A Quasi-Experimental Design to Evaluate the Use of PythonTutor on Programming Laboratory Session

by Oscar Karnalim Mewati Ayub

A Quasi-Experimental Design to Evaluate the Use of PythonTutor on Programming Laboratory Session

https://doi.org/10.3991/ijoe.v14i02.8067

Oscar Karnalim(✉), Mewati Ayub
Maranatha Christian University, Bandung, Indonesia
oscar.karnalim@it.maranatha.edu

Abstract—Educational tool is one of the prominent solutions for aiding students to learn course material in Information Technology (IT) domain. However, most of them are not used in practice since they do not properly fit student necessity. This paper evaluates the impact of an educational tool, namely PythonTutor, for completing programming laboratory task regarding data structure materials. Such evaluation will be conducted in one semester by implementing a quasi-experimental design. As a result, six findings can be deducted which are: 1) PythonTutor might positively affect student performance when the students have used such tool before; 2) Sometimes, student perspective regarding the impact of educational tool is not always in-sync with actual laboratory result; 3) the impact of PythonTutor might be improved when similar data representation is used consequently for several weeks; 4) the correlation between the use of PythonTutor and student performance might not be significant when the control and intervened group share completely different characteristics; 5) the students might experience some difficulties when they are asked to handle a big task for the first time; and 6) the students might be able to complete a particular weekly task with a promising result if the students have understood the material well.

Keywords—quasi-experimental design, empirical evaluation, program visualization, educational tool, laboratory session

1 Introduction

According to the fact that students are one of the most influential resources in University, emerging issues on such domain are focused as research topics. Some of them are tracing alumni [1], predicting student outcome [2], detecting plagiarism among student's assignments [3, 4], and aiding students to learn course material [5]. Among these mentioned issues, we would argue that the latest one is the most urgent issue since it affects student learning performance directly.

For aiding students to learn course material, the use of educational tool has been proved to be effective, especially in Information Technology (IT) domain where most of the course materials involve abstract representation [5, 6]. However, in most cases, such tools are not applicable to be used in real learning environment since they do not fit student necessity [7]. We would argue that one of the main reasons for such unfitness is the lack of empirical evaluation on real learning environment. Consequently, this paper is intended to mitigate the gap by proposing an empirical evaluation of an educational tool. To be specific, we want to evaluate the impact of PythonTutor (i.e. an educational tool to learn programming) for completing data-structure laboratory task in general and specific perspectives in regard to student grade. The findings of this work are expected to provide a brief insight into IT lecturers who plan to incorporate PythonTutor as their supplementary learning tool.

2 Related Works

Although IT is a prominent major for undergraduate students nowadays, some IT students experience difficulties for learning IT materials, especially algorithm [8, 9] and programming [10, 11]. They feel that learning such materials is not a trivial task since most concepts are abstract and require high logical thinking for further understanding. Consequently, to handle such issue, several educational tools for learning both algorithm and programming materials are developed. They are referred as Algorithm Visualization (AV) and Program Visualization (PV) tools respectively. On the one hand, AV tools are focused on providing a brief concept of how standard algorithms work without discussing the implementation [12]. It usually relies on interactive visual and animation in order to keep the user's attention. VisuAlgo [12] and AP-ASD1 [13] are two examples which fall into this category. On the other hand, PV tools are focused on visualizing and animating program aspects based on its runtime execution [14]. It usually displays all information stored on a program in a debug-like manner. Jeliot 3 [15], JIVE [16], VILLE [17], and PythonTutor [18] are several examples which fall into this category.

PythonTutor is a PV tool that is initially aimed at assisting students to learn programming with Python [18]. Unlike other PV tools, PythonTutor is designed as a web-based application with responsive UI. It can be accessed from anywhere as long as the students are connected to the internet. In addition, it can also be used on various machines such as personal computer, laptop, tab, or smartphone.

Based on the fact that several PV tools have been evaluated on real programming courses to measure their effectiveness comprehensively [19, 5], this paper proposes a quasi-experimental design to evaluate the impact of PythonTutor for learning programming in laboratory sessions using students' grade. To our knowledge, it is the first attempt that discusses such impact on given conditions. For our case study, students from 4 classes of Basic Data Structure (BDS) course are considered as our participants. They are asked to use such tool for completing their laboratory task in half of the semesters while experiencing the absence of such tool on the other half semester. Their laboratory results are then used to statistically evaluate PythonTutor's impact for completing programming laboratory task regarding data structure materials in general and specific perspectives.

It is important to note that our work is different with works proposed in [20] and [21] that also evaluate the impact of PythonTutor. On the one hand, our work is different with a work proposed in [20] since our work is more focused on laboratory session instead of theory session. Further, the work in [20] is focused on introductory course while our work is focused on basic data structure course. On the other hand, our work is different with a work proposed in [21] since our work evaluates the impact of PythonTutor through student performance rather than student perspective. Our work will complement the work in [21] by providing a more-objective result (the result of work in [21] is rather subjective since it relies on questionnaire survey toward the students).

3 Methodology

Evaluation will be conducted by performing a quasi-experimental design [22] in 14-lecturer-weeks laboratory sessions of Basic Data Structure (BDS) course. The participants are first-year students who take BDS course on the even semester of 2016/2017. They will be split into two groups before conducting the evaluation. Since there are four classes of BDS course during that semester, we assign two classes for each group. Class A (15 students) and B (10 students) are assigned to the first group, which will act as an intervened group for even weeks, whereas class C (19 students) and D (18 students) are assigned to the second one, which will act as an intervened group for odd weeks. As a result, group 1 consists of 22 students while group 2 consists of 34 students.

One of the predefined groups will be assigned as an intervened group while another one will be assigned as a control group alternately during given laboratory sessions. For odd-week laboratory session, group 1 will be assigned to the control group and group 2 will be assigned to the intervened group. For even-week laboratory session, it will work in reverse: group 1 will be assigned to the intervened group and group 2 will be assigned to the control group. If a group is assigned to a control group, students in that group should complete their laboratory task in a conventional manner (i.e. without using PythonTutor). Otherwise, students in that group should complete their laboratory task with the help of PythonTutor if necessary.

Each group will get similar laboratory tasks for each session; each task should be completed in 80 minutes. The detail of each task, including its assigned intervened group, can be seen in Table 1. Most given laboratory tasks are about implementing data structure concepts in Python programming language to solve problems. Some weekly tasks are split into several smaller sub-tasks to mitigate the difficulty of learning such task. However, the total score for each weekly task is still 100, regardless of how many sub-tasks are involved for each week.

After 14-lecture-weeks laboratory sessions have been conducted, students' laboratory grades will be collected. Those grades will be further analyzed to evaluate PythonTutor's impact for completing programming laboratory task regarding data structure materials in general and specific perspectives. On the one hand, for the general perspective, the impact is measured by comparing the result of intervened sessions toward the un-intervened ones on the same group. This evaluation will be conducted based on paired t-test. On the other hand, for the specific (i.e. lecture week) perspec-

Table 1. Detailed Laboratory Tasks and Their Assigned Intervened Group

Lecture Week	Task Description	Group
1st	create two programs as a review and a program to implement a simple Abstract Data Type (ADT)	Group 2
2nd	create a program to implement a simple ADT and a program to show interaction of two ADT objects.	Group 1
3rd	create a program to use a standard array and a program to manipulate ADT objects in an array	Group 2
4th	create a program to manipulate an ADT of array	Group 1
5th	create a program to convert infix to postfix notation using an ADT of array-based stack	Group 2
6th	create a program to manipulate an ADT of array-based queue	Group 1
7th	create a program to manipulate an ADT of standard linked list	Group 2
8th	create a program to manipulate an ADT of standard linked list with more operations than 7th lecture week	Group 1
9th	create a program to process an intersection between two linked lists	Group 2
10th	create a program to manipulate an ADT of list-based stack and an ADT of list-based queue	Group 1
11th	create a program to manipulate an ADT of list-based priority queue	Group 2
12th	create a program to manipulate an ADT of double pointer linked list	Group 1
13th	create a program to manipulate an ADT of circular linked list	Group 2
14th	create a program to implement shell sort and merge sort using an ADT of array	Group 1

tive, the impact is measured by comparing the result of the intervened group toward the control group (i.e. a group that is not intervened by PythonTutor) for each data structure material. This evaluation will be conducted based on unpaired t-test or Mann-Whitney U test.

4 Result and Discussion

4.1 General Overview of Student Laboratory Results

The statistics of student laboratory results for each group in one semester can be seen in Figure 1. The horizontal axis represents lecture weeks while the vertical axis represents resulted score. Mean refers to the average score for submitted assignments, SD refers to the standard deviation of the score of submitted assignments, and n refers to the number of submitted assignments. All components are based on submitted assignments for each lecture week for that group. According to Figure 1, there are two findings that can be deducted. First, the students might feel some difficulties when they are asked to handle a big task for the first time. Such finding is deducted from the fact that both groups achieved the lowest mean score on the 4th week, the first week on that semester where the students were asked to handle a big task on laboratory session. Second, the students might be able to complete a weekly task with a promising result if the students have understood the material well. Such finding is deducted from the fact that the result of each week varies and some of them are higher than 80 of 100.

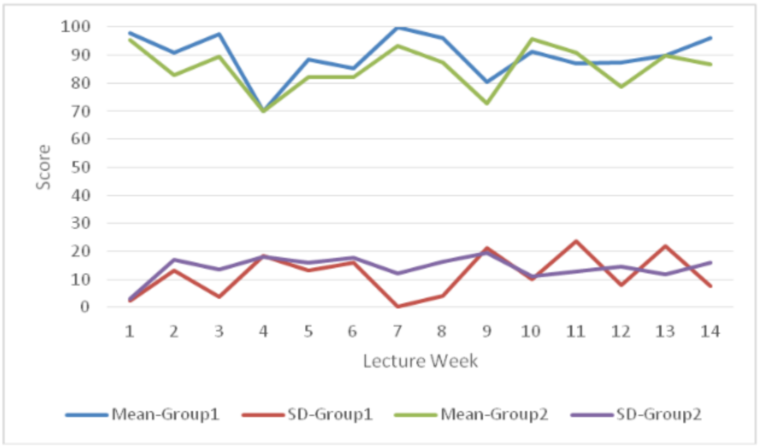

Fig. 1. The comparison of mean and standard deviation of student laboratory results.

4.2 The Results of Evaluating PythonTutor's Impact in General Perspectives

In this evaluation, PythonTutor's impact is measured by comparing the result of intervened sessions toward the un-intervened ones for the same group. In other words, for each group, the average result for the odd weeks will be compared to the average result for the even weeks. Such comparison will be measured based on paired t-test where the result for each group can be seen in Figure 2. The horizontal axis represents evaluated groups while the vertical axis represents resulted score. The p-value for the group 1 is 0.0176, while the p-value for the group 2 is 0.0004. In general, both groups experienced statistically significant difference when learning data structure materials using PythonTutor. Such finding is deducted based on their generated p-value where both are lower than 0.05.

For group 1, the use of PythonTutor is negatively correlated with student performance regarding student grade. Such finding is deducted from the fact that the mean score for the intervened sessions is lower than the score for the un-intervened ones. We would argue that such negative correlation is caused by the fact that most students on the group 1 had never used the tool before they took BDS course [20]. Therefore, they might feel that such tool mitigated them for completing their task, considering that PythonTutor UI is not intuitive enough for students [20], [21].

For group 2, the use of PythonTutor is positively correlated with student performance regarding student grade. Such finding is deducted from the fact that the mean score for the intervened sessions is higher than the score for the un-intervened ones. We would argue that such positive correlation is caused by the fact that most students from group 2 had used the tool before they took BDS course [20]. They might have been adapted to such tool, resulting in a positive impact regarding the use of Python-Tutor. This finding is supported by p-value which is less than 0.01.

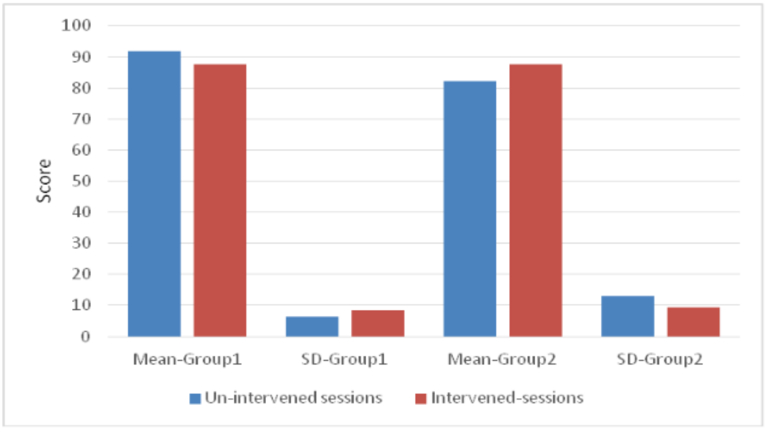

Fig. 2. The comparison of mean and standard deviation of paired tests.

4.3 The Results of Evaluating PythonTutor's Impact in Lecture Week Perspective

In this evaluation, PythonTutor's impact is measured by comparing the result of the intervened group toward the control group per lecture week. Such comparison will be measured based on unpaired tests. If scores involved in such comparison are normally distributed, an unpaired t-test will be selected as our evaluation metric. Otherwise, Mann-Whitney U (MWU) test will be used. The result of this evaluation can be seen in Table 2. Improvements for each week is generated by subtracting mean score from the intervened group with the mean score from the control group. Figure 3 shows the improvement of each lecture week of unpaired tests. Horizontal axis refers to lecture weeks while vertical axis refers to resulted improvement degree. From 14 weeks, only 2 weeks (bolded in Table 2) show that PythonTutor affects student performance significantly ($p\text{-value} < 0.05$). These weeks are the 1st and 12th week.

On the one hand, the 1st week statistically generates 2.44 mean reduction. Hence, it can be stated that the use of PythonTutor negatively affects student performance for completing laboratory task on that week. This finding contradicts the result proposed in [21] which stated that most students from group 2 (a group which acted as the intervened group on the 1st week) felt that the use of PythonTutor affected the most on that week. When discovered further, sometimes, it is natural that such contradiction exists, considering that student perspective and the actual result might not always be in-sync to each other.

On the other hand, the 12th week statistically generates 8.86 mean improvements. Hence, it can be stated that the use of PythonTutor positively affects student performance for completing laboratory task on that week. When discovered further, such finding is natural since a double-pointer linked list is a simple expansion of the standard linked list, a data structure that had been learned on several previous weeks beforehand. We would argue that, at that time, the students from group 1 (a group which acted as the intervened group on the 12th week) had been adapted to the visual representation of standard linked list on PythonTutor, resulting significant improvement in terms of student grade.

Table 2. Evaluation Result of Unpaired Tests

Lecture Week	Evaluation Metric		p-value
	Unpaired t-test	*MWU test*	
1st		✓	**0.002**
2nd	✓		0.076
3rd		✓	0.132
4th	✓		0.955
5th	✓		0.174
6th	✓		0.504
7th		✓	0.115
8th		✓	0.323
9th	✓		0.177
10th		✓	0.139
11th		✓	0.850
12th		✓	**0.024**
13th		✓	0.102
14th		✓	0.058

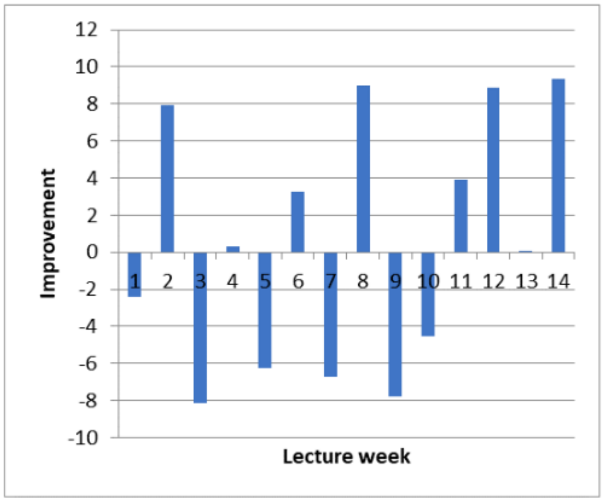

Fig. 3. The improvement of each lecture week of unpaired tests.

For the remaining 12 weeks, the correlation between the use of PythonTutor and student performance is not statistically significant. When discovered further, such finding might be caused by the high variance of student characteristics between both groups. Such variance might include student background, learning style, intelligence, and prior knowledge.

5 Conclusion and Future Work

This paper presents a quasi-experimental design to evaluate the impact of PythonTutor for learning programming in 14-lecture-weeks laboratory sessions. Such sessions were held in even semester of 2016/2017 academic year and involved 4 classes of Basic Data Structure (BDS) course. According to our evaluation, several findings can be deducted which are: 1) PythonTutor might positively affect student performance when the students have used such tool before; 2) Sometimes, student perspective is not always in-sync with actual laboratory result; 3) the impact of PythonTutor might be improved when similar data representation is used consequently for several weeks; 4) the correlation between the use of PythonTutor and student performance might not be significant when the control and intervened group share completely different characteristics; 5) the students might feel some difficulties when they are asked to handle a big task for the first time; and 6) the students might be able to complete a particular weekly task with a promising result if the students have understood the material well.

For future work, we plan to develop a PV tool that, to some extent, is similar with PythonTutor but with more comprehensive features. These features are expected to mitigate the negative feedbacks that are reported in Karnalim & Ayub's work and the results of this work. Hopefully, such tool may help students to learn programming, especially in our university.

6 Acknowledgment

The authors would like to acknowledge the financial support provided by the Maranatha Christian University Research Committee, by means of the Maranatha Christian University Grant.

7 References

[1] H. Toba, E. A. Wijaya, M. C. Wijanto and O. Karnalim, "Enhanced Unsupervised Person Name Disambiguation to Support Alumni Tracer Study," Global Journal of Engineering Education, vol. 19, no. 1, pp. 42-48, 2017.

[2] M. Ayub and O. Karnalim, "Predicting outcomes in introductory programming using J48 classification," World Transactions on Engineering and Technology Education (WTE&TE), vol. 15, no. 2, pp. 132-136, 2017.

[3] O. Karnalim, "Detecting Source Code Plagiarism on Introductory Programming Course Assignments Using a Bytecode Approach," in Proc. 10th International Conference on In-

formation & Communication Technology and Systems (ICTS), Surabaya, 2016. https://doi.org/10.1109/ICTS.2016.7910274

[4] O. Karnalim, "Python Source Code Plagiarism Attacks on Introductory Programming Course Assignments," Themes in Science and Technology Education, vol. 10, no. 1, 2017.

[5] E. Kaila, T. Rajala, M.-J. Laakso and T. Salakoski, "Effects of Course-Long Use of a Program Visualization Tool," in Australian Computing Education Conference, Brisbane, 2010.

[6] F. C. Jonathan, O. Karnalim and M. Ayub, "Extending the Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development," in Seminar Nasional Aplikasi Teknologi Informasi, Yogyakarta, 2016.

[7] J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide, "A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems," ACM Transactions on Computing Education (TOCE) - Special Issue on the 5th Program Visualization Workshop (PVW'08), vol. 9, no. 2, 2009. https://doi.org/10.1145/1538234.1538236

[8] E. Elvina and O. Karnalim, "Complexitor: An Educational Tool for Learning Algorithm Time Complexity in Practical Manner," ComTech: Computer, Mathematics and Engineering Applications, vol. 8, no. 1, pp. 21-27, 2017. https://doi.org/10.21512/comtech.v8i1.3783

[9] S. Zumaytis and O. Karnalim, "Introducing an Educational Tool for Learning Branch & Bound Strategy," Journal of Information Systems Engineering and Business Intelligence, vol. 3, no. 1, pp. 8-15, 2017. https://doi.org/10.20473/jisebi.3.1.8-15

[10] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. Kolikant, C. Laxer, L. Thomas, I. Utting and T. Wilusz, "A Multi-National, Multi-Institutional Study of Assessment of Programming Skill of First-year CS Students," ACM SIGSCE Bulletin, vol. 33, no. 4, pp. 125-180, 2001.

[11] R. Lister, S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Mostrom, K. Sanders, O. Seppala, B. Simon and L. Thomas, "A Multi-National Study of Reading and Tracing Skills in Novice Programmers," ACM SIGSCE Bulletin, vol. 36, no. 4, pp. 119-150, 2004. https://doi.org/10.1145/1044550.1041673

[12] S. Halim, Z. C. Koh, V. B. H. Loh and F. Halim, "Learning Algorithms with Unified and Interactive Web-Based Visualization," Olympiads in Informatics, vol. 6, pp. 53-68, 2012.

[13] L. Christiawan and O. Karnalim, "AP-ASD1: An Indonesian Desktop-based Educational Tool for Basic Data Structures," Jurnal Teknik Informatika dan Sistem Informasi (JuTISI), vol. 2, no. 1, pp. 21-30, 2016.

[14] S. Bentrad and D. Meslati, "Visual Programming and Program Visualization- Toward an Ideal Visual Software Engineering System -," ACEEE International Journal on Information Technology, vol. 1, no. 3, pp. 43-49, 2011.

[15] A. Moreno, N. Myller, E. Sutinen and M. Ben-Ari, "Visualizing programs with Jeliot 3," in The Working Conference on Advanced Visual Interfaces, Gallipoli, 2004. https://doi.org/10.1145/989863.989928

[16] P. Gestwicki and B. Jayaraman, "Interactive Visualization of Java Programs," in Symposia on Human Centric Computing Languages and Environments, Arlington, 2002. https://doi.org/10.1109/HCC.2002.1046375

[17] T. Rajala, M.-J. Laakso, E. Kaila and T. Salakoski, "VILLE - A Language Independent Program Visualization Tool," in Proc. 7th Baltic Sea Conference on Computing Education Research, Finland, 2007.

[18] P. J. Guo, "Online python tutor: embeddable web-based program visualization for cs education," in Proc. 44th ACM technical symposium on Computer science education, Denver, 2013. https://doi.org/10.1145/2445196.2445368

[19] S. M. Cisar, D. Radosav, R. Pinter and P. Cisar, "Effectiveness of Program Visualization in Learning Java: A Case Study with Jeliot 3," International Journal of Computers, Communications & Control, vol. 6, no. 4, pp. 668-680, 2011. https://doi.org/10.15837/ijccc.2011.4.2094
[20] O. Karnalim and M. Ayub, "The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor," CommIT (Communication and Information Technology) Journal, vol. 11, no. 2, 2017, in press.
[21] O. Karnalim and M. Ayub, "The Use of PythonTutor on Programming Laboratory Session: Student Perspectives," KINETIK Journal, vol. 2, no. 4, 2017.
[22] J. W. Creswell, Educational Research Planning, Conducting, and Evaluating Quantitative and Qualitative Research, New Jersey: Pearson, 2008.

8 Authors

Oscar Karnalim is with Faculty of Information Technology, Maranatha Christian University, Indonesia (ORCID: http://orcid.org/0000-0003-4930-6249). His research interests are Computer Science Education, Software Engineering, and Information Retrieval.

Mewati Ayub is with Faculty of Information Technology, Maranatha Christian University, Indonesia. Her research interests are Data Mining, Computer Science Education, and Software Engineering.

Article submitted 04 December 2017. Resubmitted 10 January 2018. Final acceptance 16 January 2018. Final version published as submitted by the authors.

A Quasi-Experimental Design to Evaluate the Use of PythonTutor on Programming Laboratory Session

ORIGINALITY REPORT

9%	3%	3%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Muhammadiyah Malang Student Paper	4%
2	Submitted to Universidad Militar Nueva Granada Student Paper	1%
3	www.tandfonline.com Internet Source	1%
4	Submitted to University of Newcastle upon Tyne Student Paper	<1%
5	jitecs.ub.ac.id Internet Source	<1%
6	Oscar Karnalim. "An abstract method linearization for detecting source code plagiarism in object-oriented environment", 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017 Publication	<1%

7	thinkmind.org Internet Source	<1%
8	Oscar Karnalim. "Software Keyphrase Extraction with Domain-Specific Features", 2016 International Conference on Advanced Computing and Applications (ACOMP), 2016 Publication	<1%
9	academic.oup.com Internet Source	<1%
10	comengapp.unsri.ac.id Internet Source	<1%
11	Lisan Sulistiani, Oscar Karnalim. "ES-Plag: Efficient and sensitive source code plagiarism detection tool for academic environment", Computer Applications in Engineering Education, 2018 Publication	<1%
12	Linda Mannila, Mia Peltomäki, Tapio Salakoski. "What about a simple language? Analyzing the difficulties in learning to program", Computer Science Education, 2007 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		