Block and Text Programming in Swedish High School: What do students know on their first day?

Johan Snider
NTI-Gymnasiet Uppsala
Uppsala, Sweden
johan.snider@ntig.se

Erik Bokström
NTI-Gymnasiet Uppsala
Uppsala, Sweden
kasper.davidsson@ntig.se

Kasper Davidsson
NTI-Gymnasiet Uppsala
Uppsala, Sweden
kasper.davidsson@ntig.se

Anna Eckerdal
Department of Information Technology
Uppsala University
Uppsala, Sweden
anna.eckerdal@it.uu.se

Robin Kastberg
NTI-Gymnasiet Uppsala
Uppsala, Sweden
robin.kastberg@ntig.se

***Abstract*—This research work-in-progress paper presents findings related to upper secondary students' understanding of programming concepts such as: (1) variable assignment, (2) if-statements, and (3) loops in Python and Scratch. In 2017, the Swedish education board added computer science curriculum to compulsory schooling. Students now entering upper secondary school are expected to have experience with programming. To find out how familiar first year upper secondary students are with programming, we assessed 172 students' programming knowledge with a multiple choice questionnaire with block programming questions in Scratch and text programming questions in Python. Each question required students to read a program between 4-8 lines of code and correctly identify the output of the program. The questions included basic programming concepts such as: variable assignment, if-statements and loops. Additionally, we asked students to self-report their prior experiences with programming in terms of how many hours they had spent programming inside and outside of school, as well as if they had more experience with block or text programming. As expected, the students' scores correlated positively with the amount of hours they self-reported to have spent programming inside and outside of school. In conclusion, we correlate students' self-reported programming experience with their scores on the block and text sections and reason about these results based on teachers' experience and research literature.**

***Index Terms*—Computer science**

I. Background

Computer science is increasingly important in society and accordingly many countries have introduced programming in school curricula. In Sweden, programming was introduced in 2018. For compulsory school, grade 1-9, it was decided that "programming will be introduced as a distinct element in the compulsory school, specifically in mathematics and technology" [1][our translation]. In upper secondary school programming as its own subject exists, but only as elective for students in the natural sciences and technology programs [2]. Programming, however, is not taught in mathematics courses in upper secondary school. Students are expected to have some knowledge from primary school so they can use it in mathematical problem solving.

Introducing programming in Swedish schools has offered several challenges. For example, most teachers, especially in grade 1-9, had no experience of the subject when it was included in their teaching, and resources available in different schools varied [3]. Vinnervik found that "there is a risk of inequality among schools and that the children's experience of programming becomes fragmented" (p. iii).

In addition to these challenges, numerous studies have reported programming to be difficult to learn [4] with low motivation among students [5]. Numerous attempts have been made to lower the barriers into programming. One widely used approach is to teach students block-based programming before text-based programming. In block-based programming, blocks that are syntactically correct can be dragged and dropped to form a program, which frees the learner from remembering syntax details. Block-based programming has been described as a "low threshold, high ceiling" way into programming, compared to text-based programming [6]. Several studies report learning advantages with block-based programming compared to text-based, especially for the younger students [7] [8] [9].

Programming teachers in Swedish upper secondary schools face several challenges, principally, from the gap in knowledge level between student groups from different K-12 schools, as described by Vinnervik (2021). To better understand this division, this study investigates first-year upper secondary school students' programming experiences from compulsory school. Our research questions are:

A. Research questions

1) How does experience with programming inside and outside of compulsory school affect the students' abilities to answer programming questions in block and text correctly?
2) How does students' knowledge of programming compare with respect to block programming syntax in Scratch and text programming syntax in Python?

Finansierat av ULF-projektet
Thanks to teachers and students at NTI Gymnasiet Uppsala

3) How does students' previous experience with programming affect their understanding of variable assignment, if-statements and loops?

II. Related Works

Introducing programming in school through block-based programming environments like Scratch (https://scratch.mit.edu/) and Alice (https://www.alice.org/) has become increasingly common in primary and secondary school [7]. Even though the number of research studies on the effects of block-based programming are growing, still many questions remain [8]. Most previous research reports benefit from teaching school children block-based programming compared to text-based.

Weintrop and Wilensky [9] in a study on high school students learning block-based programming compared to text-based, found that the former students "showed greater learning gains and a higher level of interest in future computing courses" while they found no difference between the groups "with respect to confidence or enjoyment" (p. 3:1). Moors et al. [7] discuss that block-based programming is motivating, especially for young students, and reduce the need to memorize syntax details. Hu et al. [8] performed a meta-analysis on 29 empirical studies, finding "a small to medium significant positive overall mean effect size" on students' academic achievement for block-based programming (p. 1467). Some research has used an environment where students had access to both block-based programming and the corresponding code in text. Weintrop and Hobert [10, p. 638] conclude: "While much of the discussion around the design of introductory programming has been focused on debating which is better for learners – blocks or text, this paper shows the answer may be: why not both?"

We found a few studies that report mixed or negative results from teaching block-based programming. Parsons and Haden [11] found "minimal transfer from the imperative context to the visual environment" (p. 213). Moors et al. [7] report that transitions from block-based to text-based programming can be difficult for students by reducing their confidence and making them unwilling to do the switch. However, if the students choose to continue study programming, they need to transition to text-based programming since that is what is used in industry. Lewis [12] compared "attitudinal and learning outcomes of sixth grade students programming in either Logo [text-based programming] or Scratch ... in interpreting loops and conditional statements" (p. 346). She found that students learning Scratch were better at interpreting conditional statements, but no difference between the groups on their ability to interpret loops. Students who had learned Logo on the other hand, demonstrated higher confidence on average than the other group.

Contrary to this, Mladenović et al. [13], comparing elementary school students learning Scratch, Python, or Logo, found less misconceptions about loops in the Scratch group compared to the other groups, specifically on complex tasks like nested loops. Additionally, some studies, however, implemented in higher education, report no significant difference between students using text-based or block-based programming [14] [15].

Hu et al. [8] discuss the differences in the reported academic achievement between block-based and text-based programming suggesting this could be due to measurement of different educational stages, different block-based programming tools, different ways the experiments were conducted, or differences in schools. Nonetheless, most previous research seems to identify more benefits with teaching block-based programming to young novices compared to teaching text-based.

III. Methodology

In this study, we surveyed 172 first year upper secondary students to assess their knowledge of block and text programming. The study was conducted at NTI High School in Uppsala, Sweden. NTI high school in Uppsala is one of 29 NTI schools across Sweden, all of which have a focus in STEM subjects and follow national curricula [16]. The NTI high schools are privately owned and receive funding from the Swedish government in a system similar to American charter schools. As such this study should not be seen as representative for Swedish high schools in general.

A multiple choice questionnaire was administered to all first year students in September and October 2021, before the students received any formal programming instruction. Students were informed that participating in the study was optional and the results would not effect their grades. Additionally, students were informed the results would be used in research with the aim to improve the quality of programming education. Students were encouraged to take the questionnaire seriously and to make their best effort even if they were unsure about the correct answers. The students were also informed they could skip questions if they did not have a guess as to which answer would be correct. Blank scratch paper and pencils were handed out to the students with the instructions to use the scratch paper to keep track of intermediate results, if needed. The scratch paper was discarded after the survey and not included in the analysis. The students were informed that they had 25 minutes to answer the 20 programming questions, 10 of which were block programming questions in Scratch and 10 of which were text programming questions in python.

The questionnaire was developed in Google Forms and consisted of three different sections. The first section consisted of background questions where students were asked to self-report how many hours they had spent in school programming, as well as, how many hours they had spent programming in their free time. Students were also asked if they had more experience with block or text programming. The following two sections consisted of programming questions. Every question followed the same format where a variable would be: created, given a value, possibly changed, and printed out at the end of the program. Students were asked to select the correct value that was printed out from four possible alternatives where

Hours of programming experience	Number of students
Less than one hour (<1h)	28 (16%)
Between one and five hours (1-5h)	81 (47%)
Between six and ten hours (6-10h)	33 (19%)
More than 10 hours (>10h)	25 (15%)
Did not respond	5 (3%)

TABLE I
HOURS OF PROGRAMMING EXPERIENCE FROM COMPULSORY SCHOOL

three options were created manually with the intent of being reasonable answers to the question.

In both sections, two questions focused on variable assignment and reassignment. Two questions focused on conditional logic using if or if-else statements. These questions used either a less-than or greater-than operator with either a single if statement or a paired if-else statement. Four loop questions were included. Two questions focused on non-conditional loops. In Scratch, the non-condition loop block is labeled 'repeat' and shows the number of times the loop will execute, as shown in Figure 3. In Python, the non-condition loop is the for loop statement. The two additional loop questions were about conditional loop. In Scratch, the conditional loop block is labeled 'repeat until' followed by a conditional value. In Python, the conditional loop is the while loop statement. The last two questions were built to be the most complex questions. These questions were combinations of the previous questions and contained a non-conditional loop with a nested if or if-else statement inside the loop. These questions were designed to see if students could combine the looping logic with the conditional logic to correctly answer the questions.

Both the block and text sections had the same type of questions with only the names and values of the variables changed. The study was designed in this way to be able to compare if students answered more block or text questions correctly. In total the students could get 20 scores, 10 in each section.

The analysis was done using Python which produced the present tables and figures.

IV. RESULTS

The results show the number of hours spent programming inside and outside of school correlated with higher average scores on the questionnaire. Table I shows the responses for the number of self reported hours of programming experience from compulsory school. Roughly half the students (47%) self reported between one and five hours of programming experience.

Figure 1 shows a box plot of the scores for the data in Table I. Students who self-reported more hours of programming instruction during compulsory school had higher averages scores on the questionnaire. The same result held for students who self-reported more hours of programming outside of compulsory school (graphic not shown).

These results can be further broken down into separate scores for block and text. Figure 2 shows the scores on the block questions in light blue and the text questions in dark blue, for the varying amount of hours worked in compulsory

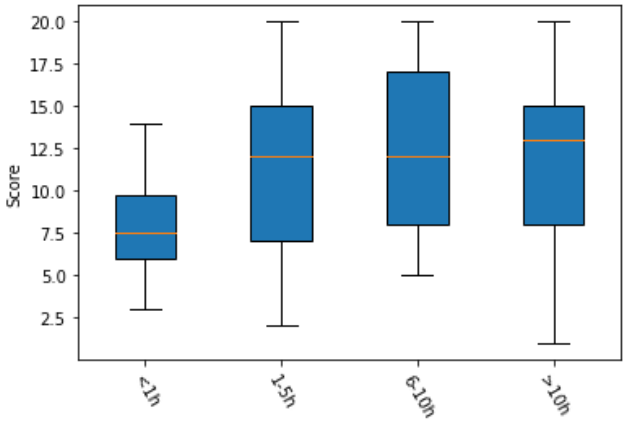

Fig. 1. Score for varying amount of hours worked in compulsory school

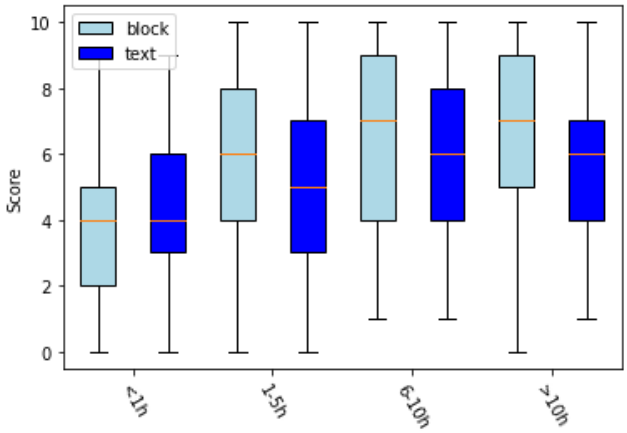

Fig. 2. Block and text scores for varying amount of hours worked in compulsory school

school as shown in Table I. This shows that in general students did better on the block sections than text sections if they had spent more than one hour with programming in compulsory school. The students with less than one hour of programming experience did slightly better on the text section, however, their scores were lowest overall.

Students were also asked if they had done more block or text programming in compulsory school. Table II shows that roughly one-third of the students only had previous experience with block programming, and that the majority of students mostly had block programming experience.

The correct response rate for different types of questions

Response options	Number of students
Only block programming	58 (33%)
Mostly block programming	28 (16%)
Half block and half text programming	30 (17%)
Mostly text programming	14 (8%)
Only text programming	23 (13%)
Did not respond	19 (10%)

TABLE II
PREVIOUS EXPERIENCE WITH BLOCK AND TEXT PROGRAMMING

Question type	% correct in Scratch	% correct in Python
variable assignment	91%	92%
if-statements	57%	75%
loops	55%	51%
if-statement inside loop	45%	34%

TABLE III
PROGRAMMING CONCEPTS WITH AVERAGE RESULTS IN SCRATCH AND PYTHON

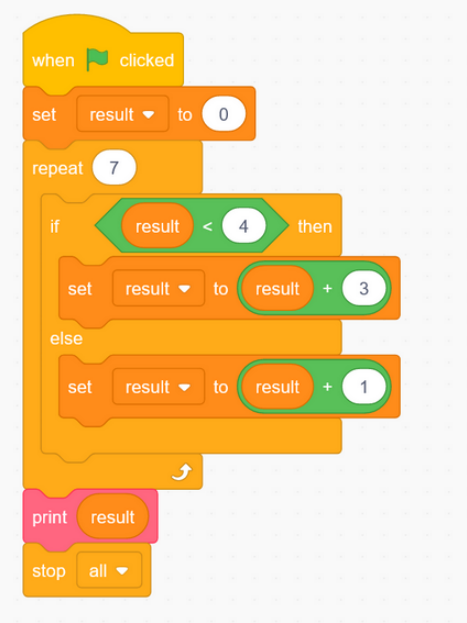

Fig. 3. Example of a block programming question in Scratch

also gives insights into the students' knowledge of programming. Table III shows the questions about variable assignment were answered correctly more than 90% of the time in both Scratch and Python. Questions with if-statements were answered more correctly on average in Python, and that loop questions were answered at similar rates in Scratch and Python. More complex questions with an if-statement inside a loop were more frequently answered incorrectly. This indicates that compound questions involving if statements inside loops were more difficult to answer than separate question with only if statements or loops by themselves. Figure 3 shows one such block programming question in Scratch. Figure 4 shows the corresponding text programming question in Python. Despite having the same form, the block question was answered correctly around 10% more often.

V. DISCUSSION

When comparing our results with previous research we found that for questions on if statements, students gave more correct answers with Python than with Scratch, which is contrary to Lewis' results [12]. For the loop-questions the rates were similar which is in line with Lewis' results as well as Mladenović et al. [13]. For the complex questions with an if-statement inside a loop, Scratch gave better answering rate than Python. Neither Lewis and Mladenović et al. included

```
result = 0

for i in range(5):
    if result < 3:
        result = result + 1
    else:
        result = result + 3

print(result)
```

Fig. 4. Example of a text programming question in Python

this type of question in their studies. Likely explanations to these divergent results are discussed by Hu et al. [8], saying that the studies are difficult to compare due to such factors as the different ages on the participants.

We conclude that most students in the study received between one and five hours of programming instruction during compulsory school, most of which was only or mostly block programming. As would be expected, these students performed better on block programming questions than text programming questions. This shows that students come to upper secondary school mostly prepared for block programming, but lack the knowledge to understand more complex programming examples like a loop with an if statement inside it.

In Sweden, text based programming is a requirement for the course Programming 1. This is problematic because of the knowledge gap between block programming and text programming as described by Vinnervik [3]. Teachers need to spend time in the beginning of the Programming 1 course to bridge the gap that exists between block programming and text programming. If students are mostly familiar with drag-and-drop style block programming, it will take time to learn text based programming constructs which can use indentation or parenthesis. Additionally, even though students can be familiar with some programming concepts such as if statements and loops, that does not mean that they have an in-depth understanding of the topics or can necessarily use them together. For educators involved in compulsory school, we recommend spending at least a few hours programming in either block or text environments. Previous research indicates that block programming is easier and more motivating for younger students [7]–[9]. From our results we see that a few hours of instruction can make a difference in students understanding of programming topics. Last but not least, complementing block programming with text programming can also serve to prepare students better for programming in upper secondary school, as suggested by [10].

As programming in compulsory schools becomes more established, we hope to see better results and potentially a need to adjust our curriculum to account for students having more experience. Additionally, in the future, we would like to look at students attitudes around programming to see how this relate to how they manage Programming 1 since previous research has pointed to learning text-based programming give students better confidence than learning block-based [7], [17].

REFERENCES

[1] Regeringskansliet. (2017) Stärkt digital kompetens i läroplaner och kursplaner. [Online]. Available: https://www.regeringen.se/pressmeddelanden/2017/03/starkt-digital-kompetens-i-laroplaner-och-kursplaner/

[2] F. Heintz, L. Mannila, and T. Färnqvist, "A review of models for introducing computational thinking, computer science and computing in k-12 education," in *2016 IEEE Frontiers in Education conference (FIE)*. IEEE, 2016, pp. 1–9.

[3] P. Vinnervik, "När lärare formar ett nytt ämmnesinnehåll: Intentioner, förrutsättningar och utmaningar med att införa programmering i skolan," Ph.D. dissertation, Umeå universitet, 2021.

[4] A. V. Robins, "12 novice programmers and introductory programming," *The Cambridge handbook of computing education research*, p. 327, 2019.

[5] P. Kinnunen and L. Malmi, "Why students drop out cs1 course?" in *Proceedings of the second international workshop on Computing education research*, 2006, pp. 97–108.

[6] E. A. Kyza, Y. Georgiou, A. Agesilaou, and M. Souropetsis, "A cross-sectional study investigating primary school children's coding practices and computational thinking using scratchjr," *Journal of Educational Computing Research*, vol. 60, no. 1, pp. 220–257, 2022.

[7] L. Moors, A. Luxton-Reilly, and P. Denny, "Transitioning from block-based to text-based programming languages," in *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*. IEEE, 2018, pp. 57–64.

[8] Y. Hu, C.-H. Chen, and C.-Y. Su, "Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis," *Journal of Educational Computing Research*, vol. 58, no. 8, pp. 1467–1493, 2021.

[9] D. Weintrop and U. Wilensky, "Comparing block-based and text-based programming in high school computer science classrooms," *ACM Transactions on Computing Education (TOCE)*, vol. 18, no. 1, pp. 1–25, 2017.

[10] D. Weintrop and N. Holbert, "From blocks to text and back: Programming patterns in a dual-modality environment," in *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*, 2017, pp. 633–638.

[11] D. Parsons and P. Haden, "Programming osmosis: Knowledge transfer from imperative to visual programming environments," in *Procedings of The Twentieth Annual NACCQ Conference*. Hamilton New Zealand, 2007, pp. 209–215.

[12] C. M. Lewis, "How programming environment shapes perception, learning and goals: logo vs. scratch," in *Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, pp. 346–350.

[13] M. Mladenović, I. Boljat, and Ž. Žanko, "Comparing loops misconceptions in block-based and text-based programming languages at the k-12 level," *Education and Information Technologies*, vol. 23, no. 4, pp. 1483–1500, 2018.

[14] C. Kyfonidis, N. Moumoutzis, and S. Christodoulakis, "Block-c: A block-based programming teaching tool to facilitate introductory c programming courses," in *2017 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2017, pp. 570–579.

[15] C. Mihci and N. Ozdener, *Programming Education with a Blocks-Based Visual Language for Mobile Application Development*. ERIC, 2014.

[16] N. Gymnasiet. (2022) Upper secondary education in ict/tech, science, design business. [Online]. Available: https://www.ntigymnasiet.se/nti-in-english/

[17] K. Powers, S. Ecott, and L. M. Hirshfield, "Through the looking glass: teaching cs0 with alice," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, 2007, pp. 213–217.