Learning to Program: an In-service Teachers' Perspective

1st Majid Rouhani *, 2ndMiriam Lillebo †, 3rd Veronica Farshchian‡, 4th Monica Divitini §
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
*majid.rouhani@ntnu.no †miriam.s.lillebo@ntnu.no ‡veronfa@ntnu.no §divitini@ntnu.no

2022 IEEE Global Engineering Education Conference (EDUCON) | 978-1-6654-4434-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/EDUCON52537.2022.9766781

***Abstract*—Learning to program is often reported as challenging. Difficulties might be connected to, e.g., acquisition of core concepts like iteration, specific language constructs, and program design. Additionally, mastery of a programming language's constructs does not consequently translate into solving new programming problems. These challenges have to be taken into account in the context of in-service teachers' professional development (PD). In this paper, we address challenges teachers face when learning to program, considering these closely related questions: What do teachers perceive as difficult and how does this impact the perception of challenges their students will face? How does this influence the perception of teaching programming? The paper is based on the analysis of 178 reflection notes delivered by in-service teachers attending a university-level course on programming to identify issues related to the research questions. Out of the topics that we have selected from the literature as challenging for newcomers, the most difficult ones for teachers seem to be writing code and pseudocode to solve a specific problem. The easiest ones are understanding variable initialization and if statements. Also, teachers express that pupils will struggle with many aspects of programming and point to the complexity and students' misconceptions inherent in several programming concepts such as loops. Our results suggest that it is not necessarily difficult to learn to program if in-service teachers are given paid time to learn.**

***Index Terms*—In-service teachers, programming, professional development, challenges**

I. Introduction

While there has been a great interest in making computer science a core K-12 academic subject in many countries [1]–[3], there is a shortage of computer science teachers to implement computer science courses successfully. Addressing this challenge requires to provide professional development to in-service teachers, i.e., teachers who are already working in schools and have already completed their main teacher training. Therefore, many organizations offer training programs through teacher professional development to enhance computer science teacher capacity [4], which is essential for any successful change in educational practice [5], including changes related to integrating programming into K-12 schools.

However, there is a consensus in the research community that learning programming may be difficult and challenging [6], [7]. When this is to be done through online educational environments [8] by students who are in-service teachers with limited time and prior programming knowledge, it might become even more challenging [9]. Additionally, instructors generally don't understand how to prepare and incorporate the gained information in their educational settings [10]. Several studies identify challenges with teachers' self-efficacy regarding learning how to program, apply their programming skills to problem-solving, and teach it [11], [12]. Positive self-efficacy is connected to increased student and teacher outcomes, and it has a positive influence on teachers' psychological well being [13]. It is essential to understand the challenges met specifically by teachers and how this can impact their work.

In the context of an introductory programming course for in-service teachers, we investigate teachers' challenges and difficulties concerning programming and the challenges they expect concerning their teaching and their students' learning. We are focusing on teachers who are learning to program and then using it in their education. The data consist of completed questionnaires with closed (quantitative) and open (qualitative) items. We address the following main research question: what do teachers perceive as difficult in learning programming? Given that in-service teachers have been given some free time to learn to program, we want to take a closer look at what they find challenging when they learn to program. While learning the basics of programming, teachers may reflect on which challenges they expect their pupils will face. Our intention is not to find out what challenges the pupils actually have when they learn to program but rather to look at what teachers think about this since this might affect how they plan to teach programming. Based on what they find challenging and think students will struggle with, we now want to know how this impacts their view on teaching programming. Hence the objectives of this study are to identify the most critical challenges in learning and to teach programming in the context of in-service teachers' professional development.

We have organized this paper as follows: The next section (II) presents a literature review on related work. Section III presents the case and explains our methodology. Section IV analyzes the quantitative- and qualitative data around the main themes identified in the dataset. Section V discusses the main findings in this study, while Section VI presents the conclusions and further work.

II. Literature Review

Programming is often described as complicated and challenging to learn, especially in text-based programming lan-

guages [6], [7], [14]. Programming courses are usually considered as difficult and often have the most considerable dropout rates. Besides, mastery of a programming language's constructs does not consequently translate into solving new programming problems. It involves various cognitive activities and mental representations related to program design, program understanding, modifying, debugging, and documenting. Even at the level of computer literacy, it requires conceptual knowledge and structuring basic operations into schemas and plans; developing strategies flexible enough to derive benefits from programming environment, and methods [15]. Reference [16] introduces an overall view of the elements involved in programming, which are: (a) Problem domain; (b) Programming language domain; (c) Programming paradigm domain; and (d) Solution orchestration. Hence, programming consists of a set of skills. The skills form a hierarchy, and a programmer will be using many of them at any point in time. Programming is not only more than a single skill; it also involves more than one distinct process [17]. The picture is complex, and in this study, we investigate how teachers who might have extended knowledge of different school subjects and related pedagogical knowledge, are experienced teachers but considered programmer beginners, experience this process.

Previous research has shown that novice students may find learning programming difficult in terms of programming concepts and program design [18]. Novice developers experience a broad range of difficulties and deficiencies [19]. Mastering syntax is one of the earliest challenges facing the novice programmer [20]. Writing syntactically correct code is a challenge that regularly confronts the novice programmer [21], [22].

Learning programming is not only developing knowledge and skills but the feeling of mastering it, developing self-efficacy both in learning and teaching programming. One of the measurements of the success of the PD program might be the change in confidence that participants feel concerning their programming ability and their ability to teach programming and computation to others. Teachers' self-efficacy is essential in teaching, according to both theoretical and practical investigations. Low self-efficacy, on the other hand, has a negative impact on teaching effectiveness and performance [23].

TPACK (Technological Pedagogical and Content Knowledge) [24] is a term that refers to knowledge about the intricate relationships between technology, pedagogy, and content that enables teachers to develop proper and context-specific teaching approach [25]. TPACK contains dimensions such as content knowledge (CK), pedagogical knowledge (PK) and pedagogical content knowledge (PCK). In the context of programming, CK represents knowledge of concepts and practices in programming, and PK refers to general pedagogical knowledge. The answers to the main questions of the PCK for programming are: Why teach programming? What should be taught? What are the learning difficulties? How to teach programming? [26].

Teachers may find it challenging to separate learning programming from teaching it. For example, when presenting certain concepts/constructs, addressing pedagogical (PK) challenges related to learning them may improve the course's relevance, which is known to be important in professional development [27]. It is essential to identify the topics that are more difficult to grasp to provide access to quality in-service teachers' professional development.

Teachers learn to program not because they want to be programmers but with the intention of teaching programming, and their main objective is to apply it in their field of study to increase the students' subject understanding. In addition, they must carry this out in addition to ordinary teaching and thus the pressure of time. In this study, we seek to understand what challenges are highlighted by teachers, how the teachers' challenges affect their perceptions of what their students will find challenging and whether the teachers' challenges and beliefs about their students then impact their teaching. To the best of our knowledge, there is little research on teacher training programs for in-service teachers where both learning and teaching programming must be seen in context.

III. Case description and research design

A. The Context

The study presented in this paper is conducted in the context of a program consisting of two courses of 7,5 ECTS credits (European Credit Transfer and Accumulation System) each over two semesters. The first course, that is the focus of this paper, deals with learning the basics of programming (variables, operators, if- and loop statements, lists, functions and libraries). The second course is focusing on pedagogical aspects of programming and it is beyond the scope of this paper. The program is entirely online with no physical gatherings with web-based lectures and weekly activities.

More than 90% of the teachers in this study are teaching in secondary schools, with the large majority in upper secondary schools (In the national educational system, this corresponds to the last three years of the K-12 system). More than 80% of the teachers are teaching STEM-related subjects. Others teach in different subjects such as language, history, music, arts and crafts, etc. 24% of participants have already taught programming in their schools. 61% of participants had some level of programming knowledge before starting the first course. The participants are from the same country but coming from different schools and districts. All schools follow the same national curriculum. Participants vary in terms of teaching experience, including teachers who just started their practice and others who have been teaching for many years. The motivation to take these subjects courses also varies. Most say they are motivated, while others are more sceptical about the inclusion of programming in their subjects. Overall, the participants are diverse in their background knowledge in programming, their interests in learning and teaching it, and the school level. We have not addressed demographic information in this study as we aim to include a large group with different backgrounds, needs, desires, and motivations. Students participate in the course with the support of their school, which is committing to provide some free time (one

day in a week) to teachers to complete the course, though they continue their primary duties during the two semesters. The additional costs for the schools are partly covered by a national program of the Ministry of Education[1]. This support leads to a high completion rate. The assessment form is project-based, with a final grade passed/failed. In the school year 2020-21, 87% successfully completed the program.

B. The Course

We use the textbook "Starting out with Python" [28]. There are three compulsory exercises defined as well as a reflection note and a mini-project in the subject. In addition, students can work on specified assignments from the book. These are defined as voluntary submissions. The mandatory exercises are delivered individually. In the reflection task before starting the mini-project, teachers reflect on how they can use programming to solve relevant problems related to what they teach themselves. The mini-project is carried out over two weeks, and teachers can collaborate in groups of a maximum of three. The mini-project is about identifying competence goals in the subjects to introduce programming and program a simple problem statement. The teaching is online, emphasising interactive learning with weekly activities such as online lectures and regular compulsory work requirements (assignments). Online collaboration and guidance are carried out in social spaces. It is not mandatory to participate in lectures/webinars in real-time. However, recordings are made available for watch afterwards. Recording teaching allows students to repeat the subject matter and makes the teaching more accessible.

C. Research Design

1) Method: In this study, we used a questionnaire to collect both quantitative and qualitative data. The questionnaire was part of a mandatory assignment with a pass/failed grade. Participants had the option of opting for being included in the research. Data collection was approved by NSD, the Norwegian Agency regulating data collection and management for research. The quantitative part consists of a set of claims related to learning and teaching programming. For statements 1-7, the range of answers was 1=Completely Disagree, 2= Disagree, 3=Neutral, 4=Agree, 5=Completely Agree. For statements 8-16, the range of answers was 1=Very Difficult, 2= Difficult, 3=Neutral, 4=Easy, 5=Very Easy. We received a total of 178 reflection notes and excluded 17 of them from the collection since they were not valid. This resulted in a dataset of 161 reflection notes.

Quantitative. Based on the curriculum defined in the programming courses in this context, we searched the literature for common difficulties and challenges that novice programmers may face when learning to program. We formulated statements 1-16 to get teachers' perspectives on these challenges and sought answers to our main research question: (ST 1) Programming has not been an easy subject for me to learn, (ST 2) Programming requires proper understanding of abstract concepts (eg functions / methods, etc.), (ST 3) I may have learning difficulties due to the nature of the subject, (ST 4) I may have challenges in designing lessons so that they will be beneficial to my students, (ST 5) It will be exciting to teach programming courses, (ST 6) I knew little or no programming BEFORE starting the course, (ST 7) I've learned enough programming now to be able to teach my students, (ST 8) Finding bugs in programs are, (ST 9) Dividing a problem into sub-problems (program functions) is, (ST 10) Using libraries in programs is, (ST 11) Understanding variable initialization is, (ST 12) Understanding loops is, (ST 13) Understanding how loops should be used is, (ST 14) Understanding a select-statement is, (ST 15) Understanding how to use a select-statement is, and (ST 16) Creating a program to solve a specific problem is.

Statements 1-3 are related to the challenges of learning to program [6], [7]. Statements 4, 5, and 7 are related to the teacher's self-efficacy in teaching programming [29], [30]. Statements 8-15 are related to teachers' challenges when learning specific programming concepts such as variables, libraries, loops, select-statements, etc [6]. Statement 16 is about the process of creating programs to solve a particular problem [31].

Qualitative. In the qualitative part, participants reflected on questions related to learning and teaching programming challenges from their perspective as teachers: (Q1) What do you find challenging to learn (including language constructions, programming concepts process related issues), and why? (Q2) What parts of programming do you think are most challenging for your pupils to learn? (Q3) What challenges do you think could arise by teaching your pupils specific programming concepts? (Q4) What challenges do you think could arise by teaching your pupils specific language constructions? (Q5) What challenges do you think could arise to motivate your pupils to learn to program?

We used inductive thematic analysis [32] to methodically detect, categorise, and provide insight into patterns of themes across our data set. The units of analysis are defined as paragraphs to capture as much as possible about each statement. The resulting categories (based on sentence entities) are shown in Table III-VII.

2) Data Preparation and Coding: The questionnaire's quantitative data at the beginning of each reflection document were imported in Excel by one researcher and checked by a second researcher. Seven responses were missing answers to one of the statements, and three responses had duplicated scores on four statements. These answers are considered invalid and removed from the analysis. Seventeen responses had duplicated scores on one statement, and an average score was calculated and used in the analysis instead.

The research team processed the questionnaire's qualitative data in several steps: (1) All the researchers familiarized themselves with the data by reading some of the documents. Researchers independently coded five documents; (2) Researchers discussed the codebooks during a meeting. Afterwards, two of the researchers made an independent effort to

[1]https://www.udir.no/kvalitet-og-kompetanse/

merge the codebooks into a common one; (3) Two of the researchers worked together to code four documents, using the agreed-upon codebook to test it and gain a common understanding of the codebook and the coding process; (4) The two researchers coded two common documents to check inter-coder reliability[2]. Since we only use two coders, they used the method "Percent Agreement for Two Raters". We manually compared the codes and gave 1 to the same and 0 to the different ones. The number of 1's divided by the total of codes compared was inter-rater reliability. The results from the inter-coder reliability check were between 75 and 88.4% and were considered sufficient; (5) The remaining document in the data set has been divided into two and coded independently by the two researchers; (6) After individually coding approximately a third of the documents from the divided data sets, another check of inter-coder reliability was performed. The two researchers then coded two common documents before calculating the reliability. The reliability was again considered sufficient, and the coding could keep going; (7) A third and last calculation of the inter-coder reliability was done after all the documents were coded. These results corresponded with the two previous checks; (8) The individual NVivo [33] projects from the two researchers were merged into one project; and (9) The researchers discussed the merged project during a meeting and decided that it would be reasonable to merge some codes from the codebook together. Afterwards, one of the researchers merged the codes and redistributed the final project to the other researchers.

IV. Results

For the quantitative results, table I shows the response to statements 1-16, and the calculation of average (Avg), standard deviation (SD), standard error (SE) and confidence interval (Con) for all responses (n=158). A 95% confidence is calculated ($\alpha = 5\%$). Table II shows the correlation between statements 1-7 and statements 1 to 16. In statements 1-7, we asked teachers to reflect on learning and teaching programming challenges based on their experience from the course. In contrast, other statements seek teachers reflection on language constructs, programming concepts and creating a program to solve a specific problem.

For the qualitative data, we identified the following main themes (which are tightly connected to our research questions) during the thematic analysis of reflection notes: (1) Teachers' challenges when learning to program (159 reflection notes, 22 codes and 1038 code references); (2) Teacher's perspective of pupils learning to program. (159 reflection notes, 20 codes and 961 code references); (3) Teaching programming (161 reflection notes, 33 codes and 1973 code references); (4) Perceived learning (137 reflection notes, 5 codes and 373 code references); and (5) Course-specific reflections (108 reflection notes, 8 codes and 350 references). Additionally, researchers identified few other themes (learning resources/syllabus of the course and attitude towards future learning), which are not further analysed in this study since they are not directly related to our research questions.

[2]https://www.statisticshowto.com/inter-rater-reliability/

TABLE I
STATISTICAL ANALYSIS

	Avg	SD	Variance	SE	Con
ST 1	2,80	1,09	1,18	0,09	0,17
ST 2	4,27	0,62	0,39	0,05	0,10
ST 3	1,82	0,89	0,79	0,07	0,14
ST 4	2,63	1,04	1,09	0,08	0,16
ST 5	4,15	0,75	0,56	0,06	0,12
ST 6	3,46	1,43	2,05	0,11	0,22
ST 7	3,23	1,05	1,10	0,08	0,16
ST 8	2,53	0,96	0,91	0,08	0,15
ST 9	2,85	1,01	1,02	0,08	0,16
ST 10	3,15	1,04	1,07	0,08	0,16
ST 11	3,92	0,99	0,98	0,08	0,15
ST 12	3,86	0,99	0,97	0,08	0,15
ST 13	3,21	1,12	1,26	0,09	0,18
ST 14	3,73	1,06	1,13	0,08	0,17
ST 15	3,41	1,14	1,29	0,09	0,18
ST 16	2,72	0,92	0,84	0,07	0,14

TABLE II
CORRELATION BETWEEN STATEMENTS

	ST 1	ST 2	ST 3	ST 4	ST 5	ST 6	ST 7
ST 1		0,23	0,54	0,35	-0,32	0,55	-0,37
ST 2			0,15	0,06	-0,08	0,09	-0,06
ST 3				0,45	-0,24	0,44	-0,39
ST 4					-0,36	0,28	-0,33
ST 5						-0,24	0,35
ST 6							-0,30
ST 8	-0,44	-0,26	-0,40	-0,30	0,23	-0,40	0,32
ST 9	-0,44	-0,11	-0,44	-0,38	0,26	-0,43	0,30
ST 10	-0,45	-0,03	-0,34	-0,40	0,35	-0,33	0,33
ST 11	-0,58	-0,11	-0,47	-0,35	0,27	-0,43	0,34
ST 12	-0,44	-0,02	-0,52	-0,30	0,26	-0,46	0,33
ST 13	-0,48	-0,16	-0,47	-0,31	0,21	-0,48	0,30
ST 14	-0,55	-0,09	-0,54	-0,35	0,23	-0,47	0,34
ST 15	-0,60	-0,16	-0,51	-0,39	0,26	-0,51	0,36
ST 16	-0,59	-0,27	-0,45	-0,37	0,26	-0,47	0,29

It is important to note that in this study, we intended to identify trends and principal issues. It does not aim to perform a quantitative analysis of the occurrence of specific codes. From this perspective, when we report a particular code's number of times, it should be mainly interpreted to indicate the magnitude rather than an exact number.

In the following sections, we report each code in **bold**-format and use examples of citations from teachers in *italic*-format. Due to lack of space, we only show the top and bottom rows in the tables. We have marked this with a row "..." in the middle of the table.

A. Teachers' challenges when learning to program

This theme covers the challenges teachers report when learning to program, where we seek to find answers to our main research question.

Table III summarizes the results in this theme. The overall understanding of the results is that teachers find many areas of difficulties when learning to program. The most important

TABLE III
TEACHERS' CHALLENGES OF LEARNING PROGRAMMING

Code	# teachers	code frequency
Writing code and pseudocode	74	110
Loops	52	59
Practicing enough	49	59
New way of thinking	38	53
...	...	...
If-statements	24	24
Connecting theory and practice	21	21
Variables	15	16

areas are listed in this table in descending order (based on code-frequency).

Writing code and pseudocode. The coding challenges are often connected to syntax errors, logical errors, and not getting the code correctly on a level of details, e.g.: *one needs to be very careful with absolutely everything, from words down to simple characters, which I think has been most challenging. It takes so little to make a mistake somewhere.* Many teachers also describe difficulties with code structure and how to connect the different parts of a program. Some even report challenges with creating a persuasive or optimal code, e.g.: *It is challenging to find the optimal solution to a given problem. There are so many ways to solve the same problem. So, in the beginning, I experienced that I wrote long and cumbersome codes, but I also see that as I get more used to writing code, my code gets a little better too.*

Loops. Teachers who report that loops are challenging refers to difficulties with knowing what and how to use the different types of loops, e.g., *Loops were challenging to use, it's nice to understand what it is, but when to use "for" or "while", and how to use it, was challenging. Loops in loops and lists in loops made this further complicated.* The difference between "for"- and "while"-loops is mentioned as especially challenging to understand and build. Nested loops are also mentioned explicitly by eight teachers as challenging. Writing code for loops (syntax) and using loops combined with other programming concepts are frequently reported as challenging.

Practicing enough. Teachers highlight the importance of practising sufficient to learn to program. A frequently mentioned challenge is that teachers do not have enough time to learn programming properly, for example, due to personal conditions or lack of time, e.g.: *Practice leads to mastering. I did not have enough time this fall to program more than "testing myself" and doing assignments. Therefore, it does not sit in my fingers and is perceived as time-consuming and challenging. When the small details are missing, it takes a lot of time to resolve the assignments/problems*.

New way of thinking. The process of learning programming is reported as a new way of thinking. Many of them say that this might be related to the fact that programming is something entirely new to them, e.g.: *What has been challenging is that this is entirely new to me. Creating an algorithm based on a problem/task can be challenging.* Some also report that the lack of knowledge and experience makes it challenging to connect the different programming concepts and constructs in a program.

TABLE IV
TEACHER'S REPORTED CHALLENGES OF PUPILS

Code	# teachers	Code frequency
Loops	91	124
Writing code and pseudocode	67	105
New way of thinking	65	97
...	...	...
Mixing concepts	22	26
Time	13	20

B. Teacher's perspective of pupils learning to program

Table IV summarizes the results in this theme. It identifies teachers perception of challenges that their pupils will face when learning to program. Teachers discuss several challenging areas that pupils may struggle with when learning to program. Some of the most important ones are loops, error handling, a new way of thinking, the process of writing code and pseudocode, and the use of libraries and functions. Teachers also discuss that the pupils' background in terms of prior knowledge, attention span, academic skills, or reading and writing difficulties will affect their ability to learn to program. Teachers report to a large extent that learning programming means learning a lot of new things that the pupils have no relation to.

Loops. Ninety-one teachers expect that loops will challenge pupils when learning to program, e.g.: *Perhaps the most challenging thing for students will be learning loops, how they are structured, and understanding how pupils can apply loops to different problems.*

Writing code and pseudocode. The process of writing code and pseudocode is reported to be challenging for pupils, e.g.: *I think it will be most challenging for the students to learn 'the preparation phase' -to write useful pseudocode that shows what the program should do. Then write a program with exact code and good structure based on this pseudocode.*

New way of thinking. Teachers report that the new way of thinking required when learning to program will be challenging: *The first challenge will be algorithmic thinking -breaking the program down into sub-tasks and keeping track of what is happening and why. For some students, the level of abstraction will also be a challenge.*

C. Teaching programming

Table V summarizes the teachers perception of teaching programming. Teachers discuss several areas they will focus on when teaching programming.

Relation to other subjects (interdisciplinary). Many teachers mention that they had (or planned to implement) programming as part of other subjects. Many reflected on possible interdisciplinary functions for programming in their course. Many teachers feel that it is essential to apply programming in the subject to motivate pupils and see programming applications, e.g.: *I think it is essential to apply programming*

TABLE V
TEACHING PROGRAMMING

Code	#Teachers	Code frequency
Relation to other subjects	92	159
Pupils' general motivation	104	144
Prior-knowledge	73	119
...	...	...
Lack of time	57	83
Practicing enough	59	82
Teachers knowledge	55	76

TABLE VI
PERCEIVED LEARNING

Code	#Teachers	Code frequency
Prior-knowledge	86	131
Programming is challenging	67	88
Positive own learning	64	82
Programming is time consuming	45	55
Negative own learning	14	17

TABLE VII
REFLECTIONS ON THE COURSE

Code	#Teachers	Code frequency
Assignments	51	71
Negative feedback	42	62
Progression	42	52
Suitability for beginners	39	47
Syllabus relevance	32	37
Positive feedback	26	36
Course was time consuming	15	24
Participation in learning activities	19	20

in topics such as mathematics or science. In this way, the use of relevant examples can show the usage of loops and conditions related to the subject.

Pupils general motivation. Many teachers reflected on the general motivation of the pupils in the class. There seems to be a large variety of different motivation levels, fluctuating between schools, subjects and fields of study. A common consensus amongst many teachers seemed to be that some pupils are motivated to learn to program, whilst others fail to see the value/importance, e.g.: *I teach both language and mathematics and notice many differences in motivation among my students in the respective subjects. While students in language subjects are motivated mainly by the joy, mastery and usefulness of learning another language, many of the students struggle to see the benefit of the typical theoretical subjects in mathematics. Only a handful in each class is motivated by problems where the known algorithms alone do not suffice.* A few teachers were concerned about teaching programming in vocational classes, as they report that many pupils in the field struggle to learn theory (as well as some failing to see the importance of learning programming for their future line of work), e.g.: *Motivation of students in vocational subjects is usually always a challenge - regardless of theoretical subjects.*

Prior-knowledge. Seventy-three teachers reflected on the pupils' prior-knowledge. The consensus amongst these teachers was that the pupils generally have little to no prior-knowledge, except for a few pupils that have learned programming in their own spare time out of personal interest. Some teachers also reflected that pupils in school now have low digital competencies and little experience and that it will be challenging to teach them programming in the initial phases. However, over time, more and more pupils will learn programming, causing the class's prior-knowledge to be more homogeneous, making it easier to teach programming without adapting to the large gap in knowledge currently prevalent in most new courses, e.g.: *In the first years after programming is introduced in education, we in upper secondary school will have a challenge because the various secondary schools have not focused as much on programming. It will give us a challenge that the student's prior knowledge will be quite different, but I reckon that it will be quite similar in the various schools as time goes on.*

D. Perceived Learning

This theme is about reflections teachers have made about their learning, that is, whether they think they have improved their skills or whether they have learned enough to teach. It gives a general impression of the teacher's perception of their knowledge, influencing how they perceive learning to program and teach it. Table VI shows an overview of their reflections. Teachers have reflected both positively and negatively on their learning. However, far more teachers are optimistic about their learning. E.g., *Programming is actually something I can accomplish!* And many teachers felt that they were competent enough to teach programming in school after taking the course, e.g., *I think I will have learned enough programming to teach my students programming in mathematics.*

E. Course Reflections

The curriculum, defined syllabus, and how the programming course has been conducted may have influenced teachers' experience of challenges. This section examines the teacher's response to the study and puts it in context with the previous themes' results. It might help to detect cases where a challenge relates to a specific course weakness rather than being an inherent challenge of the subject.

Assignments. Fifty-one teachers found the assignments in the course to be somewhat challenging. Many teachers reflected that the third assignment was difficult and time-consuming and too demanding. This exercise contained several real-life problems that students had to use their algorithmic thinking capabilities to create a solution and code in Python. Also, it included the use of libraries such as NumPy and time. Several teachers also found the formulation of the assignments to be confusing or difficult to understand, e.g.: *But what is the biggest problem for me is how the language is in the assignments. I struggle to know what is meant by the task. Expressions and formulations that I am not used to being used. I spend an incredible amount of time understanding the assignment. The reason for this, I think, is because programmers have their professional language that I am not familiar with.*

Negative feedback. Forty-two teachers had negative feedback on the course in their reflection notes. A majority of the negative feedback was related to the fact that teachers experienced the learning curve in the program being too steep and feeling that the reading material was not relevant to the course or sufficient for their learning. Some teachers also thought that the course did not prepare them well enough for teaching programming in school, e.g.: *I believe we have learned a lot in this course which is not relevant for my students as I only teach math; as I teach 1P this year, I still do not quite know what to learn to program, but in general, I hoped that this course should have been a little more closely linked to math.*

V. Discussion

This section discusses the results presented in the previous sections in connection to the research questions. In addition, we identify some implications for the design of professional development for in-service teachers.

Teachers learning programming. Learning programming is often labelled as challenging [18], [19]. However, some researchers are questioning this reputation. Recent research suggests that problems might relate to how the subject is taught, rather than programming is inherently complicated [34]. Trying to cover too much in introductory courses and setting unrealistic expectations on students might be the core of the problem. It is also known that techniques or activities may enhance learning by improving how the content is delivered by teachers and experienced by students [35]. Recent research is also warning about the potential damage of generic statements that might negatively affect learning environments, especially in terms of inclusiveness [36]. Our results are also questioning the perception of programming as difficult.

The answers to the questionnaires indicate that teachers do not perceive programming as very difficult. Only 47 teachers agree or completely agree with the statement that learning programming has not been easy (ST1), while 67 disagree or disagree entirely, and 44 are neutral, with an average of 2.8. The average is even lower (1,82) for the statement connected to learning difficulties (ST3), with only seven teachers agreeing with the statement. The result might partly be influenced but the fact that 61% of the teachers had already some previous programming knowledge. However, there is only a moderate correlation with the knowledge level before the beginning of the course. The positive perspective on programming is partly confirmed by the qualitative data, with 64 teachers explicitly reporting a positive learning experience. In the words of a teacher: *This is my first course in programming, and I have found it rewarding. I knew minimal programming before I started, but this course has opened the door to a new and exciting value for me.* However, some teachers explicitly refer to programming as challenging, especially at the beginning, e.g., *I think it has been demanding to learn to program, it is something I have not done before.*

Out of the topics that we have selected from the literature as challenging for newcomers (ST8-16) [7], [37], the most difficult and the easiest ones are shown in Table III. More in general, based on the results, we can define three levels of complexity: (1) Understanding basic concepts; (2) Understanding how to apply basic concepts; and (3) Developing working programs to solve specific problems. It is also interesting to note that all the topics are correlated to the perception of the difficulty level of learning programming and learning challenges. Still, no elements stand out. In their reflection notes, teachers elaborate on different aspects and introduce new ones. Specifically, the use of pseudocode is one of the topics that is addressed most. However, most often, when talking about basic concepts, teachers relate to challenges connected with their use rather than their basic understanding. It is also interesting to underlay that challenges might be connected to the context more than the subject, with 49 teachers explicitly referring to challenges connected with practising enough. In the words of a teacher: *The time factor has also been a challenge. A hectic teacher's everyday life has become even more hectic this autumn due to new curricula and corona. If you want to be good at programming, you have to take the time to get acquainted with and ponder over various issues. Unfortunately, I have not had enough of that.*

It is known from the literature [14], [38] that learning programming requires time. Only a minority of teachers have explicitly discussed their future learning. Still, of those who do – optimism and positivity are dominant, with 29 teachers reporting that they are motivated to learn more programming and stating that they are looking forward to increasing their knowledge. In the words of a teacher: *I look forward to learning more programming and didactics in the subject. During this fall, I have enjoyed working with programming. It has given some sparks. I have been very motivated to learn something new, and I am looking forward to working further with programming.*

Teachers perspective on pupils. The questionnaire did not include any question connected to pupils, but teachers were asked to reflect on their pupils' expected challenges. The results are briefly summarized in Section 4.2. Here we want to compare the teachers' reflection on their learning and the challenges that they experienced with their reflection on pupils' learning. If we compare table III (including the codes for teachers), with table IV (the codes for pupils), we can note several differences. The topics that are discussed are mainly the same. However, there are many differences about which themes are in focus. Considering that teachers of this course are beginner learners, one could have expected a greater alignment in the discussion. However, when talking about pupils, teachers expect pupils to face difficulties already at a lower level, e.g., basic understanding concepts. So, for example, variables mentioned by only 15 teachers about their learning are mentioned by 53 when discussing pupils. Teachers might have a broader background knowledge helping them to understand basic concepts, especially for STEM teachers. However, these differences are worrying because they might indicate a misconception of pupils' real challenges. Teachers might use too much time to convey basic concepts rather

than facing higher-level challenges. It is also interesting to note that practising enough is one of the most discussed topics by teachers when discussing their learning. However, only two teachers mention it in connection with pupils. There are, however, 13 teachers who discuss issues connected with time, especially in connection with competing requirements from other subjects. In the words of one teacher: *The biggest challenge I see is time. Time to learn languages, concepts and applications in subjects. To learn programming, continuity and maturation are needed. In a hectic school day, there is a lot more to learn.* Again, this might be worrying because one can expect that pupils would need as much time as teachers for practising, and maybe even more if we accept the teachers' perspective that pupils might be struggling with even more basic concepts. However, it should also be noted that many teachers reflect on this issue when discussing teaching.

Impact on view on teaching. Seventy-four of the teachers disagree that they have learned enough for teaching (ST7, average 3.23). At the same time, only 42 expect they will have challenges with designing lessons for their pupils. So, even if teachers feel that they have not learned enough, they think they have learned enough to design lessons. These results correlate only moderately with the previous knowledge of programming. In their reflection, when discussing the teaching of specific language constructs, many teachers mention issues connected with the logic of the language. In the words of one teacher: *The challenges here will be that the students do not naturally understand which concept they should use for the various assignments. They get logical errors in their programming, and it can be challenging to find a solution. There is also a lot to think about and many different concepts to get control over.* However, many reflections around teaching are not specifically connected to teaching specific concepts, despite this being explicitly asked, but rather relating to broader pedagogical aspects. In the analysis, we identified three additional themes in the reflection notes: (i) issues connected to individual pupils, e.g., their expectations and interests; (ii) issues connected to motivating pupils, e.g., through controlling physical objects and cooperation; (iii) generic pedagogical issues, like concerns about creating learning resources and adaptation of the teaching to individual needs. An issue that is discussed by many teachers connects with time. Challenges with time are connected with subject and context aspects. On one side, programming is seen as a subject requiring time to master and it is often stated, as when reflecting on own learning, that it requires time to practice. At the same time, it is only one of the subjects in school and, in many cases, it is integrated in other subjects. In this way, there is a double competition for time with other subjects and with other topics to cover in the subject where it is integrated. One hundred thirty-three teachers agree that it will be exciting to teach programming in school and the connected statement (ST5) has an average of 4.2. This is an excellent result. Despite the challenges that they might face in learning a new subject within the time constraints of a busy working context or the challenges they are aware they will face when teaching it, teachers look positively at their future teaching practice.

Implications for professional development. Based on the results of our study, we claim that it is important to break the narrative of programming as challenging. This might be demotivating for both the ones who are teaching the course and the participating teachers. Especially considering that teachers will have the responsibility to help others to learn to program, a positive approach is paramount [36]. With this claim, we do not want to underestimate the challenges connected to teaching and learning specific topics. However, this study shows that when teachers are provided (i) access to quality training and (ii) time for it, overall, they do not perceive programming as challenging and are excited about teaching it. It is, therefore, a responsibility of the educational system to create these conditions. At the same time, we know from the literature that far too many teachers are left alone learning programming and how to teach it or are offered limited training [39].

It is essential to identify which are the topics that are more difficult to grasp and for which teachers, to be able to provide access to quality professional development. For example, the teachers of our study did not seem to have problems learning basic concepts, like variables and if-statements. Their difficulties started in understanding how to apply them and building programs, as discussed in section V. However, it should be considered that our cohort consisted mainly of STEM teachers, who might be familiar with some of the concepts from other subjects, such as variables in mathematics. Feedbacks about the course also point to the need to create the correct progression and be careful to avoid a steep learning curve. The results also show that some teachers might have problems understanding learning programming from their pupils' perspectives. This could be addressed more clearly by inviting the teachers to explicitly reflect on this aspect and using the course to showcase good pedagogical practices. One teacher said: *I think the way to organize the course with exercises and supporting video was good, and I can reasonably consider using something similar*. We think that the course would benefit from promoting reflection, possibly at the collaborative level, about how pupils will face different topics, which is the most suitable pedagogical approach to address them.

Our results show that, not strangely, it is difficult for teachers to disconnect learning programming and teaching it. For example, when they express concern about their learning, teachers often refer to the fact that they have not learned enough to teach it. Most teachers do not see the need to become expert programmers but knowing enough to teach programming. Again, the course should be used to showcase good teaching practices and help the teachers reflect on learning goals and pedagogical design of the course activities that they could re-use in their teaching. This might help improve the course's relevance, which is known to be critical in professional development [27], [40]. In this context, we want to underline that when we asked teachers to reflect on teaching programming, the challenges they discussed were mainly at a very general level, as mentioned in section IV-C.

On one side, this is a source of concern because teachers might not correctly address the teaching of specific concepts. This is something that should be considered in the training organization, for example, by addressing pedagogical issues connected to learning specific concepts/constructs when introducing them. On the other side, the analysis shows the capability of in-service teachers to reflect on pedagogical issues of a broader nature. Teachers are probably building on their previous pedagogical experience in other subjects to contextualize the teaching of the new subject. This previous pedagogical knowledge is an asset that should be exploited appropriately in professional development. Overall, the results above indicate that teachers will benefit from courses tailored to them rather than generic introductory programming courses that we offered to some degree. A better coupling of learning programming with teaching would solve many of the issues and limitations that the teachers have discussed. We are fully aware that, in a context where many teachers do not get access to adequate professional development [41], a generic course might be the best option. The drawbacks of this approach might be limited by designing specific additional material, exercises, and reflection activities that can help the teachers put their learning in their teaching context. However, getting back to the beginning of the section, quality training is important, but it is not enough if teachers do not get the time to learn.

Limitations and threats. The template provided to the teachers for completing the exercise includes the first part with a structured questionnaire and a second part only indicating the main themes to reflect on. The questions in the questionnaire may have influenced teachers reflections, e.g. it might have brought the teachers' attention to specific constructs while neglecting others. However, it was essential to making sure that teachers had some concrete issues to start with, which could trigger reflection. From the analysis of the results, this did not constitute a threat to the study since teachers have openly reflected on several issues without limiting them to those explicitly mentioned in the questionnaire.

Teachers have delivered the documents that are analyzed as a mandatory exercise. One risk that we considered in terms of the study's validity is that teachers might be more reluctant to report difficulties in learning. However, this risk is limited because the exercise was not graded. This exercise was also done in the context of a professional development program promoting formative assessment and reflection rather than student performance. The choice of collecting data through an exercise that we consider fully adequate in this context might be more problematic in other contexts. The reflection notes have been divided among two authors, and most of the documents have been coded by only one coder. A double coding of all the reflection notes would have increased the reliability of the study. However, creating the codebook and checking the coding agreement between the two coders, as previously described, is reducing the risks usually associated with one coder. Also, considering the exploratory nature of the research, this limitation is not expected to impact the results significantly. The objective is to identify the main issues rather than the highly focused analysis of the interviews. Another limitation of the study that we want to underline is that all the teachers attended the same course. This impacts the study's external validity and generalization needs, therefore, to be done cautiously. Some of the challenges about learning programming might indeed be connected to the actual organization of the course, as pointed out also in the discussion chapter. This limitation is partly reduced because it is a relatively standard programming course based on widely adopted teaching material.

It should also be noted that the study is based on difficulties in learning to program as the teachers perceive them. There is no external evaluation with knowledge tests to compare with the teachers' statements. So, a teacher might report that a specific programming concept is straightforward. This does not necessarily corresponds to a fundamental understanding of the concept or an increased capability to use and teach it. The choice to focus on teachers' self-assessment and reflection has value in providing the individual perspective of the learner. It might impact how they see themselves as programmers and teachers. This perspective is often neglected in favour of more objective knowledge tests. It remains as part of future studies to relate the self-reported view of teachers to their actual performance as programmers and teachers. Our data do not link participants' success rate and views to their background knowledge, e.g. prior programming knowledge, teaching experience, etc. This must be seen as a limitation of this study.

VI. Conclusions and future work

This paper presented the results from the analysis of the reflection notes delivered by 178 in-service teachers attending a teacher's professional development program. Our results show that learning to program is not necessarily difficult if, like in our case, teachers are given access to adequate training and time to learn. In this perspective, our paper contributes to the growing body of literature challenging the reputation of programming as difficult. This does not mean that specific topics are not challenging to learn. In particular, the results show three levels of growing complexity: (1) Understanding basic concepts, (2) Understanding how to apply basic concepts, and (3) Developing working programs to solve specific problems. When talking about pupils, teachers reflect on similar topics and challenges than when reflecting on their own learning. However, they expect pupils to face difficulties already at a lower level, e.g., understanding basic concepts. Not all teachers feel that they have learned enough to start teaching programming. Many of them are also concerned with time issues connected to teaching a subject that requires time to master. However, most teachers think they will be able to design lessons for their classes and are excited about teaching programming in schools. As part of future work, we intend to conduct (1) Additional studies with teachers attending different programming courses to understand better the impact of the course on the challenges perceived by the teachers, and (2)

Follow-up studies to see how the teachers´ perception changes after they start with the actual teaching.

VII. Acknowledgements

We thank the teachers who participated in this research. The work is partly funded by the Norwegian Directorate for Education and Training (https://www.udir.no) and the Norwegian Center for Excellent IT Education (https://www.ntnu.edu/excited).

References

[1] R. S. Lindberg, T. H. Laine, and L. Haaranen, "Gamifying programming education in k-12: A review of programming curricula in seven countries and programming games," *British Journal of Educational Technology*, vol. 50, no. 4, pp. 1979–1995, 2019.

[2] S. Kjällander, A. Åkerfeldt, L. Mannila, and P. Parnes, "Makerspaces across settings: Didactic design for programming in formal and informal teacher education in the nordic countries," *Journal of Digital Learning in Teacher Education*, vol. 34, no. 1, pp. 18–30, 2018.

[3] J. Gal-Ezer and C. Stephenson, "A tale of two countries: Successes and challenges in k-12 computer science education in israel and the united states," *ACM Transactions on Computing Education (TOCE)*, vol. 14, no. 2, pp. 1–18, 2014.

[4] M. Menekse, "Computer science teacher professional development in the united states: a review of studies published between 2004 and 2014," *Computer Science Education*, vol. 25, no. 4, pp. 325–350, 2015.

[5] L. S. Shulman and J. H. Shulman, "How and what teachers learn: A shifting perspective," *Journal of Education*, vol. 189, no. 1-2, pp. 1–8, 2009.

[6] S. Abesadze and D. Nozadze, "Make 21st century education: The importance of teaching programming in schools," 2020.

[7] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *Acm sigcse bulletin*, vol. 37, no. 3, pp. 14–18, 2005.

[8] C. J. Heinrich, J. Darling-Aduana, A. Good, and H. Cheng, "A look inside online educational settings in high school: Promise and pitfalls for improving educational opportunities and outcomes," *American Educational Research Journal*, vol. 56, no. 6, pp. 2147–2188, 2019.

[9] H. Demarle-Meusel, M. Rottenhofer, B. Albaner, and B. Sabitzer, "Educational pyramid scheme–a sustainable way of bringing innovations to school," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2020, pp. 1–7.

[10] T. Hiltunen, "Learning and teaching programming skills in finnish primary schools–the potential of games," *University of Oulu, Oulu*, 2016.

[11] E. Hartell, A. Doyle, and L. Gumaelius, "Teachers' attitudes towards teaching programming in swedish technology education." *PATT 37*, p. 195, 2019.

[12] L. Mannila, L.-Å. Nordén, and A. Pears, "Digital competence, teacher self-efficacy and training needs," in *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 2018, pp. 78–85.

[13] M. Zee and H. M. Y. Koomen, "Teacher self-efficacy and its effects on classroom processes, student academic adjustment, and teacher well-being: A synthesis of 40 years of research," *Review of Educational Research*, vol. 86, no. 4, pp. 981–1015, 2016. [Online]. Available: https://doi.org/10.3102/0034654315626801

[14] M. Piteira and C. Costa, "Learning computer programming: study of difficulties in learning programming," in *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, 2013, pp. 75–80.

[15] J. Rogalski and R. Samurçay, "Acquisition of programming knowledge and skills," in *Psychology of programming*. Elsevier, 1990, pp. 157–174.

[16] R. Juárez-Ramírez, C. X. Navarro, V. Tapia-Ibarra, R. Macías-Olvera, and C. Guerra-García, "What is programming? putting all together-a set of skills required," in *2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE, 2018, pp. 11–20.

[17] R. E. Mayer, *Teaching and learning computer programming: Multiple research perspectives*. Routledge, 2013.

[18] B. S. Xia, "A pedagogical review of programming education research: What have we learned," *International Journal of Online Pedagogy and Course Design (IJOPCD)*, vol. 7, no. 1, pp. 33–42, 2017.

[19] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer science education*, vol. 13, no. 2, pp. 137–172, 2003.

[20] P. Denny, A. Luxton-Reilly, E. Tempero, and J. Hendrickx, "Understanding the syntax barrier for novices," in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, 2011, pp. 208–212.

[21] M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," in *Proceedings of the second international workshop on Computing education research*, 2006, pp. 73–84.

[22] M.-H. Nienaltowski, M. Pedroni, and B. Meyer, "Compiler error messages: What can help novices?" in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 2008, pp. 168–172.

[23] M. Zhao, M. Zhao, X.-H. Wang, and H.-L. Ma, "Promoting teaching self-efficacy in computational thinking of school teachers," in *2020 Ninth International Conference of Educational Innovation through Technology (EITT)*. IEEE, 2020, pp. 235–239.

[24] L. S. Shulman, "Those who understand: Knowledge growth in teaching," *Educational researcher*, vol. 15, no. 2, pp. 4–14, 1986.

[25] M. Koehler and P. Mishra, "What is technological pedagogical content knowledge (tpack)?" *Contemporary issues in technology and teacher education*, vol. 9, no. 1, pp. 60–70, 2009.

[26] M. Saeli, J. Perrenet, W. M. Jochems, and B. Zwaneveld, "Teaching programming in secondary school: A pedagogical content knowledge perspective." *Informatics in education*, vol. 10, no. 1, pp. 73–88, 2011.

[27] J. M. McPartland, "Organizing schools for improvement: Lessons from chicago," 2011.

[28] T. Gaddis and R. Agarwal, *Starting out with Python*. Pearson, 2015.

[29] J. Thorsnes, M. Rouhani, and M. Divitini, "In-service teacher training and self-efficacy," in *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Springer, 2020, pp. 158–169.

[30] Y. J. Reimer, M. Coe, L. M. Blank, and J. Braun, "Effects of professional development on programming knowledge and self-efficacy," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–8.

[31] S.-C. Kong, M. Lai, and D. Sun, "Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy," *Computers & Education*, vol. 151, p. 103872, 2020.

[32] V. Braun and V. Clarke, "Thematic analysis." 2012.

[33] K. Jackson and P. Bazeley, *Qualitative data analysis with NVivo*. Sage, 2019.

[34] A. Luxton-Reilly, "Learning to program is easy," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 2016, pp. 284–289.

[35] A. Luxton-Reilly, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo, "Introductory programming: a systematic literature review," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 2018, pp. 55–106.

[36] B. A. Becker, "What does saying that 'programming is hard' really say, and about whom?" https://cacm.acm.org/magazines/2021/8/254304-what-does-saying-that-programming-is-hard-really-say-and-about-whom/fulltext, (Accessed on 07/30/2021).

[37] T. Jenkins, "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, no. 2002. Citeseer, 2002, pp. 53–58.

[38] M. Piteira and C. Costa, "Computer programming and novice programmers," in *Proceedings of the Workshop on Information Systems and Design of Communication*, 2012, pp. 51–53.

[39] L. Cooke, J. Schugar, H. Schugar, C. Penny, and H. Bruning, "Can everyone code?: Preparing teachers to teach computer languages as a literacy," in *Participatory Literacy Practices for P-12 Classrooms in the Digital Age*. IGI Global, 2020, pp. 163–183.

[40] H. Borko, "Professional development and teacher learning: Mapping the terrain," *Educational researcher*, vol. 33, no. 8, pp. 3–15, 2004.

[41] J. Vahrenhold, M. Caspersen, G. Berry, J. Gal-Ezer, M. Kölling, A. McGettrick, E. Nardelli, C. Pereira, and M. Westermeier, "Informatics education in europe: Are we all in the same boat?" 2017.