National Prior Knowledge Test in Programming
How proficient are incoming higher education students?

Sondre S. Bolland

University of Bergen

sondre.bolland@uib.no

Keywords: Prior Knowledge · Test · Introduction to Programming · CS1

Table of Contents

Unfinished bits

- Analyse attitude questions
- Conclusion
- Make front page a bit nicer

1 Introduction

The implementation of Kunnskapsløftet 2020 (LK20) has made programming a mandatory part of the curriculum within mathematics subjects in secondary school [1]. Three years later, the first cohort of students with this programming experience entered higher education institutions. This existing programming competence can have significant implications for the teaching of ICT subjects at universities and colleges. If students already possess a solid understanding of basic programming elements, instruction can be intensified and reach a more advanced level as early as the first semester. However, it is necessary to investigate whether this is the case. There has been substantial variation in the quality of programming education during the implementation of LK20 [3] [5]. Teachers' knowledge of programming varies greatly, with some having computer science didactics as part of their education while others have never coded before. Therefore, it is important to assess the extent and comprehensiveness of students' prior knowledge.

The National Prior Knowledge Test in Programming is a tool for assessing students' programming skill. The test covers the fundamental elements of introductory programming taught at different universities and colleges in Norway. The test was inspired by the National Prior Knowledge Test in Mathematics which has been hosted by the Norwegian National Council of Mathematics since 1984 [2]. This test was created to monitor the quality of the prior mathematical knowledge of beginner students. However, while the mathematics test measures mainly prior knowledge in the form of basic concepts and skills taken from the syllabus in elementary school, the programming test entails curriculum objectives found in *Introduction to Programming* (CS1) [4] at universities and colleges. The goal of the test is to evaluate students' level of knowledge regarding the curriculum taught at higher education institutions, regardless of what they have learned in secondary school. By testing the students in the concepts found in CS1 we aim for instructors to be better able to develop and adapt their courses to this new found prior knowledge.

This report presents the results from the first iteration of the test administered august 2023. We start by summarizing the key findings. Following this, we present the methodology, outlining the test's content, distribution, test-taker demographics, and data processing procedures. In the results section, we initially present the primary outcome, which is the overall performance of all students and an evaluation of the impact of LK20. Subsequently, we delve into the students' specific backgrounds and assess how these factors have influenced their performance. Finally, we conduct a detailed analysis of each task within the test, aiming to gain a deeper understanding of the programming concepts in which students exhibit proficiency and identify common misconceptions.

2 Summary

The test was taken by 2,133 students (following data pruning, 1,767 individuals) at seven different higher education institutions in Norway. This collective yielded an average score of 46.1%. The students' academic backgrounds exhibited significant variations with potential correlations to their performance. As expected, a notable factor contributing to the variation in scores became evident when evaluating the students' year of completion of secondary education. By dividing the students into two cohorts, those who graduated in 2023 (n=483) and those who graduated before 2023 (n=1,280), we could discern a specific subset of students who had been exposed to the prescribed curriculum objectives of LK20, including compulsory programming within their mathematical coursework. This division in educational experiences revealed a significant difference with the 2023 graduates achieving an average score of 63.3%, while their counterparts from earlier graduation years achieved an average score of 39.4%.

However, it is crucial to acknowledge additional noteworthy factors influencing the performance of the 2023 student cohort. Among these factors, the presence of elective programming courses during their secondary school education is significant. When the students who participated in elective programming courses are excluded from the dataset the mean performance of both groups undergoes a reduction. The 2023 cohort looses a large number of their highest scoring students and their mean score sinks from 63.3% to 54.5%. Furthermore, another pivotal determinant of performance relates to the level of mathematics courses completed by these students. Within the 2023 cohort, those who have undertaken the most advanced mathematics course, Science Math 2 (R2), achieve a commendable mean score of 66.4%. In contrast, those who have completed the most fundamental concluding course, Practical Math 2 (P2), exhibit a notably lower mean score of 38.0%. A third factor is independent exploration of programming outside of the classroom. 16.1% of the students reported having at least 30 hours of experience with block or text based programming outside of formal education. Excluding these students from the 2023 cohort the mean test score drops from 63.3% to 57.3%. These observation underscore the importance of considering the extent of a student's involvement in programming, particularly their participation in elective programming courses, outside programming experience and enrollment in advanced mathematics courses, the last often being a prerequisite for admission to ICT study programs.

By excluding the factors of elective courses and outside experience, we can obtain a clearer assessment of the impact of LK20. When we restrict the analysis to students without this additional experience, the 2023 cohort still outperforms those who graduated in preceding years with a 19.4% higher mean score. This suggests that LK20 has indeed led to a significant increase in the proportion of students who possess a solid foundational understanding of programming. However, it is important to note that a considerable portion of the student population still demonstrates less proficiency in this area. The results from the National Prior Knowledge Test in Programming offer promising prospects for the advancement of computer science education in Norway. Nonetheless, it is clear that a significant number of students still commence their higher education in computer science with rudimentary knowledge.

3 Methodology

3.1 The Test

The prior knowledge test was developed in partnership with 9 higher education institutions in Norway[1]. It drew inspiration from the prior knowledge test in mathematics, incorporating several analogous elements related to student backgrounds.

The test was divided into two sections, with the first section dedicated to gathering information about students' backgrounds, while the second section was designed to evaluate their programming proficiency. In the background information section, students were surveyed on the following items:

- educational institution
- study program
- gender
- when they graduated secondary school
- which math courses they completed
- whether they completed elective programming courses
- whether they have completed a university level programming course
- programming experience outside of formal education
- their attitude towards programming

The second segment of the test encompassed 21 programming tasks, each addressing fundamental programming concepts crucial to the CS1 curriculum:

[1] Institutions developing the test: University of Bergen, The Norwegian University of Science and Technology, University of Oslo, Kristiania University College, Norwegian University of Life Sciences, University of Agder, Western Norway University of Applied Sciences, Oslo Metropolitan University and University of Stavanger.

- datatypes
- variables
- booleans
- conditionals
- loops
- lists
- functions

For each concept, a range of tasks was provided, progressively increasing in complexity. It's important to note that the specific tasks will not be disclosed in this report, as they are intended for utilization in subsequent iterations of the test.

The test was administered in Norwegian, and both test items and their responses cited here have been translated to English by the author.

3.2 Distribution

The test was distributed in the initial weeks of the 2023 fall semester. Each institution had the responsibility for making the test available and oversee it's completion under the following guidelines:

- Make no aids available, such as computers, phones or print outs,
- maximum two hour completion time,
- administer the test as early in the semester as possible. We want to map their prior knowledge, not anything they learn at university.

3.3 Data Pruning

The variables of gender, educational institution, and study program were provided by respondents in an open-text format. Consequently, a manual labeling process was employed to categorize and assign appropriate labels to these responses. For instance, terms like *jente*, *kvinne*, *ho*, etc., were all unified under the label *f* to represent the female gender. However, it is important to note that not all submissions received accurate labels, due to unintelligible responses, resulting in a reduced sample size for specific categories.

In the analysis pertaining to gender, only the binary classifications of *female* and *male* were considered, given the limited representation of other gender categories within the dataset (n=49). This decision was made due to the insufficient sample size of alternative gender identities for meaningful statistical analysis.

All blank responses have been removed from the dataset (n=57). This includes a student who answered the initial questions regarding their background, but gave no answers in the programming section. All students who reported having completed a university level course in programming were also omitted (n=252). This was done to restrict the evaluation of student programming proficiency to only that learned prior to starting higher education.

This pruning reduced the size of our dataset from 2133 responses down to 1767.

3.4 Demographic

The test was distributed at seven higher education institutions in Norway.

- University of Bergen (UiB)
- The Norwegian University of Science and Technology (NTNU)
- University of Oslo (UiO)
- Kristiania University College (Kristiania)
- Norwegian University of Life Sciences (NMBU)
- Western Norway University of Applied Sciences (HVL)
- University of Stavanger (UiS)

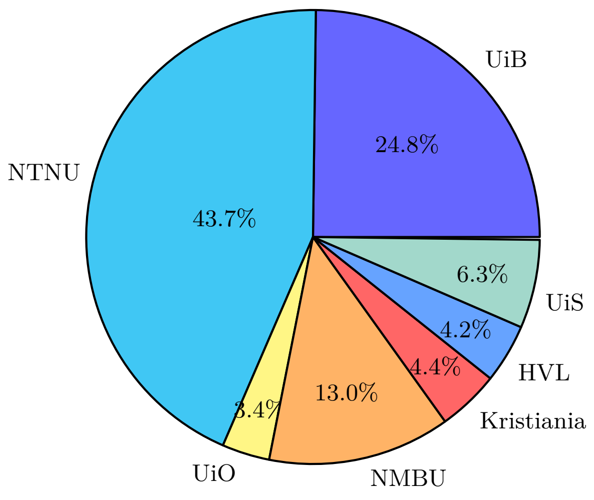


Fig. 1: Distribution of students by institution (n=1752).

The test was given to students in *Introduction to programming* courses (CS1). A large part of the students participating in the test were enrolled in computer science study programs, although there are several other study programs in other fields of study that have CS1 as a mandatory course. The students were asked what study program they were enrolled in, in a open text format. This yielded a large variety of answers, which has proven to be very hard to discern what program many of the students actually meant. Hence, we do not have an exact distribution of study programs.

The test was taken predominantly by male students. Figure 2 shows the distribution of gender based on the binary values *Female* and *Male*, in addition to those that identified outside of these two options or did not give an answer (*other*).

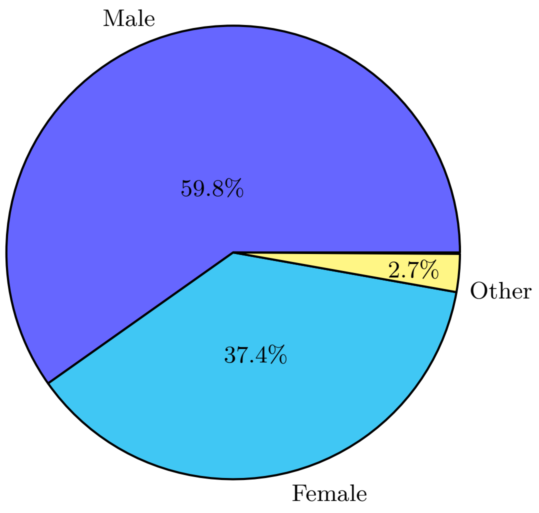


Fig. 2: Gender distribution of participating students (n=1767).

The students were queried about the year in which they completed their secondary school education. The distribution of participants according to their year of graduation can be found in Figure 3.

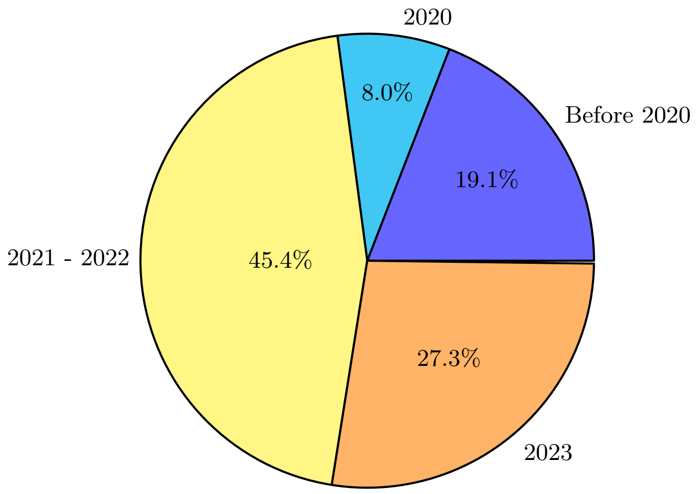


Fig. 3: Number of students of each graduating year (n=1764).

4 Results

In this section of the report, the main results are presented before a review of the background the students have on the various study paths and what connection there is between background and results. We also take a close look at how well they performed in specific programming tasks to understand their grasp of the different concepts.

4.1 Main Results

The average test score was 10.43 out of a maximum of 22.6, with a standard deviation of 6.0. Figure 4 illustrates how the scores are distributed among all students. This distribution reveals two noticeable peaks, with the majority of students scoring around 7.5. Additionally, there is a group of high-achieving students who scored 18 points or more. On the other hand, a fair amount of students at the lower end of the distribution demonstrate a limited grasp of fundamental programming concepts.

To evaluate the impact of the LK20 curriculum, we conducted a comparative analysis of two groups of students: those who completed their secondary education in 2023 (exposed to the curriculum objectives of LK20) and those who graduated in preceding years (without exposure to LK20). Furthermore, to isolate the influence of programming taught within the mathematics subjects, we controlled for other significant factors affecting test scores, specifically elective programming courses and programming experience outside of their formal education. These two factors are explored in dedicated subsections.

The notable disparity in the distributions shown in Figure 5 highlights the improvement observed in the 2023 cohort. This improvement is evident through the significantly different mean test score (Wilcoxon rank sum test, $W = 182065$, $p < 2.2e - 16$) and the larger proportion of high scoring students. These findings suggest that the implementation of LK20 has had a substantial impact on the programming proficiency of incoming higher education students. While there are still students performing at the lower end of the scale, the average student appears to possess a greater understanding of programming compared to previous cohorts. This bodes well for the future of computer science higher education in Norway, signaling the potential for enhanced instruction in the coming years.

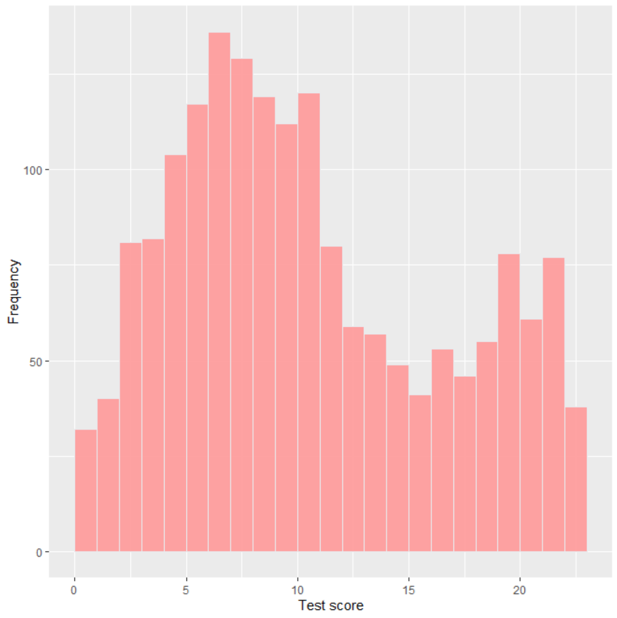
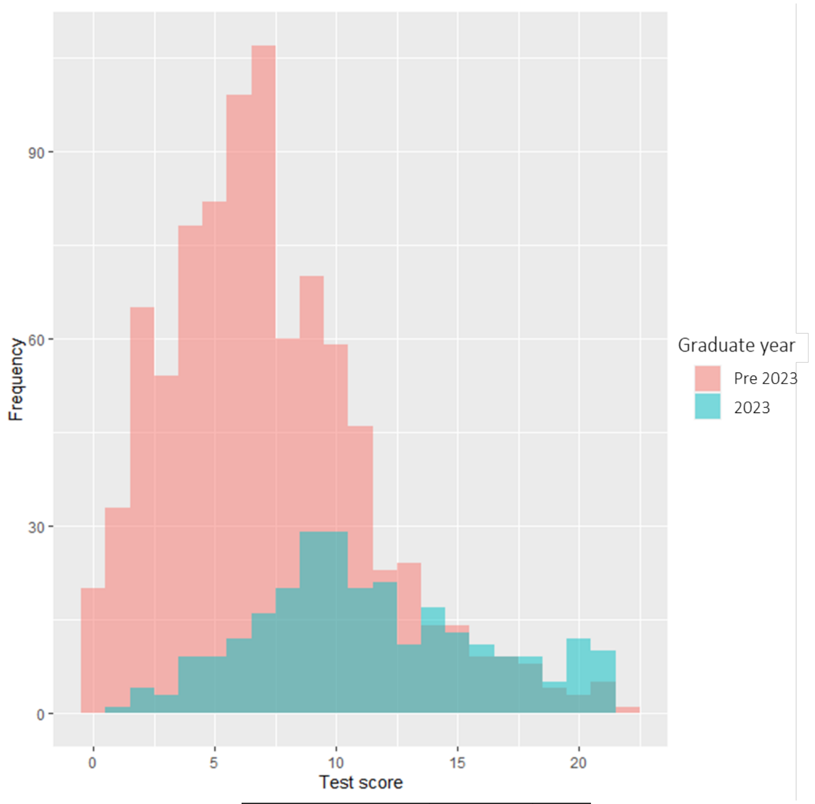


Fig. 4: Test score distribution of all students (n=1767).



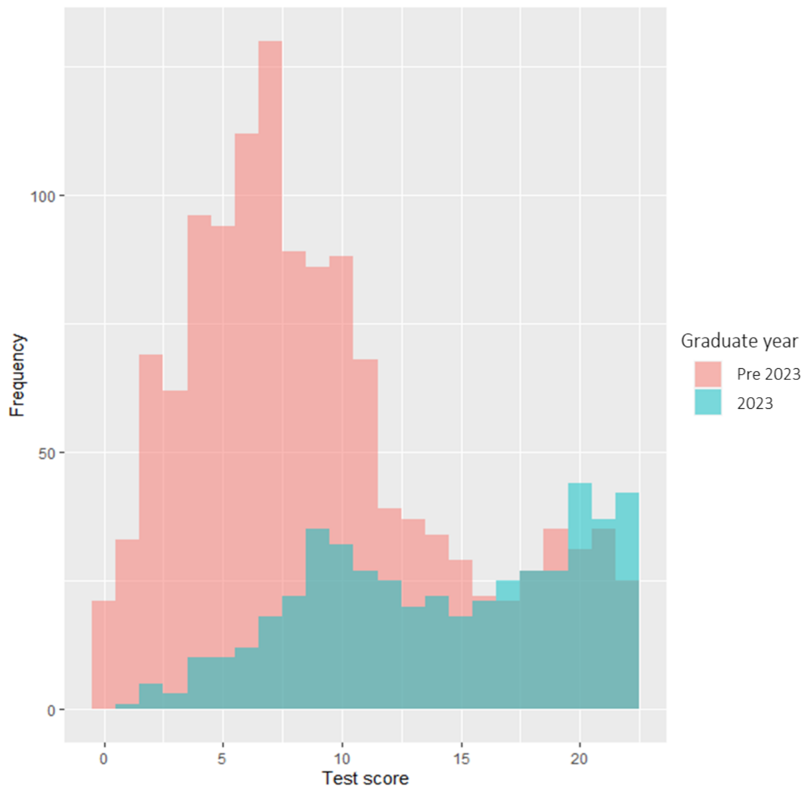
Graduate year	Pre 2023	2023
N	997	318
Mean	7.8	12.2
SD	4.9	5.1

Fig. 5: Test score distribution of the students without any electives or outside programming experience, divided by graduate year.

4.2 Prior Programming Experience in Secondary School

In the initial segment of the test, we inquired with the students regarding their prior exposure to programming before to commencing their higher education studies.

Graduation Year The educational reforms outlined in LK20 were introduced in the year 2020, resulting in programming becoming a compulsory component solely for those students who graduated in 2023 and onwards. Nonetheless, a significant portion of test participants concluded their secondary education in preceding years.



Graduate year	Pre 2023	2023
N	1280	483
Mean	8.9	14.3
SD	5.5	5.5

Fig. 6: Test score distribution divided by graduating year.

As illustrated in Figure 6, a notable distribution shift toward the right is evident when comparing students who graduated before 2023 with those who graduated in 2023. This shift corresponds to a significant increase in the mean test score for the 2023 students (Wilcoxon rank sum test, $W = 465302$, $p < 2.2e - 16$).

Do the 2023 students learn programming? In accordance with LK20 every secondary school student is meant to have programming as part of the mathematics curriculum. However, there are a large number of teachers that are not very proficient in programming, much less skilled in how to teach programming [3]. As a result, it's possible that certain students have had limited exposure to programming education, particularly if their teachers have faced challenges in effectively delivering such instruction. The students were prompted with the question *"Did you learn to program in Python in the mathematics courses?"*. While there were 483 students who graduated in 2023 only 380 answered *"Yes"*. 100 answered that they did not learn programming in the mathematics courses, and three answered that they learned another programming language. However, these three seem to have simply misunderstood the question as the programming languages they cite are the ones taught in the elective programming courses.

Although these 100 students report not having learned programming in the mathematics courses they perform significantly better than students from preceding years (Wilcoxon rank sum test, $W = 142709$, $p < 2.2e - 16$). Figure 7 shows the distribution and mean score of graduate years and self reported Python experience. Considering the vast difference in performance between *Graduated before 2023* and *Did not learn Python (2023)* it seems strange that these students have not been taught Python programming. One plausible explanation for the reported absence of programming education among the 2023 students could be that they did not perceive the level of programming instruction they received as substantial enough to confidently claim that they had learned programming.

Programming Experience	N	Mean	SD
Graduated before 2023	1283	9.0	5.5
Did not learn Python (2023)	100	13.0	6.3
Learned Python (2023)	380	14.6	5.2

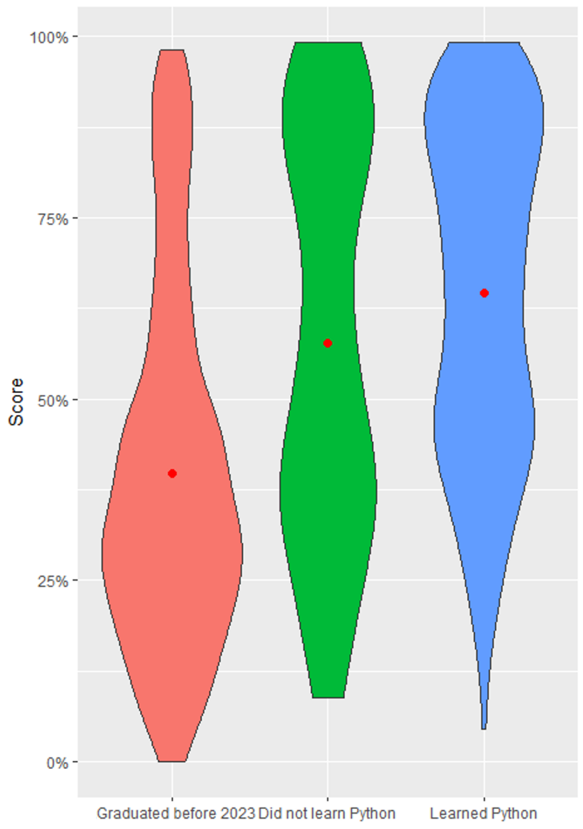


Fig. 7: Score distribution of those who graduated before 2023, and those who graduated in 2023 that reported learning Python programming and those who reported not learning programming.

Elective Programming Courses During the secondary school phase, students have the option to take three elective courses in programming: *Informasjonsteknologi 1* (IT1), *Informasjonsteknologi 2* (IT2), and *Programmering og modellering X* (PMX). These courses include, among others, the following curriculum objectives:

IT1

- develop web pages using markup language
- use algorithmic thinking and programming to explore a problem and present the result
- describe different types of algorithms and assess the effectiveness of your own program code

IT2

- apply object-oriented modeling to describe a program's structure
- develop object-oriented programs with classes, objects, methods and inheritance
- generalize solutions by developing and using reusable program code

PMX

- develop holistic and structured programs
- use and create algorithms that can be used for modelling, and assess the validity of these algorithms
- assess and use strategies for testing, troubleshooting and error handling

Although these subjects are not mandatory and are not pursued by all students, a substantial portion of students seeking admission to ICT studies opt for them. Figure 10 gives the number of students who have completed these electives.

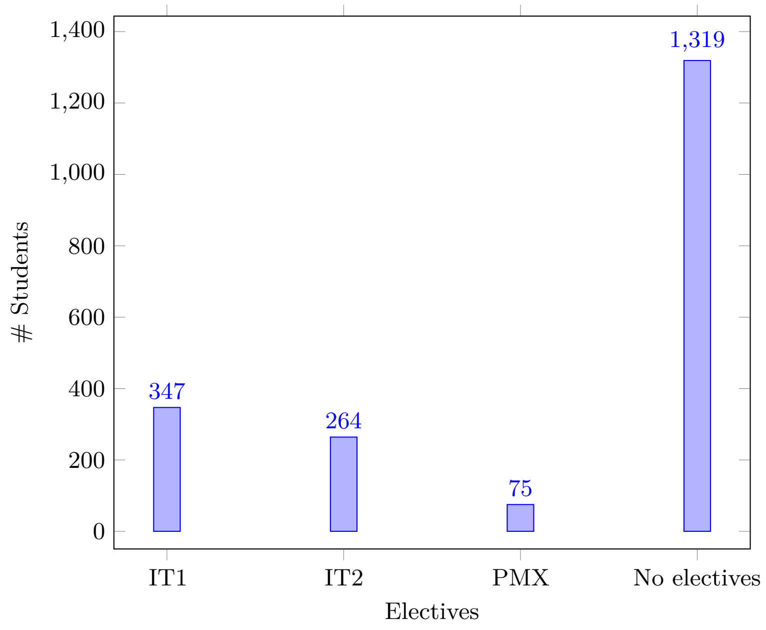


Fig. 8: Number of students who have taken elective programming courses in secondary school (n=1767).

Given that these courses emphasize programming and algorithmic thinking, it is reasonable to anticipate that students who have taken them would achieve higher test scores. Figure 11 illustrates how students' scores correlate with the elective subjects they have undertaken. The highest-scoring

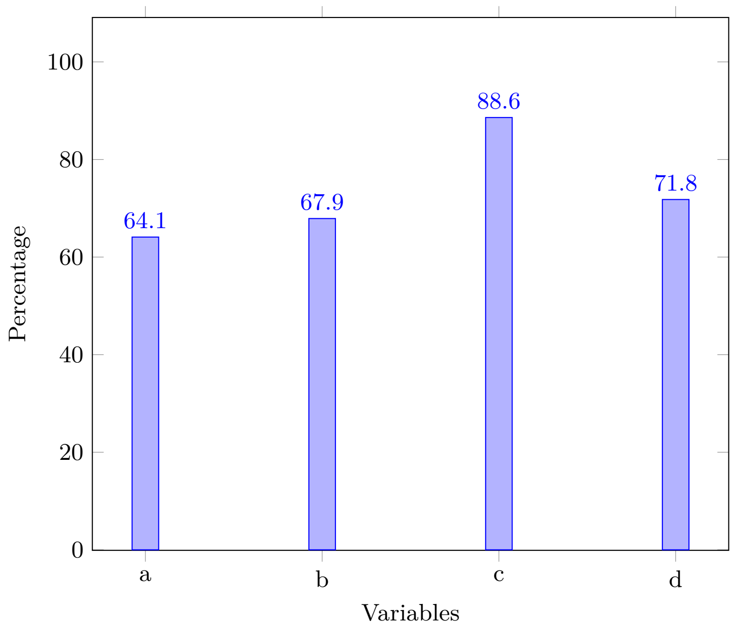


Fig. 9: Number of students who have taken elective programming courses in secondary school (n=1767).

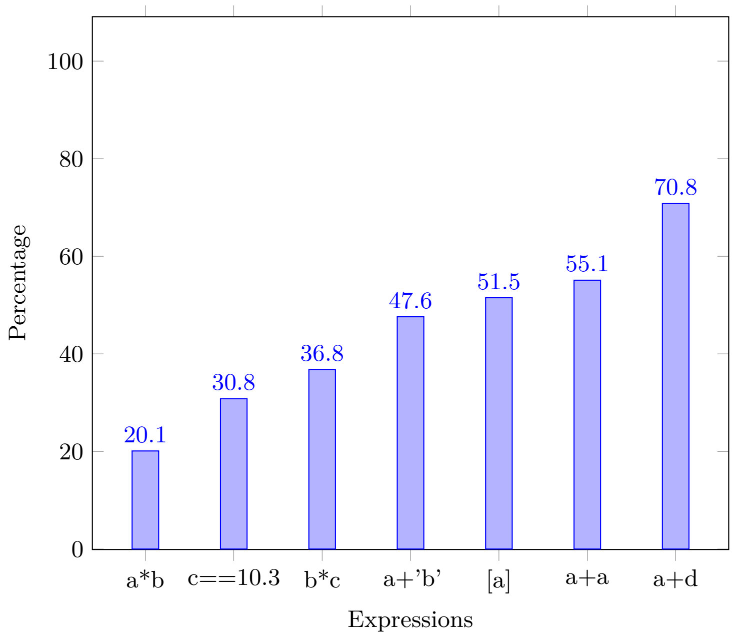
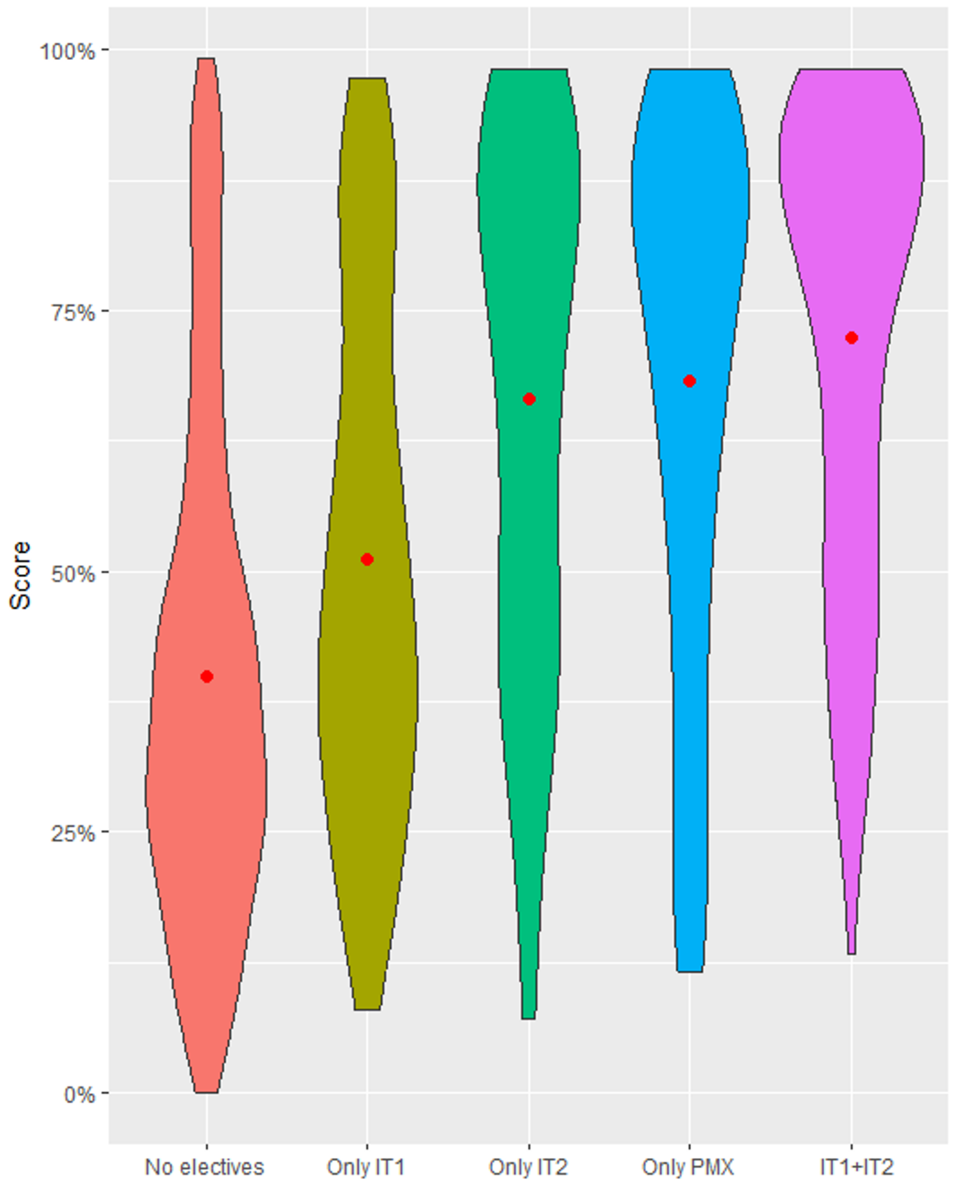


Fig. 10: Number of students who have taken elective programming courses in secondary school (n=1767).

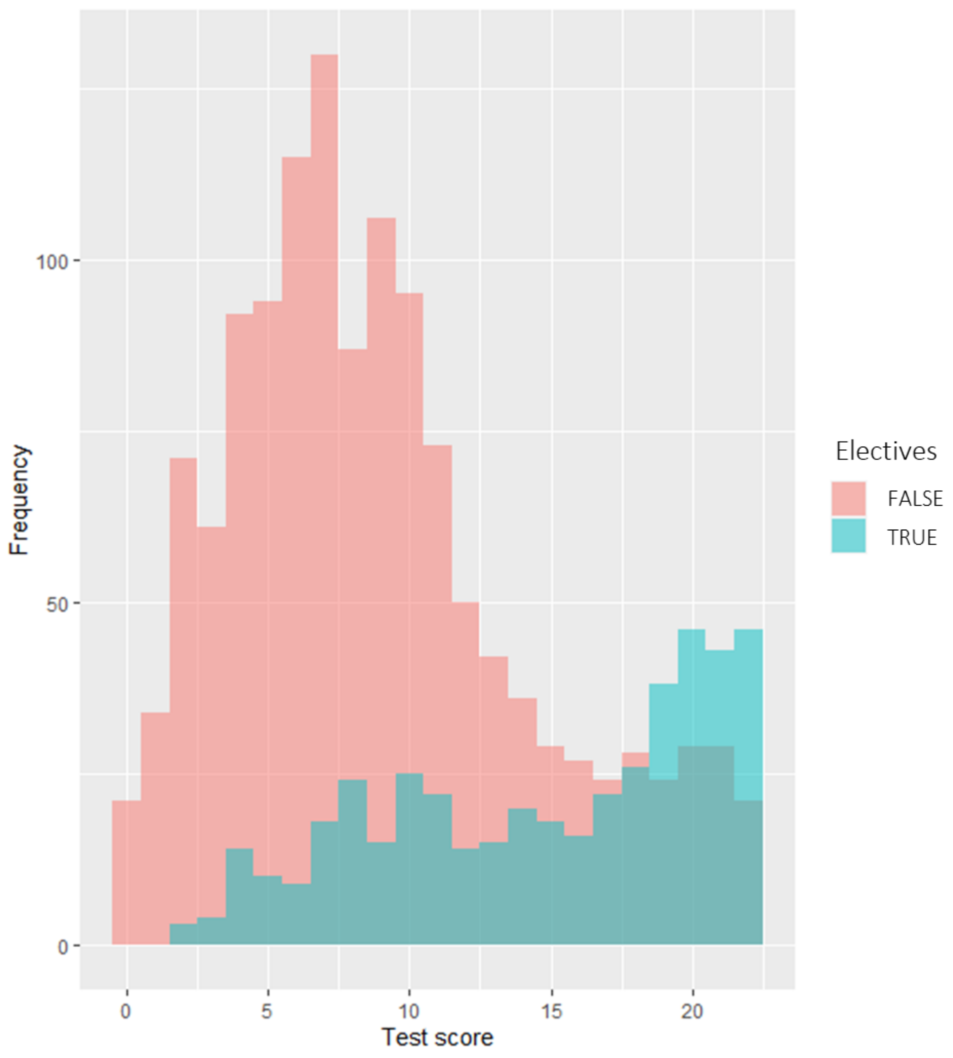


Electives	N	Mean	SD
No electives	1318	8.9	5.3
Only IT1	119	11.5	5.5
Only IT2	41	15.0	5.7
Only PMX	57	15.4	5.8
IT1+IT2	220	16.4	5.0

Fig. 11: Score distributions by elective programming course (n=1767).

students are those who have completed both IT1 and IT2. Certain subject combinations, such as IT1+IT2+PMX, have been omitted from the figure due to small sample sizes.

When we divide the students based on whether they have taken one or more electives and those who have taken none, as demonstrated in Figure 12, a significant difference in test scores is apparent (Wilcoxon rank sum test, $W = 132822$, $p < 2.2e - 16$).



Electives	True	False
N	448	1318
Mean	14.9	8.9
SD	5.6	5.3

Fig. 12: Test score distribution divided by whether the students has taken an elective programming course.

Mathematics Courses Proficiency in mathematics is frequently acknowledged as a substantial indicator of programming aptitude [**?**]. Furthermore, the higher-level mathematics courses involve more advanced programming concepts. Therefore, we surveyed the students on which mathematics courses they had completed during their secondary education. The most common mathematics courses available are the following, listed in increasing difficulty.

- Practical Math 1 (P1)
- Practical Math 2 (P2)
- Social Science Math 1 (S1)
- Social Science Math 2 (S2)
- Natural Science Math 1 (R1)
- Natural Science Math 2 (R2)

The majority of students who took the test belonged to STEM fields, where the typical admission requirement includes S1 and S2 or R1 mathematics. Certain math-intensive study programs may also demand R2 mathematics. Notably, most students had completed the Natural Science Math courses, which is the most advanced option (see Figure 13).

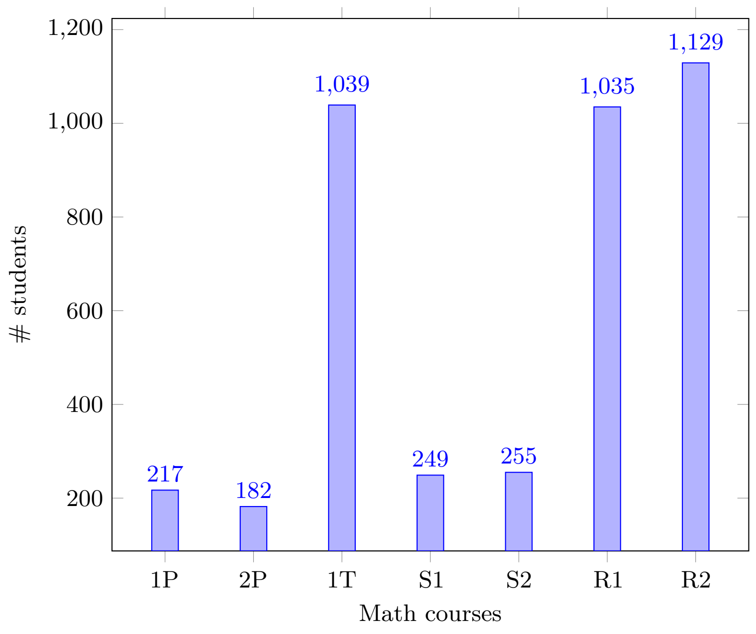
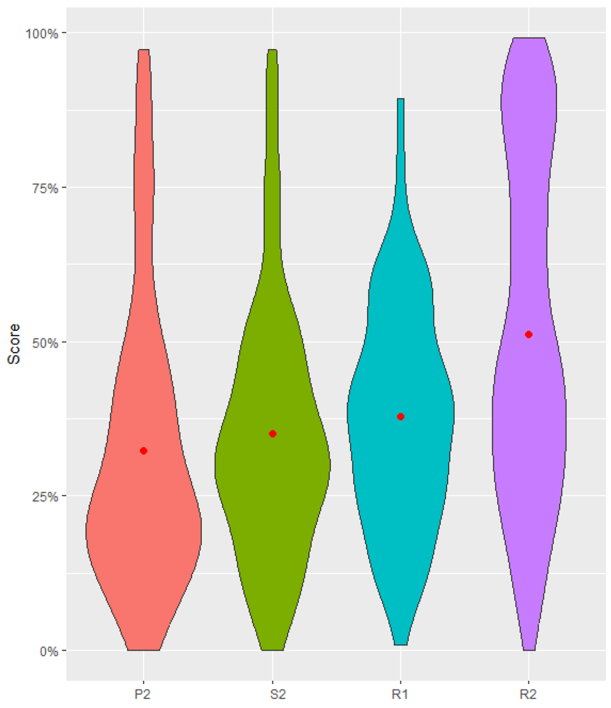


Fig. 13: Number of students taking the different mathematics courses in secondary school (n=1767).

To assess the influence of mathematics courses on test scores, we categorized the students based on the most advanced mathematics course they had completed (see Table and Figure 14). These findings are in line with our initial expectations: students who have completed more advanced mathematics courses tend to perform better in programming.



Most Advanced Math Course	N	Mean	SD
Practical Math 2	152	7.3	5.2
Social Math 2	235	7.9	4.6
Science Math 1	110	8.5	4.0
Science Math 2	1128	11.5	6.1

Fig. 14: Score by most advanced mathematics course (n=1625).

Experience outside of school A source of programming knowledge is self-directed learning outside of formal education, where individuals independently explore the field, using resources like books and online materials. We surveyed the students regarding their engagement in programming beyond the classroom, posing the following question and providing these answer options:
"What experience with programming do you have outside of school?"

- At least {30, 50, 100, 120} hours experience with text based programming,
- At least {30, 50, 100, 120} hours experience with block based programming,
- No experience outside of school.

In Figure 15, we have grouped the hours spent on different types of programming into the same categories. Approximately 16.1% of students indicated having at least 30 hours of experience with either block or text-based programming. When comparing the students with and without outside experience (see Figure 16), we notice a significant difference in mean score (Wilcoxon rank sum test, $W = 132822$, $p < 2.2e - 16$).

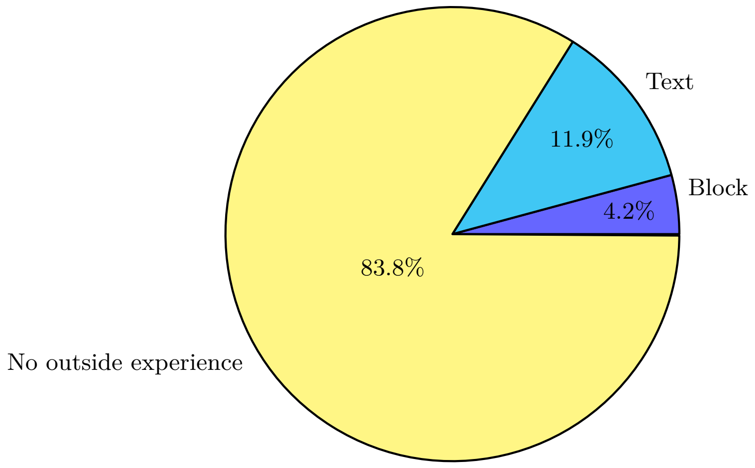
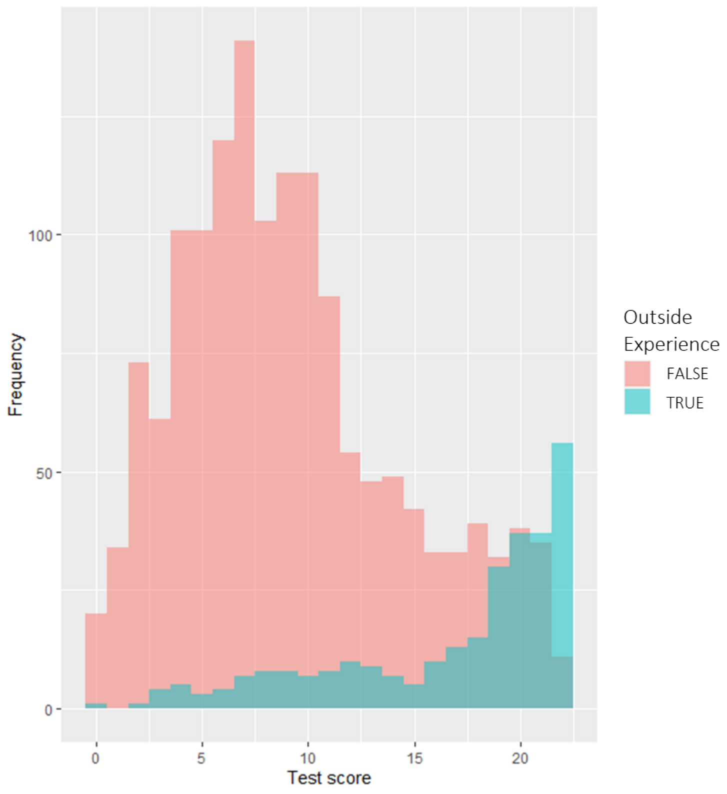


Fig. 15: Non formal programming experience (n=1767).

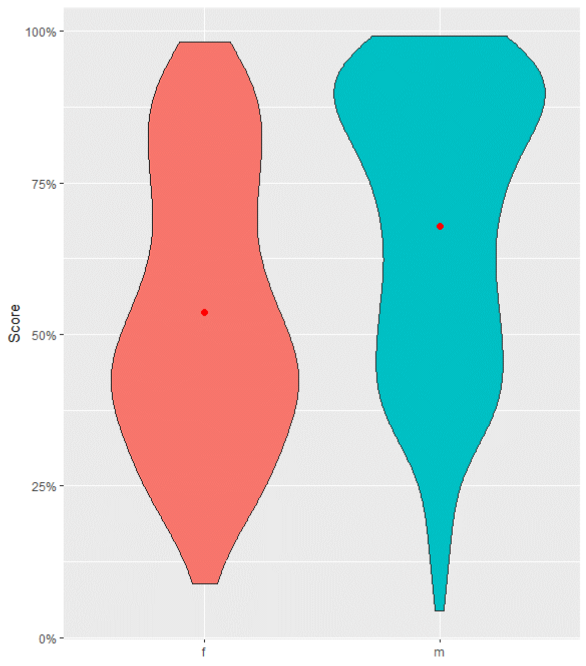


Outside experience	True	False
N	285	1481
Mean	16.8	9.2
SD	5.5	5.3

Fig. 16: Score distribution divided by whether the student had outside experience (n=1767).

4.3 Gender

We have observed a notable difference in performance when examining the variable of gender, as shown in Figure 17. Specifically, the mean test score for men is significantly higher than that for women, as shown using a Wilcoxon rank sum test, $W = 238659$, $p < 2.2e - 16$.



Gender	Male	Female
N	1056	661
Mean	11.7	8.3
SD	6.1	5.0

Fig. 17: Score distributions by gender (n=1717).

This disparity in performance appears to be influenced by variations in exposure to programming and mathematics. Within our dataset, there is a higher proportion of men who participated in elective programming courses, as indicated in Table 1.

Gender	N	No Electives	IT1	IT2	PMX
Men	1056	70%	24%	19%	5%
Women	661	82%	12%	8%	3%

Table 1: Number of students taking electives programming courses divided by gender.

Moreover, fewer women had engaged in independent exploration of programming outside of the classroom, as presented in Table 2. Additionally, a lower number of women were enrolled in the most advanced mathematics course, as outlined in Table 3. These factors likely contribute to the performance discrepancy observed between male and female students.

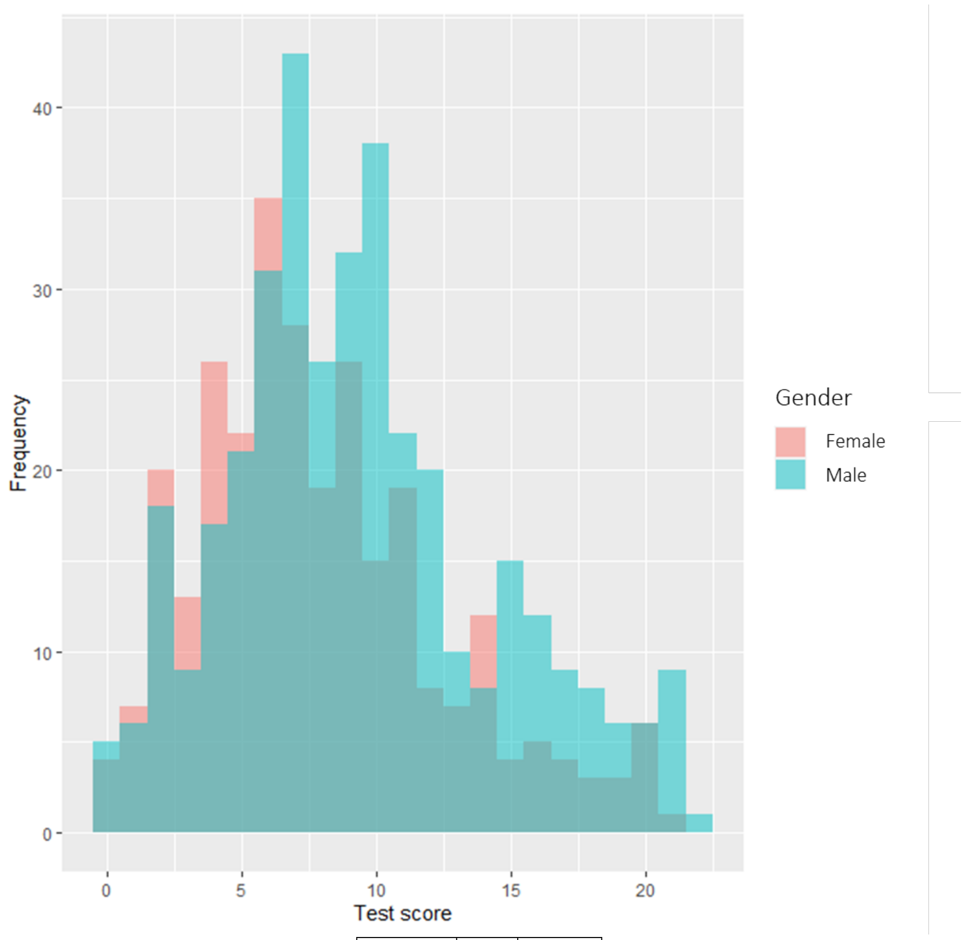
When we normalize the analysis by considering only men and women who have not taken any electives, lack outside programming experience, and have completed the R2 mathematics course, the performance gap between the two groups narrows (as depicted in Figure 18). However, it is important to note that this difference remains statistically significant (Wilcoxon rank sum test, $W = 43235$, $p = 2.819e - 05$).

Gender	N	Outside experience	No outside experience
Men	1056	22%	78%
Women	661	5%	95%

Table 2: Number of students with outside experience divided by gender.

Gender	N	P2	S2	R1	R2
Men	1056	10%	13%	5%	66%
Women	661	11%	19%	8%	59%

Table 3: Number of students taking each mathematics course divided by gender.



Gender	Male	Female
N	372	287
Mean	9.4	7.8
SD	4.1	4.5

Fig. 18: Score distribution of men and women who have had R2, no programming electives and no outside programming experience (n=659).

4.4 Programming Concepts

Every task featured in the test pertained to a designated concept category. Figure 19 gives the percentage of accurate responses achieved by all participants for each concept, subdivided by graduate year.

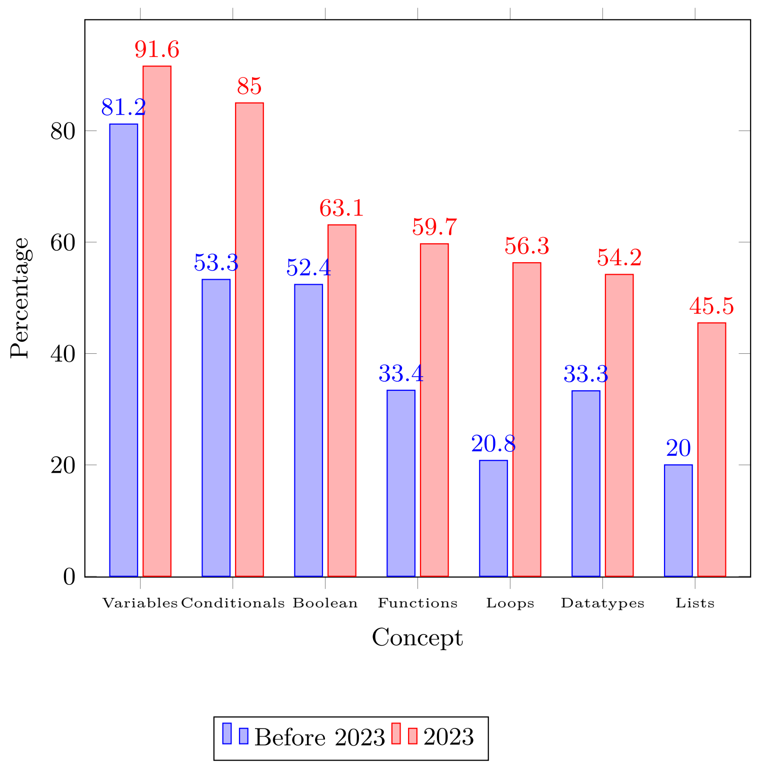


Fig. 19: Percentages of correct answers for each task concept category, by graduate year.

Performance within each category may not exclusively reflect the students' mastery of that programming concept. Variability in task difficulty plays a substantial role, with some tasks naturally being easier than others, irrespective of the underlying concept. To gain a deeper understanding of the students' knowledge and comprehension of specific programming concepts, it is important to analyze their performance on these concepts. In the subsequent sections, we will focus exclusively on data from the 2023 students, as our aim is to gain insights into what is being taught in secondary school under the LK20 curriculum. While non of the actual tasks are provided, examples are shown to closely illustrate the nature of each task.

Lists The poorest-performing concept was Lists, which included three tasks. The first two primarily focused on list indexing. The first task resembled the following example:

```
# What does this program print?

ls = [5, 9, -11, 14, 0, 3]
print(ls[3])
```

Upon examining the incorrect answers provided, it becomes evident that a significant portion of students seemed to have a misconception that lists in Python are 1-indexed. If we consider both 0-indexing and 1-indexing as correct answers, the percentage of correct responses increases substantially, moving from 34.7% to 86.6%. This trend of responses related to 1-indexing is also noticeable in the second and third tasks within the same category.

Datatypes In the datatype task, four variables were defined, each with different data types: string (str), integer (int), list, and float. The students were tasked with correctly identifying the data type for these variables and a set of expressions. The answer options included int, float, bool, list, str, and *Error* if the expression would lead to an error. Figure 20 shows the percentages of correct answers to giving the datatype of the four variables.

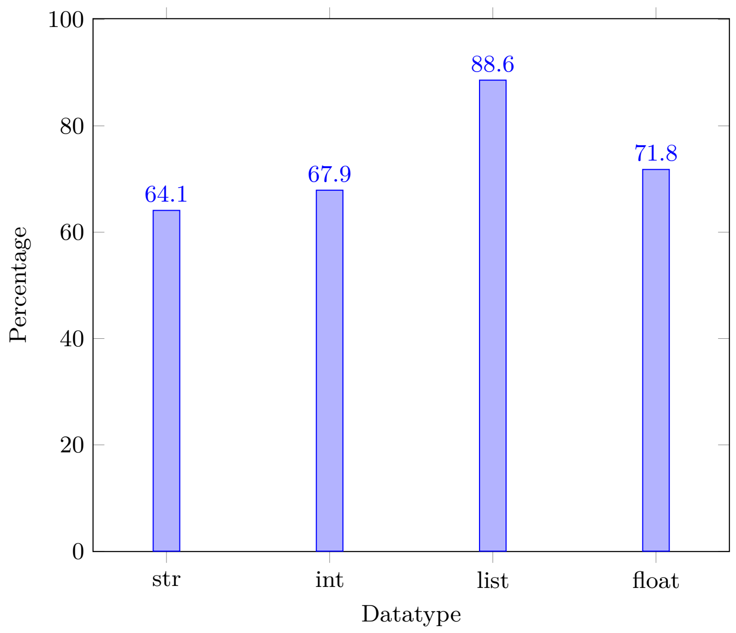


Fig. 20: Percentage of correct answers to giving the datatype of the four variables in datatype task. Data from 2023 students (n=483).

The subsequent section of this task involved a series of expressions that employed these four variables along with the following operators:

$+$ concatenation
$==$ comparison
$*$ repetition

The most common incorrect response for these expressions was *Error*. Specifically, when the expressions used the repetition or concatenation operators with strings and lists, the majority of students erroneously selected *Error* as their answer. It appears that many students were not fully familiar with the polymorphic nature of these operators, interpreting $*$ and $+$ solely in the context of multiplication and addition, respectively.

Loops In the loops category, both tasks started with a variable $n = 0$, and its value was modified within a (while/for) loop. The students were asked to determine the final value of n at the end of the program. A relatively common incorrect response (7.6% in task 1 and 2.9% in task 2) was that n equaled 0. In these instances, it appeared that the students had difficulty perceiving how the code modified the variable, and consequently, they provided its initial value as their answer.

In task 2, the loop increments the variable by two with each iteration, similar to the following code snippet:

```
n = 0
for i in range(3):
    n = n + 2
print(n)
```

The most prevalent incorrect responses to this task were 2 (15.7%) and 4 (10.7%). It appears that in these cases, students have recognized that the answer must be a multiple of two due to the line within the loop, but they were uncertain about the number of iterations it will go through.

Functions The functions category displayed a substantial diversity in task complexity, featuring a total of seven tasks that progressively increased in difficulty.

The first task, the simplest among them, required evaluating a function that added two input numbers. This function was named *sum*, and its intuitive nature made it accessible even to individuals without prior programming experience. As a result, 81.5% of the participants answered correctly. Excluding blank responses, this percentage increases to 90.5%.

Interestingly, the three final tasks, widely acknowledged as the most challenging within the entire test, presented a unique outcome. Despite their complexity, the majority of students responded correctly. These tasks involved the application of two specific functions, similar to the following examples:

```
def f(x):
    x = x + 10
    return x

def g(x):
    x = f(x) - 2
    return x
```

In the first task, students were asked to evaluate the function f with a given input. In the second task, they evaluated the function g, and in the third task, they encountered a nested function call: $f(g(input))$. These functions were relatively uncomplicated, primarily centered around addition and subtraction operations. While Task 1 was relatively straightforward, Tasks 2 and 3 introduced the intricacy of functions interacting with one another. The correct response rates for these three tasks were 58.3%, 59.4%, and 48.0%, respectively.

This pattern implies that a large part of the students excel in tracing code when the operations involved are familiar to them.

Booleans The Boolean tasks presented a number of boolean expressions to the students and gave the answer options of $True$, $False$ and *Error*. In Task 1 and Task 2, the correctness of answers exhibited variation dependent on the specific operators used. When the expressions involved operators more closely aligned with mathematical conventions, a substantial portion of students performed well, achieving correct responses at rates ranging from 88% to 95%. Examples of these types of expressions include:

```
100 == 100
(10*2) > 9
99 >= 100
(14/2) > 7
```

However, when introducing operators found outside the realm of math, correctness rates dropped to a range from 70% to 81%. Examples of these include:

```
100 != (50*2)
(5 < 7) and (4 > 5)
```

The most challenging expressions, with correctness rates between 47% to 59%, included:

```
42 != 25
not (100 == 100)
and 500 == 100
(5 < 7) or (4 > 5)
```

This variance in performance underscores the influence of specific operators on students' ability to provide correct responses in these tasks.

In Task 3, the expressions became more abstract, involving only the truth values *False* and *True* with a diverse range of operators. Examples of these abstract expressions include:

```
True and False
(False != False) == (True != True))
(not True) or False
```

The particularly low correct response rates in Task 3, ranging from 27% to 52%, can be attributed to the use of abstract truth values and a wide array of operators. These abstract expressions appeared to present more significant challenges for students when it came to accurately evaluating them. It is likely that students were less familiar with these types of expressions, as they are not commonly encountered outside of tasks designed to specifically assess one's ability to evaluate boolean expressions. These concepts might not have been extensively utilized in the context of solving mathematical problems in their secondary school education, contributing to the lower performance.

Conditionals The three tasks related to the concept of conditionals required students to determine the correct output from code snippets that included if-statements. These code snippets incorporated relatively straightforward boolean expressions, similar to those found in Task 1 within the Booleans category. The students had previously demonstrated proficiency in evaluating such expressions, and they did so once more when these expressions were incorporated into if-statements, achieving a correct response rate of 85.0%.

Variables The variables tasks were straightforward, requiring students to determine the correct output of code snippets involving various variables with integer values. These variables were combined and manipulated using basic mathematical operators. The second task, which was slightly more challenging, resembled this example:

```
a = 1
b = 9
c = a * b
d = 10
e = c + d
print(e)
```

The majority of students performed exceptionally well in these tasks, achieving a high correct response rate of 91.6%.

5 Discussion & Conclusion

References

1. Kunnskapsløftet 2020. https://www.regjeringen.no/no/dokumentarkiv/regjeringen-solberg/aktuelt-regjeringen-solberg/kd/pressemeldinger/2018/fornyer-innholdet-i-skolen/id2606028/?expand=factbox2606064, hentet: 2023-10-12
2. Matematikkrådets forkunnskapstest. https://matematikkraadet.wixsite.com/matematikkraadet/forkunnskapstesten, hentet: 2022-12-12

3. Frantsen, T.: Å vere lærar i programmering utan å kunne programmere. Master's thesis, OsloMet-storbyuniversitetet (2019)
4. Hertz, M.: What do "cs1" and "cs2" mean? investigating differences in the early courses. In: Proceedings of the 41st ACM Technical Symposium on Computer Science Education. p. 199203. SIGCSE '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1734263.1734335, `https://doi.org/10.1145/1734263.1734335`
5. Stenlund, E.: Programmering og Fagfornyelsen. Master's thesis (2021)