Exploring Scratch to Python Transfer in Norwegian Lower Secondary Schools*

Ragnhild Kobro Runde[1], Quintin Cutts[1,2], and Lars Kristian Skaarseth[1]

[1] University of Oslo, Norway
ragnhild.runde@ifi.uio.no, larsksk@ifi.uio.no
[2] University of Glasgow, Scotland
Quintin.Cutts@glasgow.ac.uk

Abstract. With an increased focus on programming in schools, more and more pupils are introduced to Scratch in primary school before learning Python in secondary school. This paper draws on an intervention approach associated with the Model of Programming Language Transfer (MPLT), and on hugging and bridging transfer techniques more generally, to consider how to ease the transition between the two contexts and to deepen conceptual learning.
An initial guess quiz, part of the MPLT intervention approach, taken by 97 pupils from mathematics classes in three lower secondary schools in Norway, indicates that in the current context where the pupils have had some, but not much, exposure to Scratch, many of the basic programming concepts seem to be abstract true carry-over concepts, i.e. the pupils are familiar with the concepts in Scratch, but do not automatically transfer that understanding to the similar Python concepts.
An MPLT-based teaching intervention was designed in order to help the pupils see the connections between the Scratch and Python concepts. The intervention used the Python turtle library in an attempt to make the two programming environments more similar. The intervention was well-received by pupils and teachers, but follow-up interviews revealed that the teachers do not currently have enough programming knowledge themselves to be able to adapt the intervention into their regular teaching. In practice, the situation is also complicated by the two programming languages being taught by different teachers in different schools, with no communication and little curricular continuity.

Keywords: programming language · transfer · secondary school · novice

1 Introduction

Transfer of conceptual understanding of programming by a near-novice, as they move from their first programming language (PL) to their second, has seen increasing attention since Tshukudu and Cutts proposed their Model of Programming Language Transfer (MPLT) [7]. Prior to this, PL transfer studies had

* Based on material collected as part of the master thesis of Lars Kristian Skaarseth [6] .

principally concerned transfer by experts, particularly when changing paradigms, e.g. [5]. The MPLT highlights how students can be facilitated to move from their first to second PL with both minimal hindrance and the opportunity to deepen their conceptual understanding of PLs generally. A successful trial of an intervention approach based on the MPLT and using bridging and hugging transfer techniques [4] is presented in [8].

The context used by Tshukudu and Cutts was the transfer from Python to Java, two textual PLs used in their own institution as the first two PLs. An increasingly prevalent transition is now taking place in the school system from Scratch to Python. However, just as the teacher principally associated with Tshukudu and Cutts's studies had not previously considered reaching back to the students' first PL when introducing their second, so school teachers are reported to often not consider Scratch knowledge when introducing Python [9]. Many teachers reported using Scratch to get the pupils engaged only, not as a first stepping stone in a progression.

A pilot study is presented here as a first step in investigating whether a stronger conceptual understanding can be fostered in learners if researched transfer techniques are incorporated into teaching Python to pupils who have learned Scratch. The pilot study was undertaken in the context of lower secondary Norwegian education, on the basis of initial Scratch exposure in primary education. The pilot study therefore has two research questions, as follows:

– RQ1. How can a transfer plan based on the MPLT be applied to help learners move from Scratch to Python?
– RQ2. How do Norwegian lower secondary teachers view the potential and challenges of integrating such a transfer plan into their own teaching?

The pilot study involved 3 schools, 4 teachers and 97 pupils. The principles of the MPLT held for the new context, although a new carryover concept category was detected, a so-called Abstract False Carryover Concept. A transfer plan related to the original approach in [8] was used successfully, with careful attention paid to the very different programming contexts and environments offered by Scratch and Python to ease transfer. However, we recognise that the highly controlled context of the original MPLT work, involving bachelor students moving between PLs in two programming courses in one institution, is very different from the school-based Scratch to Python transfer currently underway in some regions of Norway, where multiple schools, teachers and teaching approaches may have been used for the first PL.

2 MPLT and Scratch to Python

In this section, the MPLT approach is presented in more detail, together with a comparison of both the language concepts and the environments of Scratch and Python.

2.1 MPLT categories and intervention

The MPLT and its concept categories. This work aims to build on the original MPLT work in [7], which itself built on research in conceptual transfer in natural languages, showing that transfer of conceptual understanding from one PL to another is initiated when syntax elements in the two PLs match. This match is followed by the automatic transfer of conceptual and semantic understanding associated with the syntax in the first PL to the matching syntax in the second, new, PL. The semantics are the implementation of the concept in the first PL. Three categories of concept are identified, dependent on syntax and semantic matching, as follows:

- True Carryover Concept (TCC). A concept whose syntax and semantics match in both languages. The Python/Scratch conditional statement using the syntax *if* is a TCC.
- False Carryover Concept (FCC). A concept whose syntax matches in both languages but whose semantics don't. An example in Scratch/Python is the *round*-function, where values ending with 0.5 are always rounded up in Scratch, but to the nearest even integer in Python, such that e.g. *round(1.5)* equals 2 in both Scratch and Python, while *round(2.5)* equals 3 in Scratch but 2 in Python.
- Abstract True Carryover Concept (ATCC). A concept without common syntax in both languages, but with common semantics. A *forever* loop in Scratch and a Python *while (true)* loop fall into this category.

How learning is influenced by the categories. Learners' in-built syntax-based language transfer mechanism, underpinning the MPLT, will carry over conceptual/semantic understanding for both TCCs and FCCs. For TCCs, this facilitates learning; for an FCC, learners will now have the wrong semantic understanding of a concept in the new PL. For an ATCC, they will know the concept and semantics in the old PL, but will not recognise it in the new PL and will believe it is something new to learn.

MPLT guess quiz. As part of the MPLT-based intervention approach, predictions are made about which concepts will fall into which categories. A so-called *guess quiz* is then developed based on these predictions: students with no exposure to the second PL answer questions about code fragments in both first and second PLs. The collated student answers are used in the on-going transfer teaching, as explained below. Questions on the first PL are necessary to gather each student's baseline understanding – they may have the wrong understanding of a concept in the first PL and carry this over, or not, to the second PL. This can be checked when marking the quiz, and is important input data for the teacher to understand the differentiation in his/her class.

Teaching intervention. Having set the quiz for the students, the teacher marks it and shapes subsequent lessons around the collated responses: TCCs can be noted, but otherwise ignored, since semantic transfer of the students' existing knowledge will happen automatically; FCCs that learners took to be TCCs require a special focus to correct the automatic, but incorrect, transfer – a case of *bridging* [4] where related but not identical items are connected in the learner's mind; ATCCs require an explicit connection made to the pre-existing knowledge – a case of *hugging* [4] where seemingly unrelated items are brought together for the learner. In all of this, the teacher can highlight meta-cognitively the deeper conceptual understanding binding particular semantics and syntax, and also how PL transfer works, for future learning experiences.

2.2 Investigating the language environments

In du Boulay's key work on issues with learning programming [2], the programming environment is highlighted as a concern, as well as particular language issues. When considering transfer, what knowledge of the environment can be successfully transferred, and what is unique to the particular context? In considering Scratch and Python environments, the following observations are noted:

Problem domains The stage and graphical objects dominate the Scratch environment, and constructs to work with these objects are built into the language. Scratch has just one problem domain - that of graphical interactive animations. Python, by comparison, is general purpose, in that it can be used to tackle any problem domain, using specialised programming and/or a specialised library. This specific/generic split between the two languages may make them look very different. Furthermore, teachers often use simple textual I/O as a problem context when introducing Python, bearing no relation to the graphical context learners have come from. We propose that transfer can be facilitated and deeper learning achieved if the Python *turtle* graphics library is used in the first lessons. This is a *hugging* tactic. The problem domain is at least familiar, and the move to the general-purpose Python language can be motivated shortly after when another problem domain and library is explored.

Manual vs Programmed initialisation Setting up the Scratch graphical stage using drag-and-drop and building scripts does not seem that related to typing in a file of Python code. However, Scratch's stage set-up and the Python code to declare variables achieve the same result: to initialise the computational *world* on which the code operates. Pointing out this similarity, a hugging approach, using a number of examples and perhaps a visualisation tool like PythonTutor[3] to show the hidden Python world that is created at run-time, should enable a learner to understand what they know already in Scratch more deeply, while also enabling transfer of that knowledge across to Python, and seeing that it is not so different.

[3] pythontutor.org

3 Pilot Study Design

A pilot study was designed for Norwegian classrooms, where a recent curriculum change introducing programming into maths and other subjects, has lead to many schools implementing this using programming in Scratch at primary level, followed by Python in secondary mathematics classrooms. Experience seems to indicate that it varies across different school regions in Norway whether the change from block-based to text-based programming happens in lower or upper secondary school, but all pupils will meet the transfer at some point.

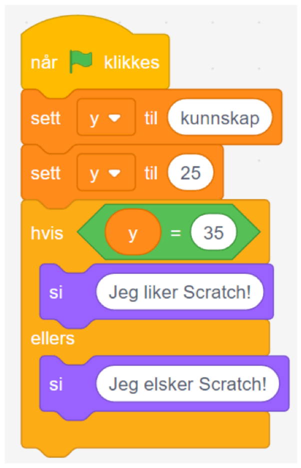
To address RQ1, a transfer intervention to be used in lower secondary schools was designed that drew on the original MPLT intervention model, as follows.

First, a guess quiz was developed, based on a categorisation of related Scratch/ Python constructs into TCCs, FCCs and ATCCs. In the guess quiz, the pupils were given some code samples and asked to explain (guess) what was happening in different parts of the code. In an attempt to avoid unintended learning effects between the two parts of the quiz, all of the Python questions were given before the Scratch questions, and similar questions were given in a different order and with slightly different details in the two languages. An example is given in Figure 1, where conditionals (*if/hvis*) were predicted to be a TCC, and variable assignment, boolean equality and output were predicted to be ATCCs.

```
x = "hei"
x = 45

if x == 50:
    print("Python er morsomt!")
else:
    print("Python er spennende!")
```

(a) Python



```
når 🏁 klikkes
sett y til kunnskap
sett y til 25
hvis y = 35
    si Jeg liker Scratch!
ellers
    si Jeg elsker Scratch!
```

(b) Scratch

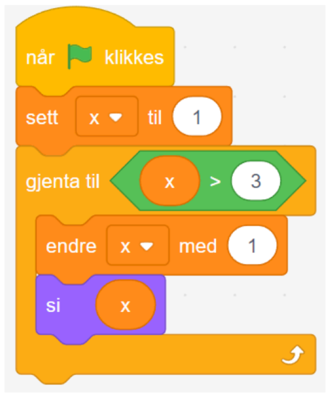
Fig. 1: Guess quiz code example using assignments, conditionals, boolean equality and output.

Another example is given in Figure 2, where the boolean operator $<$ was predicted to be a TCC, while the other new parts, the *while/gjenta* loop and the addition assignment, were both predicted to be ATCCs.

```
y = 0

while y < 2:
    y += 1
    print(y)
```

(a) Python



(b) Scratch

Fig. 2: Guess quiz code example using loops.

The predictions of how the pupils will respond to the new language is an important exercise as it encourages the teacher to think deeply about the two languages, to see where there are similarities and differences, and to consider how the pupils may respond. The quiz was delivered in the first session of Python teaching.

A teaching session was then delivered in the next available programming class, discussing transfer and introducing Python based on the guess quiz results. The session made use of the turtle package *trinket.io/turtle* since it is the same browser-based system as used by teachers for introducing Python – *trinket.io/python3*. The session was designed using a live-coding format, to encourage questions, prompt predictions on what would happen during execution, and invite input directly from the pupils, in order to personalise the session. The session leader introduced transfer concepts, and highlighted particularly the ATCC errors made by the pupils in the guess quiz. The pupils were at machines and so were following along with the code being developed by the session leader – and this enabled them to experiment themselves at the end of the session, to increase personal investment in their learning.

The third author led both guess quiz and teaching sessions. The data captured includes the guess quiz results, the third author's field notes recorded immediately after the sessions, and an interview with the relevant teacher(s) after each teaching session. These contribute to answering RQ2.

Recruitment of teachers was via cold-call to schools in the region. Before collecting any data, approval from the Norwegian Centre for Research Data (NSD) was gained for the study.

4 Pilot Study Results

For logistical reasons, teachers were not invited to take part in the study until December/January, well into the school year. Of over 20 schools contacted, only 3 of the interested schools had not started Python teaching yet and were therefore appropriate for the study. This enabled access to 114 pupils aged between 13 and 15, of whom 97 pupils had had prior exposure *only* to Scratch. It is these pupils' responses that are reported and discussed concerning the guess quiz. Four teachers were involved and provided input in the post-session interviews.

4.1 Guess quiz

The most relevant results from the guess quiz regarding various language constructs are now discussed in light of the original predictions made and the guess quiz responses. In some cases, the pupils' responses matched to the predictions, and sometimes not. In the latter case, this is crucial information for the teacher, showing where their own expectations of pupils' understanding are incorrect.

Mathematical and boolean operators. As predicted, almost all pupils understood the use of mathematical operators, both in Scratch and in Python, making this a TCC. The same applied to the boolean operators < and >, while the pupils seemed to struggle more with transferring their understanding of boolean equality, making this an ATCC.

Conditionals. Conditionals were assumed to be a TCC. As Norwegian pupils at this level are very used to English in relation to e.g. gaming and social media, they were expected to connect *if* and *else* in English to the direct Norwegian translations *hvis* and *ellers* used in Scratch. However, this seemed not to be the case, making conditionals an ATCC and not a TCC.

Assignments. The results indicate that variable (re-)assignment is also an ATCC, since the syntax is quite different and pupils were generally correct with Scratch, but not Python. While there was little difference between the understanding of assignment and re-assignment in Scratch, in Python even fewer pupils understood re-assignment than assignment. Notably, for Python, several pupils thought either that a variable could contain more than one value simultaneously, or that the values of two assignments to the same variable were somehow combined. For example, some pupil answers related to the first two lines of the Python code in Figure 1a were *"x = Hei = 45"*, *"[the result is] Hei 45"*, and *"He says Hei 45 times"*.

Loops. As expected, most pupils were familiar with loops in Scratch, but almost no-one seemed to be able to transfer their understanding from *repeat* and *repeat until* loops in Scratch to *for* and *while* loops in Python, with many pupils not even indicating that they recognized them as loop constructs at all, making this a clear ATCC.

Connecting to original problem domain. Even though we had identified the different problem domains of Scratch and Python as described in Section 2.2, we were intrigued by guess quiz results demonstrating how closely some of the pupils linked programming with animation, and how they tried to interpret much of the Python code in an animation setting. For example, one pupil explained the *if*-test in Figure 1a by *"If one pushes the x-button, . . . "*. For the assignments in Figure 1a, one pupil stated that *"It says Hei and displays it for 45 seconds"*, while another explained the assignment $x = 45$ by *"It shows how far it should go"*.

4.2 Teachers' views

Knowing pupils' programming background. In the current situation in Norway, the teachers seem to focus on learning Python themselves and figuring out where programming fits into the curriculum. The teachers have little to no idea about what previous programming knowledge the pupils have, neither what they are supposed to have learnt nor what has actually been covered in previous classes. Partly because of this, they have not explicitly considered building on the pupils' Scratch knowledge when introducing Python, even though some talk about it as an important principle:

> *"I think that in a transition phase it is important to know what the pupils know from before. And then start on that."*

Using a transfer plan. After participating in this study, the teachers generally approve of the transfer plan concept. All four mention the usefulness of knowing more about the pupils' programming background and building on that. As it differs how much and which parts of Scratch the pupils know from primary school, a well-designed transfer plan should include something similar to the guess quiz where the results form a basis for adapting the general transfer plan to each specific class. As one of the teachers put it:

> *"I think [having a transfer plan] is very optimal and seems very nice, but then it is not that realistic. [...] Maybe if there were some really short questionnaires that we could use in the planning phase."*

Another teacher emphasizes the need for help in interpreting the pupils' answers:

> *"I do not have enough knowledge about programming. So I would never have [a background test]. I would have needed a description if they answered that, what that meant, because I had no idea. But I think that is absolutely very smart."*

Using the turtle library. The teachers are in general positive towards using the turtle library, but for different reasons. Some of the teachers view the use of turtle mainly as something fun to inspire and motivate the pupils:

> *"[The pupils] were really concentrating when they wrote the code, and they thought it was really fun when they pressed "play" that something [visual] actually happened."*

Others also link it to the similarities with Scratch:

> *"If you know Scratch you could then maybe see some parallels. Python is not something completely different. [...] I think I would do something like that if I would do the same."*

A contributing factor to some of the teachers not necessarily understanding the intention behind using turtle might be that in the teaching sessions, the syntactic and conceptual similarities between Scratch and Python were only talked about and not shown explicitly to the pupils. We intend to change this in the future.

However, as commented on by the teachers relating to the guess quiz, also when it comes to using the turtle library, the teachers find it difficult in practice as they do not know enough programming to answer quite basic questions from the pupils:

> *"I was asked why do you do that? Why should there be empty parentheses? Why colons there? And those are things I do not know."*

5 Discussion

The discussion will address each of the research questions in turn. RQ1 concerns how an MPLT-based intervention can work in a Scratch to Python context. First, we discuss the specifics of Scratch to Python transfer, in relation to the MPLT. The guess quiz served its purpose well, both in validating and challenging our beliefs as teachers about how the pupils would transfer their knowledge - crucial knowledge for the teaching step. The guess quiz results indicated that there are TCCs (e.g. arithmetic operators) and ATCCs (e.g. loops) that we predicted. However, we did not expect pupils to have difficulty with conditional statements – but as noted earlier, this may be an issue with English/non-English failure to match keywords. We propose therefore that, while our predictions may not have been accurate, the same kind of syntax-based transfer that lead to the development of MPLT also was going on for learners transitioning from Scratch to Python – but that the natural language issues may be putting more concepts into the ATCC category.

Our guess quiz contained no FCCs, because our analysis showed that these are rare in Scratch/Python. Indeed, we only found one – the *round* function. In searching more extensively for other FCCs, we found string indexing where Scratch's *letter <number> of <string>* is like Python's *<string>[<number>]*, but with the indexing starting at 1 in Scratch and 0 in Python. We consider this to be a new carry-over category for the MPLT. It is clearly the same concept, but with a different syntax and semantics – an Abstract False Carryover Concept, or AFCC.

Our analysis and our results suggest quite a different sorting of concepts into carryover categories for our Scratch/Python context compared to the original Tshukudu and Cutts Python/Java context. In the latter, both languages are textual and from broadly the same lineage – and this appears to lead to mainly TCCs and FCCs, with only a very few ATCCs. However, Scratch and Python are very different syntactically on the basis of the graphics and even the English words used – and then in our context we used a non-English version of Scratch. Hence, we found many more ATCCs, with relatively few TCCs and FCCs. This suggests that while there is less danger of incorrect automatic incorrect transfer due to few FCCs, teachers in the Scratch to Python context will get little for free, due to few TCCs, and will need to explicitly make the ATCC connections between the two languages.

However, once the new curriculum has been in effect for some years and the pupils will have had more Scratch experience in primary school, some of the concepts originally predicted to be TCCs, such as conditionals, may actually become TCCs (and no longer ATCCs), e.g. due to the pupils more easily making the connection between the Norwegian and English words. For reasons such as this, it could be a good idea for teachers to continue doing a guess quiz with each new class in order to adapt how they teach the transfer from Scratch to Python based on the pupils' background knowledge and intuitive understanding of the different concepts.

RQ1 also concerns the use of an MPLT-based transfer intervention plan for this context. The sessions led with pupils were clearly very engaging, judging by the session leader's field notes. The original MPLT transfer plan was augmented with both the turtle library to maintain at least a similar problem domain, and explicit explanation to show how drag-and-drop and variable initialisation are two ways of setting up the computational world. This appears to have successfully reduced the complexity of moving between languages. In the short time available, the session leader was able to cover all the carryover concepts highlighted by the guess quiz, and the level of pupil questions and comments received or overheard during and after the sessions indicates that interest in programming was maintained or raised.

In this pilot study, a limitation in answering RQ1 is the restriction of the teaching following the guess quiz to a single session. This was caused by the late recruitment of schools into the study, with teachers unable at that point in the school year to commit more time. In future studies, we would expect to recruit teachers much further ahead, giving time to allocate more sessions to the teaching.

As an addition to the MPLT model of TCCs, FCCs, and ATCCs, we found that some pupils brought with them an understanding of programming as animation, and tried to interpret the Python code in that context. As part of future work, it should be investigated whether or not using the turtle library as part of a transfer plan in order to keep the problem domains more similar also means reinforcing the misunderstanding that programming means animation, and whether or not this is something that should be addressed more explicitly when teaching Python to pupils with a Scratch background.

RQ2 concerns the potential and challenges of using an MPLT-based transfer plan from the teachers' point of view. One key issue is the level of preparation of the maths teachers who are teaching programming. Despite having little to no formal programming training, they are all eager to make the most out of a difficult situation. The teachers describe how they find it difficult to find enough time to learn programming themselves, and how that makes it hard to make progress as they forget what they have learnt between each session. For the time being, they seem to treat programming in mathematics primarily as something that has to be overcome as quickly as possible in order to get back to what matters most, i.e. mathematics. A partial exception is the 10th grade teachers, who worry about not preparing the pupils well enough for the exam as they do not know in enough detail what level of programming will be expected of the pupils.

Conversations with teachers highlighted an issue with the MPLT-based teaching approach. The teachers saw the potential value in knowing more about the pupils' previous programming knowledge and using a transfer plan to build on that knowledge. However, they did not know Scratch themselves and so would have had trouble marking the guess quiz themselves, in order to determine what aspects to focus on, and then more trouble leading the teaching session and making connections between Scratch and Python. In both this study and the original studies around the MPLT, the guess quiz creation, marking and interpretation was carried out by the researcher. Full transfer of the MPLT teaching approach is yet to be explored.

6 Related Work

Mlanedoviȍ et al [3] explored the transfer between block-based and textual languages with 11-12 year old children, using mediated-transfer techniques, picking in particular the hugging and bridging techniques we have used as part of the intervention in this study. In their approach, the instructors determine which concepts will require special attention, whereas the guess quiz in the MPLT-based approach presented here enables a more learner-centred structure. They used a more controlled environment where they taught students who had already learned block-based programming, and in their intervention, used a second block-based environment and a text-based programming environment alongside each other.

Dann et al [1] worked with Alice and Java, with university students, again working with both learning environments at the same time. Although not directly related, Weintrop and Wilensky [10] explored the progress of students who learned programming initially either via block-based language or via an isomorphic text-based language. There was no difference in the two groups when they transitioned to Java. However, the study involved high school/upper secondary students from age 14 upwards, and the ability to work with more abstract textual languages may be simpler for these more advanced students compared to primary or lower secondary students.

7 Conclusions

The results from this pilot study clearly indicate that MPLT is a useful model also in the context of transfer from Scratch to Python at school level. In addition to the three concept categories in MPLT, we have identified the need for a fourth category - the Abstract False Carryover Concept (AFCC) where the two PLs in question have the same overall concept, but with a different syntax and a different semantics making it a risky one if the pupils learn enough to realize the connection, but without understanding the important differences in semantics.

Previous MPLT studies were conducted in a relatively controlled context with pupils transferring from first to second PL in a single institution. From this pilot study, we conclude that it is very different to consider programming transfer across levels and institutions where the curriculum is under-specified both with respect to programming language to be used and concepts to be learnt, and where the teachers do not necessarily have much general programming knowledge.

Having not explored this style of transfer in the secondary school sector before, this pilot study was essential to gain a better understanding of the context. This is the first in a series of studies aiming to explore transfer in lower secondary. In particular, in future work, we would expect to: prepare teachers to lead the process rather than a researcher; encourage transfer to be better integrated into their ongoing classroom activity; take more time over the transfer process - one session was far from enough; determine whether using Turtle after Scratch reinforces an incorrect stereotype about programming as an animation tool only; and reflect and incorporate approaches specifically tailored to the challenging transfer environment where teachers are not experienced in CS typically and the pupils' prior learning of the first PL took place in maybe numerous other schools with possibly different approaches to programming.

Progressive education systems, such as found in Norway, encourage the removal of common teaching materials used nationwide, in favour of recognising teacher expertise, encouraging teacher autonomy and supporting specialised teaching approaches to suit specific contexts. We suggest that these laudable goals may work well in long-established subjects, where teachers have both a sound content knowledge, and if teaching for some years, a strong developed pedagogical content knowledge also - but this is very much not the case for programming education in the Norwegian context.

The teachers we interacted with could see the great value of the approach we are proposing - but knew that they would not have been able to come up with appropriate materials for their own classroom. In such a situation, there is surely a case for common materials to be developed and shared among teachers. Not all secondary schools will have feeder primary schools that have used Scratch - but many will. In other contexts, specific shared materials could be created for the particular prevalent language pairing in that context. This is a situation where teachers need strong expert-led support. New teachers in traditional subjects get that expert support in their teacher education institutions, and from mentors when they start out on the job. The Norwegian mathematics teachers who are being required to teach programming do not have enough support of that kind. The study results reported here simply underline that issue.

References

1. Dann, W., Cosgrove, D., Slater, D., Culyba, D., Cooper, S.: Mediated transfer: Alice 3 to java. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education. p. 141–146. SIGCSE '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2157136.2157180
2. Du Boulay, B.: Some difficulties of learning to program. Journal of Educational Computing Research **2**(1), 57–73 (1986)
3. Mladenović, M., Žana Žanko, Granić, A.: Mediated transfer: From text to blocks and back. International Journal of Child-Computer Interaction **29**, 100279 (2021). https://doi.org/10.1016/j.ijcci.2021.100279
4. Salomon, G., Perkins, D.N.: Rocky roads to transfer: Rethinking mechanism of a neglected phenomenon. Educational psychologist **24**(2), 113–142 (1989). https://doi.org/10.1207/s15326985ep2402_1
5. Scholtz, J., Wiedenbeck, S.: Learning second and subsequent programming languages: A problem of transfer. International Journal of Human-Computer Interaction **2**, 51–72 (1990). https://doi.org/10.1080/10447319009525970
6. Skaarseth, L.K.: Investigating the transfer from Scratch to Python in Norwegian secondary school. Master's thesis, Department of Informatics, University of Oslo (2023)
7. Tshukudu, E., Cutts, Q.: Understanding conceptual transfer for students learning new programming languages. In: Proceedings of the 2020 ACM Conference on International Computing Education Research. p. 227–237. ICER '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372782.3406270
8. Tshukudu, E., Cutts, Q., Foster, M.E.: Evaluating a pedagogy for improving conceptual transfer and understanding in a second programming language learning context. In: Proceedings of the 21st Koli Calling International Conference on Computing Education Research. Koli Calling '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3488042.3488050
9. Tshukudu, E., Cutts, Q., Goletti, O., Swidan, A., Hermans, F.: Teachers' views and experiences on teaching second and subsequent programming languages. In: Proceedings of the 17th ACM Conference on International Computing Education Research. p. 294–305. ICER 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3446871.3469752
10. Weintrop, D., Wilensky, U.: Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. Computers & Education **142**, 103646 (2019). https://doi.org/10.1016/j.compedu.2019.103646