Effect of Auto-complete Function on Processing Web IDE for Novice Programmers

Motoki Miura

Department of Basic Sciences, Faculty of Engineering, Kyushu Institute of Technology
1-1 Sensui, Tobata, Kitakyushu, Fukuoka, 804-8550 JAPAN
Email: miuramo@mns.kyutech.ac.jp

***Abstract*—Web IDE is suitable for novice learners because it can reduce the troublesome of constructing programming environment. Since most of the Web IDEs utilize text-based programming, a certain level of typing skills is required for learners. For that reason, the performance of learning programming may be affected by the typing skill of the learner. In order to mitigate the influence of the typing skill, we introduced automatic code completion to the Web IDE for Processing language. We investigated how the automatic completion function was used for novices through an actual course. Since the snippet of "if" and "for" statements already contained parenthesis and curly brackets, the auto-complete function significantly reduce the typing of these special characters. From a questionnaire survey, a strong positive correlation was found between the use frequency of the function and the necessity that learner feels.**

***Index Terms*—programming editor interface, typing skills, Processing, beginners, creativity support**

I. Introduction

Web IDE (Integrated Development Environment), which can be used only with the browser, is suitable for novice learners to learn programming because it can reduce the troublesome of constructing programming environment. Therefore, many programming environments using Web IDE have been developed [1], [2]. We also focused on Processing[1], which can utilize interaction, graphics and animation with a concise grammar similar to C language. We have been conducted lectures of creative programming and thinking for undergraduate students using Processing.js[2] with a Web IDE.

Most of the Web IDEs and conventional programming lectures utilize text-based programming. Therefore, a certain level of typing skills is required for learning programming with the Web IDE. For that reason, the level of typing skill of the learner may affect the learning outcome of programming. Table I is the Pearson's correlation coefficient of typing skill and final performance in our past 4 years course. The level of typing skill (from 1 to 7) was self-declared by the student at the beginning of the lecture. The students utilized yatt[3], which is a locally developed competitive typing trainer in our institute, for measuring typing skills. The distribution of the level (from 1 to 7) is shown in Table IV at the appendix A. From the Table I, we found significance in some classes of 2016, 2015, and 2014(1). For those classes, the typing skill correlates well with the final grades (scores) of the programming lecture. However, we consider that the situations in which the level and superiority of typing skill influences the outcome of programming learning is not desirable.

[1]https://processing.org/

[2]http://processingjs.org/

TABLE I
Correlations between typing skills and performance. (* and ** represents significance of 5% and 1%, respectively)

Year(Class)	Num of Student	r	t-value	p-value
2017	45	0.024	t(43) = 0.158	0.875
2016	69	**0.389****	t(67) = 3.457	0.001
2015	72	**0.395****	t(70) = 3.599	0.001
2014 (1)	75	**0.252***	t(73) = 2.223	0.029
2014 (2)	49	0.136	t(47) = 0.943	0.351
2014 (3)	52	0.214	t(50) = 1.552	0.127

One of the methods to mitigate the influence of typing skills is introducing block-based interface such as Scratch [4] or Google Blockly [5], [6]. By using the block-based interface, the learner can avoid mistakes and errors in the text-based method. In addition, learners can recognize available function blocks in a short time by viewing the block list. However, although the block-based interface can reduce errors, the degree of freedom of programming description is lower than that of text-based method. Also, introducing block-based interface takes considerable time to master the operation especially for solving problems.

Therefore, in order to reduce the influence of input error and typing error in the programming environment of text-based programming method, we applied input support by automatic code completion function to Web IDE. Also, through actual lectures, we investigated how the automatic completion function was used for novices.

II. Automatic completion function and its implementation

The automatic completion (auto-complete) is a function of displaying a list of completion candidates while inputting. Users can reduce the amount of typing and avoid mistypes by selecting candidates. It is a standard feature in integrated development environments such as Eclipse and Visual Studio as well as classic editors such as vi and Emacs. Users can input the first few letters of function names and variable names, select from the candidates, or automatically insert

corresponding parentheses. Therefore, in addition to mitigating typing errors and improving editing efficiency, the automatic completion has the effect of lowering the barriers of relatively long names for function names and variable names.

As the implementation of our Web IDE, we utilized the Tab key for calling the auto-complete function. It is the same as the complement operation of Bash, which is the default log-in shell on many Linux. There is also a method of automatically displaying candidates while inputting normal characters. However, there was concern that candidates would be displayed even if the learner did not intend, or some part of the code would be hidden for candidate display unexpectedly. For this reason, we adopted a method that requires explicit operation.

As an example of a specific operation, we show the web editor screen when [Tab] key is pressed after [m] key in Figure 1. In this state, the learner can select the candidates by the cursor keys (Figure 2 (left)) or narrow down the candidates by typing continuous characters (Figure 2 (middle)). When the learner presses the tab key or the enter key while the candidate is selected, the candidate is confirmed. As a result, the auto-completion input is completed as shown in Figure 2 (right). After completing the complementary input, the cursor moves immediately after the input. If the learner does not press the [Tab] key, the complementary function will not be activated and behave like the conventional editor. Also, if the learner enters the [Tab] key at the beginning of a line or after a space, all candidates are displayed.

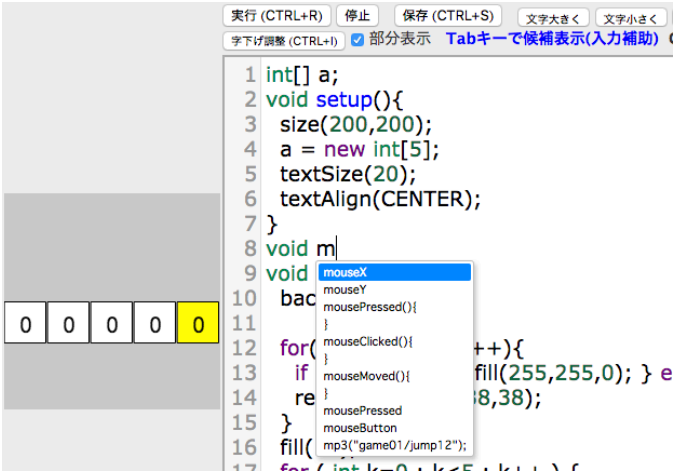

Fig. 1. Example of auto-complete: when [Tab] key is pressed after [m] key

To implement the auto-complete function, we adopted the show-hint add-on (helper) of CodeMirror editor[3]. We prepared function names, keywords, and snippets that are used frequently in Processing programming. The snippets included "if statements" and "for statements." Note that in this implementation, we did not provide dynamic completion that incorporate variables and functions names defined in the editor. Also, we did not consider the context of the source code where the

[3]http://codemirror.net/

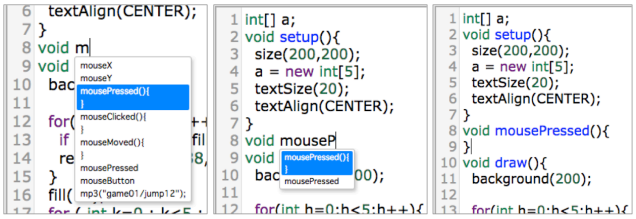

Fig. 2. (left) Select by cursor key (middle) narrow down (right)fix by Enter key

candidate is inserted. Except for these limitations, there is no noticeable difference between the web editor and the other installed IDEs. Appendix A shows a list of Processing keywords and snippets incorporated in this implementation. A web editor with auto-complete function enabled is available at the following URL. https://ist.mns.kyutech.ac.jp/miura/pjs/2018/

A. Merits

We discuss the advantages of the auto-complete function from the viewpoint of similarities and differences with block interfaces.

- It is possible to provide functions similar to "list of functions" in the block interface.
- It is possible to show usage examples of both functions and statements at the same time. It can be a substitute for reference.
- In a text-based programming environment, it can be completed only with keyboard input, so it has high affinity.
- The auto-complete can prevent keyword typing errors.
- The learners can also complement characters that are not accustomed to input, such as parentheses, curly brackets, and semicolons. Therefore, the auto-complete function lowers barriers to typing.

In the block interface, incompatible description and connection can be prevented. With the auto-complete function, however, it is difficult to feed back conformance without errors at the text-typing phase.

III. Practice and Analysis

In order to investigate how the actual learner uses the auto-complete function, and verify the amount and effect, we have implemented a Web exercise system, and used it in an actual lecture. The schedule of the lecture is shown in Table II. All learners were undergraduate students. The number of learner was 69. We analyzed only the data of 48 students who agreed to provide operation logs. Incidentally, from the fifth lecture, all learners could utilize auto-complete functions freely.

Regarding the operation log, the editor sent to the server when 100 editing operation events are accumulated or the learner performs saving operation. The editing operation events include caret moves by cursor key and mouse clicking in addition to normal key typing. We also collected the operation time. Thus the learners' process of source code editing can be completely reproduced from the operation log.

TABLE II
SCHEDULE AND OUTLINE OF INTRODUCING AUTO-COMPLETE FUNCTION

Date (week)	Setting	Contents
April 12 (1)		Introduction(drawing)
April 19 (2)		Function (mouseXXX)
April 26 (3)	Start logging	if statement
May 10 (4)		for statement
May 17 (5)	Start auto-complete	Animation
May 24 (6)		Array and for
May 31 (7)		Mid-term Exam

A. Results and analysis of the operation log

Figure 3 shows how many times the auto-complete function was used in which task (exercise, assignment, exam) for 32 learners who used the complement function at least once out of 48 learners. "Ex#" shows exercise, "As#" shows assignment, "Mx#" shows mid-term exam, and there are arranged from left to right according to the disclosure time of tasks. The learners are sorted from the top by the order of the number of times of use. Note that "Ex6" is a problem in explaining the auto-complete function first in the fifth lecture.

From the figure, we could find there are students who are using it in practice and exercises, but are not in use at the time of examination. Since the auto-complete function is not invoked without pressing the Tab key, it is suggested that some students had forgot the existence of function in high-pressured situation. In this experiment, the students were not accustomed with the auto-complete function because the function was introduced in the fifth lecture. In appendix A Table V, indicate the selected snippets and the frequency. We confirmed that "if" and "for" statements and typical drawing commands were selected frequently.

StuID	Ex1	Ex2	Ex3	As1	Ex4	Ex5	Ex6	Ex7	As2	Ex8	As3	Ex9	As4	Mx1	Mx2	Mx3	Total
1				11			12		36		11			5	14	21	110
2				5			3	3	14	1	5		29	5	9	4	78
3	12						3			1	3		29	3	7	10	68
4							6		4	2	7			11	7	8	45
5							5	1	4	1	7			7	5	10	40
6							9				4			5	8	12	38
7	4			6			2						6	16	1	1	36
8		4	2	6	2	4	1	2	10		1						32
9							3						1	10	8	8	30
10														3	7	10	20
11														5	13		18
12				3			2	2			2			1	3	1	14
13				3			6					3					12
14				1						3					1	5	10
15														1	1	3	5
16							3							1			4
17							3										3
18							3										3
19							1						2				3
20							1	1	1								3
21													2		1		3
22				2													2
23							2										2
24														1		1	2
25							2										2
26							2										2
27							1										1
28													1				1
29										1							1
30																1	1
31									1								1
32							1										1

Fig. 3. Number of AutoComplete function used per practice / exercise / exam

Next, we verified whether there is a difference in frequency of key type or not between with and without the auto-complete functions. We separated operations logs into two categories, including or not including the auto-complete function. Note that the operation log was transmitted from 48 learners, and length of the log was differed by transferring triggers (100 operation events or save operation). After that, we calculated average and standard deviation of the following key type counts for each learner: (1) cursor key (2) parentheses and curly bracket (3) semicolon (4) backspace and delete key (5) escape key (6) enter key. The values and result of Welch's t-test is shown in Table III. As a result, it was confirmed that the number of parentheses and curly brackets was significantly smaller, and the number of Enter key was significantly larger in the operation accompanied by the auto-complete function. It is thought that the input of parentheses and curly brackets can be omitted by complementing the if statement and for statement. Parentheses and curly brackets need to be entered while holding down the SHIFT key, so the threshold is slightly higher for learners with low typing skills. Therefore, being able to alleviate these key inputs is considered to have certain significance. The reason why the number of times of the Enter key is increasing is unknown, but there is a possibility that the operation of confirming the completion candidate or the operation of adding the line to the block portion (inside of the curly bracket) may be affected. Incidentally, there was marginal significance in the number of semicolons. There was no significant difference in the number of cursor key, Backspace, and Escape key. Although it is not confirmed and verified, it seems that novices often move the caret by mouse operation.

B. Questionnaire Survey

A questionnaire survey was conducted immediately after the mid-term test. The purpose of the questionnaire survey was to clarify the student's awareness of the difficulty level of the exam questions and the time setting, the student's awareness of the support function of the editor, whether or not it is used, and the student's score. Questionnaire items summarized by common options are shown in the following bullets. The "auto-indent function" was a function for collectively adjusting indentation by the button at the top center of Figure 1, which was provided from the first lecture.

- Question 1-1 / 2-1 / 3-1: How long it took you to solve the mid-term exam of Q1/2/3?
  (1) About 5 minutes (2) About 10 minutes (3) About 15 minutes (4) About 20 minutes (5) About 25 minutes (6) About 30 minutes (7) 30 minutes or more (8) I could not solve it (I think I could solve it if I had enough time) (9) I could not solve it (I think I could not solve it even if I had enough time)
- Question 1-2 / 2-2 / 3-2: How did you feel about the level of the midterm exam of Q1/2/3?
  (1) It was very easy (2) It was easy (3) I can not say either (4) It was difficult (5) It was very difficult
- Question 4: How many points of your highest score of yatt (typing training software)?
  (1) 1-60 points (2) 61 - 80 points (3) 81 - 100 points (4)

TABLE III
COMPARISON OF KEY OPERATIONS WITH AND WITHOUT AUTO-COMPLETE FUNCTION (BOLD REPRESENTS SIGNIFICANCE, OR 10% MARGINAL SIGNIFICANCE, ** REPRESENTS 1% SIGNIFICANCE)

Type	Without auto-complete Average (StdDev) n=48	With auto-complete Average (StdDev) n=36	t-value	p-value
(1) Cursor Key	5.58 (3.73)	5.51 (4.79)	t(60.74)=0.148	0.883
(2) Parentheses/Curly bracket	**0.88** (0.37)**	0.44 (0.42)	t(63.70)=3.035	0.003
(3) Semicolon	**0.38 (0.13)**	0.18 (0.23)	t(54.65)=1.970	0.053
(4) BS/Delete	6.63 (2.72)	7.16 (4.43)	t(55.44)=-1.203	0.234
(5) Escape	0.01 (0.04)	0.01 (0.04)	t(67.53)=-0.086	0.932
(6) Enter	0.71 (0.26)	**1.34** (0.78)**	t(44.59)=-3.672	0.001

101–120 points (5) 121 - 140 points (6) 141 - 160 points (7) 161 points or more

- Question 5: Did you use "auto-indent function" while editing source code? / Question 7: Have you used the "auto-complete function" while editing the source code? (1) I did not know the existence of the function (2) I knew but I did not use it (3) I occasionally used when noticed (4) I actively used
- Question 6: To what extent do you need "auto-indent function"? / Question 8: To what extent do you need "auto-complete function"? (1) Not at all necessary (2) Not much needed (3) I think either way is okay (4) Hopefully necessary (5) Very necessary

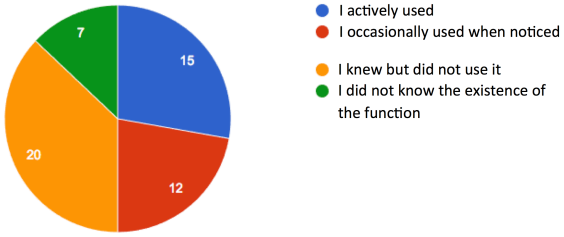

Fig. 4. Q7: Frequency of auto-complete function

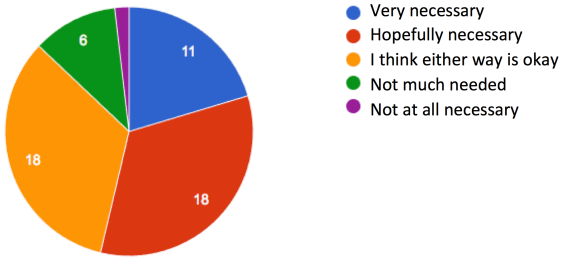

Fig. 5. Q8: Necessity of auto-complete function

There was 54 valid responses. Figure 4 and Figure 5 show the frequency (Q7) and the necessity (Q8) of the auto-complete function, respectively. About half of the learners used the auto-complete function, and they felt the necessity. However, there were many students who answered "either way is okay." We consider that they recognized lack of programming skills at the midterm test, and felt the function was designed for advanced programmers.

The correlation coefficient between the questionnaire result and the score up to the midterm test and the result of performing the uncorrelated test are shown in Figure 6. Light blue ** indicates 1% significant, green * indicates 5% significant, yellow unmarked indicates 10% marginal significance. From the result of Figure 6, we could confirm the high correlations between the problem difficulty felt by the learner, the time necessary to solve, and the score. In addition, we can also confirm the high correlation between usage frequency and necessity for auto-complete and auto-indent functions. There is a negative correlation between the time of Q2 and the necessity of indentation. This can be interpreted that the learner who took time to Q2 did not feel the necessity of auto-indent function. Q2 was a problem that can be solved in a short time by using a for statement. From the result, it can be said that learners who were not able to understand and utilize the "for" statement could not find the usefulness or convenience of the indentation function.

Although it is the result of the midterm exam of this lecture, no significant difference was found between the score and typing skill ($r = 0.21$, $t(52) = 1.56$, $p = 0.12$). This is good situation from our standpoint that we consider the typing skills should have a lower impact on results.

In this exam, more than half students felt that the difficulty level was high. Free comments also reveal there were many learners who felt insufficient understanding and lack of practice. Also, in this practice, about half of the learners felt the necessity of the auto-indent function provided from the beginning of the lecture. Therefore, it is thought that there were a certain number of learners who have not reached the understanding of the importance of writing readable and maintainable code. Blockly Processing [7] may be suitable for the learners who do not yet recognize or motivated to the text descriptions.

IV. RELATED WORKS

In order to reduce mistype caused by unfamiliar grammar, and alleviate the influence of typing skill by keyboard, there are some researches of introducing block-based interface for novice programmers and development of learning environment with input assist function. Matsuzawa et al. developed a programming environment that can mutually convert between block description and text description, and performed a practice in Java language [8]. As a result of the practice, it was confirmed that the transition gradually from the block

	Q1Time	Q1Level	Q2Time	Q2Level	Q3Time	Q3Level	Typing Skill	IndentUsed	IndentNeed	ACmplUsed	ACmpl Need	Score
Q1Time		0.59 **	0.47 **	0.32 *	0.37 **	0.24	−0.04	−0.03	−0.12	−0.03	−0.14	−0.49 **
Q1Level	0.59 **		0.26	0.60 **	0.15	0.59 **	−0.22	0.04	−0.11	0.02	0.06	−0.57 **
Q2Time	0.47 **	0.26		0.64 **	0.48 **	0.13	−0.17	−0.11	−0.38 **	−0.00	0.24	−0.37 **
Q2Level	0.32 *	0.60 **	0.64 **		0.15	0.42 **	−0.26	−0.12	−0.26	−0.12	0.15	−0.62 **
Q3Time	0.37 **	0.15	0.48 **	0.15		0.44 **	−0.03	0.07	0.05	0.02	0.15	−0.23
Q3Level	0.24	0.59 **	0.13	0.42 **	0.44 **		−0.09	−0.03	0.06	−0.08	0.08	−0.35 *
Typing Skill	−0.04	−0.22	−0.17	−0.26	−0.03	−0.09		−0.04	0.03	−0.00	−0.16	0.21
Indent Used	−0.03	0.04	−0.11	−0.12	0.07	−0.03	−0.04		0.50 **	−0.04	−0.08	0.07
Indent Need	−0.12	−0.11	−0.38 **	−0.26	0.05	0.06	0.03	0.50 **		0.04	0.03	0.14
ACmpl Used	−0.03	0.02	−0.00	−0.12	0.02	−0.08	−0.00	−0.04	0.04		0.70 **	−0.02
ACmpl Need	−0.14	0.06	0.24	0.15	0.15	0.08	−0.16	−0.08	0.03	0.70 **		−0.11
Score	−0.49 **	−0.57 **	−0.37 **	−0.62 **	−0.23	−0.35 *	0.21	0.07	0.14	−0.02	−0.11	

Fig. 6. Correlations between questionnaire survey and score (** and * denote 1% and 5% significance, respectively. Yellow background shows 10% marginal significance)

description to the text description along with the progress of programming learning. This research has similarity in carrying out long-term practice and investigation in programming exercises targeting primary scholars. However, in the text description method, there is no reference to the auto-complete function or investigation of its effect.

PEN (The programming environment for novices) [9] provides input support buttons. This buttons allows learners to easily input control structures such as conditional statements, repetitions and functions. It is considered to be an indispensable function for accurately and quickly entering the syntax of extended DNCL[4], which contains Japanese text. Since menu buttons are always displayed on the screen, it is easier to notice than the auto-complete function. In addition, the expression of menu can be intuitive for learners. It is superior to auto-complete function. On the other hand, it is a slight disadvantage that it is difficult to continue key input because mouse operation is required when selecting the menu. Also, the buttons occupies the screen.

Satav et al. compared the functions of integrated development environment for C/C++/Java languages[10]. They separated into the C/C++ environment and the Java environment, and lists whether typical function can be used or not for each IDE. The comparative study is effective for summarizing provided functions of major IDEs. On the other hands, we investigated the acceptance and effect of the auto-complete function for novice learners.

As an improvement of the programming interface not limited to beginners, there are the following studies. Brandt et al. propose a method to incorporate Web search interface into IDE [11]. Oney et al., develop an editor Codelets that can effectively retrieve and insert sample code, and conduct evaluation experiments for Web development using jQuery Mobile[12]. Ko et al. provide a framework that extends conventional code editors to introduce flexible expressions[13]. These advanced, novel programming interface and framework will be fruitful not only for professionals but also for novice programmers.

[4]DNCL is a standard test procedure description language defined by the University Entrance Examination Center in Japan.

V. Conclusion

Web IDE is superior in that learners can easily perform exercises because the environment can be provided only with the browser. We introduced auto-complete function to increase the convenience of Web IDE for Processing language.

We conducted a lecture with the enhanced Web IDE in three weeks. The frequently used completion candidates in the three weeks were if statements, for statements, and drawing functions. The frequency of using auto-complete function will be increased if we could provide the function from the beginning of lecture.

The snippet of the “if” and “for” statements already contained necessary parentheses and curly brackets. Therefore, the number of the parentheses and curly brackets was significantly lower when the learner utilized the auto-complete function. From the questionnaire immediately after the midterm exam, about half of the learners used the auto-complete function. In addition, a strong positive correlation was found between the use frequency of the function and the necessity that learner feels.

From the results, we found that the learners who are not yet conscious or motivated to clearly and carefully write the source code in text could not fully recognize the necessity of the auto-complete and auto-indent functions.

Since the target duration of the experiment was about 3 weeks, and explicit operation by [Tab] key was required, the learners might be not accustomed with the auto-complete function. Also, motivation to use the function was not strong. We would like to examine the effect of long-term experiment more than 3 month.

The concept and function of auto-completion are quite useful. Therefore, it can be used continuously not only for the beginners but also at a highly proficient stage. Therefore, we consider that the use of auto-complete function for novice

learners is important to cultivate a long-term programming skill.

REFERENCES

[1] Max Goldman, Greg Little, and Robert C Miller. Real-time collaborative coding in a web IDE. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 155–164. ACM, 2011.

[2] Vu Nguyen, Hai H Dang, Kha N Do, and Thu D Tran. Learning and Practicing Object-Oriented Programming Using a Collaborative Web-based IDE. In *Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE, 2014.

[3] Hiroshi Kimura. Yatt: Yet Another Typing Trainer. https://literacy.melt.kyutech.ac.jp/yatt.html (Visited: July 5, 2018).

[4] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, Vol. 52, No. 11, pp. 60–67, 2009.

[5] Neil Fraser, et al. Blockly: A visual programming editor. *URL: https://developers.google.com/blockly/*, 2013.

[6] Neil Fraser. Ten Things We've Learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, pp. 49–50, October 2015.

[7] Kochi University of Technology Takata Laboratory. Blockly processing editor. https://ytakata69.github.io/blockly-processing/ (Visited: June 3, 2018).

[8] Yoshiaki Matsuzawa, Takashi Ohata, Manabu Sugiura, and Sanshiro Sakai. Language migration in non-cs introductory programming through mutual language translation environment. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pp. 185–190. ACM, 2015.

[9] Tomohiro Nishida, Akira Harada, Tomoko Yoshida, Ryota Nakamura, Michio Nakanishi, Hirotoshi Toyoda, Kota Abe, Hayato Ishibashi, and Toshio Matsuura. PEN: A Programming Environment for Novices — Overview and Practical Lessons—. In *EdMedia: World Conference on Educational Media and Technology*, pp. 4755–4760. Association for the Advancement of Computing in Education (AACE), 2008.

[10] Sampada K. Satav, S. K. Satpathy, and K. J. Satao. A Comparative Study and Critical Analysis of Various Integrated Development Environments of C, C++, and Java Languages for Optimum Development. *Universal Journal of Applied Computer Science and Technology*, Vol. 1, pp. 9–15, January 2011.

[11] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R Klemmer. Example-Centric Programming: Integrating Web Search into the Development Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 513–522. ACM, 2010.

[12] Stephen Oney and Joel Brandt. Codelets: linking interactive documentation and example code in the editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2697–2706. ACM, 2012.

[13] Andrew J. Ko and Brad A. Myers. Barista: An implementation framework for enabling new tools, interaction techniques and views in code editors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 387–396. ACM, 2006.

APPENDIX

A. Distributions of typing score at the beginning of course

TABLE IV
DISTRIBUTIONS OF TYPING SCORE (NUM OF STUDENTS)

Year (class)	1〜60	61〜80	81〜100	101〜120	121〜140	141〜160	161〜
2018	26	19	10	7	4	0	0
2017	23	12	6	1	1	1	1
2016	4	3	5	27	16	5	9
2015	7	9	11	15	11	5	14
2014 (1)	19	16	18	3	3	8	8
2014 (2)	13	16	12	5	0	0	3
2014 (3)	14	15	12	5	1	1	4

B. List of snippets and keywords provided by the auto-complete function

Note: \n represents line break.

```
void                        setup(){\n}
size(w,h);                  background(200);
fill(,,);                   ellipse(cx,cy,w,h);
noFill();                   strokeWeight(3);
stroke(,,);                 rect(x,y,w,h);
textSize(50);               width
height                      mouseX
mouseY                      int
float                       textAlign(CENTER);
text("",x,y);               println();

text("",width/2,height/2);
if () {\n  } else {\n  }
else
for ( int k=0 ; k<10 ; k++ ) {\n  }

while () {\n  }             mousePressed(){\n}
mouseClicked(){\n}          mouseMoved(){\n}
keyPressed(){\n}            keyCode
frameRate(20);              line(x1,y1,x2,y2);

triangle(x1,y1,x2,y2,x3,y3);
quad(x1,y1,x2,y2,x3,y3,x4,y4);

noStroke();                 mousePressed
mouseButton                 PImage
loadImage("http://");       draw(){\n}

image(img, x, y, img.width, img.height);

class                       new
random(low,high);           floor();
ceil();                     round();
LEFT                        CENTER
RIGHT                       pushMatrix();
popMatrix();                translate(dx,dy);
rotate(radians(90));        colorMode(HSB,255);

textFont(createFont("Arial",20));

ArrayList list;             list = new ArrayList();
list.add();                 list.size();
list.get(0);                list.remove(0);
return ;                    mp3("game01/jump12");
```

C. Frequency of snippet used

TABLE V
FREQUENCY OF SNIPPET USED

count	snippet
123	`if () {\n } else {\n }`
63	`rect(x,y,w,h);`
62	`for ( int k=0 ; k<10 ; k++ ) {\n }`
51	`background(200);`
49	`int`
47	`ellipse(cx,cy,w,h);`
42	`fill(,,);`
26	`mouseX`
24	`mousePressed(){\n }`
24	`frameRate(20);`
24	`void`
21	`else`
16	`random(low,high);`
15	`text("",x,y);`
15	`float`
14	`mouseClicked(){\n }`
12	`draw(){\n }`