Chapter # 5

LEARNING ANALYTICS TO SUPPORT LOW-ACHIEVING STUDENTS IN PROGRAMMING EDUCATION

Takuya Hasegawa[1] & Yosuke Ito[2]
[1]Hyogo University of Teacher Education, Japan
[2]Naruto University of Education, Japan

ABSTRACT
In recent years, providing effective programming education has become an important issue in Japan. This study aimed to support students predicted to have low proficiency by analysing their coding processes. Programming lessons were conducted at a senior high school, and learning analytics were performed using both the frequency of tokens contained in the students' programs and their proficiency levels measured through assessments. Using machine learning with token frequencies as features, a classification model was constructed to identify the lower-achieving group effectively. Furthermore, by examining tokens with low coverage rates in this group and visualising their positions in the exemplar program, patterns were identified that may help pinpoint areas of difficulty. These findings suggest that the proposed analysis method can help support students at risk of low proficiency.

Keywords: programming education, learning analytics, coding process, tokens, machine learning.

1. INTRODUCTION

1.1. Background

Japanese school education is guided by national curriculum guidelines, known as the Course of Study, which are revised approximately every ten years. In the latest revision issued in 2018, programming education was strengthened, and Informatics I, which includes programming as one of its main contents, became a compulsory subject in senior high schools (Ministry of Education, Culture, Sports, Science, and Technology [MEXT], 2018). Consequently, all high school students are required to learn programming.

In Japan, many students take the Common Test for University Admissions. Informatics I was incorporated into this test, with the first administration held in January 2025. The total number of test-takers was 462,066, of which 279,718 took Informatics I (National Center for University Entrance Examinations, 2025). According to statistics from MEXT (2023), the number of third-year senior high school students as of May 2024 was approximately 930,000. Thus, approximately 30% of the cohort had taken the subject, highlighting the growing importance of programming education in senior high schools.

Hasegawa, one of the authors, has been engaged in teaching elective courses, including programming, at the senior high school level for over a decade. Drawing on this experience, it has been consistently observed that, relative to other domains within informatics, students encounter greater difficulty in attaining proficiency in programming. Furthermore, recent revisions to the national Courses of Study, coupled with modifications to university entrance examinations, have not only increased the complexity of programming content but also expanded the cohort of students required to acquire programming skills. Consequently, the

provision of effective programming instruction has emerged as a critical and pressing issue in the context of high school informatics education.

Learning analytics research focusing on programming education has been actively conducted overseas (Ihantola et al., 2015; Humble & Mozelius, 2019). In Japan, there is growing interest in the implementation of learning analytics and the application of its insights in programming education.

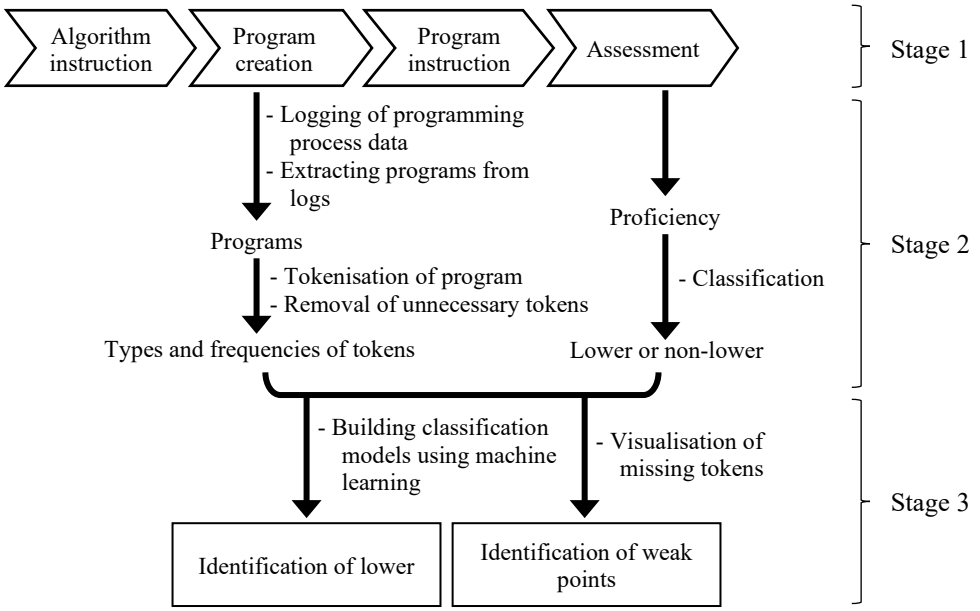
1.2. Research Purpose

Learning support is particularly important for students with lower proficiency levels. However, previous research has suggested that students with lower proficiency often struggle to explain what they do not understand and may be unable to ask teachers questions effectively (Hasegawa & Ito, 2024a). Therefore, this study has two primary aims: to identify students at risk of low proficiency, and to pinpoint the parts of code that they do not fully understand.

1.3. Overview

This study consists of three major stages (Figure 1). The first stage involves a series of teaching practices, including program creation and assessments. The second stage involves data collection and pre-processing during the teaching practice sessions. The third stage involves learning analytics to identify low achievers and the parts of code where their understanding is weak.

Figure 1.
Outline of the research process.



2. TEACHING PRACTICE

2.1. Lessons

In September 2024, Informatics I programming lessons were conducted for 135 second-year students across four classes at a private senior high school. The lessons were conducted by one of the authors, Hasegawa.

The curriculum guidelines for Informatics I specify the following learning objective for programming: "to understand and acquire skills concerning means of expressing algorithms, and methods of utilizing computers and information communication networks through programming", and "to consider algorithms according to objectives, to express them by appropriate methods, to utilize computers and information communication networks through programming, and to evaluate and improve the process" (MEXT, 2018, p. 191). Furthermore, as concrete examples, the guidelines specify: "to take up activities such as considering and expressing typical algorithms such as searching and sorting, and to deal with methods of expressing algorithms, the importance of correctly expressing algorithms, and differences in efficiency depending on algorithms" (MEXT, 2019, p. 33).

Based on the above, we conducted instructional practice on the unit "Search Programs". The educational objectives of the unit were to understand algorithms in searching and to develop thinking and expressive skills related to program creation and improvement. Lessons were conducted in accordance with the instructional plans listed in Table 1. JavaScript was used as the programming language. The focus of the analysis in this study was the learning of a binary search program. During the lessons, the teacher explained the algorithm of the target program. Students then created their programs, and the teacher explained the exemplar program. After completing the unit, students' proficiency was assessed.

Table 1.
Instructional plan of the unit "Search Programs".

Class*	Instructional content
1	Review of prior learning contents in programming, such as branching, iteration, and arrays.
	Linear search program: - Explanation of algorithm
2	- Program creation (15 minutes) - Explanation of exemplar program - Reflection
3	- Confirmation of key points
	Binary search program: - Explanation of algorithm
4	- Program creation (20 minutes) - Explanation of exemplar program - Reflection
5	- Confirmation of key points
	Summary

* The class duration is 45 minutes.

2.2. Assessment

The assessment was conducted using a written test. The assessment consisted of three components.

(a) Algorithm understanding: one item that did not require programming language expression.
(b) Program completion: four basic items assessing understanding of the exemplar program.
(c) Program improvement: one applied item assessing the exemplar program, plus the thinking and expressive skills developed in earlier lessons.

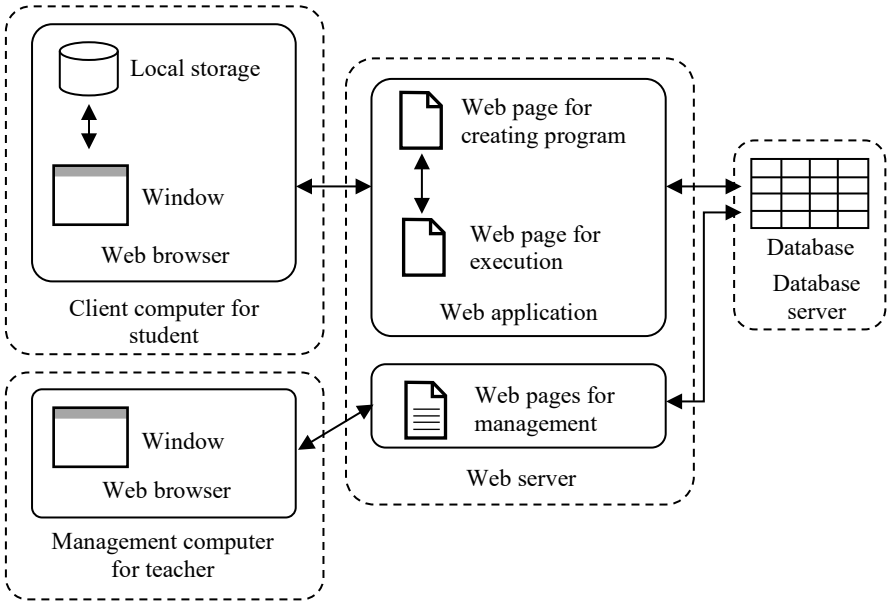
Each item was worth one point. The maximum score is six. The students' scores represent their proficiency in this unit. The purpose of the written test was to serve as a summative assessment of students' proficiency at the end of the unit.

3. DATA COLLECTION AND PRE-PROCESSING

3.1. Programming Data Collection

Students utilized a programming education support system to create their programs. This system was originally developed by Hasegawa and Ito (2024b). The system has functions for program creation and execution, as well as functions for sequentially recording the coding process for individual students. The block diagram of the system is shown in Figure 2. The system is executed using a web browser. When a student enters a key or saves or executes a program, such events are detected by the system. Immediately after detection, the type of event, timestamp, and program at the time of the event are recorded in the database as learning data.

Figure 2.
Block diagram of the programming education support system.



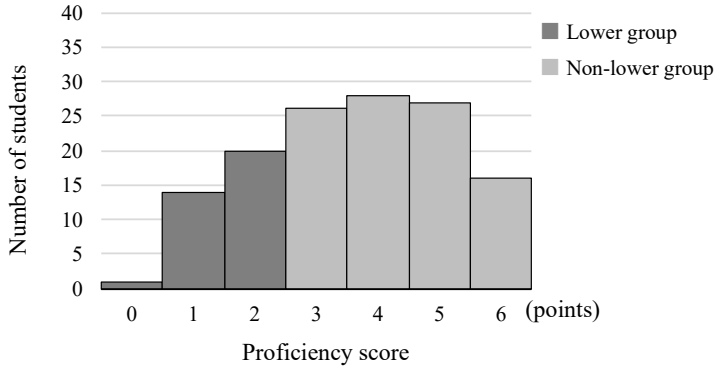
Among the 135 students enrolled in the course, 126 attended the lesson for creating a binary search program. During 20 minutes of program creation, 58,817 learning data records were obtained.

3.2. Distribution of Proficiency and Classification

Previous teaching practices concerning the binary search program showed that it was difficult to find a clear proficiency boundary between the lower and non-lower groups (Hasegawa & Ito, 2025). Therefore, the lowest third of the proficiency distribution was treated as the lower group. According to calculations based on the MEXT statistical data (2024), the average class size in Japanese senior high schools is approximately 35. One-third of this corresponds to slightly over ten students, which can be considered a manageable number for teachers to provide individual support.

A total of 132 students took the assessment. The mean proficiency score of all test-takers was 3.6 points out of 6 (SD = 1.6). The proficiency distribution in this study is shown in Figure 3. The cut-off points for the lowest third of the proficiency data was three points. Since there were more students in the non-lower group than in the lower group with a score of 3, it was considered appropriate to include students scoring 3 points in the non-lower group. As a result, 35 students were classified into the lower group, with scores of 0–2 points, and 97 in the non-lower group, with scores of 3–6 points.

Figure 3.
Distribution of proficiency in the binary search program.



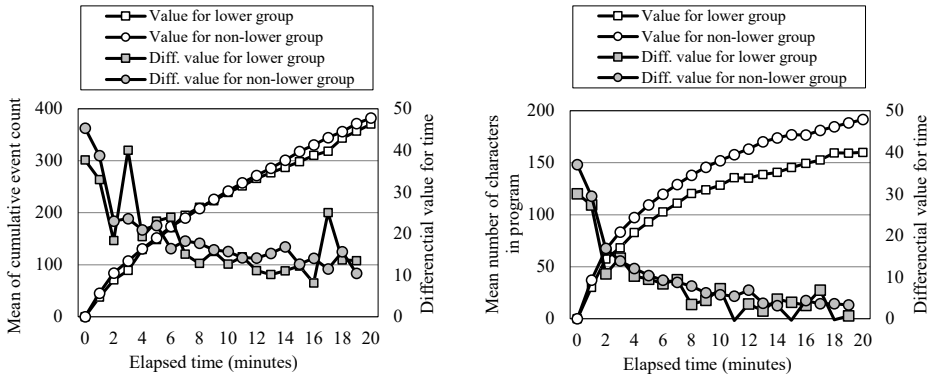
3.3. Preliminary Time-Series Analysis of Programming Activity

A total of 126 students attended both the program-creation lesson and the assessment. Following a data inspection, the learning data from one student were excluded due to an abnormal amount of text data irrelevant to the programming task. Based on the criteria described in Section 3.2, 30 of the 125 students were classified into the lower group and the remaining 95 into the non-lower group.

For each student, we aggregated the cumulative number of operational events up to each measurement point and the number of characters in the program at each measurement point and then calculated the means of the lower and non-lower groups. The time-series changes in the cumulative number of events and characters in the program for each group are shown in Figure 4.

Both groups exhibited a continuous increase in the cumulative number of operational events. At the 20-minute point, the mean number of operations was 381.9 for the non-lower group and 370.4 for the lower group, with no statistically significant difference observed using a *t*-test ($p = 0.79$). However, the increase in the number of characters slowed after approximately two minutes. An increase in deletion operations is considered to be the cause. At the 20-minute point, the mean number of characters was 160.0 in the lower group and 191.6 in the non-lower group, and a significant difference was found ($p < 0.01$).

Figure 4.
Time-series mean changes in programming data for the lower and non-lower groups.



(a) Cumulative event count (b) Number of characters in program

3.4. Tokenisation of Programs

Programs that students had created at the 20-minute mark were extracted for learning analytics. First, the programs were divided into tokens. Tokens are the smallest units of a program, and the process of dividing them into tokens is called lexical analysis or tokenisation (Python Software Foundation, 2025). Next, tokens belonging to whitespace, string literals, numeric literals, and tokens that did not appear in the exemplar program were removed, and the number of occurrences of the remaining tokens was aggregated.

4. LEARNING ANALYTICS

First, we attempted to identify students who were likely to belong to the lower-proficiency group based on the number of occurrences of each token during program creation. Second, we attempted to determine the parts of the program in which those students had weak understanding.

4.1. Identification of the Lower Group

In machine learning, predictive models have different names depending on the type of objective variable. Problems in which the objective variable is quantitative are called regression models, and those in which the objective variable is qualitative are called classification models (Hastie, Tibshirani, & Friedman, 2009). We constructed a classification model using machine learning, in which the number of token occurrences was used as a feature, and the categories of the lower and non-lower groups were used as the objective

variables. Representative methods such as logistic regression, decision trees, support vector machines, random forests, and gradient boosting were employed (Géron, 2020).

In this analysis, the dataset available for model construction consisted of only slightly more than 100 records, which was considered too small for machine learning. In general, for the performance evaluation of the models, the dataset is divided into training and validation sets. However, when the dataset is small, the evaluation by one division may be affected by chance, and the results may become unstable. Therefore, in this study, a five-fold cross-validation was adopted for performance evaluation (Géron, 2020).

The precision and recall were used as indicators to evaluate the identification performance of the classification models. Precision indicates the proportion of students who actually belong to the lower group among those predicted to belong to the lower group. Recall indicates the proportion of students predicted to be in the lower group among those who actually belonged to the lower group. There is a trade-off between precision and recall; when one is increased, the other tends to decrease. Since the purpose of this study is to detect the lower group without omissions, it is important to improve recall. Therefore, the hyperparameters in machine learning were adjusted with an emphasis on recall.

The performances of the classification models using the machine learning method are summarised in Table 2. The recall ranged from 0.74 to 0.77, and the precision ranged from 0.70 to 0.78. The method that showed the highest identification performance was gradient boosting, with a recall of 0.77 and a precision of 0.78.

Table 2.
Classification performance by machine learning models.

Model	Recall	Precision
Logistic Regression	0.76	0.71
Decision Tree	0.74	0.70
Support Vector Machine	0.76	0.70
Random Forest	0.74	0.73
Gradient Boosting	0.77	0.78

To evaluate the usefulness of the number of token occurrences as a feature, we constructed classification models using other features and compared their performances. The indicators used as other features were the cumulative number of operational events, the number of characters in the program, and the score on a written test of "Information Society" unit conducted just before the unit "Search Programs". Gradient boosting was employed as a machine learning method. The classification performance according to feature type is presented in Table 3. The recall of the classification models using other features remained in the range of 0.56–0.61, and precision remained in the range of 0.57–0.62. These results demonstrate that the number of token occurrences is a useful feature for identifying proficiency.

In general, data in which the number of samples between classes is imbalanced is called imbalanced data, and such data makes it difficult to train appropriate models. In this study, as the ratios of the lower and non-lower groups were imbalanced, it was difficult to construct a model with high identification performance. Nevertheless, by applying machine learning with gradient boosting and using the number of tokens as features, we constructed a model that identified the lower group with a certain level of recall and precision.

Table 3.
Classification performance by feature types.

Feature type	Recall	Precision
Token occurrences in created programs	0.77	0.78
Number of operational events during programming	0.57	0.57
Number of characters in the program	0.61	0.62
Score on the written test in the “Information Society”	0.56	0.59

4.2. Identification of Weaknesses in the Lower Group

In the lessons, instructions were provided with the exemplar program in mind before the students created their own programs. Therefore, it was assumed that students attempted to create programs similar to the exemplar program. We defined the coverage rate as the ratio of the number of occurrences of a given token in a student program to the number of occurrences of the same token in an exemplar program. Attention was focused on tokens with a lower coverage rate in the lower group than in the non-lower group. By visualising on the exemplar program the tokens whose coverage rate was low in the lower group programs, we attempted to identify the parts that students in the lower group did not understand.

First, the mean coverage rate of each token was calculated for the lower and non-lower groups, and the tokens with significant differences between the two groups were identified. Significance was determined using a *t*-test at a 0.01 level. Next, the entire exemplar program was displayed, and tokens with significantly lower coverage rates were highlighted. Among the tokens with low coverage rates, `while`, `/`, `Math`, `floor`, `<=`, and `>` appeared only once in the exemplar program, and therefore their positions could be identified. These tokens were visualised by reversing the background colour to black and font colour to white. Tokens that appeared more than once in the exemplar program were visualised with a grey background (Figure 5).

As a result, it was found that in the lower group, many students were unable to write the part using `while` and its condition in line 5, and the part in line 6, where the result of division by `/` is processed by `Math.floor()`.

Regarding the description in line 5, the binary search program contains a loop structure and is written using `while`. Before learning the binary search program, students had learned loop structures using `for` and `while`. Moreover, memorising how to write syntax was not a learning objective, so when creating the binary search program, explanations of how to write the `while` syntax and how it functions were presented as hints. However, in reality, many students in the lower group were unable to write a program using `while`. From this, it can be considered that more intensive instruction is required to promote understanding of loop structures using `while`.

Regarding the description in line 6, although explanations of the function and how to write `Math.floor()` were presented as hints during program creation, students had no prior experience of creating programs using `Math.floor()`. Furthermore, although the content of line 6 is short and expressed as a single line, it is a complex structure that encompasses multiple processes. Specifically, the following series of processes are summarised in one line: (1) adding the variable `L` representing the lower-bound index of the search range and the variable `R` representing the upper-bound index; (2) dividing the result of addition by 2; (3) extracting only the integer part of the result of division; and (4) assigning the integer part

to variable `i`. Expressing such a structure encompassing multiple processes in a programming language was considered difficult for students in the lower group.

As described above, by visualising the exemplar program and tokens with low coverage rates in the programs of the lower group, we obtained grounds for identifying the parts that students in the lower group did not sufficiently understand.

Figure 5.
Visualisation of tokens with low coverage rates in the lower group.

```
 1  let a = [14,17,20,22,24,26,28,30,31,33];
 2  let s = prompt();
 3  let L = 0;
 4  let R = a.length-1;
 5  while(L <= R){
 6    let i = Math.floor((L + R) / 2);
 7    if(a[i] == s){
 8      alert(s + ' is found at index ' + i + '.');
 9      break;
10    }
11    if(a[i] > s){
12      R = i - 1;
13    }
14    if(a[i] < s){
15      L = i + 1;
16    }
17  }
```

5. CONCLUSION

This study focused on exploring an approach to support students predicted to have lower proficiency. Programming lessons were conducted at a senior high school and learning analytics were performed based on both the frequency of tokens contained in students' programs and proficiency measured through assessments.

Using machine learning with token frequencies as features, a classification model was constructed to identify the lower-achieving group with reasonable accuracy. Additionally, by examining tokens with low coverage rates in this group and visualising their positions in the exemplar program, insights that may help identify areas of weaker understanding were obtained. These results provide a foundation for designing timely and data-informed support for students whose proficiency is at risk of remaining low, indicating substantial potential for practical implementation in programming education.

Although the present analysis focused on a binary search program, further investigation is needed to determine whether this method can be applied to other programming tasks. Building on these findings, future work will focus on developing concrete approaches to integrate this analytical method into instructional support.

REFERENCES

Géron, A. (2020). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.; T. Nagao, Trans.; N. Shimoda, Supervising Ed.). O'Reilly Japan. (Original work published 2019).

Hasegawa, T., & Ito, Y. (2024a, August). Quantitative text analysis of students' perceptions of programming education in Informatics I. *Proceedings of the 67th Annual Meeting of the Japan Society of Technology Education* (p. 84). Naruto, Tokushima, Japan: Japan Society of Technology Education.

Hasegawa, T., & Ito, Y. (2024b). Design and evaluation of a programming learning system with ease of operation and sequential recording function. *Journal of Computational Education, Naruto University of Education, 21*, 27-31. https://doi.org/10.24727/0002000439

Hasegawa, T., & Ito, Y. (2025). Relationship between features derived from learning process data and proficiency in programming education. *Journal of the Japan Society of Technology Education*, *67*(1), 23-31. https://doi.org/10.32309/jjste.67.1_23

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Retrieved August 19, 2025, from https://hastie.su.domains/ElemStatLearn/printings/ESLII_print12_toc.pdf.download.html

Humble, N., & Mozelius, P. (2019). Learning analytics for programming education: Obstacles and opportunities. *12th International Conference of Education, Research and Innovation (ICERI2019)* (pp. 6159-6166). IATED. https://doi.org/10.21125/iceri.2019.1483

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M. Á., Sheard, J., Skupas, B., Spacco, J., Szabo, C., & Toll, D. (2015). Educational data mining and learning analytics in programming: Literature review and case studies. In N. Ragonis & P. Kinnunen (Eds.), *Proceedings of the 2015 ITiCSE on Working Group Reports (ITiCSE-WGR '15*) (pp. 41-63). New York, NY: Association for Computing Machinery. https://doi.org/10.1145/2858796.2858798

Ministry of Education, Culture, Sports, Science and Technology. (2018). *Kotogakko gakushu shidoyoryo (Heisei 30-nen kokuji)* [Course of Study for Upper Secondary Schools (2018 Notification)]. Retrieved February 12, 2022, from https://www.mext.go.jp/content/20230120-mxt_kyoiku02-100002604_03.pdf

Ministry of Education, Culture, Sports, Science and Technology. (2019). *Kotogakko gakushu shidoyoryo (Heisei 30-nen kokuji) kaisetsu joho-hen* [Commentary on the Course of Study for Upper Secondary Schools (2018 Notification)]. Retrieved February 12, 2022, from https://www.mext.go.jp/content/1407073_11_1_2.pdf

Ministry of Education, Culture, Sports, Science and Technology. (2023). *Gakko kihon chosa* [School Basic Survey]. Retrieved February 15, 2024, from https://www.mext.go.jp/b_menu/toukei/chousa01/kihon/1267995.htm

Ministry of Education, Culture, Sports, Science and Technology. (2024). *Monbu kagaku tokei yoran (Reiwa 6-nen ban) 7. Kotogakko* [Education statistics yearbook (2024 edition) 7. High schools]. Retrieved August 16, 2025, from https://www.mext.go.jp/content/20240531-mxt_chousa01-000036236-7.xls

National Center for University Entrance Examinations. (2025). *Reiwa 7-nendo daigaku nyugaku kyotsu tesuto jisshi kekka no gaiyo* [Summary of the results of the Common Test for University Admissions, Academic Year 2025]. Retrieved August 16, 2025, from https://www.dnc.ac.jp/kyotsu/hyouka/r7_hyouka/r7_data.html

Python Software Foundation. (2025). *Python 3.10.17 documentation: 2. Lexical analysis*. Retrieved May 18, 2025, from https://docs.python.org/release/3.10.17/reference/lexical_analysis.html

ACKNOWLEDGEMENTS

This research was partially supported by the Japanese Grant-in-Aid for the Encouragement of Scientists 24H02432.

AUTHORS' INFORMATION

Full name: Takuya Hasegawa
Institutional affiliation: Hyogo University of Teacher Education
Institutional address: 942-1 Shimokume, Kato-shi, Hyogo 673-1494, Japan
E-mail address: webessan@gmail.com
Short biographical sketch: He is a full-time high school teacher with a Ph.D. (Education). He serves as a technology and informatics teacher at Kyoto Tachibana Junior and Senior High School and supervises the Robot Programming Club. He is the Chair of the Informatics Research Association of Private Junior and Senior High Schools in Kyoto Prefecture. His research interests include learning analytics in programming education.

Full name: Yosuke Ito
Institutional affiliation: Naruto University of Education
Institutional address: 748 Nakashima, Takashima, Naruto-cho, Naruto-shi, Tokushima, 772-8502, Japan
E-mail address: ito@naruto-u.ac.jp
Short biographical sketch: He is a Professor in Technology and Information Education at the Graduate School of Naruto University. He is engaged in training teachers in elementary school, junior high school technology, and senior high school informatics. His research interests focus on information science education, integrating both software and hardware perspectives, with particular emphasis on teaching methods and instructional material development related to artificial intelligence and programming.