org/10.1111/j.1467-6494.1994.tb00797.x

[19] Edward L. Deci, Richard M. Ryan, and Geoffrey C. Williams. 1996. Need satisfaction and the self-regulation of learning. *Learning and Individual Differences* 8, 3 (1996), 165–183. https://doi.org/10.1016/S1041-6080(96)90013-8 Special Issue: A Symposium on Self-Regulated Learning.

[20] Pierre Dillenbourg, Sanna Järvelä, and Frank Fischer. 2009. The Evolution of Research on Computer-Supported Collaborative Learning. In *Technology-Enhanced Learning*, Nicolas Balacheff, Sten Ludvigsen, Ton de Jong, Ard Lazonder, and Sally Barnes (Eds.). Springer Netherlands, Dordrecht, 3–19. https://doi.org/10.1007/978-1-4020-9827-7_1

[21] César Domínguez, Arturo Jaime, Jónathan Heras, and Francisco J. García-Izquierdo. 2019. The Effects of Adding Non-Compulsory Exercises to an Online Learning Tool on Student Performance and Code Copying. *ACM Trans. Comput. Educ.* 19, 3, Article 16 (Jan. 2019), 22 pages. https://doi.org/10.1145/3264507

[22] Anneli Eteläpelto. 1993. Metacognition and the Expertise of Computer Program Comprehension. *Scandinavian Journal of Educational Research* 37, 3 (Jan. 1993), 243–254. https://doi.org/10.1080/0031383930370305

[23] Katrina Falkner, Rebecca Vivian, and Nickolas J.G. Falkner. 2014. Identifying Computer Science Self-Regulated Learning Strategies. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (Uppsala, Sweden) *(ITiCSE '14)*. Association for Computing Machinery, New York, NY, USA, 291–296. https://doi.org/10.1145/2591708.2591715

[24] Vincet F. Filak and Kennon M. Sheldon. 2003. Student Psychological Need Satisfaction and College Teacher-Course Evaluations. *Educational Psychology* 23, 3 (2003), 235–247. https://doi.org/10.1080/0144341032000060084

[25] Kasper Fisker, Davin McCall, Michael Kölling, and Bruce Quig. 2008. Group Work Support for the BlueJ IDE. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (Madrid, Spain) *(ITiCSE '08)*. Association for Computing Machinery, New York, NY, USA, 163–168. https://doi.org/10.1145/1384271.1384316

[26] Dilrukshi Gamage. 2021. Scaffolding Social Presence in MOOCs. In *Asian CHI Symposium 2021* (Yokohama, Japan) *(Asian CHI Symposium 2021)*. Association for Computing Machinery, New York, NY, USA, 140–145. https://doi.org/10.1145/3429360.3468198

[27] Anabela Gomes, Wei Ke, Chan-Tong Lam, Maria José Marcelino, and António Mendes. 2018. Student motivation towards learning to program. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, San Jose, CA, USA, 1–8. https://doi.org/10.1109/FIE.2018.8659134

[28] Frederick J. Gravetter and Lori-Ann B. Forzano. 2012. *Research methods for the behavioral sciences* (4th ed ed.). Wadsworth, Australia ; Belmont, CA.

[29] Thomas D. Griffin, Jennifer Wiley, and Carlos R. Salas. 2013. *Supporting Effective Self-Regulated Learning: The Critical Role of Monitoring*. Springer New York, New York, NY, 19–34. https://doi.org/10.1007/978-1-4419-5546-3_2

[30] Allyson Hadwin, Sanna Järvelä, and Mariel Miller. 2017. Self-Regulation, Co-Regulation, and Shared Regulation in Collaborative Learning Environments. In *Handbook of Self-Regulation of Learning and Performance* (2 ed.), Dale H. Schunk and Jeffrey A. Greene (Eds.). Routledge, New York, 83–106. https://doi.org/10.4324/9781315697048-6

[31] Allyson Fiona Hadwin, Lori Wozney, and Oonagh Pontin. 2005. Scaffolding the Appropriation of Self-regulatory Activity: A Socio-cultural Analysis of Changes in Teacher–student Discourse about a Graduate Research Portfolio. *Instructional Science* 33, 5-6 (Nov. 2005), 413–450. https://doi.org/10.1007/s11251-005-1274-7

[32] Yasuhiro Hayashi, Ken-Ichi Fukamachi, and Hiroshi Komatsugawa. 2015. Collaborative Learning in Computer Programming Courses That Adopted the Flipped Classroom. In *2015 International Conference on Learning and Teaching in Computing and Engineering*. IEEE, Taipei, Taiwan, 209–212. https://doi.org/10.1109/LaTiCE.2015.43

[33] Hui-Ching Kayla Hsu, Cong Vivi Wang, and Chantal Levesque-Bristol. 2019. Reexamining the Impact of Self-Determination Theory on Learning Outcomes in the Online Learning Environment. *Education and Information Technologies* 24, 3 (May 2019), 2159–2174. https://doi.org/10.1007/s10639-019-09863-w

[34] Janet Hughes and D. Ramanee Peiris. 2006. ASSISTing CS1 Students to Learn: Learning Approaches and Object-Oriented Programming. *SIGCSE Bull.* 38, 3 (June 2006), 275–279. https://doi.org/10.1145/1140123.1140197

[35] Seiji Isotani, Akiko Inaba, Mitsuru Ikeda, and Riichiro Mizoguchi. 2009. An ontology engineering approach to the realization of theory-driven group formation. *International Journal of Computer-Supported Collaborative Learning* 4, 4 (2009), 445–478.

[36] Jeroen Janssen, Gijsbert Erkens, Paul A. Kirschner, and Gellof Kanselaar. 2009. Influence of group member familiarity on online collaborative learning. *Computers in Human Behavior* 25, 1 (2009), 161–170. https://doi.org/10.1016/j.chb.2008.08.010

[37] Tony Jenkins. 2002. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, Vol. 4. LTSN-ICS, Loughborough, 53–58.

[38] Hanna Järvenoja, Sanna Järvelä, and Jonna Malmberg. 2015. Understanding Regulated Learning in Situative and Contextual Frameworks. *Educational Psychologist* 50 (07 2015), 204–219. https://doi.org/10.1080/00461520.2015.1075400

[39] James P Lantolf. 2000. Introducing sociocultural theory. *Sociocultural theory and second language learning* 1 (2000), 1–26.

[40] Ramon Lawrence. 2004. Teaching Data Structures Using Competitive Games. *Education, IEEE Transactions on* 47 (12 2004), 459 – 466. https://doi.org/10.1109/TE.2004.825053

[41] Paul Luo Li, Andrew J. Ko, and Jiamin Zhu. 2015. What Makes a Great Software Engineer?. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, Florence, Italy, 700–710. https://doi.org/10.1109/ICSE.2015.335

[42] Dastyni Loksa, Andrew J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. 2016. Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 1449–1461. https://doi.org/10.1145/2858036.2858252

[43] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus) *(ITiCSE 2018 Companion)*. Association for Computing Machinery, New York, NY, USA, 55–106. https://doi.org/10.1145/3293881.3295779

[44] Qusay H. Mahmoud, Wlodek Dobosiewicz, and David Swayne. 2004. Redesigning Introductory Computer Programming with HTML, JavaScript, and Java. *SIGCSE Bull.* 36, 1 (March 2004), 120–124. https://doi.org/10.1145/1028174.971344

[45] Jeremy Miles. 2014. Tolerance and Variance Inflation Factor. https://doi.org/10.1002/9781118445112.stat06593

[46] Matthew B Miles, A Michael Huberman, and Johnny Saldaña. 2019. Qualitative data analysis: A methods sourcebook. https://us.sagepub.com/en-us/nam/qualitative-data-analysis/book246128

[47] Iain Milne and Glenn Rowe. 2002. Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies* 7 (03 2002), 55–66. https://doi.org/10.1023/A:1015362608943

[48] Vera Monteiro, Lourdes Mata, and Francisco Peixoto. 2015. Intrinsic motivation inventory: Psychometric properties in the context of first language and mathematics learning. *Psicologia: Reflexão e Crítica* 28 (2015), 434–443.

[49] Na'ilah Suad Nasir and Victoria M. Hand. 2006. Exploring Sociocultural Perspectives on Race, Culture, and Learning. *Review of Educational Research* 76, 4 (Dec. 2006), 449–475. https://doi.org/10.3102/00346543076004449 Publisher: American Educational Research Association.

[50] Claudia Ott, Anthony Robins, Patricia Haden, and Kerry Shephard. 2015. Illustrating performance indicators and course characteristics to support students' self-regulated learning in CS1. *Computer Science Education* 25, 2 (2015), 174–198. https://doi.org/10.1080/08993408.2015.1033129

[51] Dale Parsons and Patricia Haden. 2006. Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52* (Hobart, Australia) *(ACE '06)*. Australian Computer Society, Inc., AUS, 157–163.

[52] Erika A. Patall. 2013. Constructing motivation through choice, interest, and interestingness. *Journal of Educational Psychology* 105, 2 (May 2013), 522–534. https://doi.org/10.1037/a0030307

[53] Indrajeet Patil. 2021. Visualizations with statistical details: The 'ggstatsplot' approach. *Journal of Open Source Software* 6, 61 (2021), 3167. https://doi.org/10.21105/joss.03167

[54] Oleksandra Poquet, Vitomir Kovanović, Pieter de Vries, Thieme Hennis, Srećko Joksimović, Dragan Gašević, and Shane Dawson. 2018. Social Presence in Massive Open Online Courses. *The International Review of Research in Open and Distributed Learning* 19, 3 (July 2018), 43–68. https://doi.org/10.19173/irrodl.v19i3.3370

[55] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. 2020. What Do We Think We Think We Are Doing? Metacognition and Self-Regulation in Programming. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 2–13. https://doi.org/10.1145/3372782.3406263

[56] Roman Rädle, Midas Nouwens, Kristian Antonsen, James R. Eagan, and Clemens N. Klokmose. 2017. Codestrates: Literate Computing with Webstrates. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 715–725. https://doi.org/10.1145/3126594.3126642

[57] Johnmarshall Reeve. 2013. How students create motivationally supportive learning environments for themselves: The concept of agentic engagement. *Journal of educational psychology* 105, 3 (2013), 579.

[58] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (Nov. 2009), 60–67. https://doi.org/10.1145/1592761.1592779

[59] Michael Robey, Brian R. Von Konsky, Jim Ivins, Susan J. Gribble, Allan Loh, and David Cooper. 2006. Student Self-Motivation: Lessons Learned from Teaching First Year Computing. In *Proceedings. Frontiers in Education. 36th Annual Conference*. IEEE, San Diego, CA, USA, 6–11. https://doi.org/10.1109/FIE.2006.322363

[60] Martin Robillard, Wesley Coelho, and Gail Murphy. 2005. How effective developers investigate source code: An exploratory study. *Software Engineering, IEEE Transactions on* 30 (01 2005), 889– 903. https://doi.org/10.1109/TSE.2004.101

[61] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education* 13, 2 (2003), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

[62] Janine Rogalski and Renan Samurçay. 1990. Chapter 2.4 - Acquisition of Programming Knowledge and Skills. In *Psychology of Programming*, J.-M. Hoc, T.R.G. Green, R. Samurçay, and D.J. Gilmore (Eds.). Academic Press, London, 157–174. https://doi.org/10.1016/B978-0-12-350772-3.50015-X

[63] Alexander Ruf, Andreas Mühling, and Peter Hubwieser. 2014. Scratch vs. Karel: Impact on Learning Outcomes and Motivation. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (Berlin, Germany) *(WiPSCE '14)*. Association for Computing Machinery, New York, NY, USA, 50–59. https://doi.org/10.1145/2670757.2670772

[64] R. Ryan and E. Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *The American psychologist* 55, 1 (2000), 68–78. https://doi.org/10.1037/0003-066X.55.1.68

[65] Richard M Ryan. 1982. Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory. *Journal of personality and social psychology* 43, 3 (1982), 450.

[66] Richard M. Ryan and James P Connell. 1989. Perceived Locus of Causality and Internalizatoin: Examining Reasons for Acting in Two Domains. *Journal of Personality and Social Psychology* 57, 5 (1989), 749–761. https://doi.org/10.1037/0022-3514.57.5.749

[67] Richard M. Ryan and Edward L. Deci. 2017. *Self-Determination Theory Basic Psychological Needs in Motivation, Development and Wellness*. The Guilford Press, New York, USA. 770 pages.

[68] Richard M. Ryan and Edward L. Deci. 2020. Intrinsic and extrinsic motivation from a self-determination theory perspective: Definitions, theory, practices, and future directions. *Contemporary Educational Psychology* 61 (2020),

101860. https://doi.org/10.1016/j.cedpsych.2020.101860
[69] Johnny Saldaña. 2013. *The coding manual for qualitative researchers* (2nd ed.). SAGE, Los Angeles. OCLC: ocn796279115.
[70] Judy Sheard, S. Simon, Margaret Hamilton, and Jan Lönnberg. 2009. Analysis of Research into the Teaching and Learning of Programming. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop* (Berkeley, CA, USA) *(ICER '09)*. Association for Computing Machinery, New York, NY, USA, 93–104. https://doi.org/10.1145/1584322.1584334
[71] Joshua Shi, Armaan Shah, Garrett Hedman, and Eleanor O'Rourke. 2019. *Pyrus: Designing A Collaborative Programming Game to Promote Problem Solving Behaviors*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300886
[72] Leonardo S. Silva. 2020. Investigating the Socially Shared Regulation of Learning in the Context of Programming Education. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 575–576. https://doi.org/10.1145/3341525.3394003
[73] Brian K. Smith and Brian J. Reiser. 1998. National Geographic Unplugged: Classroom-Centered Design of Interactive Nature Films. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Los Angeles, California, USA) *(CHI '98)*. ACM Press/Addison-Wesley Publishing Co., USA, 424–431. https://doi.org/10.1145/274644.274702
[74] E. Soloway. 1986. Learning to program = learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (Sept. 1986), 850–858. https://doi.org/10.1145/6592.6594
[75] E. Soloway and James C. Spohrer. 1988. *Studying the Novice Programmer*. L. Erlbaum Associates Inc., USA.
[76] Alan L. Tharp. 1981. Getting More Oomph from Programming Exercises. In *Proceedings of the Twelfth SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA) *(SIGCSE '81)*. Association for Computing Machinery, New York, NY, USA, 91–95. https://doi.org/10.1145/800037.800968
[77] Serkan Ucan and Mary Webb. 2015. Social Regulation of Learning During Collaborative Inquiry Learning in Science: How does it emerge and what are its functions? *International Journal of Science Education* 37, 15 (Oct. 2015), 2503–2532. https://doi.org/10.1080/09500693.2015.1083634
[78] April Yi Wang, Yan Chen, John Joon Young Chung, Christopher Brooks, and Steve Oney. 2021. PuzzleMe: Leveraging Peer Assessment for In-Class Programming Exercises. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2, Article 415 (oct 2021), 24 pages. https://doi.org/10.1145/3479559
[79] Philip H. Winne and Allyson F. Hadwin. 2008. The Weave of Motivation and Self-Regulated Learning. In *Motivation and Self-regulated Learning: Theory, Research, and Applications*, Dale H. Schunk and Barry J. Zimmerman (Eds.). Routledge, New York, 297–314.
[80] Leon E. Winslow. 1996. Programming Pedagogy—a Psychological Overview. *SIGCSE Bull.* 28, 3 (Sept. 1996), 17–22. https://doi.org/10.1145/234867.234872
[81] Diyi Yang, Tanmay Sinha, David Adamson, and Carolyn Penstein Rosé. 2013. Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses. In *Proceedings of the 2013 NIPS Data-driven education workshop*, Vol. 11. Curran Associates Inc., Lake Tahoe, Nevada, USA, 14. https://cs.stanford.edu/~diyiy/docs/nips13.pdf
[82] Noor Faridatul Zainal, Shahrina Shahrani, Noor Yatim, Rohizah Abd Rahman, Masura Rahmat, and Rodziah Latih. 2012. Students' Perception and Motivation Towards Programming. *Procedia - Social and Behavioral Sciences* 59 (10 2012), 277–286. https://doi.org/10.1016/j.sbspro.2012.09.276
[83] Barry J. Zimmerman. 1986. Becoming a self-regulated learner: Which are the key subprocesses? *Contemporary Educational Psychology* 11, 4 (1986), 307–313. https://doi.org/10.1016/0361-476X(86)90027-5
[84] Barry J. Zimmerman. 2000. Chapter 2 - Attaining Self-Regulation: A Social Cognitive Perspective. In *Handbook of Self-Regulation*, Monique Boekaerts, Paul R. Pintrich, and Moshe Zeidner (Eds.). Academic Press, San Diego, 13–39. https://doi.org/10.1016/B978-012109890-2/50031-7

Received July 2022; revised January 2023; accepted March 2023