# Block and Text Programming: What do students know on their first and last day of upper-secondary school?

1st Johan Mattias Snider
*Information Technology*
*Uppsala University*
Uppsala, Sweden
johan.snider@it.uu.se

2nd Anna Eckerdal
*Information Technology*
*Uppsala University*
Uppsala, Sweden
anna.eckerdal@it.uu.se

***Abstract*—This full research paper investigates upper-secondary students programming skills. In 2021, an initial survey assessed the programming skills of first-year upper-secondary school students, focusing on their ability to solve problems using block and text-based programming languages. The survey revealed significant differences between students with mandatory programming education (Group TECH) and those without (Group SCI), with Group TECH outperforming Group SCI across all categories.**

**The follow-up survey in 2024 evaluated the same cohort to measure longitudinal changes in their programming knowledge and skills. The results showed marked improvements in Group TECH's performance, particularly in more complex programming tasks involving loops and embedded conditional statements, while Group SCI also showed progress, albeit to a lesser extent. The study found that students who engaged more frequently in programming activities outside of school demonstrated higher average scores.**

**Overall, students in the study entered upper-secondary mostly prepared for programming topics mostly related to math like variables and conditionals in block programming languages. Students that left upper-secondary, after introductory programming, were better prepared to understand loops and complex problems in text.**

***Index Terms*—Computer science education, K-12 education**

## I. Background

Around the globe, programming education is increasingly being integrated into the K-12 curriculum, responding to the growing demand for digital literacy [1]. Alongside formal education systems, non-formal educational initiatives such as after-school clubs and summer camps have proliferated, enriching the programming education landscape [2]. These diverse educational avenues have adopted various pedagogical strategies to impart foundational programming and computer science wisdom to the next generation.

In Sweden, programming was officially incorporated into the compulsory education curriculum in 2018, underlining its importance in the foundational educational stages [3]. The subject is intertwined in mathematics and technology classes, structured as follows across different educational stages [4], [5]:

- **Grades 1-3:** Constructing, describing and following unambiguous step-by-step instruction as the basis for programming. The use of symbols as instructions.
- **Grades 4-6:** Programming in visual programming environments. Constructing and using algorithms when programming.
- **Grades 7-9:** Programming in visual and textual programming environments. Creating, testing and improving algorithms when programming.

Visual programming environments, notably block programming with platforms like Scratch, are typically introduced between grades 4-6, making programming accessible and engaging through intuitive drag-and-drop interfaces [6], [7]. As students progress to grades 7-9, the curriculum shifts towards textual programming, where languages such as Python introduce students to the syntax and structure of traditional coding practices [8], [9].

The mix of both formal and informal educational resources has led to a mosaic of programming proficiency among students entering upper secondary schools. These students exhibit a wide array of skills, influenced by their varied educational backgrounds in programming [10]. This diversity presents significant challenges for educators in upper secondary schools, who must adapt their teaching strategies to meet the varied needs of their students, balancing between those who are proficient in block programming and those more adept with textual programming [11].

## II. Introduction

This study builds upon a work-in-progress paper which initially surveyed the programming skills of first-year upper secondary school students in 2021, aiming to understand the impact of their previous formal and non-formal programming education [10]. It specifically assessed their ability to answer multiple choice programming problems in both block and text formats. This paper presents the follow-up study that was conducted two and a half years later with the same students in Spring 2024, to evaluate longitudinal changes in their programming knowledge and skills. The initially 164

first year upper secondary students were surveyed in Fall 2021 and 78 of those same students participated in the follow up survey in Spring 2024. The student cohort was divided into two groups: technology students who had programming as a mandatory part of their curriculum (group TECH), and students from other disciplines such as natural sciences and social sciences, for whom programming was not compulsory (group SCI). This division allowed us to treat the group of students without mandatory programming as a control group, and those with programming as the experiment group. While not a true randomized experiment, this setup provided a basis for a semi-pseudo (or quasi) random experimental approach to evaluate the educational interventions. The follow-up allows us to measure not only the improvement in students' programming abilities but also the overall effectiveness of the programming curriculum.

### A. Research questions

1) How do the programming skills of students entering upper-secondary school vary based on their self-reported programming experience in primary school?
2) How does the correct response rate for different programming topics (variables, conditionals, and loops) vary between block and text modalities?
3) How does student performance in programming change from their first semester to their last semester of upper secondary school?

## III. Related Works

The incorporation of programming into K-12 education has generated attention worldwide [1]. As programming curricula develop globally, a prevalent strategy involves initially introducing students to block-based programming before transitioning them to text-based programming [7], [12]. This method is widely regarded as beneficial for leveraging the intuitive, less frustrating aspects of block programming to build a foundational understanding necessary for more syntactically demanding text programming.

Block-based programming, characterized by its use of colorful, draggable blocks, simplifies programming by eliminating the need for syntax, thus making it more accessible and akin to natural language [9]. This format helps students visually understand how pieces of code interact, which is especially advantageous for beginners. **Despite its benefits in early education, critics argue that block programming might create a dependency that can hinder students as they advance to text-based coding**, which requires a nuanced understanding of syntax and semantics including capitalization, punctuation, and control structures [7], [9].

Empirical studies reflect varied outcomes in this educational approach. For instance, it has been found that students generally perform better on block programming tasks, particularly in areas involving variables, conditional logic, and loops [9]. Similarly, there have been results that block programming could enhance comprehension of complex programming structures in upper secondary education better than text-based approaches [7]. However, challenges remain, particularly in interpreting loops and conditional statements, where students often struggle more with text programming compared to block programming [13], [14].

Conversely, Weintrop and Wilensky reported minimal differences in performance between students introduced to programming through block-based and text-based methods once they transition to more advanced languages like Java [12]. This suggests that the initial learning context might not have a long-term impact on programming proficiency. Furthermore, a meta-analysis of 29 studies indicates a modest but significant positive effect of block programming on learning outcomes, though this effect varies with the age of students, the tools used, and instructional methods [1].

In Sweden, despite state mandated programming curriculum, Mannila et al. found that actual experience among students remains limited, particularly as they advance through primary school. Additionally, a majority of students do not engage in programming activities outside of school [5]. This gap highlights the need for enhanced programming education strategies that not only address the transition from block to text programming but also encourage engagement outside the formal educational setting.

These insights collectively underscore the complex dynamics of programming education and the critical need for curricula that effectively bridges the gap between different programming modalities.

## IV. Methodology

This study was conducted to assess the programming knowledge and skills of first-year upper secondary students and evaluate the quality of programming instruction in primary school. A programming survey was iteratively developed in collaboration by a team of educators and researchers over two years. The survey aimed to measure the students' understanding of both block and text programming before they received any formal instruction in upper secondary school. The same survey was used and modified slightly for the follow up survey, two and a half years later.

The survey was divided into three sections, covering background questions, block programming, and text programming. Each section was crafted to evaluate different aspects of the students' programming knowledge, using multiple-choice questions to facilitate analysis.

**Survey Format:** The survey was constructed and administered using Google Forms because of requirements from the upper secondary school. The students were given 30 minutes to complete the survey, which included:

- Background questions to gauge their prior experience with block and text programming, both formally in school and informally.
- Block programming questions in Scratch
- Text programming questions in Python

The block and text programming sections were isomorphic and focused on variable assignment, conditional logic, loops, and integrated programming concepts. Each programming

question followed a consistent format: one or several variables were initialized, possibly modified through programming constructs, and used to produce an output. Students were asked to select the correct output from four options, crafted to reflect common misunderstandings and logical errors.

### A. Programming Questions Detail

- **Variable Assignment:** where a variable was assigned to a constant value or the value of another variable.
- **Conditional Logic:** using relational operators (greater-than and less-than) were incorporated.
- **Loop Constructs:** where a section of code repeats a specified number of times
- **Conditional Loop Constructs:** where a section of code repeats until a particular condition is met.
- **Complex and Embedded Problems:** more advanced questions utilizing a conditional expression embedded in a loop.

Figures 1 and 2 show examples of the complex block and text programming questions from the survey.

These figures illustrate the complexity of the programming tasks students were required to analyze. In accordance with the main goal of the project, these more complex programming questions were designed to identify how many students might not require basic explanations of these topics at the start of the introductory programming course in upper secondary school. The primary aim of this work was to evaluate the programming education from primary school and equip introductory programming teachers with valuable insights into their students' existing programming skills. The survey's findings are intended to serve as a diagnostic tool, helping educators adjust their instructional approaches based on a detailed understanding of students' prior knowledge and learning needs.

Additionally, the follow up survey was conducted to assess the quality of the programming instruction in upper secondary school and to gather data on how students would be prepared to study programming at university.

### B. Survey Instructions

Students were always encouraged to make their best effort even if they were unfamiliar with programming. The students were also told they could guess on questions if they were not sure or skip questions completely. Blank sheets of paper and pencils were distributed to all the students with the instructions to use the paper if needed. The scratch paper was not included in the analysis. The students were told they had 30 minutes to answer some background questions, the block programming questions and the text based programming questions.

**Participation and Ethics:** Participation in the survey was mandatory, however, students were given the option to opt in to have there results be anonymized and used for research. Students were informed that their grades would not be affected by their participation or performance on the survey. Gender distribution was not included in the study. (The main goal of this work was to provide useful information about the students' programming level to the introductory programming teachers.)

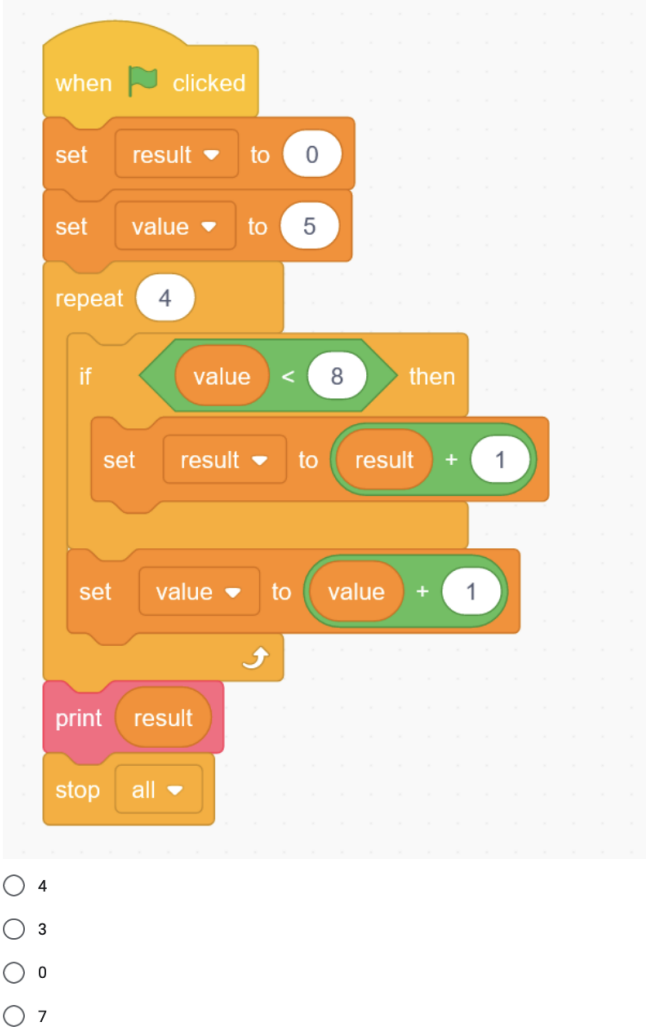


Fig. 1. Example of a complex block programming question using Scratch.

## V. Results

The initial survey in 2021 revealed significant differences between the two student groups already at the beginning of upper secondary school. Group TECH had a higher mean score of 12.41 (SD = 4.68) compared to Group SCI's mean score of 8.17 (SD = 4.13). A t-test confirmed these differences were statistically significant ($t(164) = 5.10, p < 0.001$) with a Cohen's d effect size of 0.85, indicating a substantial difference in performance.

By 2024, the follow-up survey showed a marked improvement in Group TECH's performance, with their mean score increasing to 24.89 (SD = 3.33), suggesting more consistent performance. Similarly, Group SCI improved their average score to 14.30 but also increased in variability (SD = 8.01). The difference in mean scores between the groups widened significantly over three years, supported by a stronger t-test result ($t(78) = 5.80, p < 0.001$) and a larger Cohen's d effect size of 1.75. This nearly doubled effect size suggests a

```
result = 0
count = 5

for i in range(5):
    if count < 2:
        result = result + 1
    count = count - 1

print(result)
```

- ○ 0
- ○ 1
- ○ 2
- ○ 5

Fig. 2. Example of a complex text programming question in Python.

more pronounced difference, indicating a stronger distinction between the groups over time. Descriptive statistics are shown in Table I.

| Group | Year | Mean Score | SD | N |
|---|---|---|---|---|
| Group TECH | 2021 | 12.37 | 4.62 | 117 |
| | 2024 | 24.98 | 3.30 | 58 |
| Group SCI | 2021 | 8.45 | 3.98 | 47 |
| | 2024 | 14.30 | 8.00 | 20 |

TABLE I
DESCRIPTIVE STATISTICS FOR GROUP TECH AND GROUP SCI FOR THE YEARS 2021 AND 2024, WHERE SD IS STANDARD DEVIATION AND N IS THE NUMBER OF STUDENTS.

### A. By Programming Topics

The analysis of the programming assessment responses revealed distinct performance patterns between Group TECH and Group SCI across different programming topics. The questions were grouped by topics based on their content and format: block-related (variable, conditional, loop, complex-embedded) and text-related (variable, conditional, loop, conditional-embedded). An overview of the results for 2021 are shown in Figure 3.

Overall, Group TECH outperformed Group SCI across all categories. In 2021, students were generally well-prepared for programming topics closely related to mathematics, such as variables and conditional logic. However, they were significantly less prepared for more complex topics, such as loops and embedded conditional statements. For variables and conditionals, there was not a substantial difference in performance between text and block formats. However, for loops and more complex questions, students tended to answer block format questions more correctly.

By 2024, there was an increase in correct response rates across almost all categories, particularly among Group TECH

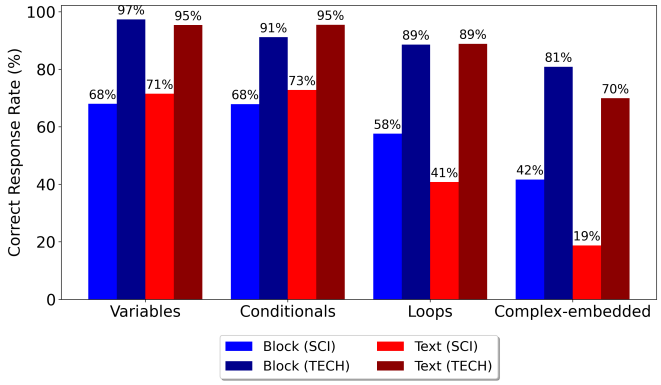


Fig. 3. Correct response rate by category and group 2021

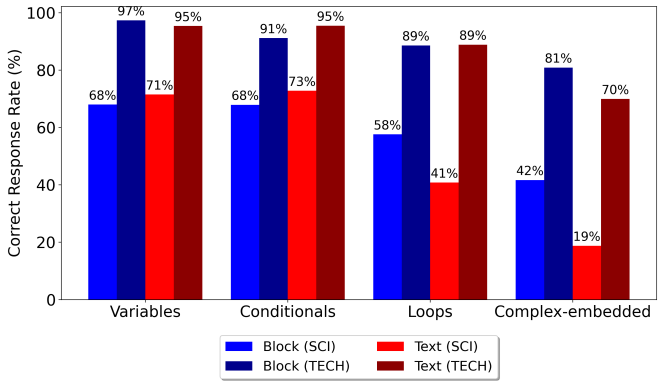


Fig. 4. Correct response rate by category and group 2024

students who studied programming. Notably, there was a significant improvement in performance on text-based loop questions and more complex questions. Despite these gains, even within the TECH group, only 70% of students answered the most complex questions correctly. This indicates that nearly one-third of the students may not be adequately prepared for such questions at the introductory programming course level at university.

### B. By Self-Reported Experience

The analysis of the 2021 survey revealed differences in average performance based on the number of self-reported hours students spent inside and outside primary school, or in formal and non-formal education.

Table II summarizes descriptive statistics for Group TECH and Group SCI based on self-reported programming hours inside and outside of primary school. In general, as the self-reported hours of programming increased, the average scores for both groups trended upwards, indicating a positive relationship between programming practice and performance. Notably, the most significant difference between the groups was not in the amount of self-reported hours in formal educational settings, where both groups reported similar hours, but in informal settings outside of school. Group TECH students reported substantially more hours spent on programming outside of

school, suggesting that additional informal practice contributed to their higher average scores.

| Group | Location | Prog. Hours | % | Mean | SD |
|---|---|---|---|---|---|
| TECH | In School | Less than 1 hr | 5.3 | 9.25 | 4.69 |
| | | 1 - 5 hrs | 46.5 | 12.55 | 4.53 |
| | | 6 - 10 hrs | 21.9 | 13.84 | 4.38 |
| | | 11 - 15 hrs | 8.8 | 12.00 | 3.13 |
| | | More than 15 hrs | 17.5 | 17.33 | 1.97 |
| | Out of School | Less than 1 hr | 44.8 | 10.33 | 4.72 |
| | | 1 - 5 hrs | 27.6 | 12.44 | 3.72 |
| | | 6 - 10 hrs | 5.2 | 14.50 | 4.04 |
| | | 11 - 15 hrs | 6.9 | 16.63 | 1.77 |
| | | More than 15 hrs | 15.5 | 16.33 | 3.20 |
| SCI | In School | Less than 1 hr | 1.9 | 6.00 | 3.30 |
| | | 1 - 5 hrs | 51.9 | 8.67 | 4.13 |
| | | 6 - 10 hrs | 15.4 | 7.88 | 3.56 |
| | | 11 - 15 hrs | 15.4 | 8.63 | 3.46 |
| | | More than 15 hrs | 15.4 | 1.00 | - |
| | Out of School | Less than 1 hr | 59.6 | 6.68 | 3.52 |
| | | 1 - 5 hrs | 23.1 | 9.75 | 3.72 |
| | | 6 - 10 hrs | 7.7 | 11.25 | 5.62 |
| | | 11 - 15 hrs | 7.7 | 9.00 | - |
| | | More than 15 hrs | 1.9 | 9.25 | 3.20 |

TABLE II
DESCRIPTIVE STATISTICS OF SELF-REPORTED PROGRAMMING HOURS FOR GROUP TECH AND GROUP SCI BASED ON PROGRAMMING HOURS INSIDE AND OUTSIDE OF PRIMARY SCHOOL IN 2021.

Figures 5 and 6 illustrate the distribution of students in Group TECH and Group SCI based on their self-reported frequency of programming outside of school in 2024. Group TECH students show a more balanced distribution across all categories, with a significant proportion engaging in programming activities regularly or several times. In contrast, Group SCI students predominantly fall into the "None or almost none" category, indicating less frequent engagement in programming outside of formal education.

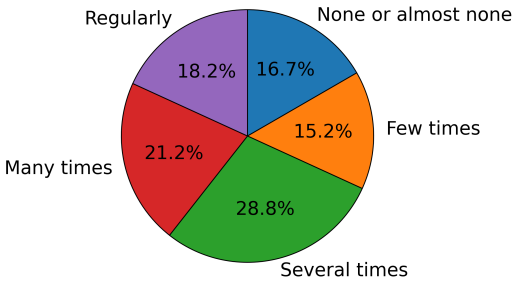


Fig. 5. Distribution of students self-reported programming experience outside of school for Group TECH in 2024

### C. 2021 to 2024 Comparison

These results underscore the substantial performance differences between Group TECH and Group SCI over time. Group TECH consistently outperformed Group SCI in both the initial and follow-up surveys. The significant improvements in

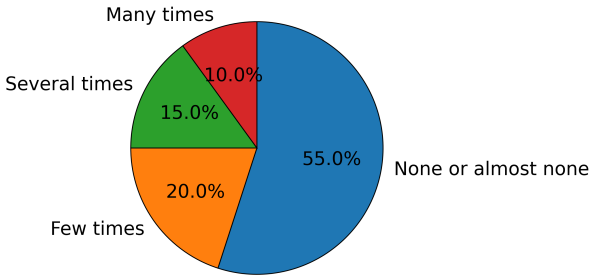


Fig. 6. Distribution of students self-reported programming experience outside of school for Group SCI in 2024

scores from 2021 to 2024 for both groups suggest progress, yet the gap between Group TECH and Group SCI remains pronounced.

A paired t-test was conducted to compare the programming assessment scores of Group TECH from 2021 and the 2024 follow-up survey. The analysis was performed on students who participated in both surveys. The results revealed a highly significant improvement in scores from 2021 to 2024 ($t(57) = -17.20, p < 0.001$). The effect size (Cohen's d = 2.30) indicates a very large improvement in performance, suggesting that the educational interventions and programming experiences over the two and a half years had a substantial positive impact.

## VI. LIMITATIONS

Some limiting and confounding factors in the study include: the multiple choice nature of the survey where students could correctly guess the answers; the time limit of the survey which could have made students feel rushed or stressed, and the fact that due to scheduling some students took the survey before lunch and some students after lunch which could have effected some students moods. Also, this survey only dealt with the *reading* or *tracing* of programming problems in block and text and not the explicit *writing* of programs.

## VII. DISCUSSION

Ultimately, this research underscores the importance of aligning educational content with students' existing competencies and learning trajectories, ensuring that programming instruction is both relevant and challenging.

These results affirm that consistent and formal programming education plays a crucial role in enhancing student proficiency, as evidenced by the improved scores and reduced variability in Group TECH's performance over the study period.

The analysis of the survey data reveals distinct patterns in how the amount of programming education during and outside of school trends with academic performance among students.

These results suggests that more extensive programming education both during and outside school hours is associated with higher academic achievement in programming-related tasks. Moreover, the consistency of higher scores with increased

exposure indicates a cumulative benefit of continued practice and education in programming.

Our results also show that students in our study arrive to upper secondary school prepared for block programming and lack the knowledge to grasp more complex programming examples, like a conditional statement inside a loop. Our results highlight that even a few hours of programming instruction in primary school can make an impact on students performance.

Thankfully, in this study we saw a positive effect from programming instruction in primary and upper secondary schools, which was not guaranteed. The scenario could have been that the amount of programming instruction correlated with lower student performance. If programming instruction at the compulsory level gives students negative experiences and attitudes towards programming, then we would expect to see this pattern. This could be the case for some students, however in this study we measured a positive effect. As programming instruction becomes more established in compulsory schools we hope to see improved results as we continue to administer this survey. Ideally, we would like to see students arriving at upper secondary school more prepared for text programming and prepare our students for more complex text programming at the university.

## Acknowledgment

Thanks to the teachers and students at NTI Gymnasiet Uppsala who contributed to this work. Special thanks to Kasper Davidsson, Erik Bokström, David Wålstedt, Mikael Björk and Robin Kastberg for their contribution and feedback on the work-in-progress paper.

## References

[1] 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.

[2] T. Gardner, H. C. Leonard, J. Waite, and S. Sentance, "What do we know about computing education for k-12 in non-formal settings? a systematic literature review of recent research," in *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*, ser. ICER '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 264–281. [Online]. Available: https://doi.org/10.1145/3501385.3543960

[3] S. N. A. for Education. (2022) Swedish national curriculum. [Online]. Available: https://www.skolverket.se/publikationsserier/styrdokument/2022/laroplan-for-grundskolan-forskoleklassen-och-fritidshemmet—lgr22

[4] 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/

[5] L. Mannila, A. Åkerfeldt, S. Kjällander, and F. Heintz, "Exploring gender differences in primary school programming," in *2022 IEEE Frontiers in Education Conference (FIE)*, 2022, pp. 1–9.

[6] MIT. (2023) Scratch. [Online]. Available: https://scratch.mit.edu/

[7] C. Szabo, J. Sheard, A. Luxton-Reilly, Simon, B. A. Becker, and L. Ott, "Fifteen years of introductory programming in schools: A global overview of k-12 initiatives," in *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3364510.3364513

[8] P. Foundation. (2023) python. [Online]. Available: https://www.python.org/

[9] 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.

[10] J. Snider, E. Bokström, K. Davidsson, A. Eckerdal, and R. Kastberg, "Block and text programming in swedish high school: What do students know on their first day*f*," in *2022 IEEE Frontiers in Education Conference (FIE)*, 2022, pp. 1–5.

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

[12] D. Weintrop and U. Wilensky, "Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms," *Computers & Education*, vol. 142, p. 103646, 2019.

[13] 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.

[14] 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.